North Carolina Agricultural and Technical State University

Aggie Digital Collections and Scholarship

Theses

Electronic Theses and Dissertations

2011

Building And Implementing An Accessbased Computational System That Produces And Plots Conditional Production Possibilities Frontier For Corn And Soybean Production Using Vba Programming Language

Sidibe Mariama Oumarou

North Carolina Agricultural and Technical State University

Follow this and additional works at: https://digital.library.ncat.edu/theses

Recommended Citation

Oumarou, Sidibe Mariama, "Building And Implementing An Accessbased Computational System That Produces And Plots Conditional Production Possibilities Frontier For Corn And Soybean Production Using Vba Programming Language" (2011). *Theses*. 41.
https://digital.library.ncat.edu/theses/41

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at Aggie Digital Collections and Scholarship. It has been accepted for inclusion in Theses by an authorized administrator of Aggie Digital Collections and Scholarship. For more information, please contact iyanna@ncat.edu.

BUILDING AND IMPLEMENTING AN ACCESS-BASED COMPUTATIONAL SYSTEM THAT PRODUCES AND PLOTS CONDITIONAL PRODUCTION POSSIBILITIES FRONTIER FOR CORN AND SOYBEAN PRODUCTION USING VBA PROGRAMMING LANGUAGE.

by

Mariama Oumarou Sidibe

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Department: Computational Science and Engineering
Major: Computational Science and Engineering
Major Professor: Dr. Lyubov Kurkalova

North Carolina A&T State University
Greensboro, North Carolina
2011

School of Graduate Studies
North Carolina Agricultural and Technical State University

This is to certify that the Master's Thesis of

Mariama Oumarou Sidibe

has met the thesis requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2011

Approved by:

Dr. Lyubov Kurkalova Major Professor	Dr. Marwan Bikdash Committee Member
Dr. Ken Flurchick Committee Member	Dr. Marwan Bikdash Department Chairperson

Dr. Sanjiv Sarin
Dean of Graduate Studies

DEDICATION

This thesis is dedicated to my husband Amadou Mounkaila Yacouba. You are the most generous person I ever met in my life. Thank you for your love, support and patience since we have been together. I am eternally grateful.

Also, I dedicate this work to my son Cheick Mouslim, for understanding all of the times when I could not be there due to projects and academic obligations. May God bless you! It is also dedicated to my lovely mother Haouaou Altine for her unconditional love and support, my brothers and sisters and to all my family and friends who supported and helped me accomplish my goal in getting higher education.

BIOGRAPHICAL SKETCH

Mariama Oumarou Sidibe was born on September 14, 1980 in Niamey, Niger, West Africa. Speaking four languages: French, Djerma, Haouassa and Fulfulde, Mariama graduated from Lycee Mariama High School in 2000 and attended Mohamed V- Souissi University of Rabat, Morocco where she earned her Bachelor of Science degree in Economics in 2005.

After graduating from college, Mariama worked as an agent of credit in Niamey, Niger for a period of one year.

In December, 2006 Mariama immigrated to the United States of America and stayed between Oklahoma and New York for a period of eighteen months to learn English as a fifth language for graduate study, then transferred to North Carolina where she attended the Forsyth community college as a transfer student from August 2008 to May 2008.

She joined North Carolina Agricultural and Technical State University in 2009 for Master's degree in Computational Science and Engineering. Mariama is married to Amadou Mounkaila Yacouba, a man from her country and they have a three years old son named Cheick Mouslim Amadou Mounkaila.

ACKNOWLEDGMENTS

I would like to thank almighty God for always leading me through my life, granting me the opportunity and providing me with the strength to accomplish this thesis.

I would like to express deep gratitude to my advisor, Dr. Lyubov Kurkalova, for her wonderful support and guidance throughout my graduate career. I am very much thankful to her for giving me this opportunity to work for such a challenging project.

I would like to thank Dr. Marwan Bikdash and Dr. Ken Flurchick for agreeing to chair the defense committee. My special thanks to Dr. Ken Flurchick for introducing me to the world of programming and always providing feedback when asked. I extend my thanks to Dr. Ram Mohan for referring me to my major advisor.

I am very much grateful to Dr. Ajit Kelkar, whose guidance and support has helped me complete this thesis successfully. I am also grateful to Ms. Lydia Leak of the CSE program.

I would like to thank my son and my husband for their love, support and seemingly endless editing of this thesis. Their pride and belief in my abilities and capabilities were of immeasurable importance to the completion of my research.

Last but definitely not least, my greatest recognition to my family and friends whose love, encouragement and support have always been with me. Without their support, this work would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADO	ActiveX Data Objects
CSR	Corn Suitability Rating
cPPF	conditional Production Possibility Frontier
C1C1sr	Continuous corn with conventional and with stover harvesting
C2C2sr	Continuous corn with mulch and with stover harvesting
C3C3sr	Continuous corn with no-till and with stover harvesting
C1C1ns	Continuous corn with conventional and without stover harvesting
C2C2ns	Continuous corn with mulch and without stover harvesting
C3C3ns	Continuous corn with no-till and without stover harvesting
C1S1sr	Corn after soybeans with conventional and with stover harvesting
C2S2sr	Corn after soybeans with mulch and with stover harvesting
C3S3sr	Corn after soybeans with mulch and with stover harvesting
C1S1ns	Corn after soybeans with conventional and without stover harvesting
C2S2ns	Corn after soybeans with mulch and without stover harvesting
C3S3ns	Corn after soybeans with mulch and without stover harvesting
C1C1S1sr	Corn-corn-soybeans rotation with conventional and with stover harvesting
C2C2S2sr	Corn-corn-soybeans rotation with mulch and with stover harvesting
C3C3S3sr	Corn-corn-soybeans rotation with mulch and with stover harvesting
C1C1S1ns	Corn-corn-soybeans rotation with conventional and without stover harvesting

C2C2S2ns	Corn-corn-soybeans rotation with mulch and without stover harvesting
C3C3S3ns	Corn-corn-soybeans rotation with mulch and without stover harvesting
DAO	Data Access Objects
DBMS	Database Management System
IDE	Integrated Development Environment
PPF	Production Possibility Frontier
RDBMS	Relational Data Base Management Systems
SQL	Structured Query Language
VBA	Visual Basic Application
y_c	Corn per acre yield
y_s	Soybean per acre yield
y_{stover}	Stover per acre yield

ABSTRACT

Oumarou Sidibe, Mariama. BUILDING AND IMPLEMENTING AN ACCESS-BASED COMPUTATIONAL SYSTEM THAT PRODUCES AND PLOTS CONDITIONAL PRODUCTION POSSIBILITIES FRONTIER FOR CORN AND SOYBEAN PRODUCTION USING VBA PROGRAMMING LANGUAGE. (**Major Advisor: Dr. Luba Kurkalova**), North Carolina Agricultural and Technical State University.

The objective of the present study is to build and implement a database application system that produces and plots conditional Production Possibilities Frontier (cPPF) for corn and soybean including a user friendly interface, where the user can easily interact with the data. It is an effort to provide economists with an accessible and easy to use data management tool which will facilitate the future analyses of the economy-wide impacts of bioenergy production and policy.

The study uses the economic modeling systems operating on field-level, GIS-based cropping history and soils data developed for the state of Iowa [1].

The database was implemented based on single-tier application using the Microsoft Access application. A single-tier application is one where the entire application is contained and runs on a single computer [2]. The main reason for choosing Microsoft Access over others database management system (DBMS) products such as Oracle, SQL Server or MySQL was the user's natural choice: Microsoft Office Access is a great engine that is easy to learn and cost effective [3], the user wished to have a user interface that is easy to interact with. Therefore Microsoft Office Access with its great user interface design served fine.

CHAPTER 1

INTRODUCTION

This chapter describes the motivation of research, followed by a brief definition of conditional production possibilities frontier, then the data source is introduced. This section presents also a detailed overview of the goals of the present study. Finally the chapter concludes with an outline of the remaining chapters organized in this thesis.

1.1 Motivation of Research

Cellulosic ethanol production technologies have undergone an accelerated development in the United States and now are expected to lead to the establishment of viable markets for corn residues (stover), which are comprised of corn stalks, cobs, and leaves left in the field after grain harvest. With corn stover being a by-product of corn production in essence, a large, viable market for stover will alter the profitability of corn relative to other traditional row crops and may affect significantly the supplies of both corn and other crops. Empirical estimates of the stover market impact on the production and acreage under traditional row crops are scarce [4]. Therefore, an analysis to show how farmers would respond to the market and policy incentive in providing such feedstock for cellulosic biofuels is urgently needed [5], thus the capability to provide an accessible and easy to use data management tool which would determine how much corn and soybeans can be producing for any given resources constitutes an essential feature to the future analyses of the economy-wide impacts of bioenergy production and policy.

1.2 Production Possibilities Frontier (PPF)

A production-possibility frontier (PPF), sometimes called a production-possibility curve or product transformation curve, is a graph that shows the different rates of production of two goods and/or services that an economy can produce efficiently during a specified period of time with a limited quantity of productive resources, or factors of production. The production possibility frontier shows the maximum amount of one commodity that can be obtained for any specified production level of the other commodity (or composite of all other commodities), given the society's technology and the amount of factors of production available [6].

In the present study the production possibility frontier (PPF) is the boundary between the sets of feasible and infeasible output combinations of corn and soybean production. It is explicitly conditional on the factors that the user specifies the level of inputs (labor, diesel, LP gas, fertilizer), thus the name conditional production possibility frontier (cPPF). The cPPF tells us the maximum amount of each product corn or soybean that can be produced given a conditional level of user inputs and how the frontier could shift if the user changes the level of inputs.

1.3 Economic Modeling

The study uses the economic modeling systems operating on field-level, GIS-based cropping history and soils data developed for the state of Iowa [1], given that the state of Iowa possesses the largest quantity of corn stover in the United States [7]. Therefore, the state of Iowa is posed to play a major role in the bioeconomy [8].

1.4 Research objectives

The present research effort aims towards building and implementing a Microsoft Office Access database system using the Visual Basic Application programming language. The specific objectives of the present investigation were as follows:

- Designing a user friendly interface where the user can easily interact with the data.
- Pulling the data from the user's inputs (price of diesel, price of corn Stover, price of soybean and percentage of Stover collected).
- Output of the table that relates alternative corn-to-soybean price ratios to corn production and soybean production.
- Producing the two-dimensional plot of cPPF (corn production along the x-axis, and soybean production along the y-axis).
- Producing the three-dimensional plot of cPPF (corn production along the x-axis, soybean production along the y-axis, and stover price along the z-axis)

1.5 Overview of the chapters

Mainly, this thesis is divided into six major chapters. The next chapter discusses the fundamentals and concepts of a database. Chapter 3 deals with the different computational formulas and requirements specifications. Chapter 4 discusses the design of the database. This chapter also presents the technologies used to implement the database. Chapter 5 discusses the building and implementation details of the database 'creation, including building table structures, user interface, the VBA code, the queries and plotting the conditional production possibilities frontier. The final chapter presents conclusion and recommendations for future work.

CHAPTER 2

DATABASE FUNDAMENTALS AND CONCEPTS

This chapter presents an overview of database. This includes a brief review of database concepts and terminology.

2.1 Definition of Database

A database is an organized storehouse of data, it is defined as a collection of data which is structured in a particular format and which is logically related. The size and complexity of a database may vary as the situation warrants.

The raw data are the building blocks of information. A well formed database is a very good source of information and hence it requires a planned and systematic approach to designing a proper database.

A database system consists of two major parts namely the Database Management System and the Database Application. The Database Management System (DBMS) is the program that organizes the data and maintains the information. The Database Application is the program that lets the user view, retrieve and update the information stored in the DBMS.

2.2 Database Management System (DBMS)

DBMS (Data base management systems) [9] is a collection of programs that enable the user to create and maintain a database. It helps in defining, constructing and manipulating a database. Defining a database involves specifying the data types and the various applicable constraints. Constructing a database involves entering the data into the memory, and manipulation includes various queries made to the database.

The primary objective of a database management system is to provide a convenient environment to store and retrieve database information. Database systems support single user and multiple user scenarios. While the single user system allows only one person to access the database at a given time, multiple user system, allows many users to simultaneously access the database.

The intended use and capabilities of a DBMS are [10]:

2.2.1 Controlling redundancy

Redundancy in storing the same data multiple times can lead to several problems such as duplication of effort, wastage of storage space, inconsistent data, difficulty in management and manipulation of the database. The DBMS software makes sure that the data entered is not redundant; the various triggers such as when to validate an item are activated when any redundant data is being inputted.

2.2.2 Restricting unauthorized access

The Database Administrator can program the DBMS to restrict unauthorized access.

2.2.3 Providing multiple user interfaces

Various users are involved in accessing a database and running different queries. The DBMS helps provide different user interfaces such as a query language for the user, programming language interface for the application programmer and forms for the end user

2.2.4 Providing backup and recovery

Backup and recovery is an essential requirement of a DBMS. In case of a system failure during the middle of a complex program it is essential that the database be restored to the state it was in, and the recovery system must ensure that the program can be effectively resumed.

2.2.5 Ability to represent complex relationships between data items

Any database can include a large amount of data, which can be interrelated in many possible ways. The RDBMS (Relational Data Base Management Systems) should be able to display these relationships effectively so it is able to assist the end user in understanding the various relationships existing in the data.

2.2.6 Enforcing integrity constraints

Most databases require certain integrity constraints to be enforced. The integrity constraints are essential to ensure the accuracy of the data retrieval from the database.

There are two rules for data integrity: entity integrity and referential integrity. Entity integrity merely states that the primary key cannot have an empty value. Referential integrity ensures that every instance of a foreign key matches with the primary key value in a relationship.

2.3 Database Models

DBMSs have evolved through a number of technological stages since their introduction in the 1960s. The most significant change has been the type of model used to represent and access the content of the physical data store [11]. Two major categories have been widely used

2.3.1 Object based logical models

The object based logical model can be defined as a collection of conceptual tools for describing data, data relationships and data constraints.

2.3.2 Record based logical models

The Record based logical model describes the data structure and access techniques of a database management system. There are three types of record based logical database models. They are as follows:

2.3.2.1 Hierarchical model

The Hierarchical database model is exactly as the name suggests. The data are organized in a parent-child relationship structure. The database can be thought of as a logical tree where the origin of data is the root. The data located at different levels along a particular branch from the root is called the node. The last node in the series is called the leaf. This model supports only a "one to many" relationship. The main disadvantage in this system is that a new level in data cannot be inserted easily and requires a change in the tree structure. Also there is a tendency to have multiple copies of data in different levels thereby causing data redundancy.

2.3.2.2 Network model

The Network model brings about the "many-to-many" relationship in the data. The relationship between many data items is called sets. This is a slightly advanced system and uses data pointers to locate specific records. However, when the size and volume of data stored in a network model increases it becomes increasingly difficult to locate data as all individual models apply pointers and it becomes very complex with so many pointers.

2.3.2.3 Relational model

To avoid these inherent disadvantages and complexities, Dr. E.F. Codd of IBM's San Jose Research Laboratories developed the relational database model in1970.This relational model allows all data to be represented in a simple row-column format. Each data field is considered as a column and each record a row of the table.

The relational database model is implemented through a very sophisticated relational database management system (RDBMS). Examples of RDBMS systems are database products from Microsoft (Access), Oracle (Oracle 8i), and IBM (DB2). RDBMS systems are widely used in corporations, small business, and personal databases; it performs the same basic functions provided by the hierarchical and network DBMS system plus a host of other functions that make the relational database model easier to understand and to implement. It creates transitory virtual pointers to records of relational tables in memory. Virtual pointers appear as they are needed to relate (join) tables and are disposed of when the relation is no longer required by a database application. Joins are created between primary key fields and foreign key fields of relational tables. A primary key is a unique identifier for a row in a table. A foreign key, however, is a primary key that is duplicated onto another table. Matching the keys from the two tables forms a relationship.

The relational database is a single data repository in which data independence is maintained. However, the relational database model should obey Codd's twelve rules for it to be relational. The twelve rules are briefly discussed below [12]:

- **The Information rule**: All information is explicitly and logically represented in tables as data values.
- **The rule of guaranteed access:** Every item of data must be logically addressable with the help of table name, primary key value and column name. From this it is clear that any individual record can be retrieved with the use of a table name primary value of the row and the column name where it is to be found.
- **The Systematic treatment of all null values**: The DBMS must be able to support null values to represent missing or inapplicable information. They must be distinct from zeros and spaces. Null values for all data types must be the same. One of the most important aspects that must be noted here is that there is a vast difference between a null value and zero and a space.
- **The database description rule**: A description of the database is maintained using the same logical structures with which data has been defined by the DBMS. These are accessible to users with appropriate authority and are stored in the data dictionary.

- **Comprehensive data sub language**: According to this rule, the system must support the following: Data definition, View definition, Data manipulation, Integrity constraints, Authorization and Transaction management operations
- **The view updating rule**: All views that are theoretically updateable must also be updateable by the system.
- **The insert and update rule**: A single operand must hold good for all retrieval, update, delete and insert activities. This rule implies that all the data manipulation commands must be operational on sets of rows in relation rather than on a single row.
- **The physical independence rule**: Application programs must remain unimpaired when any changes are made in storage representation or access methods.
- **The logical data independence rule**: The changes that are made should not affect the user's ability to work with the data. The change can be splitting the table into many more tables.
- **The Integrity independence rule**: The integrity constraints should be stored in the system catalog or in the database as a table.
- **The Distribution rule**: The system must be able to access or manipulate the data that is distributed in other systems.
- **The Nonsubversion rule**: The nonsubversion rule states that different levels of the language cannot subvert or bypass the integrity rules and constraints. Simply put, if an RDBMS supports a lower level language then it should not bypass any integrity constraints defined in the higher level.

CHAPTER 3

COMPUTATIONAL FORMULAS AND REQUIREMENTS SPECIFICATIONS

This chapter gives a detailed explanation of the different computational formulas used to implement the VBA code. Those formulas were provided by Dr. Luba Kurkalova. Additionally, this chapter also specifies the requirements the user expects from the database.

3.1 Computational Formulas

For a given set of corn and stover prices, the computations are first conducted on the individual CSR (corn Suitability Rating values 1 through 100) and then the state-total quantities are derived. The cPPF is derived as the collection of state-total quantities for varying corn and stover prices.

3.1.1 CSR-level computations

The corn Suitability Rating (CSR) measures land's productivity in crop production. It is an index from zero to one hundred and each CSR represents a farmer with the number of acres represented by the CSR acreage. Each farmer makes the rotation-tillage-stover harvesting choice based on the highest profit. There are a total of eighteen choices:

- Six choices correspond to continuous corn with conventional, mulch, or no-till with stover harvesting (C1C1sr, C2C2sr, C3C3sr) and those without stover harvesting (C1C1ns, C2C2ns, C3C3ns),

- Six choices correspond to corn after soybeans with conventional, mulch, or no-till with stover harvesting (C1S1sr, C2S2sr, C3S3sr) and those without stover harvesting (C1S1ns, C2S2ns, C3S3ns), and
- Six choices correspond to corn-corn-soybeans rotation with conventional, mulch, or no-till with stover harvesting (C1C1S1sr, C2C2S2sr, and C3C3S3sr) and those without stover harvesting (C1C1S1ns, C2C2S2ns, and C3C3S3ns).

For each of the choices, the profits are computed as the yearly-average difference between revenue and the costs of production, with the average taken over the length of rotation (one year for continuous corn, two years for corn after soybeans, and three years for corn-con-soybean rotation).

3.1.1.1 Revenues

For each year of rotation, the expected revenue is the product of crop price and expected per acre yield. If the crop is corn and stover is harvested, then the revenue increases by the product of Stover price and expected stover yield times the percentage of Stover collected.

The crop yield (bu/ac) is computed as 2.25*CSR for corn and 0.67*CSR for soybean, and then adjusted for the yield drag due to rotation and tillage. The final crop yield formulas are as follows:

For corn: $y_c(CSR, prev.crop, tillage) = 2.25 \times CSR \times drag$

For soybeans: $y_s(CSR, prev.crop, tillage) = 0.67 \times CSR \times drag$

The values of the yield drag are provided in the Table 3.1 below.

Table 3.1 Yield Drag Assumptions

Crop	**Previous Crop**	**Tillage**	**Yield Drag**
Corn	Corn	1	0.95
Corn	Corn	2	0.9
Corn	Corn	3	0.8
Corn	Soybeans	1	1
Corn	Soybeans	2	1
Corn	Soybeans	3	0.95
Soybeans	Corn	1	1
Soybeans	Corn	2	1
Soybeans	Corn	3	0.8

The stover per acre yield (kg/ac) is estimated as a multiple of corn yield:

$$y_{stover}(y_c, sr, ssr) = 21.5 \cdot sr \cdot ssr \cdot y_c$$

Here *sr* stands for a dummy variable that takes on the value 1 if Stover has been chosen to be collected and zero otherwise, and *ssr* is the proportion of Stover collected.

3.1.1.2 Costs

For each year of rotation, the costs are the costs associated with the crop grown. If the crop is corn and Stover is harvested, then the costs increase by the cost of Stover collection. The costs of crop production have been estimated as follows:

$$C_{i\ following\ j, tillage\ t} = a_0 + a_1 \cdot y_i + a_2 \cdot p_{dsl} + a_3 \cdot p_{LPG} \cdot y_i$$
$$+ a_4 \cdot p_N \cdot r_{N,i,j} + a_5 \cdot p_P \cdot r_{P,i,j} + a_6 \cdot p_K \cdot r_{K,i,j}$$

Where

$C_{i\ following\ j,\ tillage\ t}$ is the per acre cost of growing crop *i* following crop *j* using the tillage system *t*, \$ per acre.

i is the crop grown (corn or soybeans),

j is the crop grown previous year (corn or soybeans),

t is the tillage system used (conventional, mulch, or no-till),

y*i* is the expected yield for the current year crop (bu/ac),

p_{dsl} is the price of diesel fuel, \$ per gallon,

p_{LPG} is the price of LP Gas, \$ per gallon,

p_N is the price of Nitrogen fertilizer, \$ per pound,

$r_{N,i,j}$ is the rate of Nitrogen fertilizer used to grow the *i*-th crop after the j-th crop, lb per acre,

p_p is the price of Phosphate fertilizer, \$ per pound,

$r_{p,i,j}$ is the rate of Phosphate fertilizer used to grow the *i*-th crop after the *j*-th crop, pounds per acre,

p_k is the price of Potash fertilizer, \$ per pound, and

$r_{k,i,j}$ is the rate of Potash fertilizer used to grow the *i*-th crop after the *j*-th crop, pounds per acre.

The prices of fertilizer and LP gas are estimated from the price of diesel according to the following formulas:

$$p_{Nitrogen}\left(\$ / lb\ Nitrogen\right) = 0.069 + 0.089 \cdot p_{diesel}\left(\$ / gal\right)$$

$$p_{Phosphate}\left(\$ / lb\ Phosphate\right) = 0.315 + 0.064 \cdot p_{diesel}\left(\$ / gal\right)$$

$$p_{Potash}\left(\$ / lb\ Potash\right) = 0.120 + 0.0561 \cdot p_{diesel}\left(\$ / gal\right)$$

$$p_{LPG}(\$ / gal) = 0.058 \quad + 0.680 \cdot p_{dsl}(\$ / gal)$$

The budgets are summarized in Table 3.2.

Table 3.2 Parameters of costs of production of individual crops

Crop *i*	**Previous crop *j***	**Tillage *t***	a_0	a_1	a_2	a_3	a_4	a_5	a_6
Corn	Corn	Conv.	229.85	0.15	4.92	0.10	1.05	1.05	1.05
Corn	Corn	Mulch	222.17	0.15	3.76	0.10	1.05	1.05	1.05
Corn	Corn	No-till	212.25	0.15	2.76	0.10	1.05	1.05	1.05
Corn	Soybeans	Conv.	222.17	0.15	3.76	0.10	1.05	1.05	1.05
Corn	Soybeans	Mulch	215.12	0.15	3.18	0.10	1.05	1.05	1.05
Corn	Soybeans	No-till	212.25	0.15	2.76	0.10	1.05	1.05	1.05
Soybeans	Corn	Conv.	142.85	0.06	4.34	0	0	1.05	1.05
Soybeans	Corn	Mulch	152.40	0.06	2.76	0	0	1.05	1.05
Soybeans	Corn	No-till	145.34	0.06	2.18	0	0	1.05	1.05

Assumptions on fertilizer rates are detailed in Table 3.3.

Table 3.3 Assumed fertilizer rates

Crop *i*	**Previous crop *j***	**Tillage *t***	$\mathbf{r_{N,i,j}}$	$\mathbf{r_{P,i,j}}$	$\mathbf{r_{K,i,j}}$
Corn	Corn	Conv.	175	55	45
Corn	Corn	Mulch	175	55	45
Corn	Corn	No-till	175	55	45
Corn	Soybeans	Conv.	120	60	50
Corn	Soybeans	Mulch	120	60	50
Corn	Soybeans	No-till	120	60	50
Soybeans	Corn	Conv.	0	40	75
Soybeans	Corn	Mulch	0	40	75
Soybeans	Corn	No-till	0	40	75

The costs of stover collection are given as follows:

$$\begin{aligned} C\ (\$/ac) = {} & 12.125 \\ & + 1.1 \cdot p_{diesel} \\ & + 21.5 \cdot y_c \times (0.01020851 + \\ & + 0.00891947 \cdot p_{Nitrogen} \\ & + 0.00263124 \cdot p_{Phosphate} \\ & + 0.01114933 \cdot p_{Potash}) \end{aligned}$$

3.1.2 State-level computations

There are two steps for calculating the total crop production as detailed below.

3.1.2.1 Calculation of Effective Crop Yield

Calculation of effective crop yield takes into account land quality (CSR), the crop yield as calculated before, and the assigned rotation. To account for the rotation factor, if land is predicted to be in CS rotation, then ½ of every acre is assumed to be in corn, and the other ½ acre is assumed to be in soybeans. Similarly, if land is in CCS rotation, then 2/3 of every acre is assumed to be in corn, and the other 1/3 acre is assumed to be in soybeans. Finally, if land is predicted to be in continuous corn (CC), then every acre is assumed to be in corn and no soybean production is assumed on that land. The effective crop yield formulas are as follows:

$$ey_c(CSR, rotation) = y_c \times 0.5^{C1S1+C2S2+C3S3} \times (1/3)^{C1C1S1+C2C2S2+C3C3S3}$$

$$\begin{aligned} ey_s(CSR, rotation) = {} & y_s \times (1 - C1C1 - C2C2 - C3C3) \\ & \times 0.5^{C1S1+C2S2+C3S3} \times (1/3)^{C1C1S1+C2C2S2+C3C3S3} \end{aligned}$$

Here *y* denotes yield in bu/ac as calculated for each CSR, and subscripts *c* and *s* denote corn and soybeans, respectively. The *rotation-tillage choice* is captured by a set of dummy variables as follows. *C1C1* stands for a dummy variable that takes on the value 1 if the predicted rotation-tillage choice is continuous corn and conventional tillage, and 0 otherwise. The dummy variables *C2C2* and *C3C3* are defined similarly with tillage code 2 corresponding to mulch till, and tillage code 3 corresponding to no-till.

The other rotation-tillage choice dummies are defined in a similar manner: *C1S1* stands for a dummy variable that takes on the value 1 if the predicted rotation-tillage choice is CS and conventional and 0 otherwise and *C1C1S1* stands for a dummy variable that takes on the value 1 if the predicted rotation-tillage choice is CCS and conventional and 0 otherwise. The dummy variables *C2S2, C3S3, C2C2S2* and *C3C3S3* are defined similarly with tillage code 2 corresponding to mulch till, and tillage code 3 corresponding to no-till.

3.1.2.2 The Total Crop Production

The total crop production is then calculated by multiplying the corresponding effective yields by the number of acres in each CSR category and summing the numbers over all CSRs:

$$pcrop = \sum_{CSR} ey(CSR) \times acres(CSR)$$

Here *pcrop* denotes the total corn or soybean production in bu, e*y* denotes the corresponding crop's effective yield in bu/ac, and *acres* are the number of acres in the corresponding CSR.

3.1.3 Construction of cPPF

To calculate the cPPF, the CSR and state-level calculations for alternative corn and soybean prices were repeated. The alternative corn and soybean prices are chosen so that the soybean-to-corn price ratio varies from 1.5 to 3.5, with a step of 0.5, i.e., for the ratio values of 1.5, 2.0, 2.5, 3.0, and 3.5. For the purposes of these calculations, assume that the corn price is fixed at $4/bu.

3.2 Requirements Specifications

Before designing any application, the first step is to specify the requirements that the end user of the application expects from the system. Following are the requirements needed to design the database.

- Users should be able to input (price of diesel, price of corn stover, percentage of Stover collected), store the data and also browse through the stored data easily through a user-friendly interface.
- The system should be able to pull the data from the user's inputs (price of diesel, price of corn stover, percentage of stover collected).
- The system should output the table that relates alternative corn-to soybean price ratios to corn production and soybean production.
- The system should be able to produce the two-dimensional plot of cPPF (corn production along the x-axis, and soybean production along the y-axis).
- The system should be able to produce the three-dimensional plot of cPPF (corn production along the x-axis, soybean production along the y-axis, and Stover price along the z-axis).
- The data should be presented to the user in a simple and useful manner.

CHAPTER 4

DESIGING THE DATABASE

This chapter presents an overview of the application used to implement the database, the reasons and benefits of choosing that application. This section provides also a blue-print of the database model.

4.1 Choosing the Right Application

As mentioned in the abstract, Microsoft Access was chosen over other database software packages because of its widespread availability in the market and the fact that it gets the job done efficiently without advanced expertise in database management.

4.1.1 Microsoft Office Access

Microsoft Access is a popular relational database management system for creating and managing desktop and client/server database applications that run under the Windows operating system. It is packaged with Microsoft Office Professional which combines the relational Microsoft Jet Database Engine with a graphical user interface [13]. It allows relatively quick development because all database tables, queries, forms, and reports are stored in the database.

One of the benefits of Access from a programmer's perspective is its relative compatibility with some programming languages that can be used within its environment to add additional features to the applications namely:

4*.1.1.1 Macros*

The Access macros programming language is useful but limited, it provides a limited (though still useful) set of tools for automating database actions. Macros have a limited ability to respond to errors or other conditions out of the ordinary [14].

4.1.1.2 SQL

Structured Query Language, more commonly called SQL is the language that Access uses to store database queries [15].

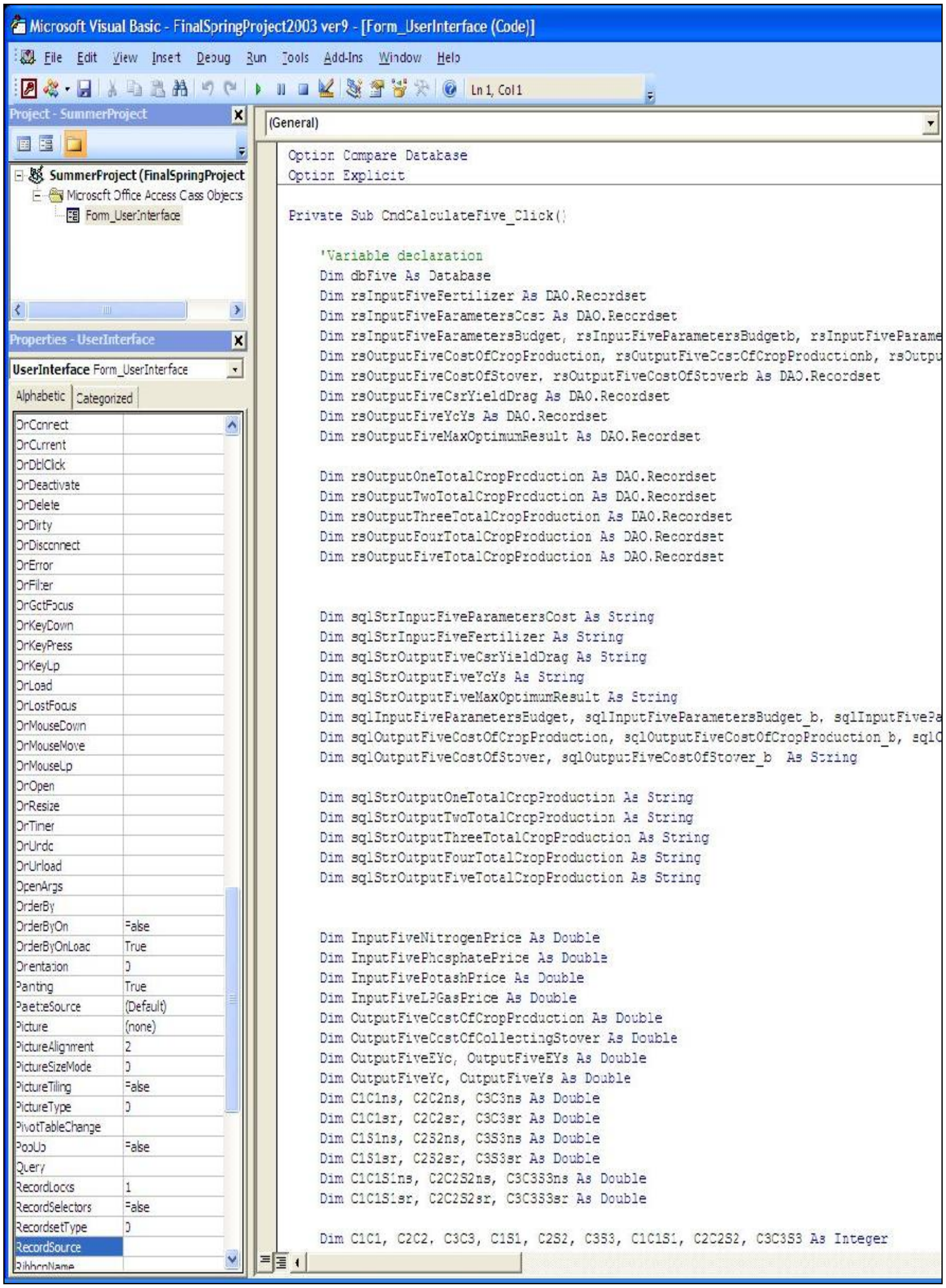
SQL queries may be viewed and edited as SQL statements, and SQL statements can be used directly as Macros or as VBA Modules to manipulate Access tables [16].

4.1.1.3 VBA

Finally, there is VBA, the programming language used the most in this thesis, allowing the construction of this complex application based in the Access interface.

VBA is an acronym that stands for Visual Basic for Applications. VBA is included as part of several Microsoft products, including Access, Word and Excel [17].

VBA is a lightweight version of the full-fledged Visual Basic programming language which is a standalone tool for creating separate software components, such as executable programs. VBA offers the same powerful tools as Visual Basic in the context of an existing application. VBA provides a complete integrated development environment (IDE) that features the same elements familiar to developers using Microsoft Visual Basic, including a Project window, a Properties window, and debugging tools [18]. Screenshot of VBA integrated development environment is shown in Figure 4.1


Microsoft Visual Basic - FinalSpringProject2003 ver9 - [Form_UserInterface (Code)]
File Edit View Insert Debug Run Tools Add-Ins Window Help
Ln 1, Col 1
Project - SummerProject
SummerProject (FinalSpringProject
Microsoft Office Access Class Objects
Form_UserInterface
Properties - UserInterface
UserInterface Form_UserInterface
Alphabetic Categorized
(General)

OnConnect	
OnCurrent	
OnDblClick	
OnDeactivate	
OnDelete	
OnDirty	
OnDisconnect	
OnError	
OnFilter	
OnGotFocus	
OnKeyDown	
OnKeyPress	
OnKeyUp	
OnLoad	
OnLostFocus	
OnMouseDown	
OnMouseMove	
OnMouseUp	
OnOpen	
OnResize	
OnTimer	
OnUndo	
OnUnload	
OpenArgs	
OrderBy	
OrderByOn	False
OrderByOnLoad	True
Orientation	0
Painting	True
PaletteSource	(Default)
Picture	(none)
PictureAlignment	2
PictureSizeMode	0
PictureTiling	False
PictureType	0
PivotTableChange	
PopUp	False
Query	
RecordLocks	1
RecordSelectors	False
RecordsetType	0
RecordSource	

```
Option Compare Database
Option Explicit

Private Sub CmdCalculateFive_Click()

    'Variable declaration
    Dim dbFive As Database
    Dim rsInputFiveFertilizer As DAO.Recordset
    Dim rsInputFiveParametersCost As DAO.Recordset
    Dim rsInputFiveParametersBudget, rsInputFiveParametersBudgetb, rsInputFiveParame
    Dim rsOutputFiveCostOfCropProduction, rsOutputFiveCostOfCropProductionb, rsOutpu
    Dim rsOutputFiveCostOfStover, rsOutputFiveCostOfStoverb As DAO.Recordset
    Dim rsOutputFiveCsrYieldDrag As DAO.Recordset
    Dim rsOutputFiveYcYs As DAO.Recordset
    Dim rsOutputFiveMaxOptimumResult As DAO.Recordset

    Dim rsOutputOneTotalCropProduction As DAO.Recordset
    Dim rsOutputTwoTotalCropProduction As DAO.Recordset
    Dim rsOutputThreeTotalCropProduction As DAO.Recordset
    Dim rsOutputFourTotalCropProduction As DAO.Recordset
    Dim rsOutputFiveTotalCropProduction As DAO.Recordset


    Dim sqlStrInputFiveParametersCost As String
    Dim sqlStrInputFiveFertilizer As String
    Dim sqlStrOutputFiveCsrYieldDrag As String
    Dim sqlStrOutputFiveYcYs As String
    Dim sqlStrOutputFiveMaxOptimumResult As String
    Dim sqlInputFiveParametersBudget, sqlInputFiveParametersBudget_b, sqlInputFivePa
    Dim sqlOutputFiveCostOfCropProduction, sqlOutputFiveCostOfCropProduction_b, sqlO
    Dim sqlOutputFiveCostOfStover, sqlOutputFiveCostOfStover_b  As String

    Dim sqlStrOutputOneTotalCropProduction As String
    Dim sqlStrOutputTwoTotalCropProduction As String
    Dim sqlStrOutputThreeTotalCropProduction As String
    Dim sqlStrOutputFourTotalCropProduction As String
    Dim sqlStrOutputFiveTotalCropProduction As String


    Dim InputFiveNitrogenPrice As Double
    Dim InputFivePhosphatePrice As Double
    Dim InputFivePotashPrice As Double
    Dim InputFiveLPGasPrice As Double
    Dim OutputFiveCostOfCropProduction As Double
    Dim OutputFiveCostOfCollectingStover As Double
    Dim OutputFiveEYc, OutputFiveEYs As Double
    Dim OutputFiveYc, OutputFiveYs As Double
    Dim C1C1ns, C2C2ns, C3C3ns As Double
    Dim C1C1sr, C2C2sr, C3C3sr As Double
    Dim C1S1ns, C2S2ns, C3S3ns As Double
    Dim C1S1sr, C2S2sr, C3S3sr As Double
    Dim C1C1S1ns, C2C2S2ns, C3C3S3ns As Double
    Dim C1C1S1sr, C2C2S2sr, C3C3S3sr As Double

    Dim C1C1, C2C2, C3C3, C1S1, C2S2, C3S3, C1C1S1, C2C2S2, C3C3S3 As Integer
```

Figure 4.1. Screenshot of VBA IDE

To use the VBA language for the database environment, Microsoft Office Access employs two main ways: via Data Access Objects (DAO) or via ActiveX Data Objects (ADO).

Data Access Objects (DAO) is the first programming interface between VBA and the Microsoft Jet database engine that allows programmers to directly connect to Microsoft Office Access tables as well as other databases through Open Database Connectivity (ODBC). Data Access Objects are suited best for either single system applications or for small, local deployments [19]. Data Access Objects remain a viable technology for interacting with Microsoft Access databases as it is faster than ADO for that purpose; however, ADO is more flexible [20].

ADO which stands for Microsoft ActiveX Data Objects enables us to write an application to access and manipulate data in a database server through an OLE DB provider. It's part of Microsoft's overall Component Object Model (COM) strategy and, as such, works in a variety of environments ranging from Visual Basic to Active Server Pages. ADO's primary benefits are data source independence, high speed, ease of use, low memory overhead, and a small disk footprint [21].

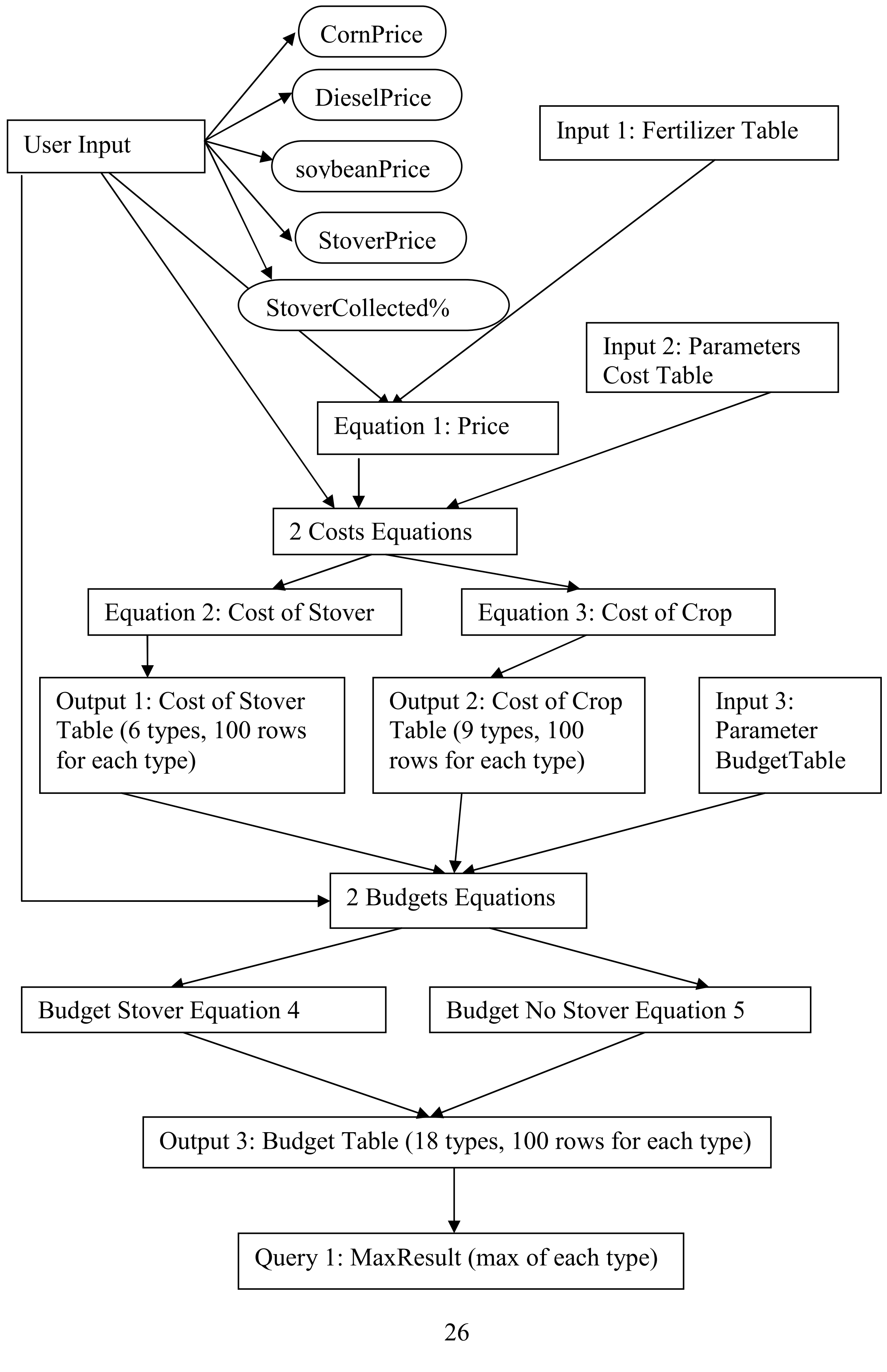
In this thesis, we use the DAO technology to interact with the Microsoft Access data from our VBA code as it is faster than ADO for that purpose.

4.2 Modeling the Application Design

After gathering all the requirements needed to create the database, the next step is to start up with the designing. Modeling is best defined as the process of documenting one or more parts of an application on paper (or with an electronic tool). A variety of modeling techniques can be used to accomplish the end result: modeling the flow of activities through the system, modeling the way the code will be structured, and so on [22].

Regardless of the modeling techniques one decides to use, the objective is to come up with a complete roadmap for building the system before writing a single line of code.

Based on the above requirements specifications and computational formulas, the database model was drafted as shown below in Figure 4.2. This is a sample draft for the entire model.


CornPrice
DieselPrice
User Input
Input 1: Fertilizer Table
soybeanPrice
StoverPrice
StoverCollected%
Input 2: Parameters Cost Table
Equation 1: Price
2 Costs Equations
Equation 2: Cost of Stover
Equation 3: Cost of Crop
Output 1: Cost of Stover Table (6 types, 100 rows for each type)
Output 2: Cost of Crop Table (9 types, 100 rows for each type)
Input 3: Parameter BudgetTable
2 Budgets Equations
Budget Stover Equation 4
Budget No Stover Equation 5
Output 3: Budget Table (18 types, 100 rows for each type)
Query 1: MaxResult (max of each type)

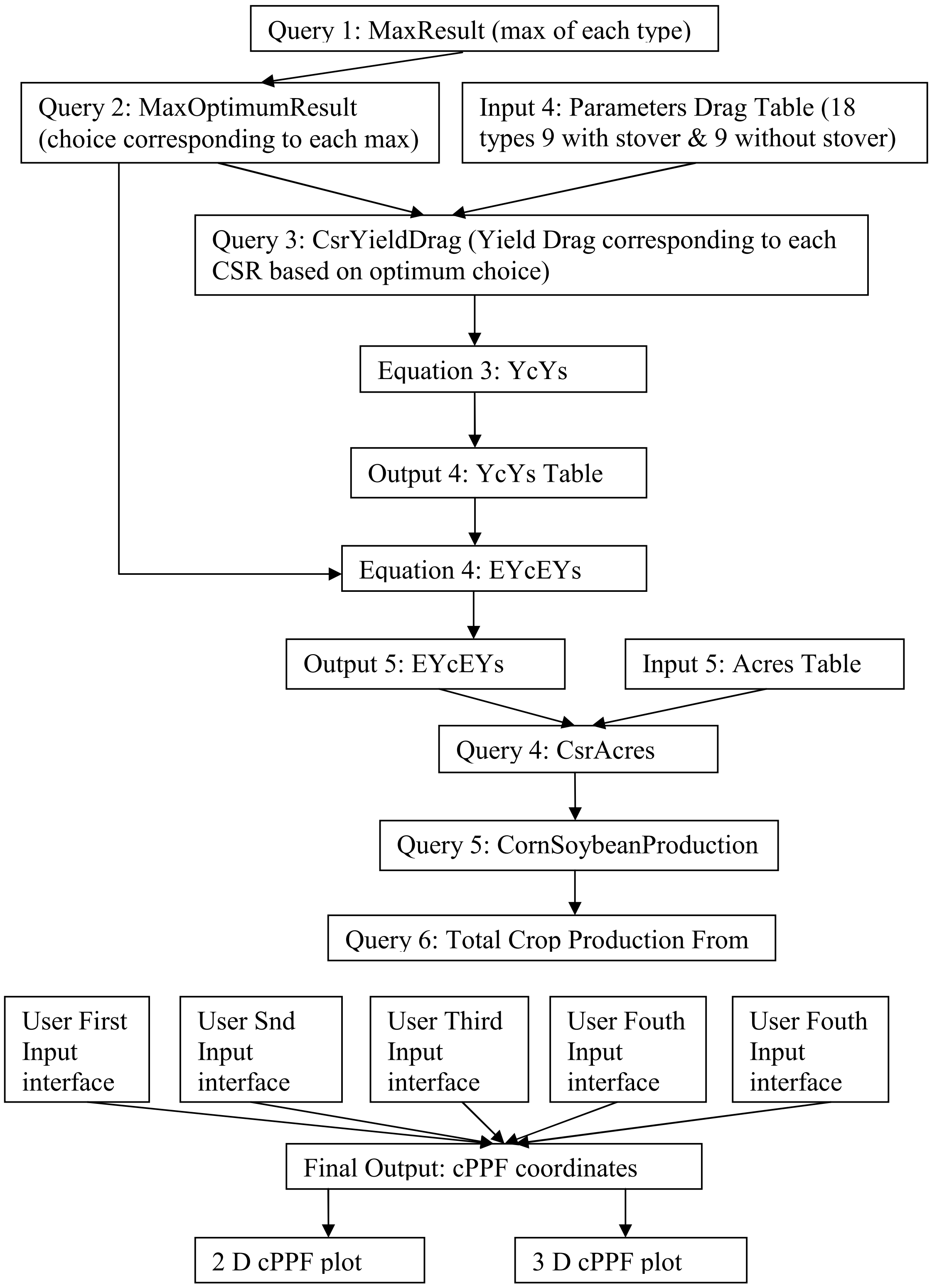

Query 1: MaxResult (max of each type)
Query 2: MaxOptimumResult (choice corresponding to each max)
Input 4: Parameters Drag Table (18 types 9 with stover & 9 without stover)
Query 3: CsrYieldDrag (Yield Drag corresponding to each CSR based on optimum choice)
Equation 3: YcYs
Output 4: YcYs Table
Equation 4: EYcEYs
Output 5: EYcEYs
Input 5: Acres Table
Query 4: CsrAcres
Query 5: CornSoybeanProduction
Query 6: Total Crop Production From
User First Input interface
User Snd Input interface
User Third Input interface
User Fouth Input interface
User Fouth Input interface
Final Output: cPPF coordinates
2 D cPPF plot
3 D cPPF plot

Figure 4.2. Database draft

Given the above model, our data was broken down into two major parts as follows:

4.2.1 The Input Parameters

The database takes input from the user inputs and from the parameters input tables.

4.2.1.1 The User Inputs

The User Inputs are the information that the user enter into the database. A tab control has been used on an Access form to present five pages of information that was needed to plot both 2D and 3D cPPF.

4.2.1.2 Parameters Input Tables

As showed in the database model, the parameters input tables consists of five tables for each page of information in the tab control from the user interface as follows:

- The Fertilizer table.
- The Parameters Cost table.
- The Parameters Budget table.
- The Parameters Drag table.
- The Acres table.

4.2.2 The Parameters Output Tables

The output tables were stored in five different tables and six queries for each page of information in the tab control from the user interface as follows:

- The Cost Of Stover table.
- The Cost Of Crop Production table.

- The Budget table.
- The YcYs table (corn and soybean production).
- The EYcEYs table (effective corn and soybean).
- The MaxResult query.
- The MaxOptimumResult query.
- The CsrYieldDrag query.
- The CsrAcres query.
- The CornSoybeanProduction query.
- The TotalCropProduction query.

CHAPTER 5

BUILDING AND IMPLEMENTING THE DATABASE

In the following section the building and implementation details of the database are discussed.

5.1 Building the Table Structures

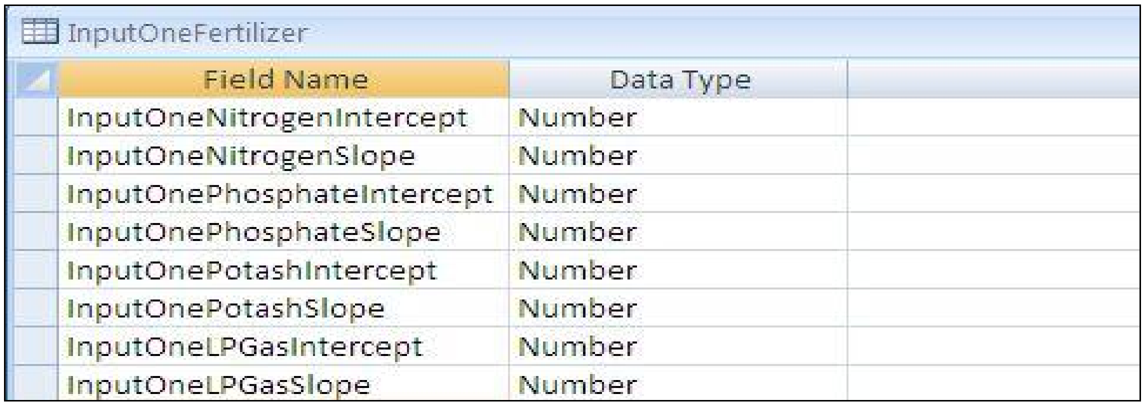
According to the database model, we created for each page of information five input tables and five output tables. For each table we determined the fields names and data type needed for each field. Since all the five pages hold the same controls of information, Figure 5.1 through 5.10 shows only the fields (Datasheet View) and the appropriate data types for each of the fields (Design View) in the tables for the first page labeled “UserFirstInputs”.

The InputOneFertilizer table holds the different slopes and intercepts of the simple linear regression equation computed from the data on the different prices such as diesel fuel price, LP Gas price and fertilizer price (Nitrogen and Potash price).

The InputOneFertilizer table contains eight fields as follows:

- InputOneNitrogenSlope.
- InputOneNitrogenIntercept.
- InputOnePhosphateSlope.
- InputOnePhosphateIntercept.
- InputOnePotashSlope.

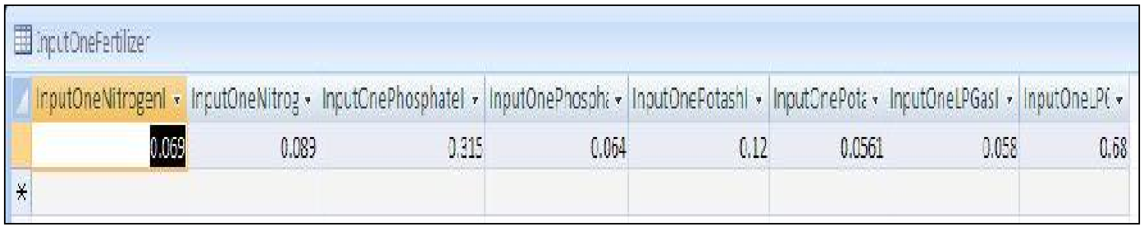
- InputOnePotashIntercept.
- InputOneDieselSlope.
- InputOneDieselIntercept.



InputOneFertilizer

Field Name	Data Type
InputOneNitrogenIntercept	Number
InputOneNitrogenSlope	Number
InputOnePhosphateIntercept	Number
InputOnePhosphateSlope	Number
InputOnePotashIntercept	Number
InputOnePotashSlope	Number
InputOneLPGasIntercept	Number
InputOneLPGasSlope	Number

Figure 5.1a. InputOneFertilizer table in Design View



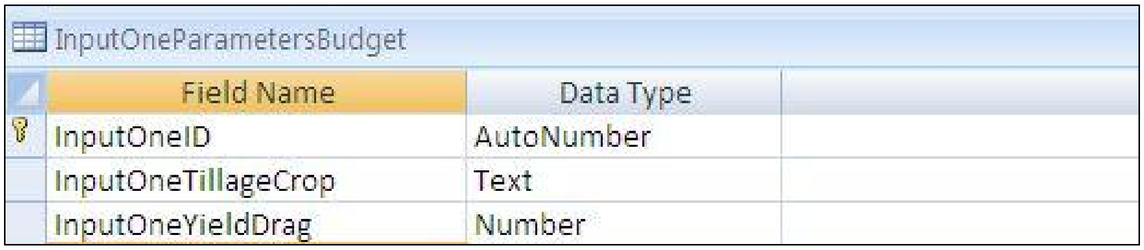
InputOneFertilizer

InputOneNitrogenI	InputOneNitrog	InputOnePhosphateI	InputOnePhosph	InputOnePotashI	InputOnePota	InputOneLPGasI	InputOneLPG
0.069	0.083	0.315	0.064	0.12	0.0561	0.058	0.68

Figure 5.1b. InputOneFertilizer table in Datasheet View

The InputOneParametersBudget table consists of brief details about the budget as follows:

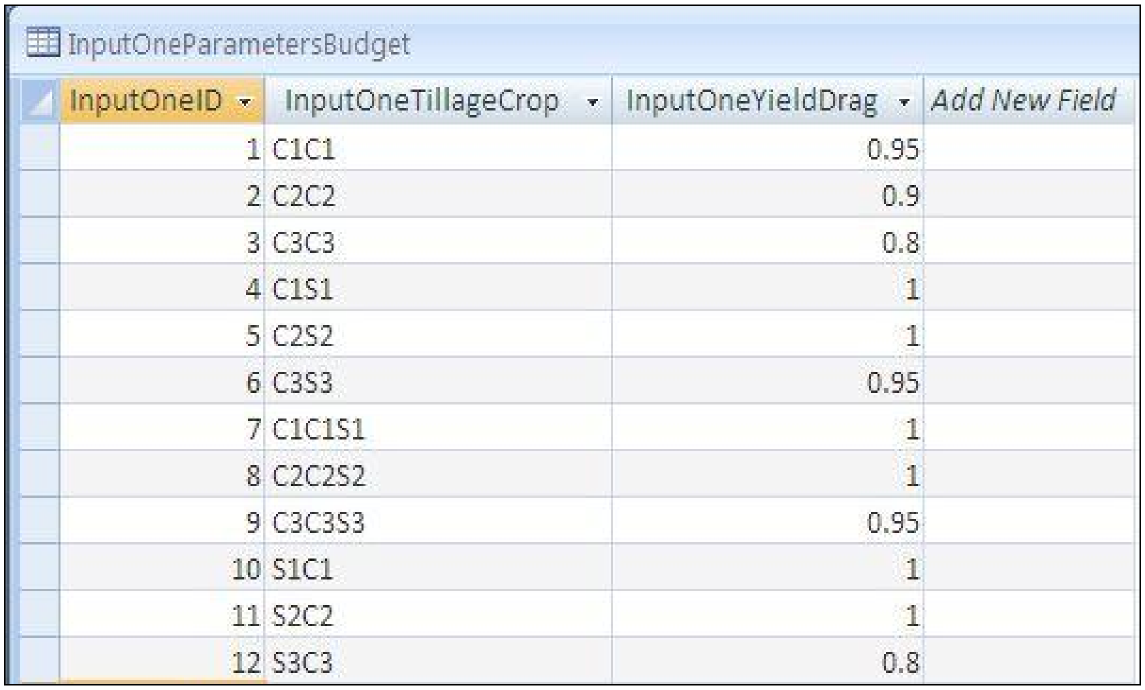
- InputOneID designed as primary key for the table.
- InputOneTillageCrop (rotation tillage types).
- InputOneYieldDrag assumptions.



InputOneParametersBudget

Field Name	Data Type
InputOneID	AutoNumber
InputOneTillageCrop	Text
InputOneYieldDrag	Number

Figure 5.2a. InputOneParametersBudget table in Design View



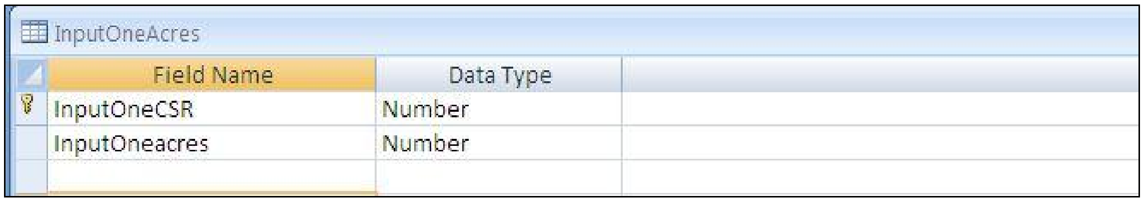
InputOneParametersBudget

InputOneID	InputOneTillageCrop	InputOneYieldDrag	Add New Field
1	C1C1	0.95	
2	C2C2	0.9	
3	C3C3	0.8	
4	C1S1	1	
5	C2S2	1	
6	C3S3	0.95	
7	C1C1S1	1	
8	C2C2S2	1	
9	C3C3S3	0.95	
10	S1C1	1	
11	S2C2	1	
12	S3C3	0.8	

Figure 5.2b. InputOneParametersBudget table in Datasheet View

The InputOneAcres table is structured in a way as to hold the acres details as follows:

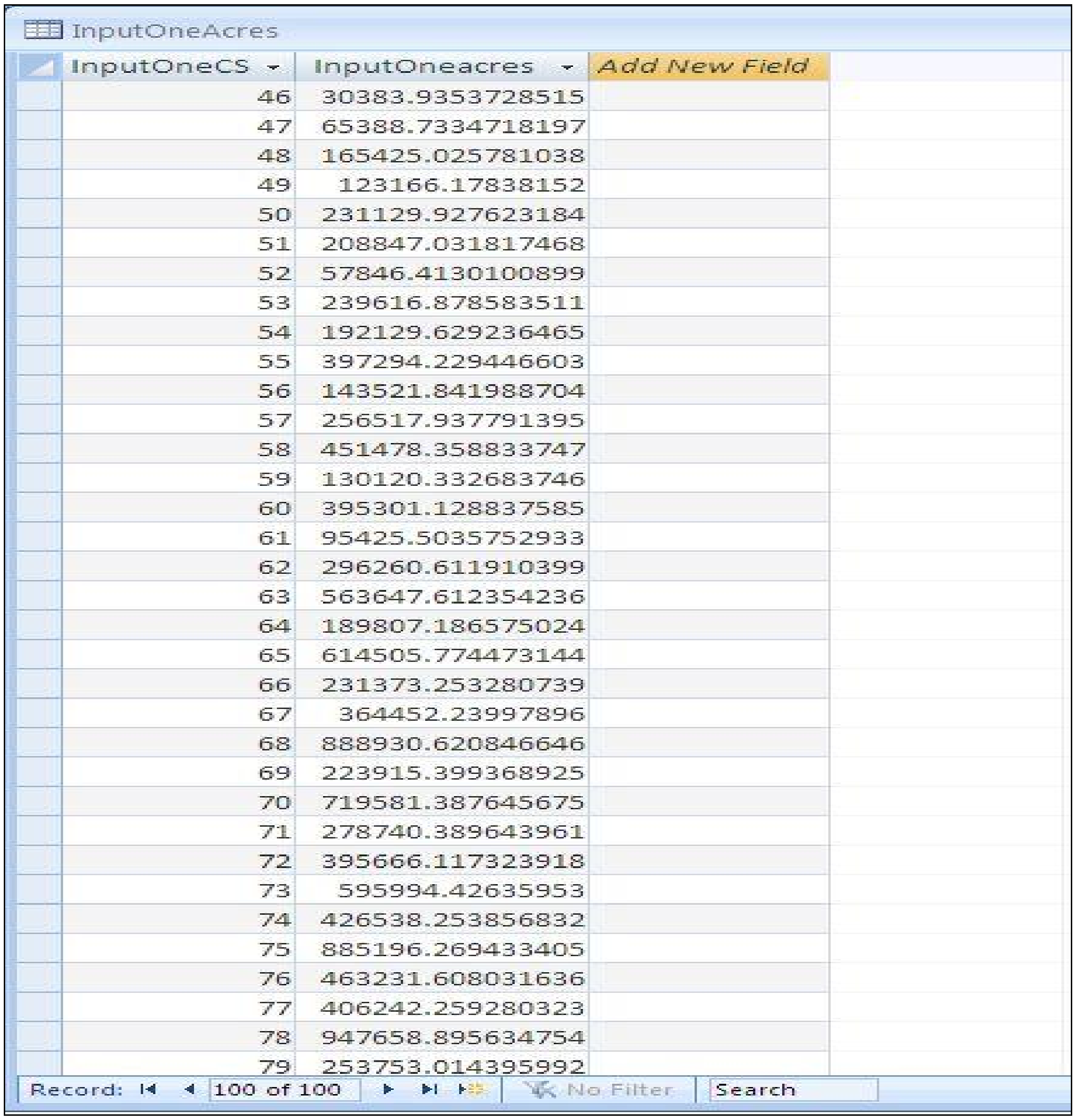
- InputOneCSR designed as primary key for the table.
- InputOneAcres values, one hundred rows and each row corresponding at each CSR value.



InputOneAcres

Field Name	Data Type
InputOneCSR	Number
InputOneacres	Number

Figure 5.3a. InputOneAcres table in Design View



InputOneAcres

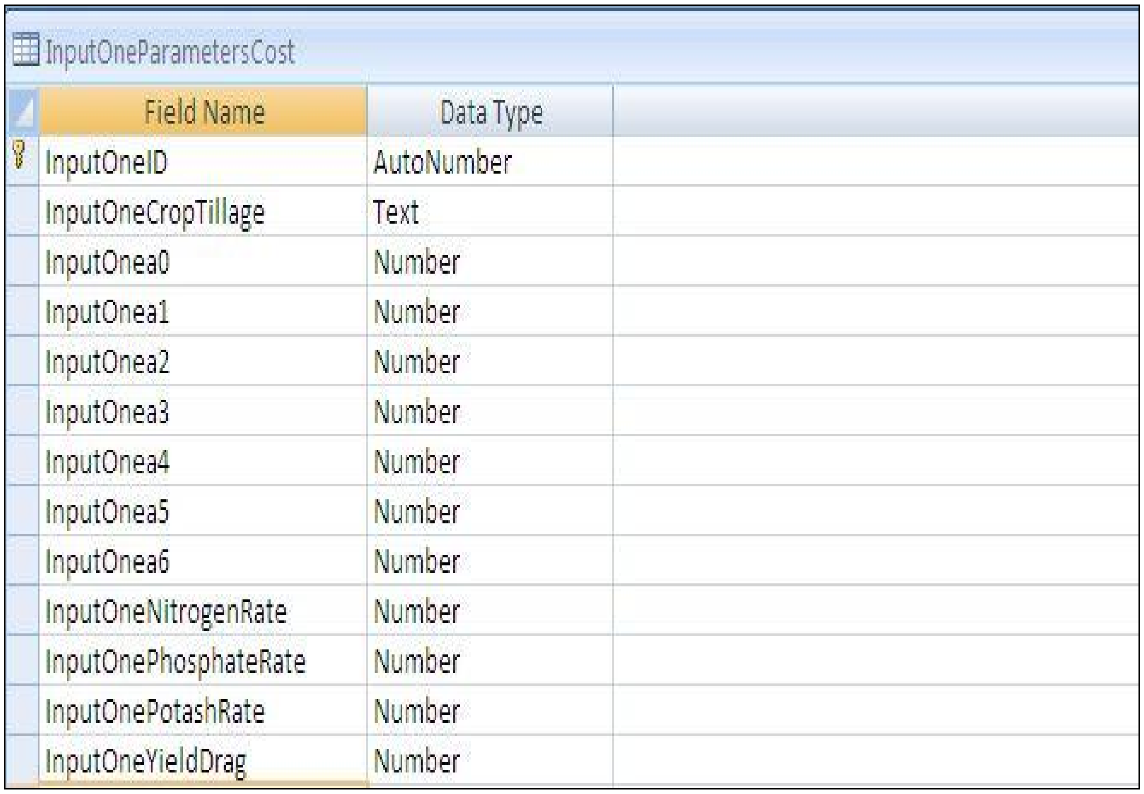
InputOneCS	InputOneacres	Add New Field
46	30383.9353728515	
47	65388.7334718197	
48	165425.025781038	
49	123166.17838152	
50	231129.927623184	
51	208847.031817468	
52	57846.4130100899	
53	239616.878583511	
54	192129.629236465	
55	397294.229446603	
56	143521.841988704	
57	256517.937791395	
58	451478.358833747	
59	130120.332683746	
60	395301.128837585	
61	95425.5035752933	
62	296260.611910399	
63	563647.612354236	
64	189807.186575024	
65	614505.774473144	
66	231373.253280739	
67	364452.23997896	
68	888930.620846646	
69	223915.399368925	
70	719581.387645675	
71	278740.389643961	
72	395666.117323918	
73	595994.42635953	
74	426538.253856832	
75	885196.269433405	
76	463231.608031636	
77	406242.259280323	
78	947658.895634754	
79	253753.014395992	

Record: 100 of 100 No Filter Search

Figure 5.3b. InputOneAcres table in Datasheet View

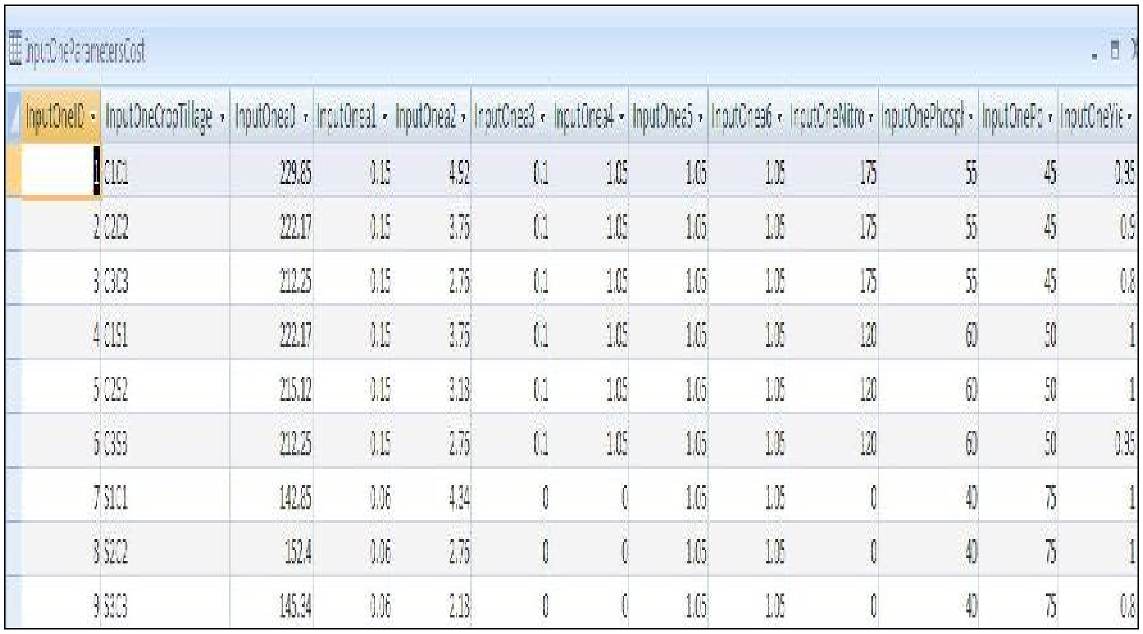
The InputOneParametersCost table is the master table; it contains all the information about the costs of production of individual crop such as:

- InputOneID designed as primary key for the table.
- InputOneCropTillage (rotation tillage types).
- InputOnea$_0$ through InputOnea$_6$ (parameters from the cost equation).
- InputOneNitrogenRate.
- InputOnePhosphateRate.
- InputOnePotashRate.
- InputOneYieldDrag assumptions.


InputOneParametersCost

Field Name	Data Type
InputOneID	AutoNumber
InputOneCropTillage	Text
InputOnea0	Number
InputOnea1	Number
InputOnea2	Number
InputOnea3	Number
InputOnea4	Number
InputOnea5	Number
InputOnea6	Number
InputOneNitrogenRate	Number
InputOnePhosphateRate	Number
InputOnePotashRate	Number
InputOneYieldDrag	Number

Figure 5.4a. InputOneParametersCost table in Design View

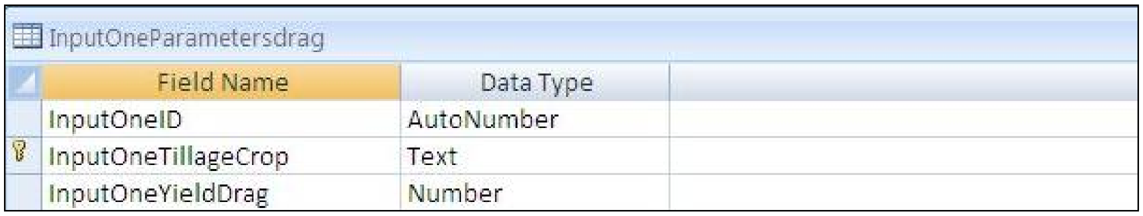

InputOneParametersCost

InputOneID	InputOneCropTillage	InputOnea0	InputOnea1	InputOnea2	InputOnea3	InputOnea4	InputOnea5	InputOnea6	InputOneNitro	InputOnePhospl	InputOnePo	InputOneYie
1	C1C1	229.65	0.15	4.92	0.1	1.05	1.65	1.05	175	55	45	0.35
2	C2C2	222.17	0.15	3.75	0.1	1.05	1.65	1.05	175	55	45	0.9
3	C3C3	212.25	0.15	2.75	0.1	1.05	1.65	1.05	175	55	45	0.8
4	C1S1	222.17	0.15	3.75	0.1	1.05	1.65	1.05	120	60	50	1
5	C2S2	215.12	0.15	3.13	0.1	1.05	1.65	1.05	120	60	50	1
6	C3S3	212.25	0.15	2.75	0.1	1.05	1.65	1.05	120	60	50	0.35
7	S1C1	142.65	0.06	4.34	0	0	1.65	1.05	0	40	75	1
8	S2C2	152.4	0.06	2.75	0	0	1.65	1.05	0	40	75	1
9	S3C3	145.34	0.06	2.13	0	0	1.65	1.05	0	40	75	0.8

Figure 5.4b. InputOneParametersCost table in Datasheet View

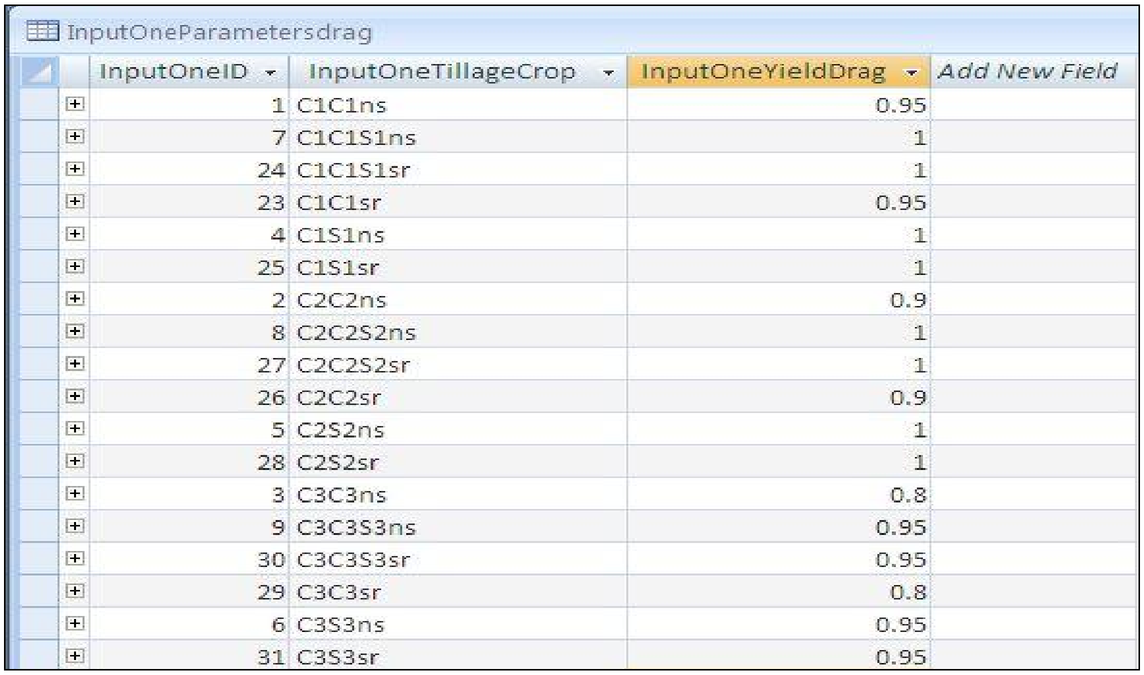
The InputOneParametersDrag table captures information describing assumptions about expected yield drag for all the rotation-tillage-Stover and no Stover harvesting choice as follows:

- InputOneID.
- InputOneTillageCrop designed as primary key for the table.
- InputOneYieldDrag assumptions.


InputOneParametersdrag

	Field Name	Data Type
	InputOneID	AutoNumber
🔑	InputOneTillageCrop	Text
	InputOneYieldDrag	Number

Figure 5.5a. InputOneParametersDrag table in Design View



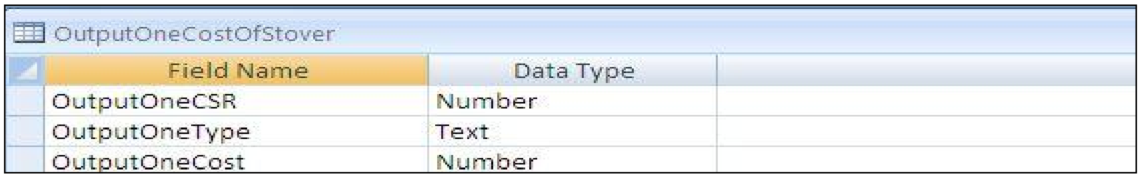
InputOneParametersdrag

InputOneID	InputOneTillageCrop	InputOneYieldDrag	Add New Field
1	C1C1ns	0.95	
7	C1C1S1ns	1	
24	C1C1S1sr	1	
23	C1C1sr	0.95	
4	C1S1ns	1	
25	C1S1sr	1	
2	C2C2ns	0.9	
8	C2C2S2ns	1	
27	C2C2S2sr	1	
26	C2C2sr	0.9	
5	C2S2ns	1	
28	C2S2sr	1	
3	C3C3ns	0.8	
9	C3C3S3ns	0.95	
30	C3C3S3sr	0.95	
29	C3C3sr	0.8	
6	C3S3ns	0.95	
31	C3S3sr	0.95	

Figure 5.5b. InputOneParametersDrag table in Datasheet View

The OutputOneCostOfStover table is used to store the six rotation-tillage types as follows:

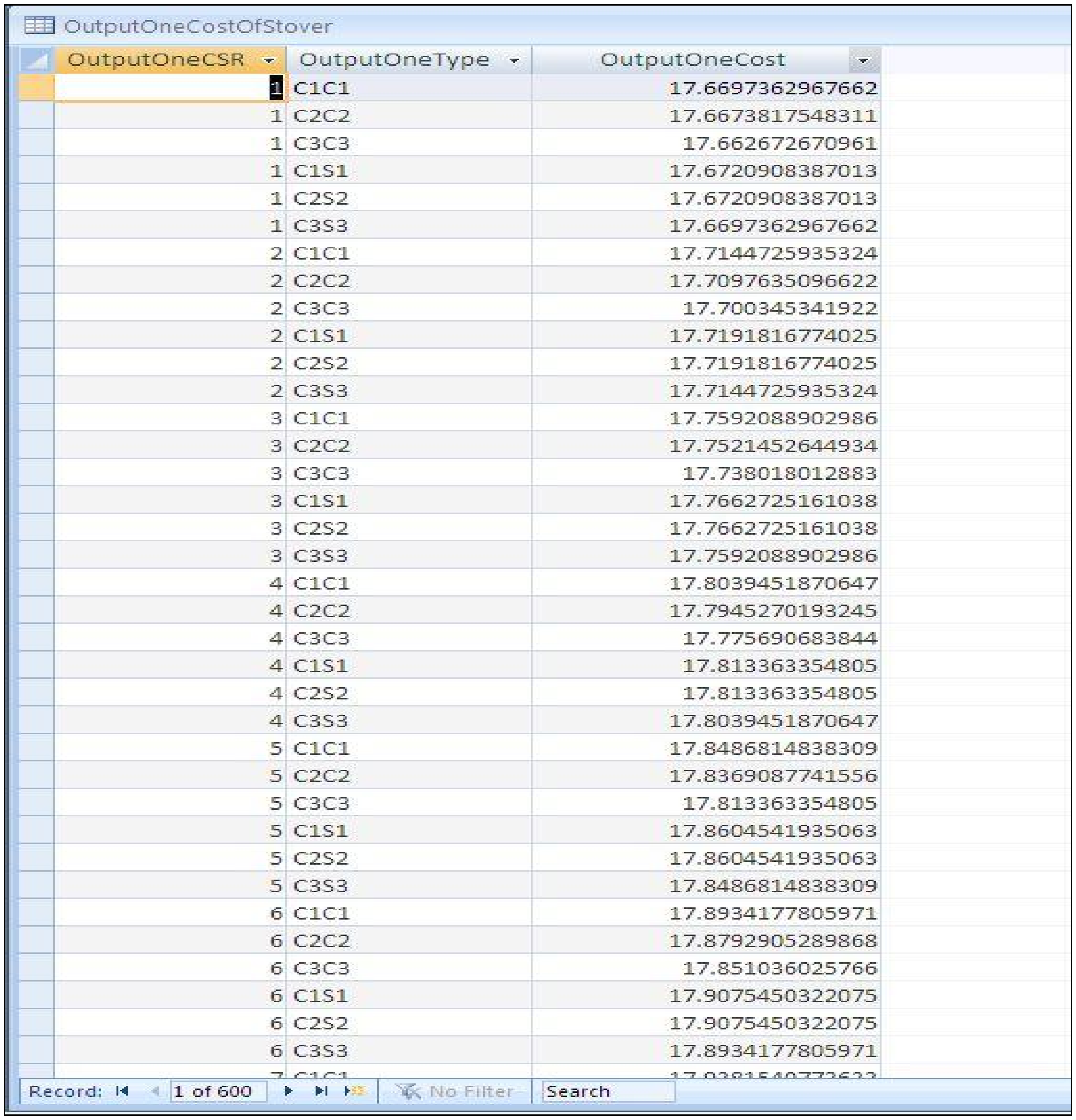
- OutputOneCSR value, from one to one hundred.
- OutputOneType, one hundred rows for each type.
- OutputOneCost, six hundred rows since each type has one hundred rows corresponding at each CSR value (one to one hundred).



OutputOneCostOfStover

Field Name	Data Type
OutputOneCSR	Number
OutputOneType	Text
OutputOneCost	Number

Figure 5.6a. OutputOneCostOfStover table in Design View



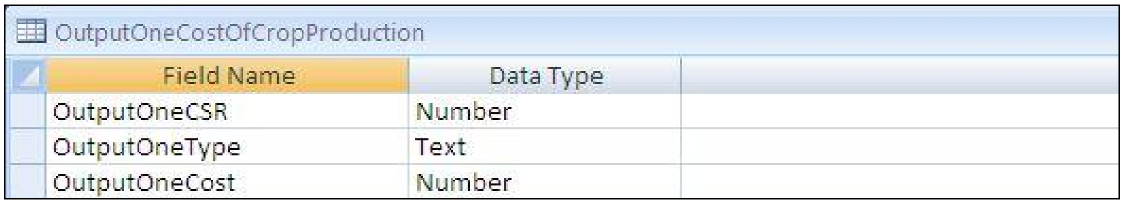
OutputOneCSR	OutputOneType	OutputOneCost
1	C1C1	17.6697362967662
1	C2C2	17.6673817548311
1	C3C3	17.662672670961
1	C1S1	17.6720908387013
1	C2S2	17.6720908387013
1	C3S3	17.6697362967662
2	C1C1	17.7144725935324
2	C2C2	17.7097635096622
2	C3C3	17.700345341922
2	C1S1	17.7191816774025
2	C2S2	17.7191816774025
2	C3S3	17.7144725935324
3	C1C1	17.7592088902986
3	C2C2	17.7521452644934
3	C3C3	17.738018012883
3	C1S1	17.7662725161038
3	C2S2	17.7662725161038
3	C3S3	17.7592088902986
4	C1C1	17.8039451870647
4	C2C2	17.7945270193245
4	C3C3	17.775690683844
4	C1S1	17.813363354805
4	C2S2	17.813363354805
4	C3S3	17.8039451870647
5	C1C1	17.8486814838309
5	C2C2	17.8369087741556
5	C3C3	17.813363354805
5	C1S1	17.8604541935063
5	C2S2	17.8604541935063
5	C3S3	17.8486814838309
6	C1C1	17.8934177805971
6	C2C2	17.8792905289868
6	C3C3	17.851036025766
6	C1S1	17.9075450322075
6	C2S2	17.9075450322075
6	C3S3	17.8934177805971

Figure 5.6b. OutputOneCostOfStover table in Datasheet View

The OutputOneCostOfCropProduction table is used to store the nine rotation-tillage types as follows:

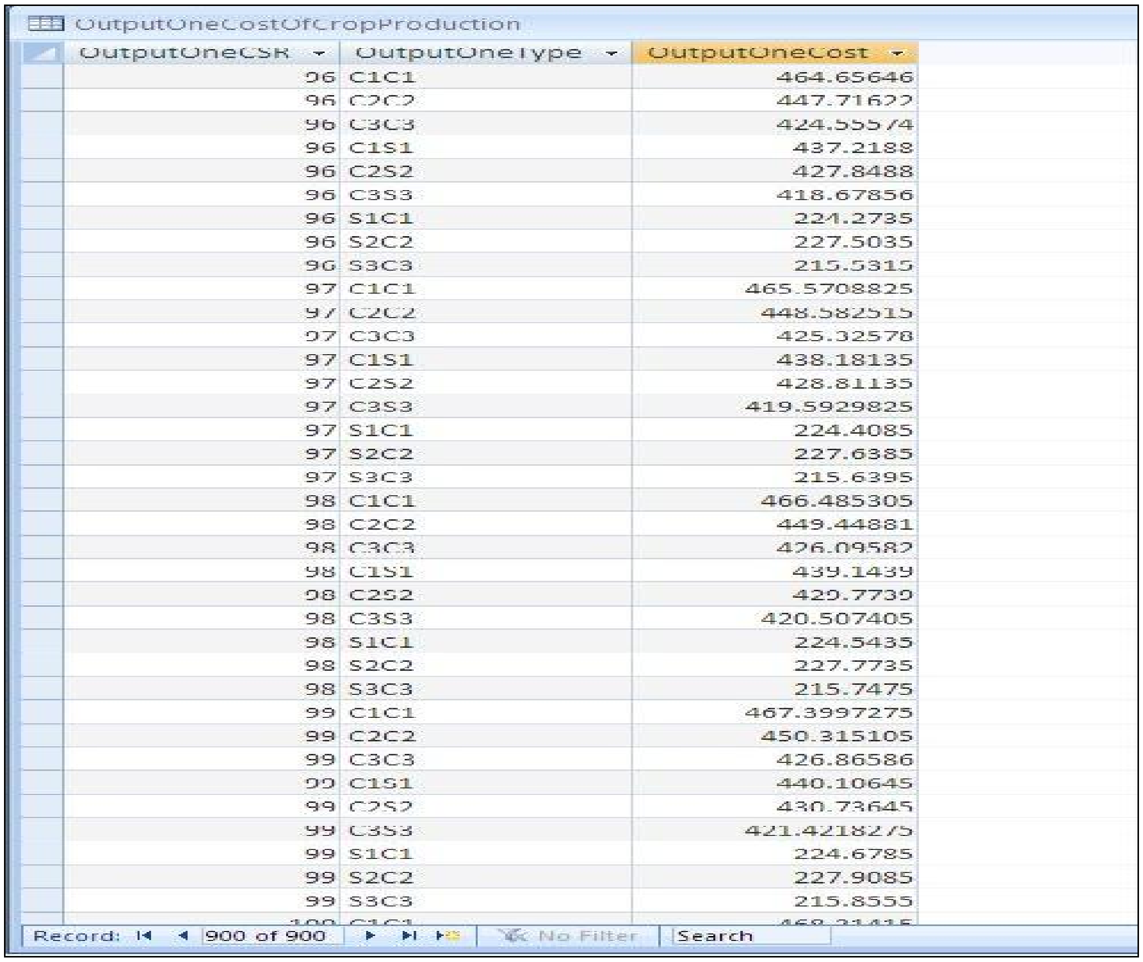
- OutputOneCSR value, from one to one hundred.
- OutputOneType, one hundred rows for each type.

- OutputOneCost, nine hundred rows since each type has one hundred rows corresponding at each CSR value (one to one hundred).


OutputOneCostOfCropProduction

Field Name	Data Type
OutputOneCSR	Number
OutputOneType	Text
OutputOneCost	Number

Figure 5.7a. OutputOneCostOfCropProduction table in Design View


OutputOneCostOfCropProduction

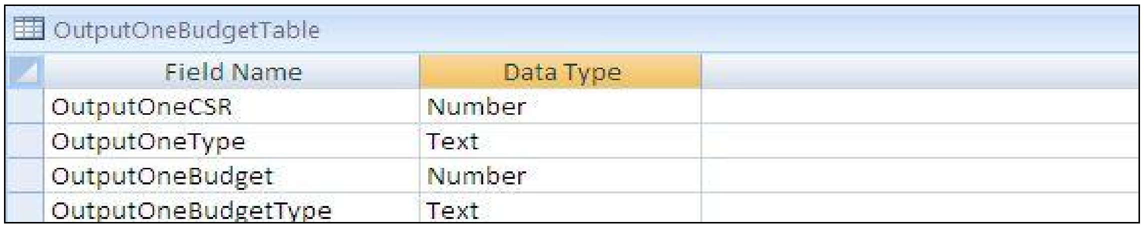
OutputOneCSR	OutputOneType	OutputOneCost
96	C1C1	464.65646
96	C2C2	447.71622
96	C3C3	424.55574
96	C1S1	437.2188
96	C2S2	427.8488
96	C3S3	418.67856
96	S1C1	224.2735
96	S2C2	227.5035
96	S3C3	215.5315
97	C1C1	465.5708825
97	C2C2	448.582515
97	C3C3	425.32578
97	C1S1	438.18135
97	C2S2	428.81135
97	C3S3	419.5929825
97	S1C1	224.4085
97	S2C2	227.6385
97	S3C3	215.6395
98	C1C1	466.485305
98	C2C2	449.44881
98	C3C3	426.09582
98	C1S1	439.1439
98	C2S2	429.7739
98	C3S3	420.507405
98	S1C1	224.5435
98	S2C2	227.7735
98	S3C3	215.7475
99	C1C1	467.3997275
99	C2C2	450.315105
99	C3C3	426.86586
99	C1S1	440.10645
99	C2S2	430.73645
99	C3S3	421.4218275
99	S1C1	224.6785
99	S2C2	227.9085
99	S3C3	215.8555
100	C1C1	468.31415

Record: 900 of 900 No Filter Search

Figure 5.7b. OutputOneCostOfCropProduction table in Datasheet View

The OutputOneBudget table outputs the nine rotation-tillage types as follows:

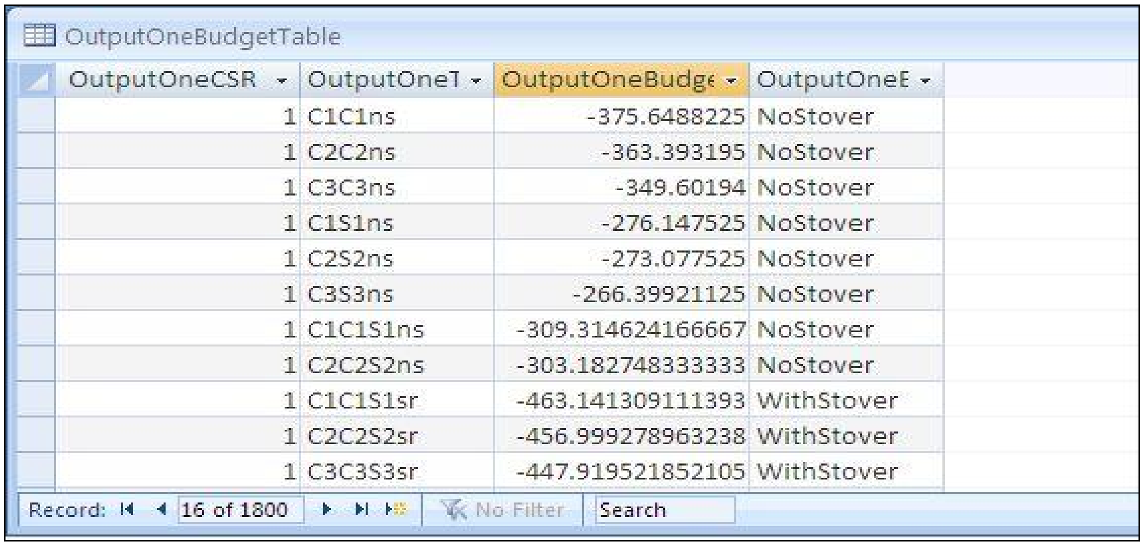
- OutputOneCSR value, from one to one hundred.
- OutputOneType, each type with Stover harvesting and no Stover harvesting one hundred rows for each type.
- OutputOneBudget, eighty hundred rows since each type has one hundred rows corresponding at each CSR value (one to one hundred).
- OutputOneBudgetType corresponding at each budget value.



OutputOneBudgetTable

Field Name	Data Type
OutputOneCSR	Number
OutputOneType	Text
OutputOneBudget	Number
OutputOneBudgetType	Text

Figure 5.8a. OutputOneBudget table in Design View



OutputOneBudgetTable

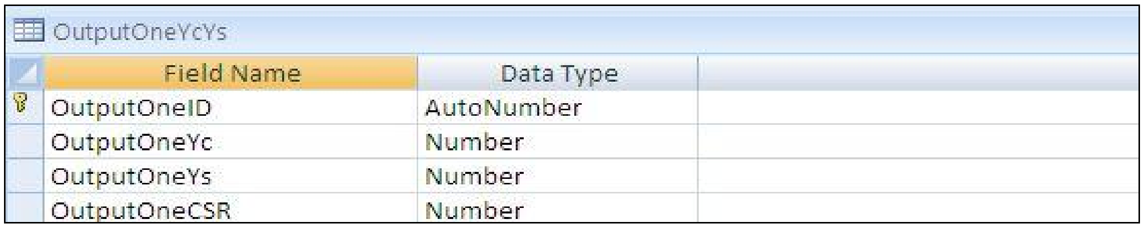
OutputOneCSR	OutputOneT	OutputOneBudge	OutputOneE
1	C1C1ns	-375.6488225	NoStover
1	C2C2ns	-363.393195	NoStover
1	C3C3ns	-349.60194	NoStover
1	C1S1ns	-276.147525	NoStover
1	C2S2ns	-273.077525	NoStover
1	C3S3ns	-266.39921125	NoStover
1	C1C1S1ns	-309.314624166667	NoStover
1	C2C2S2ns	-303.182748333333	NoStover
1	C1C1S1sr	-463.141309111393	WithStover
1	C2C2S2sr	-456.999278963238	WithStover
1	C3C3S3sr	-447.919521852105	WithStover

Record: 16 of 1800 No Filter Search

Figure 5.8b. OutputOneBudget table in Datasheet View

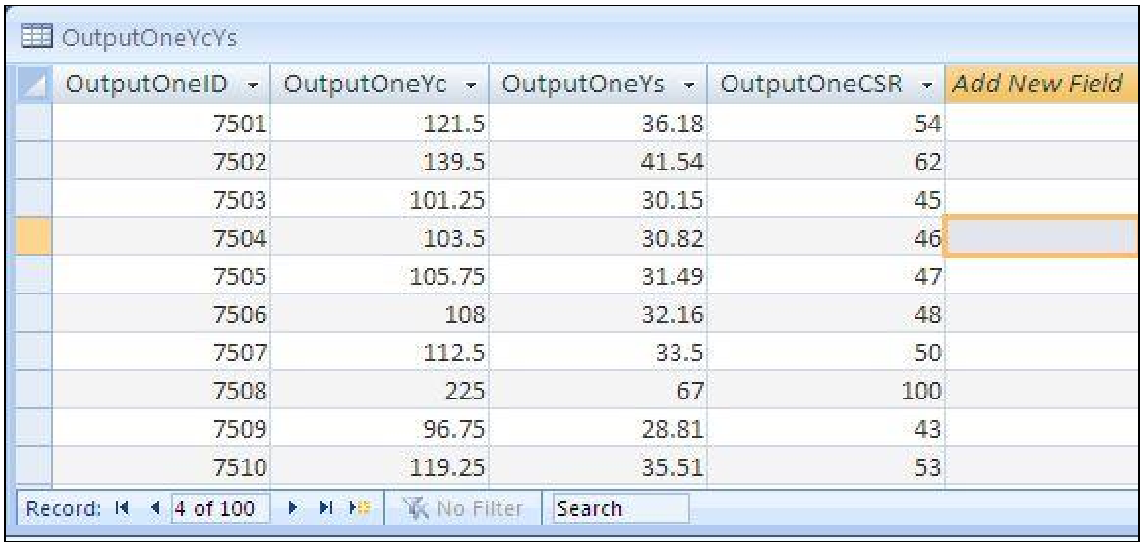
The OutputOneYcYS table is used to store the adjusted corn and soybean yield for the yield drag due to rotation and tillage as follows:

- OutputOneID, designed as primary key for the table.
- OutputOneYc corresponding to the expected revenue for corn.
- OutputOneYs corresponding to the expected revenue for soybean.
- OutputOneCSR, the corresponding value is based on the optimum choice of rotation and tillage type.


OutputOneYcYs

Field Name	Data Type
OutputOneID	AutoNumber
OutputOneYc	Number
OutputOneYs	Number
OutputOneCSR	Number

Figure 5.9a. OutputOneYcYS table in Design View

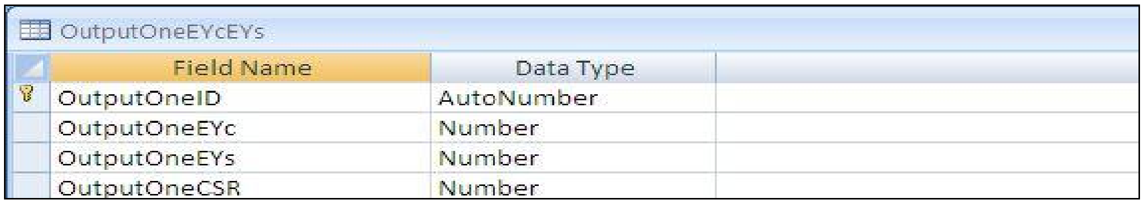

OutputOneYcYs

OutputOneID	OutputOneYc	OutputOneYs	OutputOneCSR	Add New Field
7501	121.5	36.18	54	
7502	139.5	41.54	62	
7503	101.25	30.15	45	
7504	103.5	30.82	46	
7505	105.75	31.49	47	
7506	108	32.16	48	
7507	112.5	33.5	50	
7508	225	67	100	
7509	96.75	28.81	43	
7510	119.25	35.51	53	

Figure 5.9b. OutputOneYcYS table in Datasheet View

The OutputOneEYcEYS table holds information describing corn and soybean yield for the yield drag due to rotation and tillage as follows:

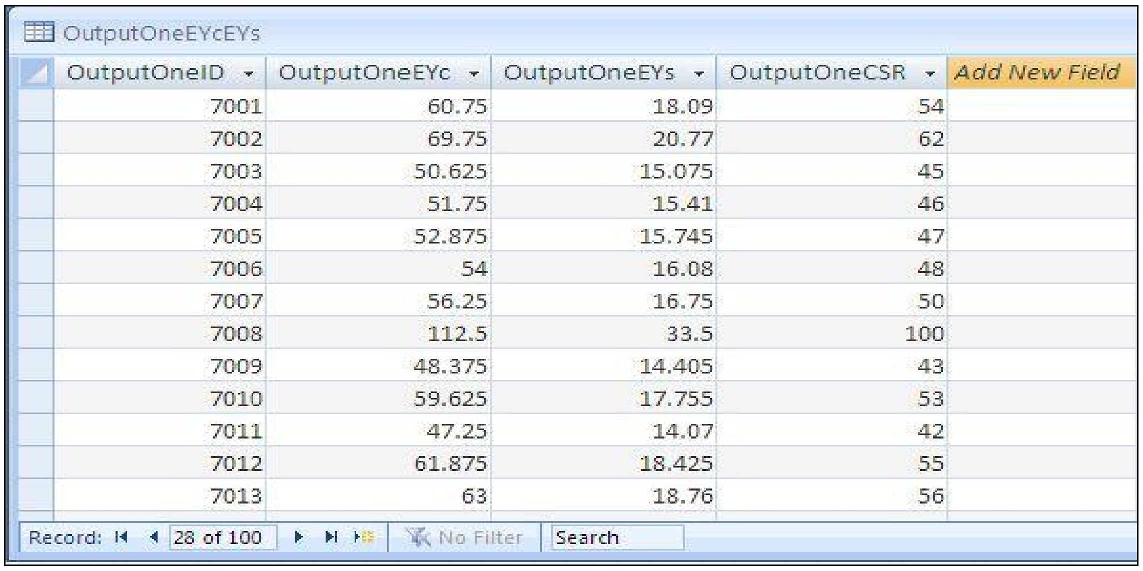
- OutputOneID, designed as primary key for the table.
- OutputOneEYc corresponding to the effective yield for corn.
- OutputOneEYs corresponding to the effective yield for soybean.
- OutputOneCSR, the corresponding value is based on the optimum choice of rotation and tillage type.



OutputOneEYcEYs

Field Name	Data Type
OutputOneID	AutoNumber
OutputOneEYc	Number
OutputOneEYs	Number
OutputOneCSR	Number

Figure 5.10a. OutputOneEYcEYS table in Design View



OutputOneEYcEYs

OutputOneID	OutputOneEYc	OutputOneEYs	OutputOneCSR	Add New Field
7001	60.75	18.09	54	
7002	69.75	20.77	62	
7003	50.625	15.075	45	
7004	51.75	15.41	46	
7005	52.875	15.745	47	
7006	54	16.08	48	
7007	56.25	16.75	50	
7008	112.5	33.5	100	
7009	48.375	14.405	43	
7010	59.625	17.755	53	
7011	47.25	14.07	42	
7012	61.875	18.425	55	
7013	63	18.76	56	

Figure 5.10b. OutputOneEYcEYS table in Datasheet View

5.2 Building the User Interface

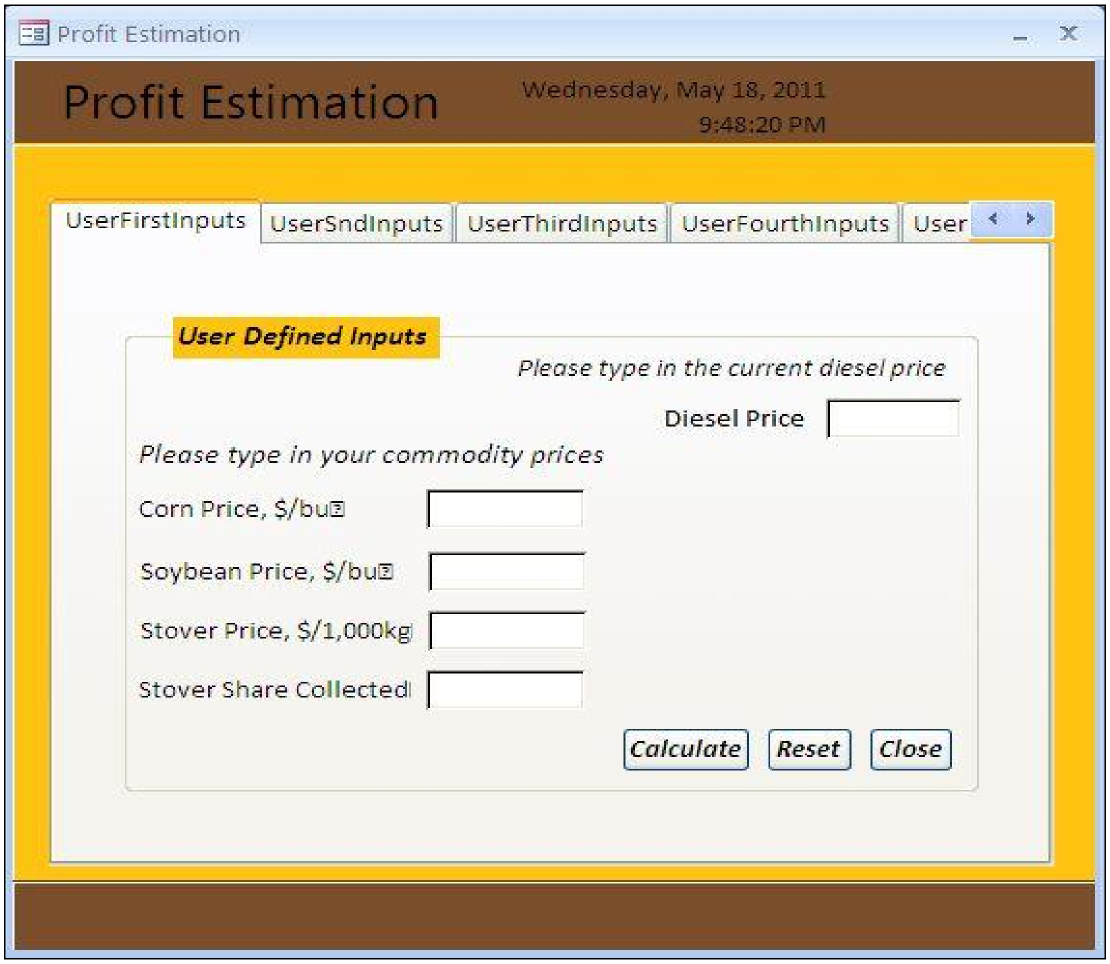
Once the tables had been created and populated, the next step was to design the program interface.

The Graphical User Interface (GUI) has been developed and rendered using Access unbound form allowing the user to input data into the database. Then, through VBA code, a number of steps will be taken to get the data and plot both 2D and 3D cPPF.

In order to create a user-friendly interface and for better data presentation and manipulation, a tab control has been used on the blank form to present five pages of information about that single form. Each page in the tab control holds eights controls. The first fives controls of each page are text boxes, which will be utilized to enter the user's commodities prices (for each commodity a text box has been created) and the three others are Command buttons. Theses Command buttons are as follows:

- **Calculate button**: When clicked, it will delete all the olds data, update and save the news ones.
- **Reset button**: When clicked, it will clear the data enter by the user, allowing the user to rectify the data.
- **Close button**: Will provide to the user, the option to close the form.

The form designed to accommodate this data entry is displayed in Figure 5.11 below.



Please type in the current diesel price

Diesel Price	

Please type in your commodity prices

Corn Price, $/bu	
Soybean Price, $/bu	
Stover Price, $/1,000kg	
Stover Share Collected	

Figure 5.11. UserInterface in form view

5.3 Building the VBA Code

The action for our database logic happens behind the click event of the Calculate Button, the Reset Button and the Close Button. The basic requirements for our VBA code are as follow:

- Pull the data from the form and ensure appropriate values have been entered.
- If values were not entered from one text box, notify the user and wait for the value.

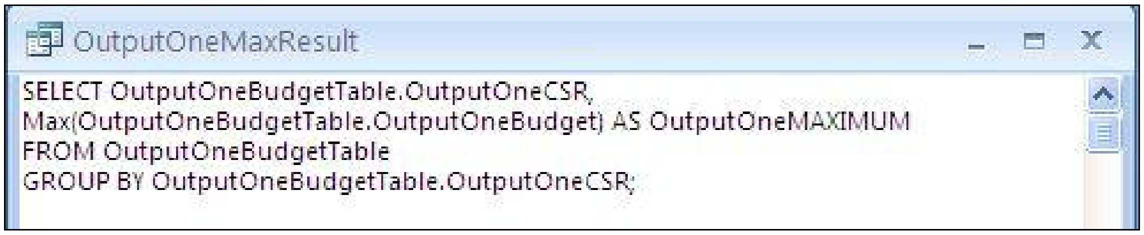
- Create an ADO connection to the database.
- Open connection to current Access database.
- Delete old data from the output tables.
- Creating a recordset for input tables.
- Close the recordset for input tables.
- Close the connection.
- Accessing data and do the calculation and update the output.
- Notify the user that the operation has been done successfully.
- Update both two and three dimensional plots for the cPPF.

The Visual Basic Application (VBA) code behind which supports the functioning of the application is shown in appendix.

5.4 Building the queries

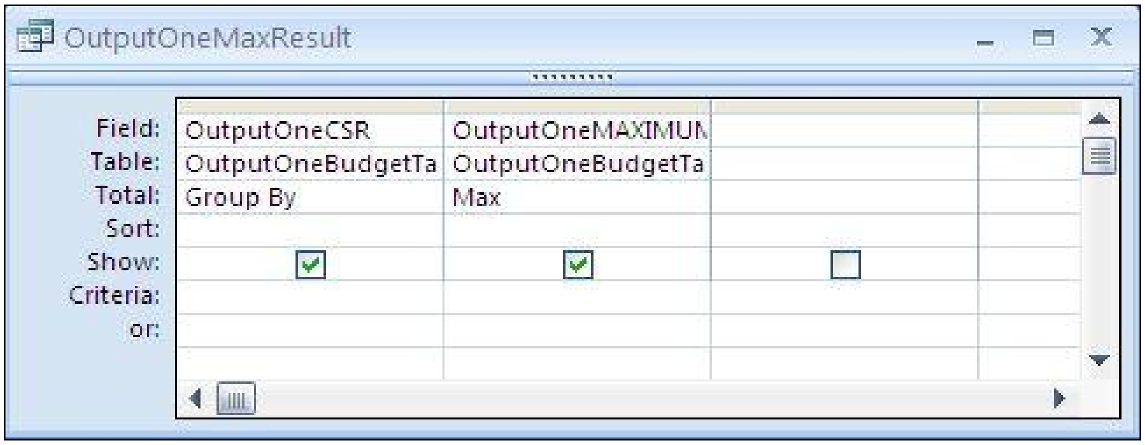
Based on the above database model, six queries were needed for each page of information to retrieve and store data into the database. Again since all the five pages hold the same controls of information, Figure 5.12 through 5.18 shows only the queries in SQL, Design and Datasheet Views for the first page labeled “UserFirstInputs”.

The OutputOneMaxResult query is set up to retrieve OutputOneCSR field and to maximum the OutputOneBudget field, giving results of the OutputOneCSR and the OutputOneMAXIMUM value corresponding to each CSR value from the above OutputOneBudgetTable.



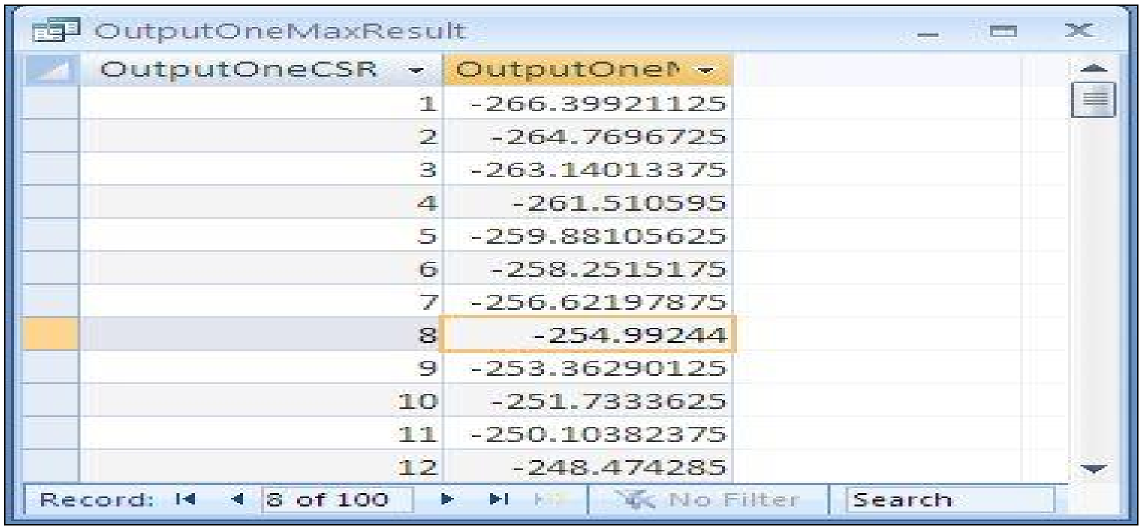
```
SELECT OutputOneBudgetTable.OutputOneCSR,
Max(OutputOneBudgetTable.OutputOneBudget) AS OutputOneMAXIMUM
FROM OutputOneBudgetTable
GROUP BY OutputOneBudgetTable.OutputOneCSR;
```

Figure 5.12a. OutputOneMaxResult query in SQLView



Field:	OutputOneCSR	OutputOneMAXIMUN	
Table:	OutputOneBudgetTa	OutputOneBudgetTa	
Total:	Group By	Max	
Sort:			
Show:	✓	✓	☐
Criteria:			
or:			

Figure 5.12b. OutputOneMaxResult query in Design View

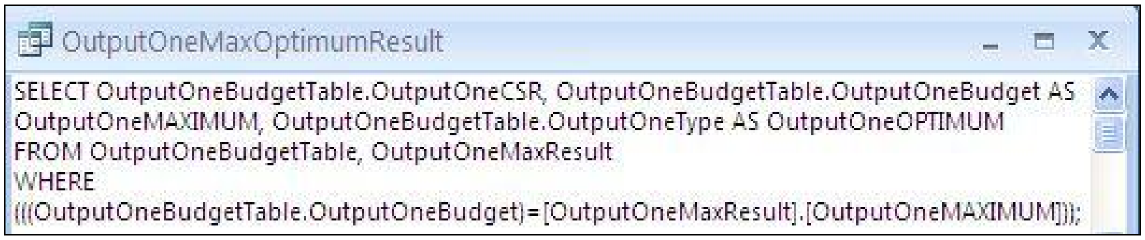


OutputOneCSR	OutputOneM
1	-266.39921125
2	-264.7696725
3	-263.14013375
4	-261.510595
5	-259.88105625
6	-258.2515175
7	-256.62197875
8	-254.99244
9	-253.36290125
10	-251.7333625
11	-250.10382375
12	-248.474285

Record: 8 of 100

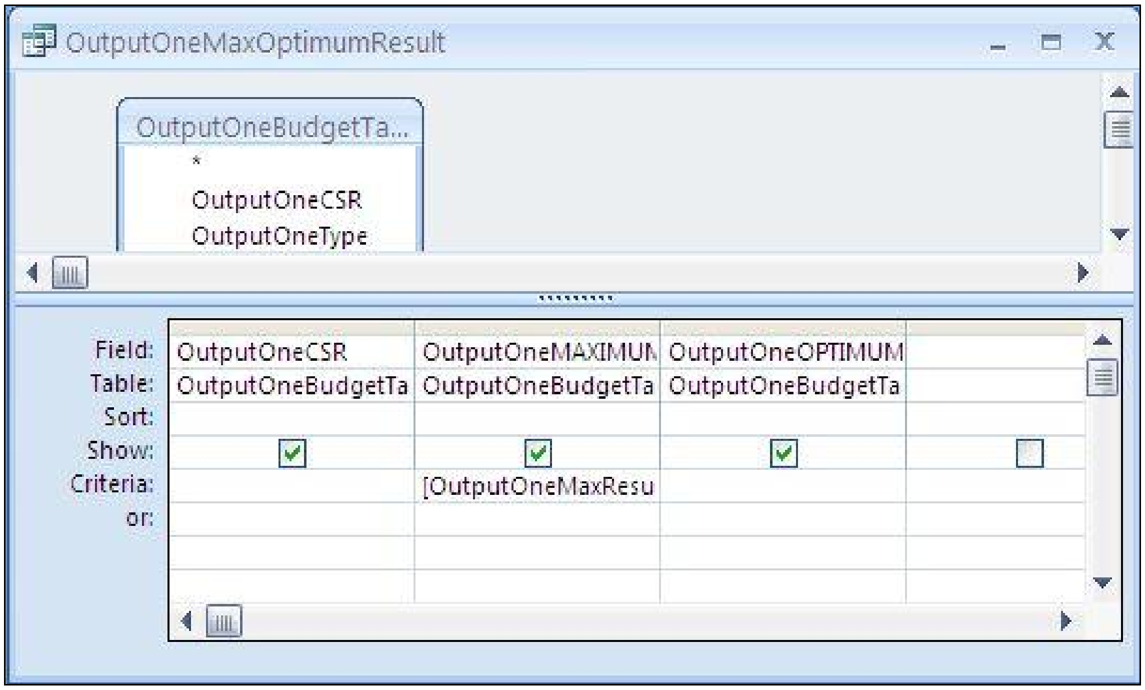
Figure 5.12c. OutputOneMaxResult query in Datasheet View

The OutputOneMaxOptimunResult query selects fields (OutputOneCSR, OutputOneBudget and OutputOneType) from previous OutputOneBudgetTable and then from both OutputOneBudgetTable and OutputOneMaxResult query, gets the optimum choice associated to each maximum value, giving results of the OutputOneCSR, the OutputOneMAXIMUM and the OutputOneOPTIMUM fields.



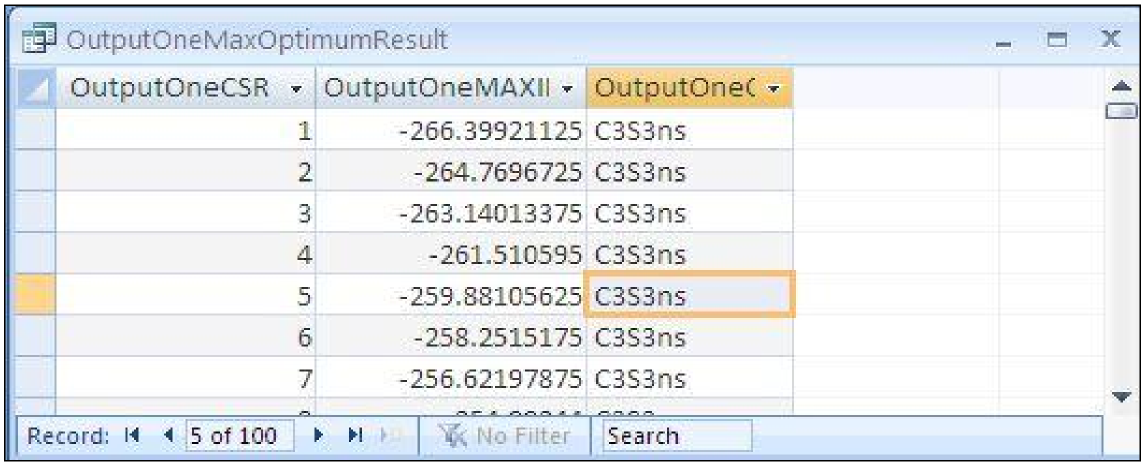
```
SELECT OutputOneBudgetTable.OutputOneCSR, OutputOneBudgetTable.OutputOneBudget AS
OutputOneMAXIMUM, OutputOneBudgetTable.OutputOneType AS OutputOneOPTIMUM
FROM OutputOneBudgetTable, OutputOneMaxResult
WHERE
(((OutputOneBudgetTable.OutputOneBudget)=[OutputOneMaxResult].[OutputOneMAXIMUM]));
```

Figure 5.13a. OutputOneMaxResult query in SQLView



Field:	OutputOneCSR	OutputOneMAXIMUN	OutputOneOPTIMUM	
Table:	OutputOneBudgetTa	OutputOneBudgetTa	OutputOneBudgetTa	
Sort:				
Show:	✓	✓	✓	☐
Criteria:		[OutputOneMaxResu		
or:				

Figure 5.13b. OutputOneMaxResult query in Design View

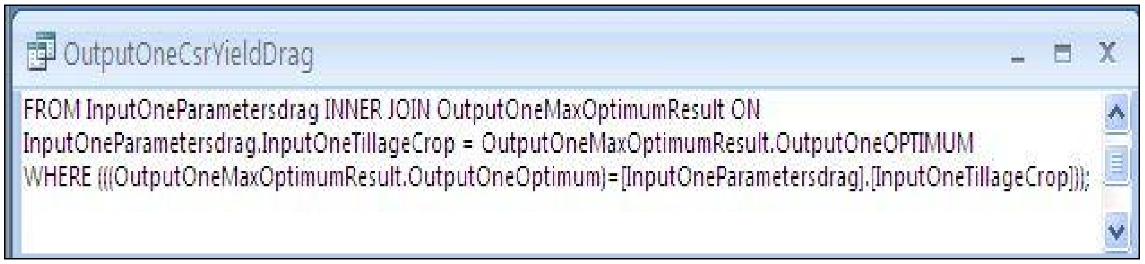

OutputOneMaxOptimumResult

OutputOneCSR	OutputOneMAXII	OutputOne(
1	-266.39921125	C3S3ns
2	-264.7696725	C3S3ns
3	-263.14013375	C3S3ns
4	-261.510595	C3S3ns
5	-259.88105625	C3S3ns
6	-258.2515175	C3S3ns
7	-256.62197875	C3S3ns

Record: 5 of 100 No Filter Search

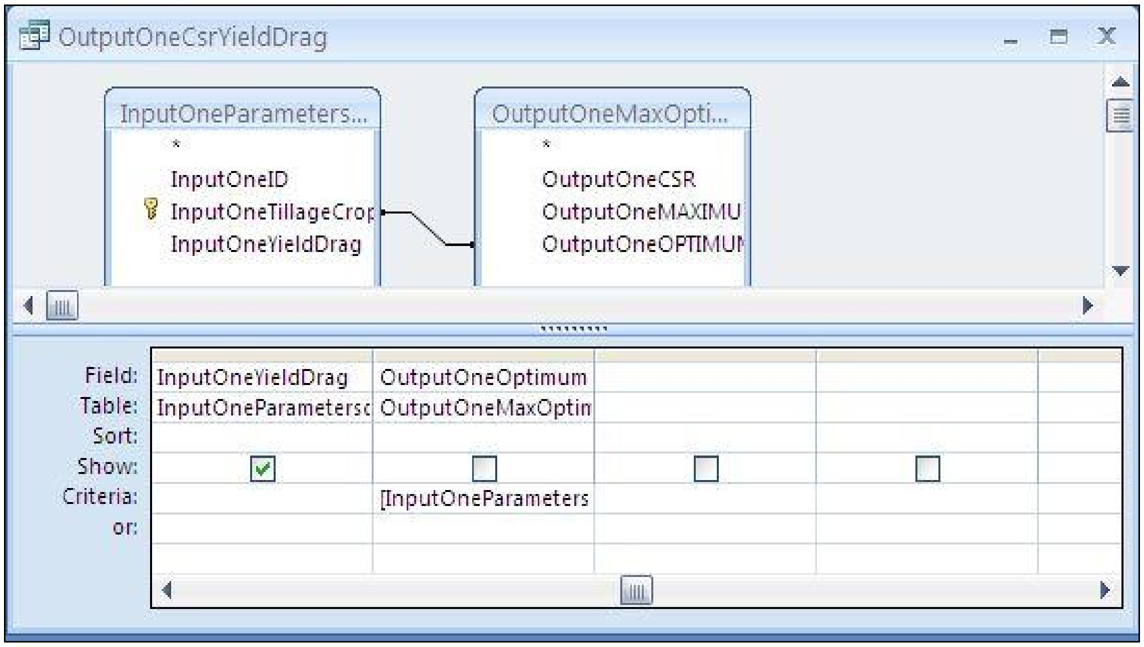
Figure 5.13c. OutputOneMaxResult query in Datasheet View

The OutputOneCsrYieldDrag query is a query with an Inner join; it joins InputOneParametersDrag table and OutputOneOptimumResult query using the InputOneTillageCrop field of InputOneParametersDrag table and OutputOneOptimum field of OutputOneOptimumResult query; where OutputOneOptimum type matches InputOneTillageCrop type. Thus, based on the join-predicate, the OutputOneCsrYieldDrag query retrieves OutputOneCSR field value from OutputOneOptimumResult query and the corresponding InputOneYieldDrag field value from InputOneParametersDrag table.


OutputOneCsrYieldDrag

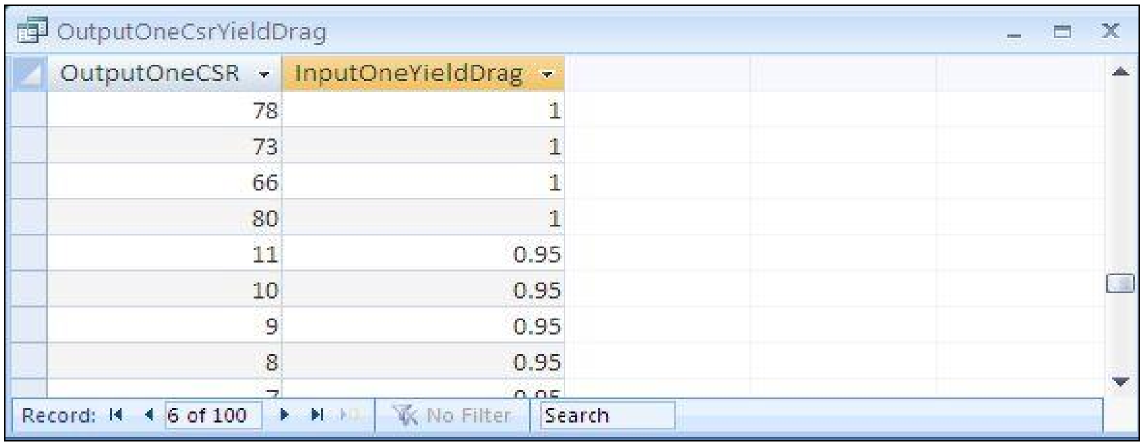
```
FROM InputOneParametersdrag INNER JOIN OutputOneMaxOptimumResult ON
InputOneParametersdrag.InputOneTillageCrop = OutputOneMaxOptimumResult.OutputOneOPTIMUM
WHERE (((OutputOneMaxOptimumResult.OutputOneOptimum)=[InputOneParametersdrag].[InputOneTillageCrop]));
```

Figure 5.14a. OutputOneCsrYieldDrag query in SQLView



Field:	InputOneYieldDrag	OutputOneOptimum		
Table:	InputOneParametersc	OutputOneMaxOptim		
Sort:				
Show:	✓	☐	☐	☐
Criteria:		[InputOneParameters		
or:				

Figure 5.14b. OutputOneCsrYieldDrag query in Design View

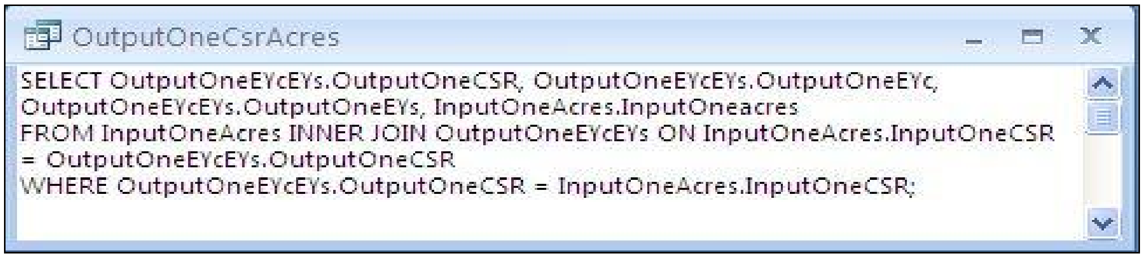


OutputOneCSR	InputOneYieldDrag
78	1
73	1
66	1
80	1
11	0.95
10	0.95
9	0.95
8	0.95
7	0.95

Figure 5.14c. OutputOneCsrYieldDrag query in Datasheet View

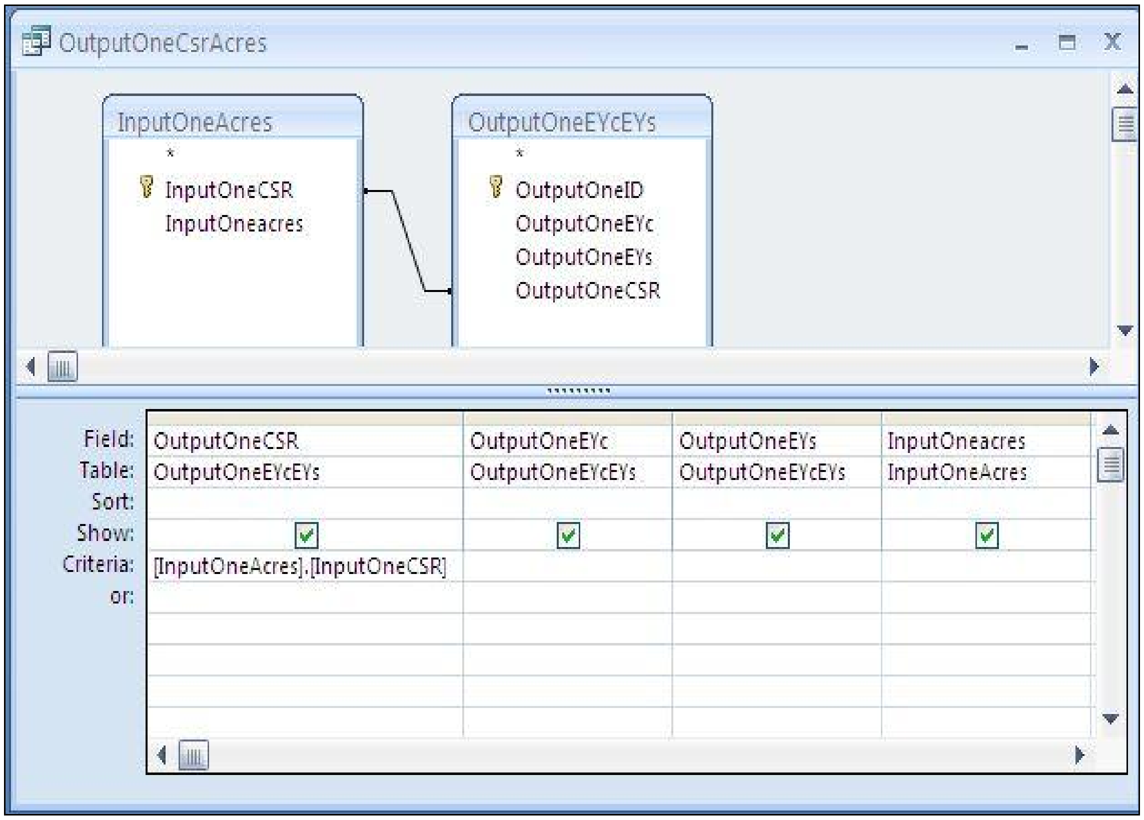
The OutputOneCsrAcres query inner joins InputOneAcres table on OutputOneEYcEYs table using the InputOneCSR field of InputOneAcres and OutputOneCSR of OutputOneEYcEYs; where OutputOneCSR matches InputOneCSR

Thus, based on the join-predicate, the OutputOneCsrAcres query retrieves OutputOneCSR field, OutputOneEyc and OutputOneEys from OutputOneEYcEYs table and their corresponding InputOneacres field value from InputOneAcres table.



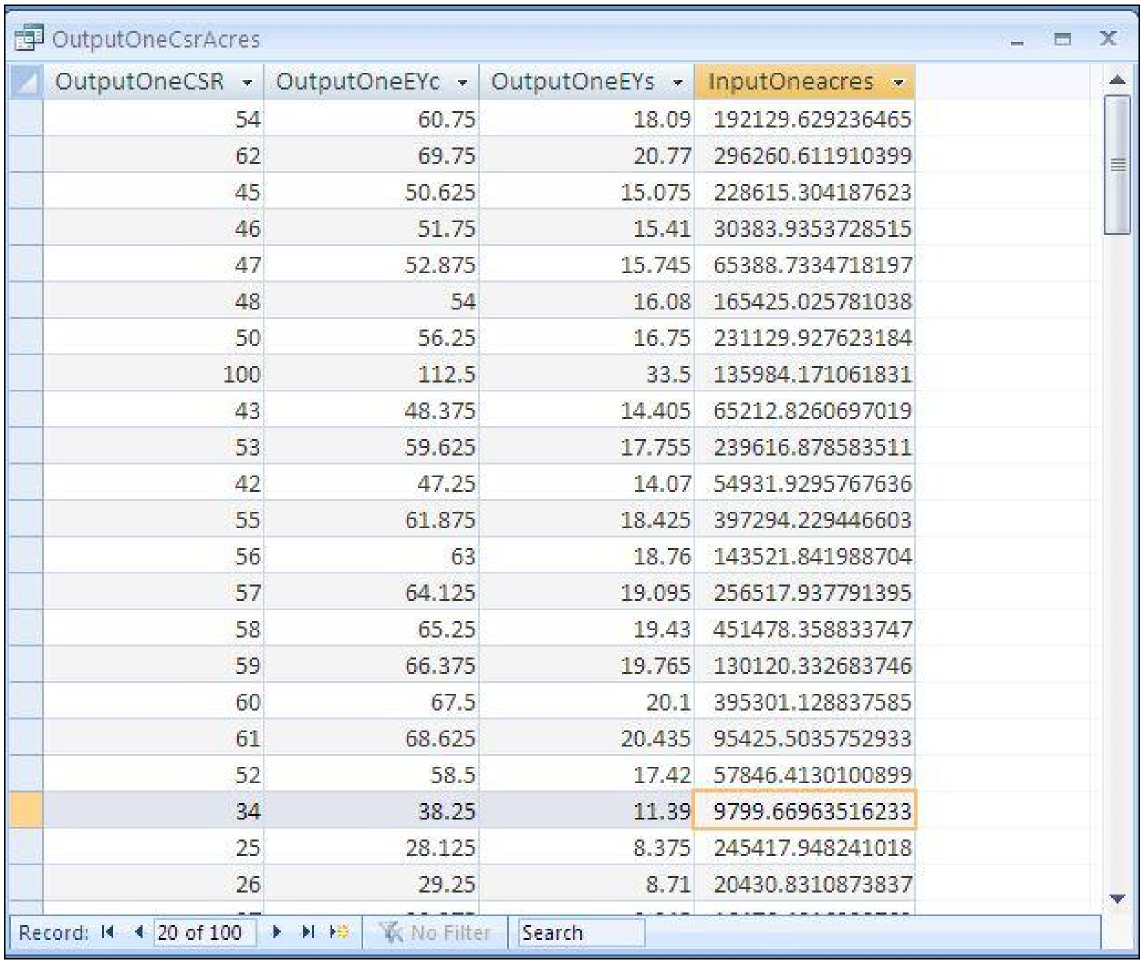
```
SELECT OutputOneEYcEYs.OutputOneCSR, OutputOneEYcEYs.OutputOneEYc,
OutputOneEYcEYs.OutputOneEYs, InputOneAcres.InputOneacres
FROM InputOneAcres INNER JOIN OutputOneEYcEYs ON InputOneAcres.InputOneCSR
= OutputOneEYcEYs.OutputOneCSR
WHERE OutputOneEYcEYs.OutputOneCSR = InputOneAcres.InputOneCSR;
```

Figure 5.15a. OutputOneCsrAcres query in SQLView



Field:	OutputOneCSR	OutputOneEYc	OutputOneEYs	InputOneacres
Table:	OutputOneEYcEYs	OutputOneEYcEYs	OutputOneEYcEYs	InputOneAcres
Sort:				
Show:	✓	✓	✓	✓
Criteria:	[InputOneAcres].[InputOneCSR]			
or:				

Figure 5.15b. OutputOneCsrAcres query in Design View



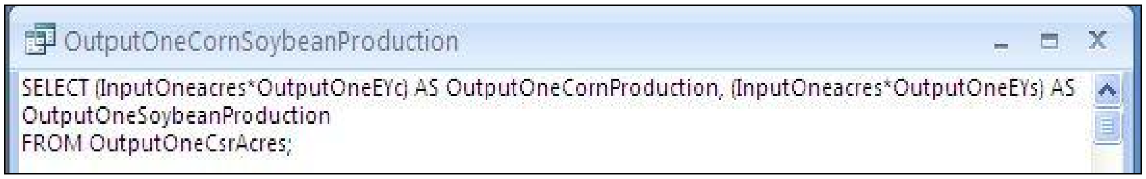
OutputOneCsrAcres

OutputOneCSR	OutputOneEYc	OutputOneEYs	InputOneacres
54	60.75	18.09	192129.629236465
62	69.75	20.77	296260.611910399
45	50.625	15.075	228615.304187623
46	51.75	15.41	30383.9353728515
47	52.875	15.745	65388.7334718197
48	54	16.08	165425.025781038
50	56.25	16.75	231129.927623184
100	112.5	33.5	135984.171061831
43	48.375	14.405	65212.8260697019
53	59.625	17.755	239616.878583511
42	47.25	14.07	54931.9295767636
55	61.875	18.425	397294.229446603
56	63	18.76	143521.841988704
57	64.125	19.095	256517.937791395
58	65.25	19.43	451478.358833747
59	66.375	19.765	130120.332683746
60	67.5	20.1	395301.128837585
61	68.625	20.435	95425.5035752933
52	58.5	17.42	57846.4130100899
34	38.25	11.39	9799.66963516233
25	28.125	8.375	245417.948241018
26	29.25	8.71	20430.8310873837

Record: 20 of 100 No Filter Search

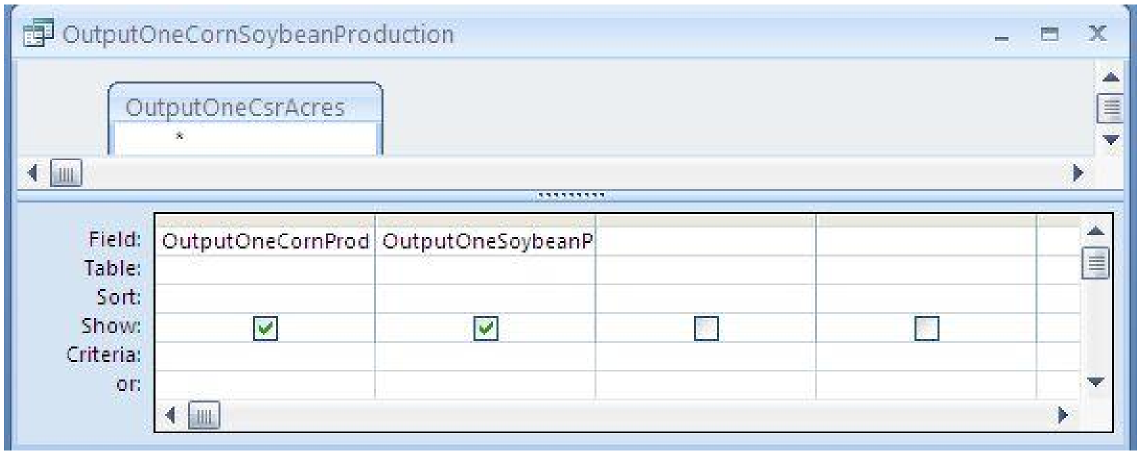
Figure 5.15c. OutputOneCsrAcres query in Datasheet View

The OutputOneCornSoybeanProduction query is a simple select query that retrieves fields (InputOneacres, OutputOneEYc and OutputOneEYs) from the OutputOneCsrAcres query and creates new fields (OutputOneCornProduction and OutputOneSoybeanProduction) for the OutputOneCornSoybeanProduction query ; giving that OutputOneCornProduction and OutputOneSoybeanProduction are equal respectively to InputOneacres multiplied by OutputOneEYc and InputOneacres multiplied by OutputOneEYs.



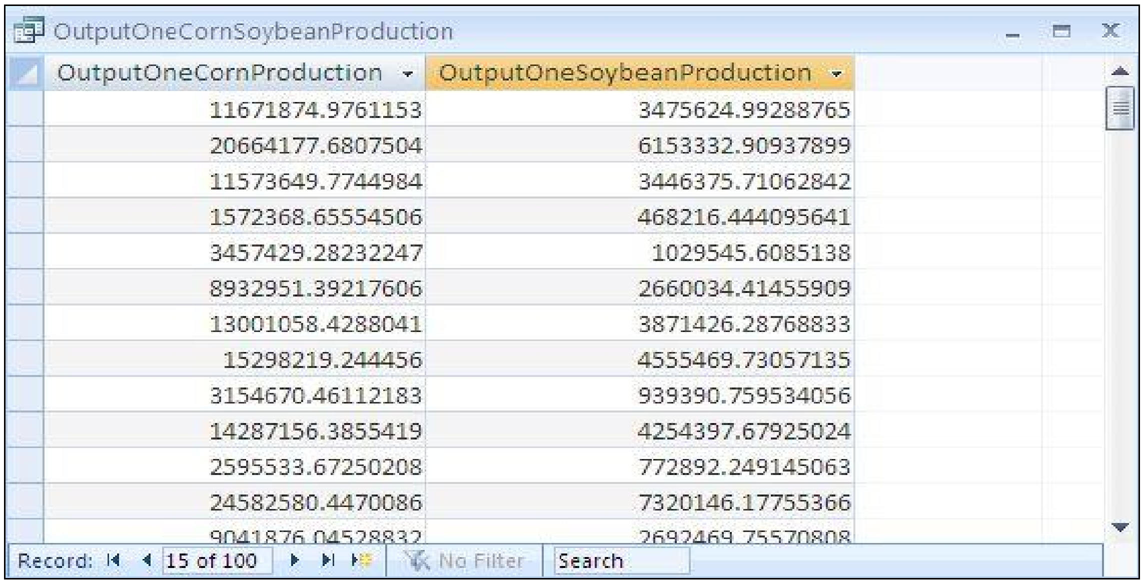
```
SELECT (InputOneacres*OutputOneEYc) AS OutputOneCornProduction, (InputOneacres*OutputOneEYs) AS
OutputOneSoybeanProduction
FROM OutputOneCsrAcres;
```

Figure 5.16a. OutputOneCornSoybeanProduction query in SQLView



Field:	OutputOneCornProd	OutputOneSoybeanP		
Table:				
Sort:				
Show:	✓	✓	☐	☐
Criteria:				
or:				

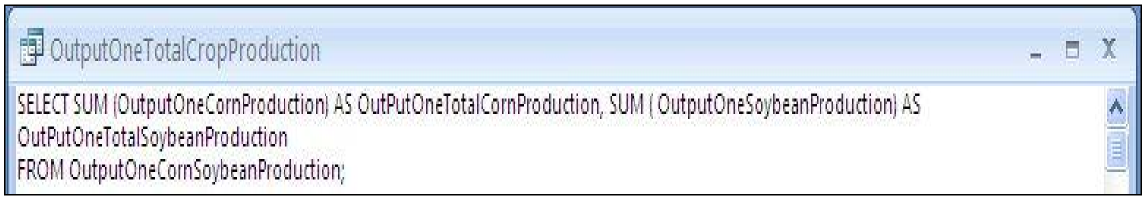
Figure 5.16b. OutputOneCornSoybeanProduction query in Design View



OutputOneCornProduction	OutputOneSoybeanProduction
11671874.9761153	3475624.99288765
20664177.6807504	6153332.90937899
11573649.7744984	3446375.71062842
1572368.65554506	468216.444095641
3457429.28232247	1029545.6085138
8932951.39217606	2660034.41455909
13001058.4288041	3871426.28768833
15298219.244456	4555469.73057135
3154670.46112183	939390.759534056
14287156.3855419	4254397.67925024
2595533.67250208	772892.249145063
24582580.4470086	7320146.17755366
9041876.04528832	2692469.75570808

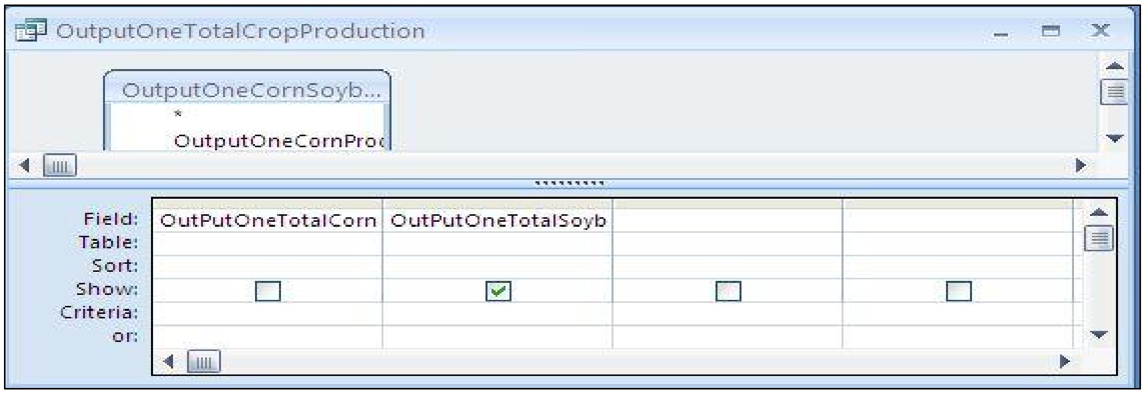
Figure 5.16c. OutputOneCornSoybeanProduction query in Datasheet View

The OutputOneTotalCropProduction query sums up the values in the two fields: [Output One Corn Production] and [Output One Soybean Production] from OutputOneCornSoybeanProduction query. For clarity, the resulting fields are named OutputOneTotalCornProduction and OutputOneTotalSoybeanProduction.



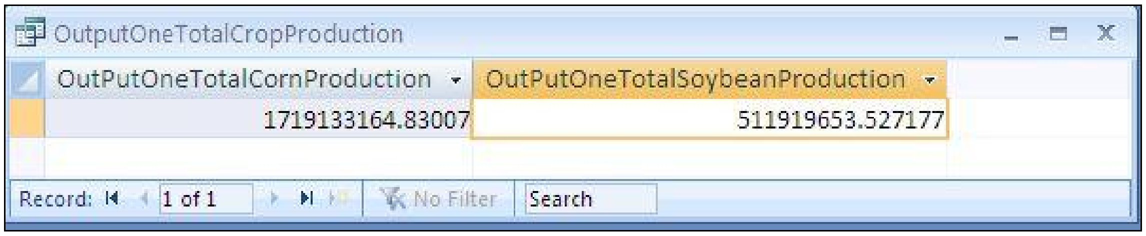
```
SELECT SUM (OutputOneCornProduction) AS OutPutOneTotalCornProduction, SUM ( OutputOneSoybeanProduction) AS
OutPutOneTotalSoybeanProduction
FROM OutputOneCornSoybeanProduction;
```

Figure 5.17a. OutputOneTotalCropProduction query in SQLView



Field:	OutPutOneTotalCorn	OutPutOneTotalSoyb		
Table:				
Sort:				
Show:	☐	✓	☐	☐
Criteria:				
or:				

Figure 5.17b. OutputOneCornSoybeanProduction query in Design View



OutPutOneTotalCornProduction	OutPutOneTotalSoybeanProduction
1719133164.83007	511919653.527177

Figure 5.17c. OutputOneCornSoybeanProduction query in Datasheet View

5.5 Plotting the Conditional Production Possibilities Frontier

In Microsoft Office Access, reports are utilized for enhanced record output. They allow us to represent the data through text and/or charts, thus both 2D and 3D conditional Production Possibilities Frontier have been plotted in two different reports as shown in Figure 5.18 and Figure 5.19.

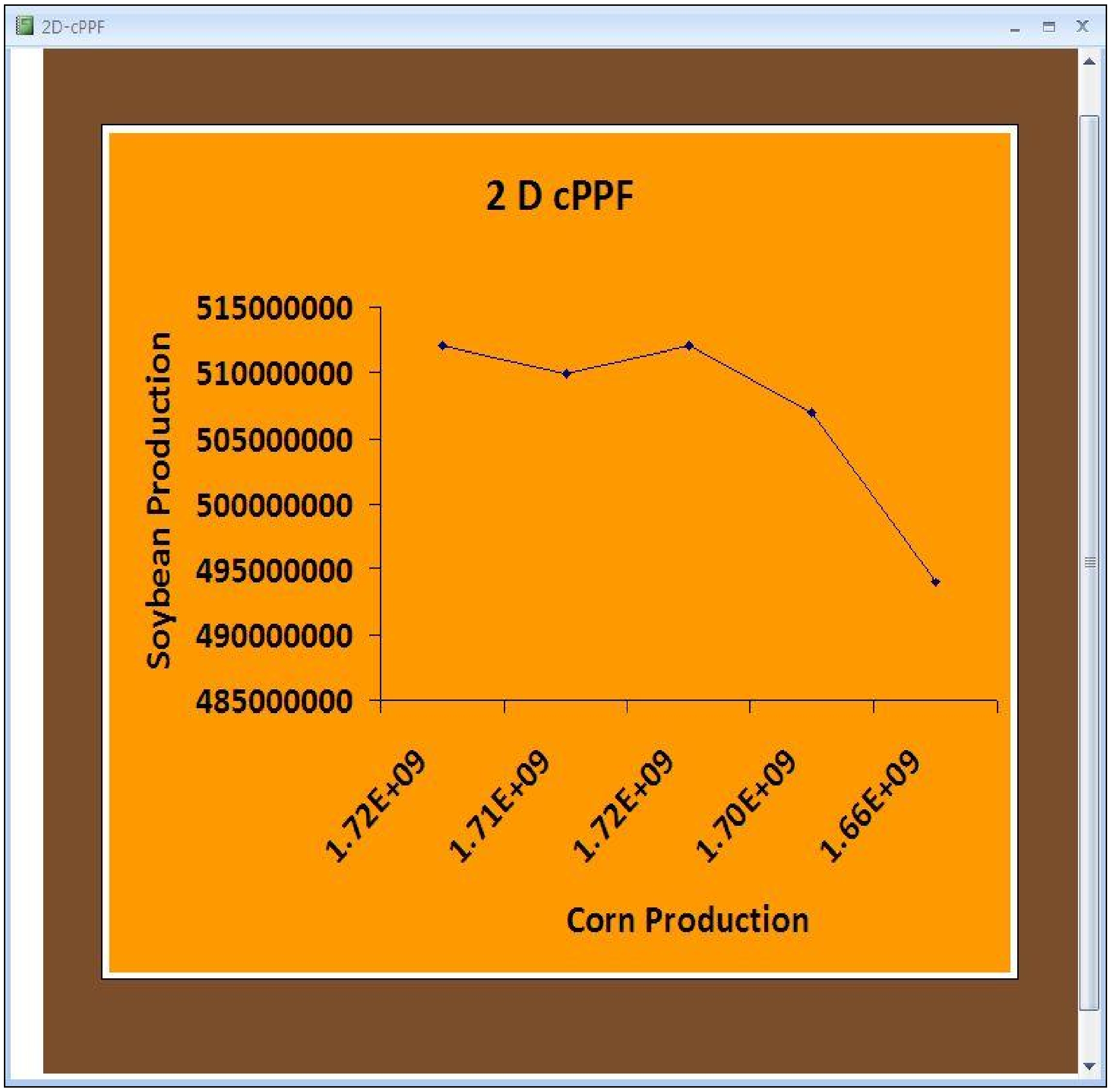


Figure 5.18. Screenshot of 2D cPPF plot in report View

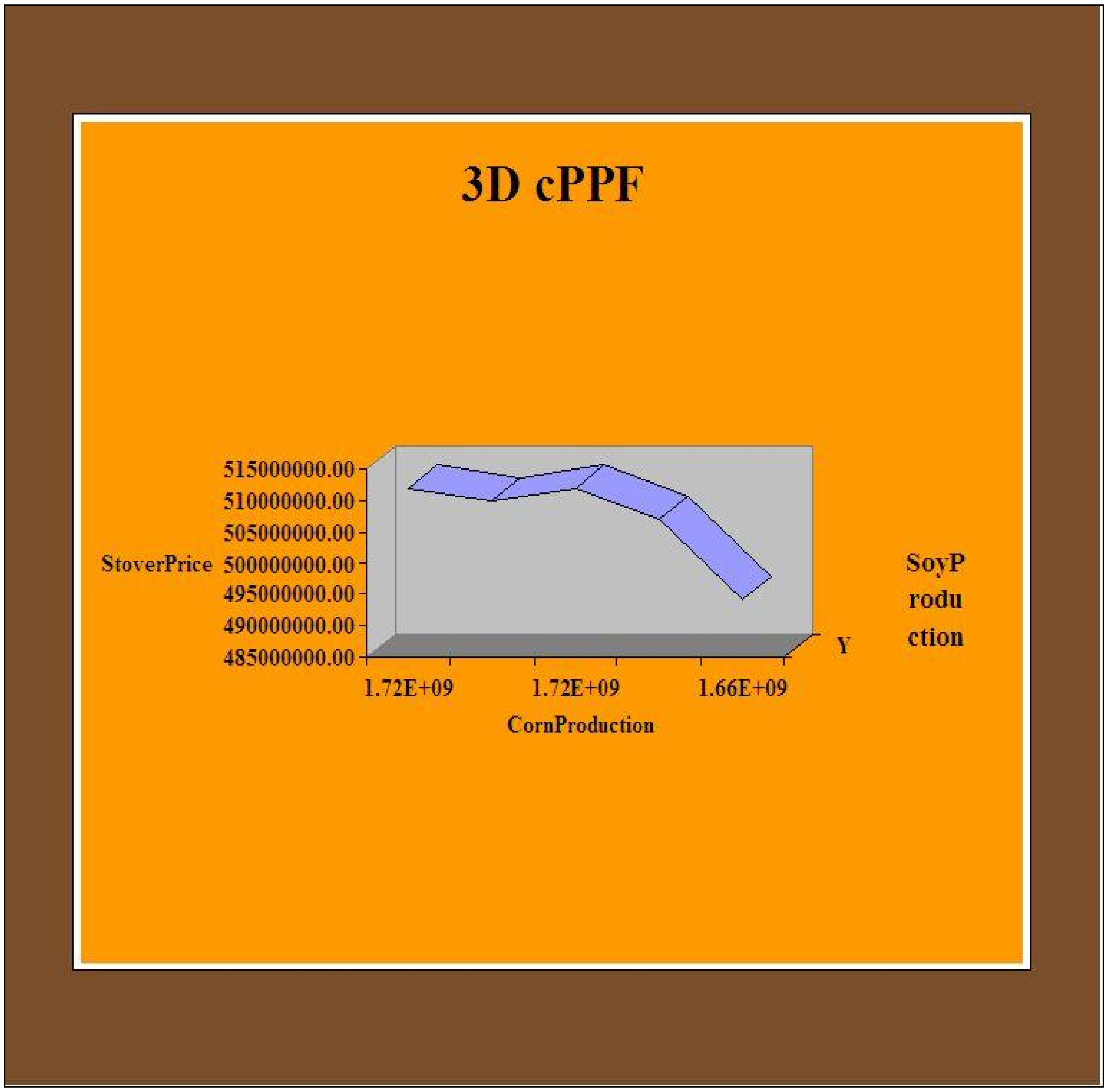
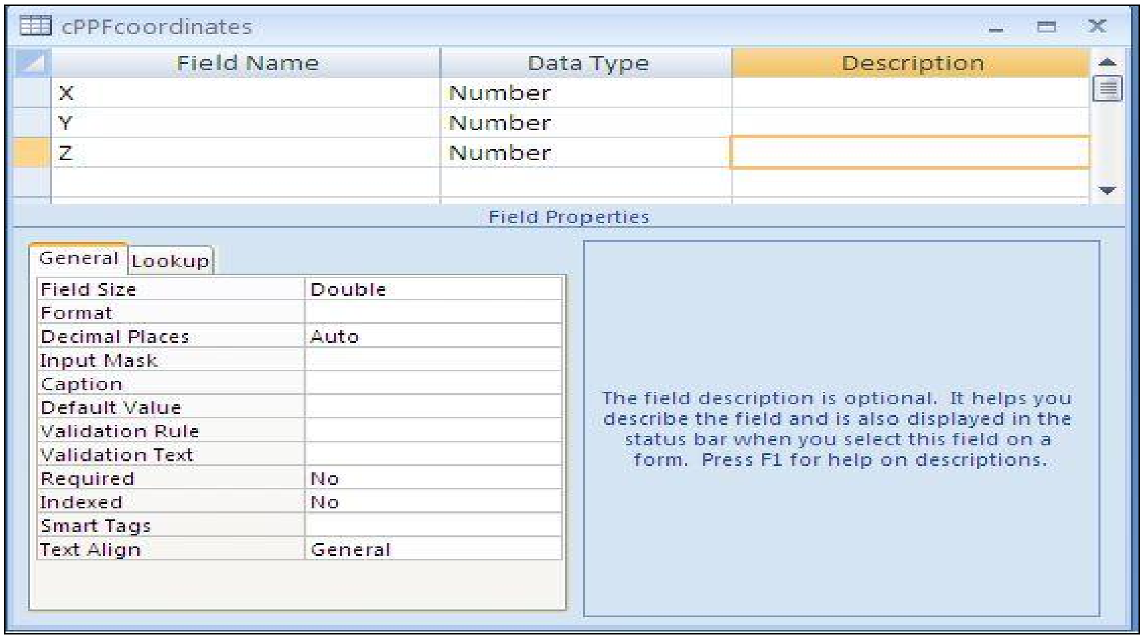


Figure 5.19. Screenshot of 3D cPPF plot in report View

Both reports get data from the cPPFcoordinates table. This table holds the *x y* and *z*-axes coordinates where total corn production along the *x*-axis, total soybean production along the *y*-axis, and Stover price along the *z*-axis.

The cPPFcoordinates table is the final table and it is structured in a way as to group total corn production, total soybean production and stover price resulting from all the five pages of information giving that each page of information corresponds to one point of coordinate *x, y* and *z* for the 3D and *x, y* for the 2D plots. Figure 5.20 (a) and Figure 5.20 (b) shows the cPPFcoordinates table in design and datasheet view.


cPPFcoordinates

Field Name	Data Type	Description
X	Number	
Y	Number	
Z	Number	

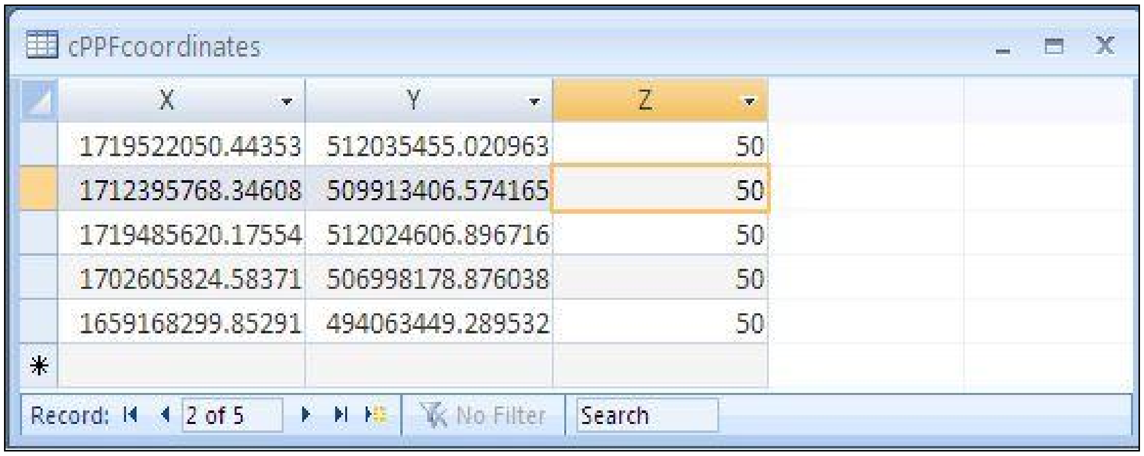
Field Properties

General | Lookup

Field Size	Double
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	No
Smart Tags	
Text Align	General

The field description is optional. It helps you describe the field and is also displayed in the status bar when you select this field on a form. Press F1 for help on descriptions.

Figure 5.20a. cPPFcoordinates table in Design View


cPPFcoordinates

X	Y	Z
1719522050.44353	512035455.020963	50
1712395768.34608	509913406.574165	50
1719485620.17554	512024606.896716	50
1702605824.58371	506998178.876038	50
1659168299.85291	494063449.289532	50

Record: 2 of 5 No Filter Search

Figure 5.20b. cPPFcoordinates table in Datasheet View

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis describes the design and implementation of an access based computational system that produces two-dimensional (corn production along the x-axis, and soybean production along the y-axis) and three-dimensional (corn production along the x-axis, soybean production along the y-axis, and Stover price along the z-axis) plots of conditional production possibilities frontier for corn and soybeans using Visual Basic Application programming language.

The process of creating this database took some time in implementing because the Microsoft Access software that is used is not very robust and has several limitations such as performance deficiencies when dealing with complex data. This database can be built rather quickly and with much greater ease using SQL Server or Oracle. However, the database has turned out to be a very useful tool for the future analyses of the economy-wide impacts of bioenergy production and policy. The database is friendly to use and can determine the quantity of corn and soybeans that can be producing for any given resources (prices of diesel, fertilizer).

Once again, just to maintain simplicity in the project, the Microsoft Access database has been used, since the volume of data which is being handled is not huge.

In order to get the maximum advantage from such database application, it is essential to keep them updated at regular intervals and also extreme care should be taken to avoid unsystematic data handling, thus user level security has been integrated in this application to provide a safe ground for producing and plotting conditional production possibilities frontier for corn and soybeans.

REFERENCES

[1] L. A. Kurkalova, S. Secchi, and P.W. Gassman, “Corn Stover Harvesting: Potential Supply and Water Quality Implications,”in *Handbook of Bioenergy Economics and Policy*, Springer: New York, 2010, vol. 33, pp. 307-323.

[2], [3] D. M. Gosnell, “Beginning Access 2003 VBA,” *Programmer to programmer,* Wiley: Indianapolis, c2004, pp. 12.

[4] B.C. English, R. J. Menard, D. G. De La Torre Ugarte, and M. Walsh, “Economic impacts of ethanol production from maize stover in selected Midwestern States,” in *Agriculture as a Producer and Consumer of Energy*. Edited by J. Outlaw, K.J. Collins, and J.A. Duffield, CABI Publishing, MA, 2005, pp. 218-231.

[5] J. Yong and S. M. Swinton, “Market interactions, farmers’ choices, and the sustainability of growing advanced biofuels: a missing perspective?” in *International Journal of Sustainable Development & World Ecology,* Vol. 16, No. 6, December 2009, pp. 438-450.

[6] “Production Possibility Frontier”.
URL: http://en.wikipedia.org/wiki/Production- possibility_frontier

[7] J. C. Tyndall, E. J. Berg, J. P. Colletti, “Corn stover as a biofuel feedstock in Iowa’s bio-economy: An Iowa farmer survey,” Biomass *and Bioenergy (2010),* doi:10.1016/j.biombioe.2010.08.049, pp. 1.

[8] L. A. Kurkalova, S. Secchi, and P.W. Gassman, “Corn Stover Harvesting: Potential Supply and Water Quality Implications,”in *Handbook of Bioenergy Economics and Policy*, Springer: New York, 2010, vol. 33, pp. 307.

[9] M.E Porter and V.E Millar, “How Information gives you competitive Advantage,” *Harvard Business Review*, July/August 1985,pp. 149-160.

[10] “Design & Development of a Database as a Part Data Management System” URL: http://www.min.uc.edu/robotics/papers/theses2000/chirag.PDF.

[11] J. W. Satzinger, R. B. Jackson, and S. D. Burd, “Systems&Analysis Design In a Changing World,” Course Technology, 2008, pp. 490.

[12] E. F. Codd's “12 rules for defining a fully relational database,” URL: http://www.cse.ohio-state.edu/~sgomori/570/coddsrules.html.

[13] C. N. Prague, M. Irwin, R. and J. Reardon, “Access 2003 Bible,” Wiley: New York, 2003, pp.4.

[14], [15] H. S. Sales and G. Mike, “Automation Microsoft Access with VBA,” Que Publishing, 2004, pp.12.

[16] J.L. Viescas, “Building Microsoft Access Applications*,”* Pap/Cdr edition ed, 2005: Microsoft Press, pp. 21.

[17] D. M. Gosnell, “Beginning Access 2003 VBA,” *Programmer to programmer,* Wiley: Indianapolis, c2004, pp. 1.

[18] J. Noel, “Microsoft Office Access 2003 Professional Results,” McGraw-Hill Osborne Media, August 2003, pp. 316.

[19] “ADO Compared with RDO and DAO”.
URL: http://msdn.microsoft.com/en-us/library/aa261340%28v=vs.60%29.aspx

[20] Visual Basic & ADO Tutorial”.URL: http://www.vb6.us/tutorials/database-access-ado-vb6-tutorial

[21] “Microsoft ActiveX Data Objects (ADO)”.
http://msdn.microsoft.com/en-us/library/ms675532%28v=vs.85%29.aspx

[22] D. M. Gosnell, “Beginning Access 2003 VBA,” *Programmer to programmer,* Wiley: Indianapolis, c2004, pp. 4.

APPENDIX

This appendix contains a sample of the Visual Basic Application code used for the database creation. Since all the five pages from the user interface hold the same controls of information, we only show the VBA code behind the first page labeled “UserFirstInputs”.

VBA code behind the click event of the Calculate Button.

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "Form_UserInterface"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Compare Database
Option Explicit

'***********************************************************
***********************************************************

' PROJECT: Impact of Biofuel
' AUTHOR: Mariama Oumarou Sidibe
' Student from North Carolina A & T State University
' Computational Science and Engineering Program
' DATE: 05/29/2010
' CONTACT: sidibemamy@hotmail.com, moumarou@ncat.edu
' DESCRIPTION: Implement  an access based computational
system that produces and plots 2D and 3D cPPF using VBA.

'***********************************************************
***********************************************************
```

```
Private Sub CmdCalculateOne_Click()

    'Variable declaration
    Dim db As Database
    Dim rsInputOneFertilizer As DAO.Recordset
    Dim rsInputOneParametersCost As DAO.Recordset
    Dim rsInputOneParametersBudget As DAO.Recordset
    Dim rsInputOneParametersBudgetb As DAO.Recordset
    Dim rsInputOneParametersBudgetc As DAO.Recordset
    Dim rsOutputOneCostOfCropProduction As DAO.Recordset
    Dim rsOutputOneCostOfCropProductionb As DAO.Recordset
    Dim rsOutputOneCostOfCropProductionc As DAO.Recordset
    Dim rsOutputOneCostOfStover As DAO.Recordset
    Dim rsOutputOneCostOfStoverb As DAO.Recordset
    Dim rsOutputOneCsrYieldDrag As DAO.Recordset
    Dim rsOutputOneYcYs As DAO.Recordset
    Dim rsOutputOneMaxOptimumResult As DAO.Recordset

    Dim sqlStrInputOneParametersCost As String
    Dim sqlStrInputOneFertilizer As String
    Dim sqlStrOutputOneCsrYieldDrag As String
    Dim sqlStrOutputOneYcYs As String
    Dim sqlStrOutputOneMaxOptimumResult As String
    Dim sqlInputOneParametersBudget As String
    Dim sqlInputOneParametersBudget_b As String
    Dim sqlInputOneParametersBudget_c As String
    Dim sqlOutputOneCostOfCropProduction As String
    Dim sqlOutputOneCostOfCropProduction_b As String
    Dim sqlOutputOneCostOfCropProduction_c As String
    Dim sqlOutputOneCostOfStover As String
    Dim sqlOutputOneCostOfStover_b As String

    Dim InputOneNitrogenPrice As Double
    Dim InputOnePhosphatePrice As Double
    Dim InputOnePotashPrice As Double
    Dim InputOneLPGasPrice As Double
    Dim OutputOneCostOfCropProduction As Double
    Dim OutputOneCostOfCollectingStover As Double
    Dim OutputOneEYc As Double
    Dim OutputOneEYs As Double
    Dim OutputOneYc As Double
    Dim OutputOneYs As Double
    Dim C1C1ns As Double
    Dim C2C2ns As Double
    Dim C3C3ns As Double
    Dim C1C1sr As Double
    Dim C2C2sr As Double
    Dim C3C3sr As Double
    Dim C1S1ns As Double
    Dim C2S2ns As Double
    Dim C3S3ns As Double
    Dim C1S1sr As Double
    Dim C2S2sr As Double
    Dim C3S3sr As Double
    Dim C1C1S1ns As Double
    Dim C2C2S2ns As Double
    Dim C3C3S3ns As Double
    Dim C1C1S1sr As Double
    Dim C2C2S2sr As Double
    Dim C3C3S3sr As Double

    Dim C1C1 As Integer
    Dim C2C2 As Integer
    Dim C3C3 As Integer
    Dim C1S1 As Integer
    Dim C2S2 As Integer
    Dim C3S3 As Integer
    Dim C1C1S1 As Integer
    Dim C2C2S2 As Integer
    Dim C3C3S3 As Integer
    Dim intCSROne As Integer
    Dim OutputOnecsrEYcEYs As Integer
    Dim csrOutputOneYcYs As Integer

    'Open connection to current Access database
    Set db = CurrentDb()

    'delete old data from the
OutputOneCostOfCropProduction, OutputOneCostOfStover and
OutputOneBudgetTable tables

db.Execute "DELETE * FROM OutputOneCostOfCropProduction"
db.Execute "DELETE * FROM OutputOneCostOfStover"
db.Execute "DELETE * FROM OutputOneBudgetTable"

    'Creating a record set for InputOneFertilizer table

sqlStrInputOneFertilizer = "select*from InputOneFertilizer"
```

```
Set rsInputOneFertilizer =
db.OpenRecordset(sqlStrInputOneFertilizer)

    'Creating a record set for InputOneParametersCost table

sqlStrInputOneParametersCost = "select * from
InputOneParametersCost"

Set rsInputOneParametersCost =
db.OpenRecordset(sqlStrInputOneParametersCost)

    'Getting data and do the calculation and write the
output

    If rsInputOneFertilizer.EOF = False Then
        If rsInputOneParametersCost.EOF = False Then

              'Calculating the prices

            InputOneNitrogenPrice =
rsInputOneFertilizer("InputOneNitrogenIntercept") +
(rsInputOneFertilizer("InputOneNitrogenSlope") *
TextDieselPriceOne.Value)

            InputOnePhosphatePrice =
rsInputOneFertilizer("InputOnePhosphateIntercept") +
(rsInputOneFertilizer("InputOnePhosphateSlope") *
TextDieselPriceOne.Value)

            InputOnePotashPrice =
rsInputOneFertilizer("InputOnePotashIntercept") +
(rsInputOneFertilizer("InputOnePotashSlope") *
TextDieselPriceOne.Value)

            InputOneLPGasPrice =
rsInputOneFertilizer("InputOneLPGasIntercept") +
(rsInputOneFertilizer("InputOneLPGasSlope") *
TextDieselPriceOne.Value)

         'Calculating the cost + the loopining 100 times
for each

             Do While Not rsInputOneParametersCost.EOF
                For intCSROne = 1 To 100
    'The equation for the cost of crop production for the 9
types

                    OutputOneCostOfCropProduction = 0

                    OutputOneCostOfCropProduction =
rsInputOneParametersCost("InputOnea0") +
(rsInputOneParametersCost("InputOnea1") * 2.25 * intCSROne
* rsInputOneParametersCost("InputOneYieldDrag")) +
(rsInputOneParametersCost("InputOnea2") *
TextDieselPriceOne.Value) +
(rsInputOneParametersCost("InputOnea3") *
InputOneLPGasPrice * 2.25 * intCSROne *
rsInputOneParametersCost("InputOneYieldDrag")) +
(rsInputOneParametersCost("InputOnea4") *
InputOneNitrogenPrice *
rsInputOneParametersCost("InputOneNitrogenRate")) +
(rsInputOneParametersCost("InputOnea5") *
InputOnePhosphatePrice *
rsInputOneParametersCost("InputOnePhosphateRate")) +
(rsInputOneParametersCost("InputOnea6") *
InputOnePotashPrice *
rsInputOneParametersCost("InputOnePotashRate"))

                    db.Execute "INSERT INTO
OutputOneCostOfCropProduction (OutputOneCSR, OutputOneType,
OutputOneCost) VALUES (" & intCSROne & ",'" &
rsInputOneParametersCost("InputOneCropTillage") & "'," &
OutputOneCostOfCropProduction & ");"

                Next
                rsInputOneParametersCost.MoveNext
            Loop
            rsInputOneParametersCost.MoveFirst

            Do While Not rsInputOneParametersCost.EOF
            If rsInputOneParametersCost("InputOneID") < 7
Then
                For intCSROne = 1 To 100

                    'The equation for the cost of stover
for the 6 types

                    OutputOneCostOfCollectingStover =
12.125 + (1.1 * TextDieselPriceOne.Value) + (2.25 *
intCSROne * rsInputOneParametersCost("InputOneYieldDrag") *
(0.01020851 + (0.00891947 * InputOneNitrogenPrice) +
(0.00263124 * InputOnePhosphatePrice) + (0.01114933 *
InputOnePotashPrice)))

                    db.Execute "insert into
OutputOneCostOfStover values (" & intCSROne & ",'" &
rsInputOneParametersCost("InputOneCropTillage") & "'," &
OutputOneCostOfCollectingStover & ");"

                Next
            End If

            rsInputOneParametersCost.MoveNext
            Loop

            '********************** BUDGET NO STOVER START
HERE *****************************************************

            ' ****** FIRST THREE EQUATIONS from the 9 ARE
THE SAME -- START ****************************************

            '1/9 ************ C1C1 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1C1'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            If rsInputOneParametersBudget.EOF = False Then
                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1C1'"

                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)
                If rsOutputOneCostOfCropProduction.EOF =
False Then
                    intCSROne = 0
                    Do While Not
rsOutputOneCostOfCropProduction.EOF
                            intCSROne = intCSROne + 1

                            C1C1ns = (2.25 * intCSROne *
rsInputOneParametersBudget("InputOneYieldDrag") *
```

```
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost")

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1C1ns'," & C1C1ns & ",'NoStover');"

rsOutputOneCostOfCropProduction.MoveNext

                    Loop
                    rsOutputOneCostOfCropProduction.Close

                Else
                    MsgBox "No Record for C1C1 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"

            End If

            '2/9 ************ C2C2 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2C2'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C2C2'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                If rsOutputOneCostOfCropProduction.EOF =
False Then
                    intCSROne = 0
                    Do While Not
rsOutputOneCostOfCropProduction.EOF
```

```
                            intCSROne = intCSROne + 1

                            C2C2ns = (2.25 * intCSROne *
rsInputOneParametersBudget("InputOneYieldDrag") *
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost")

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2C2ns'," & C2C2ns & ",'NoStover');"

rsOutputOneCostOfCropProduction.MoveNext

                    Loop
                    rsOutputOneCostOfCropProduction.Close

                Else
                    MsgBox "No Record for C2C2 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"

            End If

            '3/9 ************ C3C3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3C3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            If rsInputOneParametersBudget.EOF = False Then
                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3C3'"

                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)
                If rsOutputOneCostOfCropProduction.EOF =
False Then
```

```
                    intCSROne = 0
                    Do While Not
rsOutputOneCostOfCropProduction.EOF

                            intCSROne = intCSROne + 1

                            C3C3ns = (2.25 * intCSROne *
rsInputOneParametersBudget("InputOneYieldDrag") *
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost")

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3C3ns'," & C3C3ns & ",'NoStover');"

                   rsOutputOneCostOfCropProduction.MoveNext
                   Loop
                   rsOutputOneCostOfCropProduction.Close

                Else
                    MsgBox "No Record for C3C3 is found"
                End If

            rsInputOneParametersBudget.Close
            Else
                MsgBox "No data are found in the Budget
Parameter table"

            End If

            '******FIRST THREE EQUATIONS ARE THE SAME  END

            '*********** OTHER EQUATIONS from the 9 ARE THE
SAME -- START ***************************************

            '4/9 ************ C1S1 equation

             sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1S1'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S1C1'"
```

```
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            If rsInputOneParametersBudget.EOF = False Then


                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1S1'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S1C1'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF

                            intCSROne = intCSROne + 1

                            C1S1ns = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1S1ns'," & C1S1ns & ",'NoStover');"
rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
```

```
                    Loop
                    rsOutputOneCostOfCropProduction.Close

                Else
                    MsgBox "No Record for C1S1 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"

            End If

            '5/9 ************ C2S2 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2S2'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S2C2'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            If rsInputOneParametersBudget.EOF = False Then
          sqlOutputOneCostOfCropProduction = "select * from
OutputOneCostOfCropProduction where OutputOneType='C2S2'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)
                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S2C2'"

                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False Then
                    intCSROne = 0
                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF
```

```
                            intCSROne = intCSROne + 1

                            C2S2ns = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2S2ns'," & C2S2ns & ",'NoStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close

                Else
                    MsgBox "No Record for C2S2 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '6/9 ************ C3S3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3S3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S3C3'"
```

```
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3S3'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S3C3'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF

                            intCSROne = intCSROne + 1

                            C3S3ns = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3S3ns'," & C3S3ns & ",'NoStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
```

```
                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close

                Else
                    MsgBox "No Record for C3S3 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '7/9 ************ C1C1S1 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1S1'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S1C1'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1C1'"
            Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1S1'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S1C1'"
```

```
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

              sqlOutputOneCostOfCropProduction_c = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1C1'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF

                            intCSROne = intCSROne + 1

                            C1C1S1ns = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1C1S1ns'," & C1C1S1ns & ",'NoStover');"
rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    Loop
                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close

                Else

                    MsgBox "No Record for C1S1 or C1C1 or
S1C1 is found"
                End If

            rsInputOneParametersBudget.Close
            rsInputOneParametersBudgetb.Close
            rsInputOneParametersBudgetc.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '8/9 ************ C2C2S2 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2S2'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S2C2'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2C2'"
            Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C2S2'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)
```

```
                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S2C2'"

                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfCropProduction_c =
"select * from OutputOneCostOfCropProduction where
OutputOneType='C2C2'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF

                            intCSROne = intCSROne + 1

                            C2C2S2ns = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2C2S2ns'," & C2C2S2ns & ",'NoStover');"
rsOutputOneCostOfCropProduction.MoveNext
```

```
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close

                Else
                    MsgBox "No Record for C2S2 or S2C2 or
C2C2 is found"
                End If

            rsInputOneParametersBudget.Close
            rsInputOneParametersBudgetb.Close
            rsInputOneParametersBudgetc.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '9/9 ************ C3C3S3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3S3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S3C3'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3C3'"
            Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

            If rsInputOneParametersBudget.EOF = False Then
                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3S3'"
```

```
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S3C3'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfCropProduction_c =
"select * from OutputOneCostOfCropProduction where
OutputOneType='C3C3'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF

                            intCSROne = intCSROne + 1

                            C3C3S3ns = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3C3S3ns'," & C3C3S3ns & ",'NoStover');"
```

```
rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close

                Else
                    MsgBox "No Record for C3S3 or S3C3 or
C3C3 is found"
                End If

            rsInputOneParametersBudget.Close
            rsInputOneParametersBudgetb.Close
            rsInputOneParametersBudgetc.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '********* OTHER EQUATIONS ARE THE SAME - END
***********************************************************

            '***************** BUDGET NO STOVER END HERE
***********************************************************


            '******************BUDGET STOVER start HERE
***********************************************************

            '1/9 ************ C1C1 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1C1'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)
            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1C1'"
```

```
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C1C1'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfStover.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfStover.EOF

                            intCSROne = intCSROne + 1

                            C1C1sr = (2.25 * intCSROne *
rsInputOneParametersBudget("InputOneYieldDrag") *
TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1C1sr'," & C1C1sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
                    rsOutputOneCostOfStover.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfStover.Close

                Else
                    MsgBox "No Record for C1C1 is found"
                End If
            rsInputOneParametersBudget.Close
            Else
                MsgBox "No data are found in the Budget 
Parameter table"
            End If

        '2/9 ************ C2C2 equation

            sqlInputOneParametersBudget = "select * from 
InputOneParametersBudget where InputOneTillageCrop='C2C2'"
            Set rsInputOneParametersBudget = 
db.OpenRecordset(sqlInputOneParametersBudget)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select 
* from OutputOneCostOfCropProduction where 
OutputOneType='C2C2'"
                Set rsOutputOneCostOfCropProduction = 
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfStover = "select * from 
OutputOneCostOfStover where OutputOneType='C2C2'"
                Set rsOutputOneCostOfStover = 
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF = 
False And rsOutputOneCostOfStover.EOF = False Then

                    intCSROne = 0

                    Do While Not 
rsOutputOneCostOfCropProduction.EOF And Not 
rsOutputOneCostOfStover.EOF

                            intCSROne = intCSROne + 1

                            C2C2sr = (2.25 * intCSROne * 
rsInputOneParametersBudget("InputOneYieldDrag") * 
TextCornPriceOne.Value) + 
(TextStoverShareCollectedOne.Value * 0.0215 * 
TextStoverPriceOne.Value) - 
rsOutputOneCostOfCropProduction("OutputOneCost") - 
(TextStoverShareCollectedOne.Value * 
rsOutputOneCostOfStover("OutputOnecost"))
```

```
                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2C2sr'," & C2C2sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
                    rsOutputOneCostOfStover.MoveNext
                    Loop
                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfStover.Close

                Else
                    MsgBox "No Record for C2C2 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '3/9 ************ C3C3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3C3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            If rsInputOneParametersBudget.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3C3'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C3C3'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfStover.EOF = False Then
                    intCSROne = 0
                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfStover.EOF

                         intCSROne = intCSROne + 1

                            C3C3sr = (2.25 * intCSROne *
rsInputOneParametersBudget("InputOneYieldDrag") *
TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3C3sr'," & C3C3sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
                    rsOutputOneCostOfStover.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfStover.Close

                Else
                    MsgBox "No Record for C3C3 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '4/9 ************ C1S1 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1S1'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)
```

```
            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S1C1'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False Then
                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C1S1'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S1C1'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C1S1'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfStover.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfStover.EOF

                            intCSROne = intCSROne + 1

                            C1S1sr = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1S1sr'," & C1S1sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext

                    rsOutputOneCostOfStover.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfStover.Close

                Else
                    MsgBox "No Record for C1S1 or S1C1 is
found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '5/9 ************ C2S2 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2S2'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S2C2'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)
            If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False Then
```

```
                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C2S2'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S2C2'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C2S2'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfStover.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfStover.EOF

                            intCSROne = intCSROne + 1

                            C2S2sr = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2S2sr'," & C2S2sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext

                    rsOutputOneCostOfStover.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfStover.Close

                Else
                    MsgBox "No Record for C2S2 or S2C2 is
found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '6/9 ************ C3S3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3S3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S3C3'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3S3'"
```

```
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S3C3'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C3S3'"

                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfStover.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfStover.EOF

                            intCSROne = intCSROne + 1

                            C3S3sr = 0.5 * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost"))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3S3sr'," & C3S3sr & ",'WithStover');"
```

```
rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext

                  rsOutputOneCostOfStover.MoveNext

                  Loop

                  rsOutputOneCostOfCropProduction.Close
                  rsOutputOneCostOfCropProductionb.Close
                  rsOutputOneCostOfStover.Close

              Else
                  MsgBox "No Record for C3S3 or S3C3 is
found"
              End If

          rsInputOneParametersBudget.Close

          Else
              MsgBox "No data are found in the Budget
Parameter table"
          End If

          '7/9 ************ C1C1S1 equation

          sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1S1'"
          Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

          sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S1C1'"
          Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

          sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C1C1'"
          Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

          If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False And
rsInputOneParametersBudgetc.EOF = False Then
        sqlOutputOneCostOfCropProduction = "select * from
OutputOneCostOfCropProduction where OutputOneType='C1S1'"
```

```
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S1C1'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfCropProduction_c =
"select * from OutputOneCostOfCropProduction where
OutputOneType='C1C1'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C1S1'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                sqlOutputOneCostOfStover_b = "select * from
OutputOneCostOfStover where OutputOneType='C1C1'"

                Set rsOutputOneCostOfStoverb =
db.OpenRecordset(sqlOutputOneCostOfStover_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False And
rsOutputOneCostOfStover.EOF = False And
rsOutputOneCostOfStoverb.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF And Not
rsOutputOneCostOfStover.EOF And Not
rsOutputOneCostOfStoverb.EOF

                            intCSROne = intCSROne + 1

                            C1C1S1sr = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStoverb("OutputOneCost")))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C1C1S1sr'," & C1C1S1sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    rsOutputOneCostOfStover.MoveNext
                    rsOutputOneCostOfStoverb.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close
                    rsOutputOneCostOfStover.Close

                    rsOutputOneCostOfStoverb.Close

                Else
                    MsgBox "No Record for C1S1 or C1C1 or
S1C1 is found"
                End If
            rsInputOneParametersBudget.Close
            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '8/9 ************ C2C2S2 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2S2'"

            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S2C2'"
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C2C2'"
            Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

            If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False And
rsInputOneParametersBudgetc.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C2S2'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S2C2'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfCropProduction_c =
"select * from OutputOneCostOfCropProduction where
OutputOneType='C2C2'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)
```

```
                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C2S2'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                sqlOutputOneCostOfStover_b = "select * from
OutputOneCostOfStover where OutputOneType='C2C2'"
                Set rsOutputOneCostOfStoverb =
db.OpenRecordset(sqlOutputOneCostOfStover_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False And
rsOutputOneCostOfStover.EOF = False And
rsOutputOneCostOfStoverb.EOF = False Then

                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF And Not
rsOutputOneCostOfStover.EOF And Not
rsOutputOneCostOfStoverb.EOF

                            intCSROne = intCSROne + 1

                            C2C2S2sr = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStoverb("OutputOneCost")))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C2C2S2sr'," & C2C2S2sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    rsOutputOneCostOfStover.MoveNext
                    rsOutputOneCostOfStoverb.MoveNext

                    Loop

                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close
                    rsOutputOneCostOfStover.Close
                    rsOutputOneCostOfStoverb.Close

                Else
                    MsgBox "No Record for C2S2 or C2C2 or
S2C2 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '9/9 ************ C3C3S3 equation

            sqlInputOneParametersBudget = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3S3'"
            Set rsInputOneParametersBudget =
db.OpenRecordset(sqlInputOneParametersBudget)

            sqlInputOneParametersBudget_b = "select * from
InputOneParametersBudget where InputOneTillageCrop='S3C3'"
```

```
            Set rsInputOneParametersBudgetb =
db.OpenRecordset(sqlInputOneParametersBudget_b)

            sqlInputOneParametersBudget_c = "select * from
InputOneParametersBudget where InputOneTillageCrop='C3C3'"
            Set rsInputOneParametersBudgetc =
db.OpenRecordset(sqlInputOneParametersBudget_c)

            If rsInputOneParametersBudget.EOF = False And
rsInputOneParametersBudgetb.EOF = False And
rsInputOneParametersBudgetc.EOF = False Then

                sqlOutputOneCostOfCropProduction = "select
* from OutputOneCostOfCropProduction where
OutputOneType='C3S3'"
                Set rsOutputOneCostOfCropProduction =
db.OpenRecordset(sqlOutputOneCostOfCropProduction)

                sqlOutputOneCostOfCropProduction_b =
"select * from OutputOneCostOfCropProduction where
OutputOneType='S3C3'"
                Set rsOutputOneCostOfCropProductionb =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_b)

                sqlOutputOneCostOfCropProduction_c =
"select * from OutputOneCostOfCropProduction where
OutputOneType='C3C3'"
                Set rsOutputOneCostOfCropProductionc =
db.OpenRecordset(sqlOutputOneCostOfCropProduction_c)

                sqlOutputOneCostOfStover = "select * from
OutputOneCostOfStover where OutputOneType='C3S3'"
                Set rsOutputOneCostOfStover =
db.OpenRecordset(sqlOutputOneCostOfStover)

                sqlOutputOneCostOfStover_b = "select * from
OutputOneCostOfStover where OutputOneType='C3C3'"
                Set rsOutputOneCostOfStoverb =
db.OpenRecordset(sqlOutputOneCostOfStover_b)

                If rsOutputOneCostOfCropProduction.EOF =
False And rsOutputOneCostOfCropProductionb.EOF = False And
rsOutputOneCostOfCropProductionc.EOF = False And
rsOutputOneCostOfStover.EOF = False And
rsOutputOneCostOfStoverb.EOF = False Then
```

```
                    intCSROne = 0

                    Do While Not
rsOutputOneCostOfCropProduction.EOF And Not
rsOutputOneCostOfCropProductionb.EOF And Not
rsOutputOneCostOfCropProductionc.EOF And Not
rsOutputOneCostOfStover.EOF And Not
rsOutputOneCostOfStoverb.EOF

                            intCSROne = intCSROne + 1

                            C3C3S3sr = (1 / 3) * ((2.25 *
intCSROne * rsInputOneParametersBudget("InputOneYieldDrag")
* TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProduction("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStover("OutputOnecost")) + (0.67 *
intCSROne *
rsInputOneParametersBudgetb("InputOneYieldDrag") *
TextSoybeanPriceOne.Value) -
rsOutputOneCostOfCropProductionb("OutputOneCost") + (2.25 *
intCSROne *
rsInputOneParametersBudgetc("InputOneYieldDrag") *
TextCornPriceOne.Value) +
(TextStoverShareCollectedOne.Value * 0.0215 *
TextStoverPriceOne.Value) -
rsOutputOneCostOfCropProductionc("OutputOneCost") -
(TextStoverShareCollectedOne.Value *
rsOutputOneCostOfStoverb("OutputOneCost")))

                            db.Execute "INSERT INTO
OutputOneBudgetTable (OutputOneCSR, OutputOneType,
OutputOneBudget, OutputOneBudgetType) VALUES (" & intCSROne
& ",'C3C3S3sr'," & C3C3S3sr & ",'WithStover');"

rsOutputOneCostOfCropProduction.MoveNext
rsOutputOneCostOfCropProductionb.MoveNext
rsOutputOneCostOfCropProductionc.MoveNext

                    rsOutputOneCostOfStover.MoveNext
                    rsOutputOneCostOfStoverb.MoveNext

                    Loop
                    rsOutputOneCostOfCropProduction.Close
                    rsOutputOneCostOfCropProductionb.Close
                    rsOutputOneCostOfCropProductionc.Close
                    rsOutputOneCostOfStover.Close
                    rsOutputOneCostOfStoverb.Close

                Else
                    MsgBox "No Record for C3S3 or C3C3 or
S3C3 is found"
                End If

            rsInputOneParametersBudget.Close

            Else
                MsgBox "No data are found in the Budget
Parameter table"
            End If

            '********************** BUDGET STOVER End HERE
***********************************************************

            '********************** OutputOneCsrYieldDrag
start HERE ************************************************

            'delete old data from the OutputOneYcYs table
            db.Execute "DELETE * FROM OutputOneYcYs"

     'Creating a record set for OutputOneCsrYieldDrag table

            sqlStrOutputOneCsrYieldDrag = "select * from
OutputOneCsrYieldDrag"
            Set rsOutputOneCsrYieldDrag =
db.OpenRecordset(sqlStrOutputOneCsrYieldDrag)

            'Getting data and do the calculation and write
the output

            Do While Not rsOutputOneCsrYieldDrag.EOF

                    'Calculating the prices

                    OutputOneYc = 2.25 *
rsOutputOneCsrYieldDrag("InputOneYieldDrag") *
rsOutputOneCsrYieldDrag("OutputOneCSR")
```

```
                    OutputOneYs = 0.67 *
rsOutputOneCsrYieldDrag("InputOneYieldDrag") *
rsOutputOneCsrYieldDrag("OutputOneCSR")

                    csrOutputOneYcYs =
rsOutputOneCsrYieldDrag("OutputOneCSR")

                    db.Execute "INSERT INTO OutputOneYcYs
(OutputOneYc, OutputOneYs, OutputOneCSR) VALUES (" &
OutputOneYc & "," & OutputOneYs & "," & csrOutputOneYcYs &
");"
                    rsOutputOneCsrYieldDrag.MoveNext

            Loop

            rsOutputOneCsrYieldDrag.Close

            'After using the record set, free the memory it
was using by assigning Noting to it.

            Set rsOutputOneCsrYieldDrag = Nothing

            '********************** OutputOneCsrYieldDrag
End HERE **************************************************

            '********************** OutputOneEYcEYs start
HERE ******************************************************

            'delete old data from the OutputOneEYcEYs table
            db.Execute "DELETE * FROM OutputOneEYcEYs"

            'Creating a record set for OutputOneYcYs table
            sqlStrOutputOneYcYs = "select * from
OutputOneYcYs"
            Set rsOutputOneYcYs =
db.OpenRecordset(sqlStrOutputOneYcYs)

            sqlStrOutputOneMaxOptimumResult = "select *
from OutputOneMaxOptimumResult"
            Set rsOutputOneMaxOptimumResult =
db.OpenRecordset(sqlStrOutputOneMaxOptimumResult)

            'Getting data and do the calculation and write
the output
            Do While Not rsOutputOneMaxOptimumResult.EOF
```

```
                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C1C1ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C1C1sr" Then
                            C1C1 = 1
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C2C2ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C2C2sr" Then
                            C1C1 = 0
                            C2C2 = 1
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C3C3ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C3C3sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 1
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If
                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C1S1ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C1S1sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 1
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C2S2ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C2S2sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 1
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") = "C3S3ns"
Or rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C3S3sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 1
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If
                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C1C1S1ns" Or
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C1C1S1sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 1
                            C2C2S2 = 0
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C2C2S2ns" Or
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C2C2S2sr" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 1
                            C3C3S3 = 0
                    End If

                    If
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C3C3S3ns" Or
rsOutputOneMaxOptimumResult("OutputOneOPTIMUM") =
"C3C3S3ns" Then
                            C1C1 = 0
                            C2C2 = 0
                            C3C3 = 0
                            C1S1 = 0
                            C2S2 = 0
                            C3S3 = 0
                            C1C1S1 = 0
                            C2C2S2 = 0
                            C3C3S3 = 1
                    End If

                    Do While Not rsOutputOneYcYs.EOF

                    OutputOneEYc =
rsOutputOneYcYs("OutputOneYC") * 0.5 ^ (C1S1 + C2S2 + C3S3)
* (1 / 3) ^ (C1C1S1 + C2C2S2 + C3C3S3)

                    OutputOneEYs =
rsOutputOneYcYs("OutputOneYS") * (1 - C1C1 - C2C2 - C3C3) *
0.5 ^ (C1S1 + C2S2 + C3S3) * (1 / 3) ^ (C1C1S1 + C2C2S2 +
C3C3S3)
                    OutputOnecsrEYcEYs =
rsOutputOneYcYs("OutputOneCSR")

                    db.Execute "INSERT INTO OutputOneEYcEYs
(OutputOneEYc, OutputOneEYs, OutputOneCSR) VALUES (" &
OutputOneEYc & "," & OutputOneEYs & "," &
OutputOnecsrEYcEYs & ");"

                    rsOutputOneYcYs.MoveNext
                    Loop
                    rsOutputOneMaxOptimumResult.MoveNext

            Loop

            rsOutputOneYcYs.Close
            rsOutputOneMaxOptimumResult.Close

            'After using the record set, free the memory it
was using by assigning Noting to it.

            Set rsOutputOneYcYs = Nothing
            Set rsOutputOneMaxOptimumResult = Nothing

            '********************* OutputOneEYcEYs End
HERE *****************************************************

            MsgBox "The operation has been done
successfully"
        Else
            MsgBox "No data are found in Parameter table"
End If
    Else
```

```
        MsgBox "No data are found in InputOneFertilizer
table"
    End If

    rsInputOneFertilizer.Close
    rsInputOneParametersCost.Close

    'After using the record set, free the memory it was
using by assigning Noting to it.
    Set rsInputOneFertilizer = Nothing
    Set rsInputOneParametersCost = Nothing
    Set rsInputOneParametersBudget = Nothing
    Set rsInputOneParametersBudgetb = Nothing
    Set rsInputOneParametersBudgetc = Nothing
    Set rsOutputOneCostOfCropProduction = Nothing
    Set rsOutputOneCostOfCropProductionb = Nothing
    Set rsOutputOneCostOfCropProductionc = Nothing
    Set db = Nothing
End Sub
```

VBA code behind the click event of the reset Button.

```
Private Sub CmdResetThree_Click()
    TextDieselPriceThree = ""
    TextCornPriceThree = ""
    TextSoybeanPriceThree = ""
    TextStoverPriceThree = ""
    TextStoverShareCollectedThree = ""
End Sub
```

VBA code behind the click event of the close Button.

```
Private Sub CmdCloseOne_Click()
    DoCmd.Close acForm, "UserInterface", acSavePrompt
    Beep
End Sub
```

VBA code behind the click event of the diesel price text box.

```
Private Sub TextDieselPriceOne_Exit(Cancel As Integer)
    If IsNull(TextDieselPriceOne) Then
        MsgBox "You must enter the diesel price."
        TextDieselPriceOne.SetFocus
        Cancel = True
    End If
End Sub
```

VBA code behind the click event of the soybean price text box.

```
Private Sub TextSoybeanPriceOne_Exit(Cancel As Integer)
    If IsNull(TextSoybeanPriceOne) Then
        MsgBox "You must enter the soybean price."
        TextSoybeanPriceOne.SetFocus
        Cancel = True
    End If
End Sub
```

VBA code behind the click event of the stover price text box.

```
Private Sub TextStoverPriceOne_Exit(Cancel As Integer)
    If IsNull(TextStoverPriceOne) Then
        MsgBox "You must enter the stover price."
        TextStoverPriceOne.SetFocus
        Cancel = True
    End If
End Sub
```

VBA code behind the click event of the stover share collected price text box.

```
Private Sub TextStoverShareCollectedOne_Exit(Cancel As
Integer)
    If IsNull(TextStoverShareCollectedOne) Then
             MsgBox "You must enter the stover share
             collected price."
        TextStoverShareCollectedOne.SetFocus
        Cancel = True
    End If
End Sub
```

VBA code behind the click event of the corn price text box.

```
Private Sub TextCornPriceOne_Exit(Cancel As Integer)
    If IsNull(TextCornPriceOne) Then
        MsgBox "You must enter the corn price."
        TextCornPriceOne.SetFocus
        Cancel = True
    End If
End Sub
```