More On Supplemental Materials For Database Management System Knowledge And Skills In The Accounting Information Systems Course

Ceil Moran Pillsbury, Ph.D. (E-mail: ceil@uwm.edu), University of Wisconsin-Milwaukee

Ting J. Wang, Ph.D. (E-mail: tjwang@uwm.edu), University of Wisconsin-Milwaukee

Abstract

This paper presents educational materials to supplement the DBMS training in the Accounting Information System (AIS) course. It is a continuation of our work in "Supplemental Materials for Database Management System Knowledge and Skills in the Accounting Information Systems Course" (Pillsbury and Wang, 2001) in which we reviewed the changing nature of database education in AIS classes and provided supplemental active learning instructional aids to enhance the conceptual database coverage in AIS texts. In this paper, we provide a general discussion of Entity-Relationship based AIS architecture, Database Management Systems (DBMS) and the importance of database normalization. We then present guidelines for constructing queries, which are the real power behind any database management system. Finally, we provide instructional materials for teaching students how to prepare more complex queries. The "hands-on" query exercises we present are illustrated using MS Access 2000 but could be utilized with any other relational database program.

1. Introduction

Recent years have witnessed an increase in the usage and importance of relational database systems. Correspondingly, the coverage of database topics in the undergraduate Accounting Information Systems curriculum has also assumed greater importance. Many AIS textbooks now include strong conceptual coverage of database topics, but could be enhanced by more "hands-on" student learning materials. Our previous work presented hands-on exercises that covered tables, forms, simple queries and macros (Pillsbury and Wang, 2001). This current paper addresses the greater challenge for students, that is, how to properly construct more complex queries.

We proceed with a general discussion of data-based AIS architecture, Database Management Systems (DBMS) and the need for, and forms of, database normalization. We then proceed to the main purpose of this paper, which is to present general guidelines for constructing queries and the query exercises. These exercises are sufficiently robust to provide students with a fundamental understanding of query capabilities. The nature of the exercises is such that they are intended for instructors who lack the time or faculty reward structure to prepare DBMS application materials on their own, or for those who have little or no formal training in DBMS. For those professors wanting a more demanding set of complicated query building materials we refer you to Borthick et al. (2001) and Geerts and Waddington (2000).

Readers with comments or questions are encouraged to contact the authors via email.

2. Entity-Relationship Based AIS Architecture

Conventional entity-relationship (E-R) database systems provide many widely recognized advantages compared to the file-based computerized accounting systems companies traditionally used. These advantages include:

- Minimizes data redundancy which
 - Results in lower costs
 - Reduces errors during the input and storage stages
 - Avoids duplicate processing
 - And simplifies maintenance
- Centralization of data management and security
- Greater flexibility of the system including
 - Simplified report modifications and updates
 - Multiple uses of data and cross-functional data analysis
 - The ability to record non-financial information such as machine downtime, employee skills

The subject of this paper is another significant advantage of a database system--that is, the ability of such systems to provide routine and ad-hoc query capability. Queries are the method by which data is retrieved and organized from the database system in order to answer relevant management inquiries such as "What are sales for the month of June?" and "How much do we owe Vendor X?" Database systems easily accommodate routine information requests such as these, but they also allow the ability to create answers to ad-hoc queries, such as "How many white gloves did Tommy Bonds sell to ladies over the age of 25 in the city of Chicago during the month of December?" This query ability is one of the most important features of a database system.

In the conventional entity-relationship database system, all of the potential information about business activities is first identified and then organized and stored in entities. The decision on which data to store is made by analyzing all the potential financial and non-financial information needs of management and internal/external users of the company. The database system typically identifies entities that can be defined as any Person, Place, Object, Event, or Concept (McFadden, Hoffer, and Prescott, 1999) or, more specifically, identifies Resources, Events, and Agents in the REA model¹ (McCarthy, 1982). An example of an entity (an Agent in the REA model) would include, but is not limited to, a customer, vendor, supplier, employee, etc. An entity (an Event in the REA model) could also be a credit sale, cash sale, cash payment, customer payment, etc.

In the typical database system, data is stored in tables by entity. Thus one would expect to find a customer table, a vendor table, a cash receipts table, a sales table, etc. Relationships among entities are then established based on the limitations imposed in the business rules, in order to link the data stored in the various entities. As a result, data about business activities are organized and stored in the entities and linked by the relationships. Information stored in two or more related entities is retrieved through the use of a query language, which is discussed in greater detail later in the paper. Figure 1 demonstrates the contrast between traditional AIS architecture and E-R based AIS architecture.

Figure 1. Traditional vs. E-R-based AIS architecture

3. Database Normalization

The fundamental storage mechanism of a database is the table, often called a file. Of critical importance in the database is the concept of normalization. Database normalization is the process of ensuring that each table contains data about only ONE concept. Without database normalization serious operational errors can occur in the database, including but not limited to, insertion/deletion/update anomalies. Insertion anomalies refer to problems occurring when inserting new items into the database, deletion anomalies refer to the unintentional deletion of an item from the database and update anomalies refer to the problems that occur when updating certain values (e.g. price of an item) within the database.

In order to achieve database normalization, certain forms (also called properties or constraints) must be imposed on the entities (tables) in the database. The least restrictive is called the first normal form followed by the second, third, Boyce-Codd, fourth, fifth, and Domain/Key forms. Most accounting systems require the use of the first three normal forms (Perry and Schneider 2000). The first normal form restricts repeating attributes (or fields) and divisible data in an attribute, the second normal form requires dependency of the non-key attribute(s) on the key attribute(s), and the third normal form eliminates transitive dependency which means that non-key attributes can not be dependent on any other non-key attributes (for more detailed information, see Gelinas, Sutton, and Oram, 1999).

Although systems designers, not accountants, are typically responsible for the process of normalizing a database, an accountant must still understand the normalization process and be able to tell if a table has been properly normalized (Hall 2001). The issues an accountant can experience without proper database normalization are significant: "The update anomaly can generate conflicting and obsolete database values; the insertion anomaly can result in unrecorded transactions and incomplete audit trails; and the deletion anomaly can cause the loss of accounting records and the destruction of audit trails" (Hall 2001, p. 482). Hence, the proper normalization of the database is of critical importance to the accountant.

Once the data is stored in properly normalized entities (tables), it cannot be retrieved unless relationships between the appropriate entities are created. Three different types of relationships can be established between two entities: One-to-One, One-to-Many (or Many-to-One), and Many-to-Many. The types of relationships required are based on the particular business' rules about Person, Event, Object, Place, and Concept imposed in the organization. For example, if a company allows a salesperson (Person) to process many sales transactions (Event), but a particular sales transaction can only be handled by no more than one salesperson, then a One-to-Many relationship is required between Salesperson and Sales tables. However, if the company had instead structured transactions so that more than one salesperson is allowed to be associated with a sales transaction then a Many-to-Many relationship should be used instead.

In order to establish relationships between entities, common attribute(s) must be included in the entities (i.e., there must be identical data fields in the related tables). For this reason, data redundancy cannot be totally eliminated in the database. Without the relationships between entities, for example, information about Person (as entities), Place, Object, Event, and Concept are all isolated from each other, not only between the types of entities, but also within the type of entity (for example, sales event and inventory, and sales event and cash receipt event, respectively). As a result, information stored in more than one entity can be properly extracted only when relationships are built between these entitiesⁱⁱ. For example, information about "Event" involving "Object" and being carried out by "Person" at "Place" can only be retrieved from the database system if relationships are properly created between these "Event", "Object", "Person", and "Place" entities.

4. Database Management Systems

A database management system (DBMS) is a set of integrated programs designed to simplify the tasks of creating, accessing, and managing a database (for more detailed information, see Romney and Steinbart, 2000). An important aspect of the DBMS for accountants is that it has many embedded controls to ensure proper functionality of the database. Examples would include database lockout and concurrency controls that ensure that multiple users processing similar data do not compromise the accuracy of the database. Another essential element of a DBMS is the query language that permits users and programmers to directly access data in the database without the need for conventional programs and programmers. For example, IBM's Structured Query Language (SQL, pronounced sequel) has emerged as the standard query language for both mainframe and microcomputer DBMSs. SQL is a

fourth-generation, nonprocedural language with many textual commands that allow users to input, retrieve, and modify data easily (Hall 2001, p. 459). Even more helpful for non-technical users is Query By Example (QBE), a visual query language that employs graphical and grid-like query windows into which users enter examples of the form of the desired answer. The QBE then generates the appropriate SQL thereby allowing a non-technical user to bypass the SQL programming language altogether and directly access the database. Microsoft Access 2000 utilizes QBE, but it also allows the user to view the SQL directly.

5. Guidelines For Constructing Queries

Students' familiarity with query construction is critical because queries represent the real power of a database. We have developed general guidelines for constructing queries below. Figure 2 illustrates the logic flow of these guidelines.

5.1. Understand The Request Of Information Customers

The first step involved in constructing queries is to know WHAT information is being requested. If the person who constructs the query is not the user who requests the information, he or she has to communicate clearly with the user and understand exactly the information requested. Depending on the needs of the users, different data fields may need to be included. For example, an accounts receivable (A/R) aging report would normally contain the following fields of information, in addition to the report heading: invoice date, invoice number, customer name, invoice amount, and days overdue. The report may list the information by days overdue, by invoice number, or by customer name depending on the preferences of the information customers. The report may also cover everything to date or up to a specific date (or other criteria on different fields of information—discussed in Step 4). In addition, a user might request the contact name and phone number be included on the report. Or a different user might prefer a field of information reporting the credit rating of the customer. Clearly, a wide variety of information could be provided on the report solely dependent on the information user's preferences. As a result, the person who is constructing and executing the query, either programmer or user, must understand the exact data fields of information and criteria needed for each of the fields when constructing a query.

5.2. Identify The Sources Of Information Items

The second step in developing queries is to identify the source tables for each of the fields listed from the first step in the database. Each field of information may be retrieved directly from one of the tables, indirectly through an operation (e.g., calculation, etc.) involving one or more tables (or generated dataset), or from the system itself as constants (e.g., current date, a fixed rate, etc.). In the A/R aging report example, for instance, invoice date and number can be retrieved from the Invoice table and customer name from the Customer table. The invoice amount is most likely obtained from a calculation of quantity ordered (from the Invoice-item table) and selling price (from the Product table). The current date used in calculating the number of days old can be provided as a system constant. Sometimes the information needed is found in a virtual (or result) tableⁱⁱⁱ obtained from another query that becomes the source of information for the query being developed. Obviously, the construction of queries requires a detailed understanding of the table structures as well as the formulas involved in the data fields. When constructing queries, tables containing the appropriate information items are identified. It is very important to make sure that unnecessary tables are not identified since this may adversely affect the query results (discussed next). In addition, it is important to make sure that proper relationships exist between the tables or the query results will be unstable.

5.3. Be Aware of the Effects of Joining When Involving Two or More Tables

From the previous steps, data fields are listed and sources of the data fields are identified. If the data fields are stored in more than one table, a join of the tables will be defined/established. A specific type of join may be required depending on the information we want to obtain from the tables. The following are the types of joins that are commonly used:

- **Natural Join**
This join includes instances (records or rows) where the joining data fields from these tables are identical/equal. In our previous A/R aging report example, a natural join will be required between Invoice and Customer tables in order to retrieve the customer names from the Customer table (according to the key values from the Customer table stored in the Invoice table), assuming that they have Many-to-One relationship^{iv}. In fact, a natural join is called upon when the joining tables have a Many-to-One relationship and the basis of information requested is from the table that has the “many” relationship in the relationship^v (e.g., the Invoice table in our example).
- **Outer Join**
This join includes all instances from one table and only those instances from the other table where the joining fields are identical. There are left and right outer joins according to the side the first table is on. Suppose that we want to obtain information about A/R subsidiary ledger accounts. Specifically, we want to include all customers regardless of the number of invoices (stored in the Invoice table) with which they are involved. As a result, an outer join will be performed between Invoice and Customer tables, the same tables involved in the A/R aging report example. However, in this case, the type of join required will be different since the basis of the information switches from the Invoice table to the Customer table. In other words, all of the customers in the Customer table will be included in the result regardless if they have been issued an outstanding invoice or not.

- **Cartesian Product Effect**
When unrelated tables (i.e., no identical/common fields existed) are included in a query, the result table will include combinations of the instances from all tables. For example, two tables, A and B, without any common fields are included in a query. A has two fields and two records (x y: 3 4 and 5 6) and B has one field with two records (z: 7 and 8). A Cartesian Product, consisting of three fields and four records, will be resulted (i.e., x y z: 3 4 7, 3 4 8, 5 6 7, and 5 6 8).

5.4. Specify The Criteria And Apply Appropriate Aggregate Operations

Based on the previous steps, all fields of information that are identified are specified in the query. However, a screening of the information based on certain values, a specific type of summarization, and a specific sorting method on certain fields may be required. For example, in our A/R aging report case, a screening and/or rearrangement of the information may be required on the invoice date (say, the end of last month) and only for non-zero balance invoices. It may be desirable to have an aggregation based on invoice (rather than invoice items) and customer, and to have the information sorted in a particular order (e.g., a descending sort on the days overdue). In the screening of information, there are a number of arithmetic comparison relations (i.e., =, < >, <=, >=) and logical operators (i.e., and, or, not, like, between) that can be used. There are also several aggregate operators that can be used, such as Group By, Sum, Max, Min, Where, etc. Group By is particularly useful when the user wants to combine like items that are not quantities, where the Sum function would be appropriate. The information can be sorted in ascending order, descending order, or not sorted at all.

5.5. Specify The Output Formats And Layout

The final step involved in constructing a query is to know HOW information should be displayed on the screen and/or on the printed reports. Depending on the needs of the information customers, different formats of the reports and/or of the fields may be required. For example, the layout of the A/R aging report may vary from company to company. Formatting is also an important consideration for any new fields that have been created in the query. Decimal, percentage, currency are just a few examples of the many formats available to present the query output.

6. Exercises And Instructional Ideas

The exercises provide students with the opportunity to develop more complex queries. The instructions provide general directions, but do not lead the students through the queries in a “click by click” fashion. After providing two sample queries, students are required to think through the strategy required to obtain the correct solution to the remaining nine queries. Throughout the exercises, there are helpful hints to assist students with the more difficult concepts and required thought questions to make sure they comprehend what they are doing and what their results actually mean.

Although the concepts covered through the exercises are independent of the database used, we elected to illustrate the use of these queries in Microsoft *Access* 2000. We chose *Access* because of its widespread use and availability and because it is a more manageable database which is suited to the target audience of these exercises. The assignment (in *Word*) and the Database (in *Access*) can be obtained directly from the authors or from the web site <http://www.uwm.edu/~ceil/systems/paper.html>. *Word* and *Access* solutions sets are available to professors through email. (Note: The assignment is included as pages 25-29 of this paper for the convenience of the reviewer).

We assign the queries after students have already completed exercises requiring them to work with *Access* tables, forms, macros and simple queries. These earlier exercises have provided click-by-click instructions and by the time the students have completed them they are prepared for the less detailed instructions of the current assignment. We provide the answers to Queries #1 and #2 as examples.

6.1. Applying The Guidelines

The guidelines mentioned in the paper can help students create the assigned queries and construct them correctly. We will use Query #1 as an example of how to apply the guidelines. In that query, we ask students to identify the target market for the product “blender” in Query #1 by listing age, sex, occupation, and quantity sold

and sorting on quantity sold in descending order. According to the Step 1 of the guidelines, the student should understand the requested information in this question and identify the required data fields, such as Age, Gender, and Occupation of the customers, product of “blender”, and total quantity sold per customers’ characteristics specified. Based on Step 2 of the guidelines, the student should be able to identify the source tables for each one of the data fields (e.g., tbl_Customer contains Age, Gender, and Occupation, tbl_Order/Item Quantity, and tbl_Item Description) from the table structures (i.e., the REAL diagram).

Because no relationship exists between tbl_Customer and tbl_Order/Item (or tbl_Item), we would obtain a Cartesian product in the query results involving these tables according to the Step 3 of the guidelines. To avoid this Cartesian product, we need to include tbl_Order in the query to provide the linkage between tbl_Customer and tbl_Order/Item even though we don’t need to obtain any information from this table. There are One-to-Many relationships between tbl_Customer and tbl_Order, between tbl_Order and tbl_Order/Item, and between tbl_Item and tbl_Order/Item. Since tbl_Order/Item serves as a basis of the information requested (i.e., total quantity sold for a particular item per a specific sets of customer characteristics), we should perform natural joins between these tables.

We need to limit the values in the Description to “blender” as requested and aggregate Age, Sex, and Occupation using Group By and Quantity using Sum, according to the Step 4 of the guidelines. Finally, we have to check the formats and layouts of the output according to the Step 5 of the guidelines.

6.2. Other Teaching Considerations

Queries #3-#5 allow students to work on identifying appropriate information and sorting correctly. Most students will rerun Query #5 several times before accepting the null result. We encourage students to use expression builder in Queries #6 & #7. Their most common mistake in those queries is to improperly set up the formulas. Queries #8-11 are the more difficult queries and it may be desirable to provide students with partial check figures. Query #8 is the most difficult query. Students need to grapple with the profitability per hour of airtime since the airtime is in minutes. They will need to create an intermediate query to get the correct result, although they can go at the intermediate query two different ways and still come up with the correct query at the end. They also work on formatting output results in this query. Query #9 is a perfect example for students to see what can go wrong with the query results when inappropriate tables are included in the query. Students are specifically instructed to add an unnecessary table and examine the query results. Queries #10 and #11 provide additional examples of the need to create intermediate queries to obtain the final correct result. They also provide more experience with selection criteria (including date criteria options) and aggregate functions.

7. Conclusion

The role of technology in accounting education has increased tremendously over recent years. Emphasis on the understanding of database technology is one aspect of that increased attention. Given the pervasive use of database in software and related technologies, it is only natural that coverage of the topic will continue to increase. The exercises presented in this paper allow students to experience one fundamentally important application of database technology—the ability to construct more complex queries. This skill should be useful to students in their changing role of accountants as information experts.

Endnotes

- i Denna, et al. (1994) initiated the REAL model by adding a Location element to the REA model.
- ii Some information may be retrieved without proper relationships being established between entities, but special cares need to be taken. Database specialists who have advanced knowledge about the database and the query language usually perform these tasks.
- iii There are Base and Result tables. The Base tables are tables whose data are actually stored in the database. The Result tables are tables whose data are derived by a query (or queries).
- iv It will be rare if they have a relationship other than many-to-one.
- v It is often used when the basis of information is from the event types of table given that event types of table have many-to-one relationships with the other types of entities, such as object, place, person, and concept.

References

1. Borthick, A. F., D.R. Jones, and R. Kim. 2001. Developing Database Query Proficiency: Assuring Compliance for Responses to Web Site Referrals. *Journal of Information Systems* (Spring): 35-56.
2. Denna, E. L., J. Jaspersen, K. Fong, and D. Middleman. 1994. Modeling conversion process events. *Journal of Information Systems* (Spring): 43-54.
3. Geerts, G. L. and B. A. Waddington. 2000. The Belgian Chocolate Company: A Dynamic Data Modeling and Database Design Case for the Accounting Information Systems Class. *Journal of Information Systems* (Spring): 53-74.
4. Gelinas, U. J., S.G. Sutton, and A.E. Oram. 1999. *Accounting Information Systems*. Fourth Edition, Cincinnati, Ohio: Southwestern Publishing.
5. Hall, J. A. 2001. *Accounting Information Systems*. Third Edition, Cincinnati, OH: Southwestern Publishing.
6. McCarthy, W.E. 1982. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review* (July): 554-578.
7. McFadden, F. R., J. A. Hoffer, and M. B. Prescott. 1999. *Modern Database Management*. Reading, MA: Addison-Wesley.
8. Perry, J. T. and G. P. Schneider. 2000. *Building accounting systems with Access 2000*. Cincinnati, OH: Southwestern Publishing.
9. Pillsbury, C.M. and T.J. Wang. 2001. Supplemental Materials for Database Management System Knowledge and Skills in the Accounting Information Systems Course. *The Review of Business Information Systems* (Winter): 1-6.
10. Romney, M. B. and P. J. Steinbart. 2000. *Accounting information systems*. Upper Saddle River, NJ: Prentice Hall.

Acknowledgements

The authors greatly appreciate the assistance of Owen Cherrington with this project.

Appendix
MS Access Complex Query Assignment

I. Objective

The objective of this assignment is to help you learn to design more advance queries thus increasing your skill in obtaining useful information from a database. In the assignment you will be developing more sophisticated queries to aid management in making complex decisions regarding Waren Unlimited's operations. The database you need can be downloaded from the web at <http://www.uwm.edu/~ceil/systems/paper.html>

II. Company Background

Almost a year ago Tom Waren, owner of Waren Distributing, was approached about an idea for a new business venture by his twin sister Jill. Jill had been considering starting a VIDEO SHOPPING NETWORK show to distribute various products to retail customers. Jill wanted Tom to sell his products to her new business Waren Unlimited at cost and in exchange Tom would receive a 20% share of Waren Unlimited profits after one year. Jill started operations in July without Tom's participation, but he decided to join with Jill in September. Several cable television stations have broadcast the program, which demonstrates name brand products and flashes a 1-800 number that viewers can dial to order items. Waren Unlimited's product pricing is very competitive as inventories are purchased in very large quantities directly from Waren Distributing and a number of other manufacturers/distributors. Also, overhead is minimal in comparison to retail outlets.

This is a critical time in Waren Unlimited's growth. National cable network time has recently been purchased and decisions need to be made on how to best use the expensive national air-time. You have been asked to develop some relational database tables Waren Unlimited can use to collect data on operating activities. The data will be used to formulate future business plans.

Note: You will need a clean 3.5 inch high density (HD) floppy diskette.

III. Partial Real Diagram

Below you will find selected entities and relationship for Waren Unlimited. A complete database would consist of customer payment records, inventory purchases, etc. and would involve complete interactions between all aspects of the database. However, these few tables shown below will provide a representative exposure to relational databases and provide you with the understanding of how to create and use tables.

(*Many-to-Many relationships are not allowed by Access and require a relationship (or bridge entity) table which results in the **Order-Item Table**. The table titled "Order-Item" is necessary because inventory items have a many-to-many relationship with purchase orders. This means that any particular inventory item can be purchased by several customers while any one customer may purchase several different items. Therefore, the individual items included in each order are identified in the table "Order-Item". Access turns the many-to-many relationship between the Item and Order tables into a One-to-Many relationship between Item and Order-Item and One-to-Many between Order and Order-Item tables.)

The attributes (or fields) required in the tables are shown as follows:

Customer = (Customer#, Name, Address, City, State, Zip, Age, Sex, Occupation)
 Item = (Stock#, [Supplier#], Description, Unit Cost, Sales Price, QOH, Air Time)
 Order = (PO#, [Customer#], [SSN], Date)
 Order-Item = ([PO#], [Stock#], Quantity)
 Salesperson = (SSN, Name, Commission Rate)
 Supplier = (Supplier#, Name, Address, City, State, Zip)

The underline indicates the table's **primary key** and the brackets, [], designate a **foreign key**.

The primary key is the unique identifier for each item in the table. The foreign keys are derived from the relationships between the entities. The rule followed is the **primary key of the "one" is placed in the table of the "many" as the foreign key**.

IV. General Directions: Complex Queries

Hints for Developing Complex Queries

- Refer to the data model to see which tables to use in constructing your queries. Select **Queries/New/Design View** and **OK**. Identify tables for the query until all needed tables are shown, with their relationships, and close the Show Tables window. Be sure you don't include unnecessary tables as this can adversely affect your query!!!
- When using two queries and the second query uses the first query as the reference, in the "Show Tables" window click the Queries tab to display the queries instead of tables.

V. Assignment

Please see the hints for developing complex queries above BEFORE starting the assignment. ***DON'T FORGET TO CONSTANTLY SAVE YOUR DATABASE SO YOU DON'T EXPERIENCE THE FRUSTRATION OF LOSING YOUR WORK.***

1. Who is the target market for the blender? Base your recommendation on age, sex, occupation, and quantity sold. Sort in descending order on quantity sold. (A solution to this query is in your Waren subdirectory as **qry_Complex1**). **Answer this question on the Word document you will create under the heading Complex Query 1; you do not need to include the query, only the answer to the question.**
2. Which salesperson should receive the Outstanding Salesperson Award for the first few months of operations? Base the award on total gross margin for sales made by each salesperson. Your answer should show gross margin by salesperson in descending order. (A solution to this query is in your Waren subdirectory as **qry_Complex2**). Now list each salesperson's name and the total number of items each has sold (**Hint:** Use View/Totals to add a "Total" row and select Sum). Sort in descending order on Total Quantity Sold. Show the commission rate in your query. Examine the salesperson commission structure. Does the structure seem consistent with the data? What explanation can you think of that could explain how the given commission structure could be correct? You do not need to save this query. Given the results of these two queries, please state two conclusions--one regarding the nature of Patty Holmes' sales and the other regarding Chloe Steele's commission percentage.). **Answer these questions on the Word document you will create under the heading Complex Query 2; you do not need to include either query, only the answer to the questions.**
3. What products are most attractive to housewives and, therefore, should be emphasized during afternoon air-time? List occupation, total quantity per item, and product description for items sold to housewives. Sort in descending order on quantity sold. (Name this query **qry_Complex3XX**).
4. Which television salesperson personality should be used to sell item 302? Assume salespeople have equal air time with each product. List the salesperson's name, description of item #302, and the total quantity sold by each salesperson. (Name this query **qry_Complex4XX**).
5. Which television salesperson personality should be used to sell item 101? Assume salespeople have equal air time with each product. List the salesperson's name, description of item #101, and the total quantity sold by each salesperson. (Name this query **qry_Complex5XX**). What conclusion could you make about item 101?
6. How much revenue was earned on Ferrari Cell Phones (stock #1301) on PO # 1007? List the PO# and product description in addition to the revenue earned. (Name the query **qry_Complex6XX**).
7. How much total gross profit was earned on PO #1007? List the PO# and the total gross profit earned. (Name the query **qry_Complex7XX**).
8. Management wants a report that shows product profitability per hour of air-time. Note: The "air time" in the ITEM table is in minutes. List product stock number, description, and gross margin per hour of air-

- time. (**Hint:** This will probably take 2 queries, one to identify total quantity of each item sold along with gross margin, and one to query that query to figure total gross margin per hour of air time. Be sure to **format the new column** and sort in ascending order on gross margin per hour of air-time). (Label these queries **qry_Complex8AXX** and **qry_Complex8BXX**). Be sure to show the stock#, description and gross margin/per hour of airtime in your query.
9. Calculate the **TOTAL** historical cost based value of the inventory on hand at the end of the period. (**Hint:** You will need to have two queries; one should show the description of each item and its total historical cost, and the second should show only the total inventory value of all items). (Name these queries **qry_Complex9AXX** and **qry_Complex9BXX**). *This is a great query to experiment with to see the adverse effects of including a table when you shouldn't. Once you have safely saved your correct queries, call up 9A and select "Show Table". Add the Order/Item table and rerun your query. Did you notice what happens? There aren't any toasters in your query now. That's because there weren't any sales of toasters. But because you included the table in the query it causes only items that were in both tables to return. Be sure you don't save this incorrect query!!!!*
10. **Tom Waren, owner of Waren Distributing**, has asked Jill to provide him with sales numbers for items of **Waren Distributing's that have been sold through Waren Unlimited** for the **four** month period ending 12/31/05 (Please reread the previous sentence and make sure you remember that Waren Distributing is **just one** of Waren Unlimited's suppliers). Waren would like to receive a listing of the inventory items with quantities sold sorted in descending order. Include the supplier number, description and stock number of the inventory items. Label this query **qry_Complex10XX**. (**Hint:** You may want to construct this query without considering the date first and then add the date column. You will need to change "group by" in the date column to "where" and specify your criteria.)
11. Tom would also like to see the total gross profit earned on sales of the items you identified in Query #10. This will probably take you two queries to accomplish. (Label them **qry_Complex11AXX** and **BXX**).

VI. Turn-In

- Turn in your disk with Access Assignment on it.
 - Turn in for grading a printed copy of the query and your answers to the questions.
 - The easiest and least wasteful way to print out query results is to put them all in one MS Word document. While the Waren Unlimited database is open, open a session of Word. Run your query in Access. With the query result in datasheet view (looks like a spreadsheet), find the button that is located in the top-left corner of the query window. The button is directly to the left of the first column heading. Click it and the query will become highlighted. **Hit Ctrl-C or Edit/Copy** to copy the query results to the clipboard. Toggle over to Microsoft Word. Place your cursor where you want the results to appear. **Hit Ctrl-V or Edit/Paste** and Word will paste your results as a perfect table. **Using Word, type the name of the query above the previously pasted query. Points will be taken off if we have to search through your assignment to find the answers. Don't forget to include your sentence responses when appropriate.**
-