Frederick McGovern

ICT 4405 Database Design & Programming

Email – [Frederick\_McGovern@msn.com](mailto:Frederick_McGovern@msn.com)

Phone – 508-846-2857

Course Project

Table of Contents

Activity #1

Company Abstract ---------------------------------------------------------------------------- 3

Reporting Requirements --------------------------------------------------------------------- 4

Activity #2

Company Data Set --------------------------------------------------------------------------- 5

1NF and Dependency Diagram ------------------------------------------------------------ 6

2NF and Dependency Diagram ----------------------------------------------------------- 13

3NF and Dependency Diagram ----------------------------------------------------------- 14

Entity Relational Diagram ----------------------------------------------------------------- 18

Activity #3

Data Dictionary for Logical Model ------------------------------------------------------- 20

Activity #4

SQL Scripts to Implement Physical Design --------------------------------------------- 20

Activity #5

SQL Scripts to Populate Relational Design --------------------------------------------- 33

Activity #6

SQL Scripts to Meet Reporting Requirements ------------------------------------------ 60

Activity #7

User Interface Requirements --------------------------------------------------------------- 71

Activity #1

Company Abstract:

Chestnut Books and Collectibles was established by Rudolph and Norma Koeppel in 1998 and is located in the quaint seaside town of Haven, Maine. Initially known as Chestnut Books, the company was operated out of the Koeppel’s home and offered a small but eclectic selection of used books. In 2009, the company also began offering homemade crafts designed by Mrs. Koeppel and various members of her knitting club. Due to the high quality and demand for these crafts, this has dramatically increased the visibility of the company resulting in the decision to open an establishment in the tourist friendly waterfront district.

Their new location at 66 King Street allows for the expansion of not only carrying used books but to the ability to carry all current bestsellers and audiobooks. A children’s book department was also established on the buildings second floor as well as a section dedicated to local horror writer Audrey Parker. Additional merchandise now available for sale include both music CD’s and various movies in DVD and Blu-ray format. As usual, Mrs. Koeppel continues to provide handmade crafts which continued to be in high demand by both local and tourists alike.

Due to the large increase in inventory, the company now needs to be able to accurately and quickly respond to customer’s demands for products. Prior inventory control was spreadsheet based, which at the time was appropriate and manageable due to the company only having a small inventory and the fact that the company only had two employees with those being the owners. The expansion of the company has resulted in the hiring of two full time employees in addition to the owners and an inventory of products numbering in the thousands. A database solution is necessary in order to properly track inventory and to provide price control on all merchandise. Additionally, through a database solution the company employees will be able to quickly meet various customer service tasks such as product availability, product location and informational inquiries. The company would like to be able to track customer purchases, thus being able to provide recommendations for future purchases, and would also like to establish a newsletter for distribution by customer email. The development of a reader’s club discount for a nominal yearly fee has also been proposed with would give the customer an additional discount of 10% on all book purchase. All of these initiatives would require a transactional relational database design.

Reporting Requirements:

There are numerous reporting requirements for this application. They include but are not limited to the following:

* Basic product details including the quantity on hand, the price both regular and if on sale the sales price, the type of product and the department the product is located in must all be available by product id.
* For a given book, a query must be developed that will return all detailed information including the author, publisher, publication date, title of book, ISBN number if available, the type and format of the book as well as any categories that the book is associated with and its price.
* For music CD’s a query must return details such as title, release date, label, artist, category and price for a given music selection.
* For DVD’s and Blu-ray’s, a report must be developed that will return to the user details such as title, studio, release date, director and format and price.
* A report listing all handmade crafts available for sale which includes a description of the item, the category assigned to the item and the price of the item.
* For any given customer, a report needs to be created that will generate basic customer information such as name, address, phone and email as well as whether or not the customer has signed up for the monthly newsletter and whether or not they are part of the reader’s club discount program.
* For any given customer signed up for the reader’s discount club, a report that details the customers purchases either on a specific date or over a given date range should be created. This report would provide the customer number, invoice number, the product id and the price paid for that product.
* Since the company likes to emphasize the works of local horror writer Audrey Parker, a report should be created that provide the number of copies sold and total sales by title for a given month.
* Since the book club provides additional revenue it is important to remind customers when their membership is about to expire so that they can renew it. A report must therefore be generated to list all customers with expiring memberships within the next 90 days.

Activity #2

Company Data Set:

The basis for any successful business is its customers and as such, Chestnut Books would like to collect the following data items specific to customers:

Customers: Name, address, birth date, phone, e-mail, and an indicator whether the customer subscribes to the company newsletter. A customer can have more than one phone or email so the type of phone (home, work, fax, mobile etc.) and email (home, work etc.) should be listed as well as which one should be used for primary contact. An indicator of whether or not the customer is part of the Reader’s discount club, when the yearly fee was paid when their existing membership will be expiring. The company also wants to track the initial date the information was provided. Customers aren’t required to provide any information to make purchases but every effort is made to create a customer list to better provide customer service.

Of course you can’t run a successful business without returning a profit. To do so, you must have some inventory to sell and have customers who make purchases. The company would like to capture a list of all purchases made by all customers. Customers who are in fact members of the Reader’s discount club will receive periodic recommendations based on items they have previously purchased. This information should contain the following data elements:

Purchases: The product that was sold, the quantity of the product sold, the unit price per item, the customer who made the purchase, the date and time the purchase was made. If a customer has not provided any contact information, their purchases will be assigned to a default guest customer.

With regards to inventory, there are general product data points that need to be captured as well as specialized data points based on the type of product being sold. The company wishes to collect the following data items with regards to products.

General Product: Product name, the quantity on hand in inventory, the type of product, the department the product is located in, the regular price, and the current sales price if any.

Since the company sells only books, music, movies and handcraft items, there are additional details that need to be captured for each type of product. These would include the following:

Books: The author, publisher, publisher address, publication date, title of book, ISBN number if available, the type (non fiction vs fiction) of book, the format of the book (hardcover, paperback, audio) as well as any categories that the book is associated with. A book will only have one ISBN number, if it has one at all. All new books will have an ISBN but older used books may not. A book may be associated to multiple categories.

Music: The title, artist, release date, label and category for a given music selection. Like books, music can be assigned to multiple categories.

Movies: The title, studio, release date, director and format (DVD vs Blu-ray) of the movie.

Handmade Crafts: The product description, its creator and the creation date. Each craft will have a single creator.

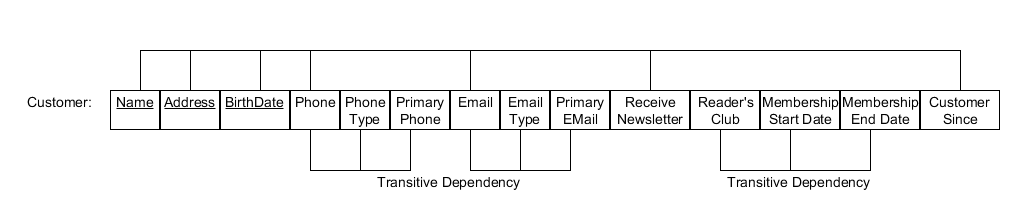
The structure of the database should be set up so as to be able to sell additional types of items by only needing to add any additional table corresponding to the type of product being sold.

1NF and Dependency Diagram:

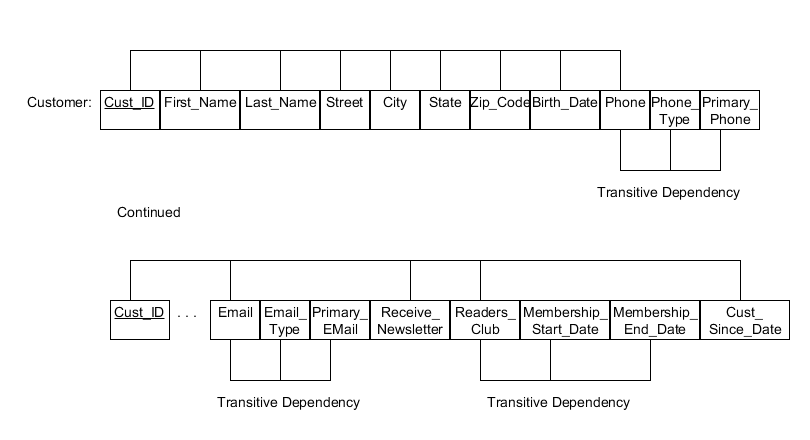
Please refer to the attached excel file Chestnut Books Raw data as needed for sample data used to make decisions with regards to normalization.

A 1NF table must contain an atomic value for each column. Thus for a table to be in 1NF, we must first eliminate repeating groups. To eliminate the repeating groups, one must simply replace all null values for a given record with a data value.

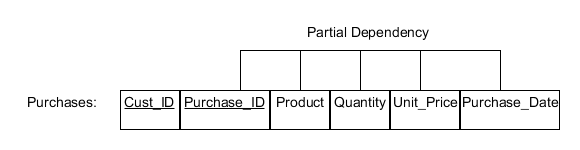
Looking at the customer worksheet in the excel file we can see that we have two repeating groups those being phone and email. These two fields are in fact multi valued attributes. The dependency diagram based on the required data points would be:



The attached excel worksheet Customer 1NF, shows the replacing of all null data points with an appropriate data value. Since we have two multi valued data points, the total number of records for a given customer is equal to the product of the number of phones listed and the number of emails listed. With the repeating groups eliminated, the final task to make put the customer table in 1NF would be to identify the primary key. The name attribute is not sufficient alone to ensure uniqueness since many people can have the same name. We can reduce the chance of a match by creating a composite key containing the attributes name, address and birth date but there are two problems with this approach. First, the attributes name and address are reducible into other attributes. For example, a name could be divided into a first name and a last name. An address would be comprised as a street, city, state and zip code. If we decided to create a composite key based on name, address and zip code, this in fact would become a composite key across seven attributes. Second, while the odds of having two people with the same name, address and birth date are extremely remote, it is still possible. Based on these two issues, the best way to ensure uniqueness in the customer entity would be to introduce a surrogate key called Cust\_ID. Based on this assignment and the decomposition of the name and address attributes, the final 1NF dependency diagram for the Customer table would be:

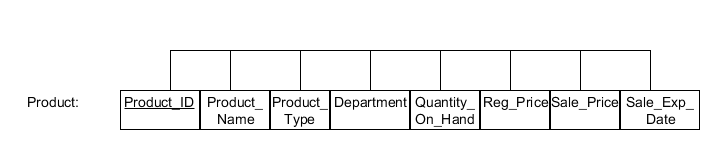


When looking at purchases, there is a relationship between the customer and the items being purchases. An example of the purchase made by a customer can be found on the worksheet Purchases. Since each customer can purchase multiple products, the attribute product is a repeating group. By replacing the nulls in the data set one can begin to get the table into 1NF. The primary key of the table needs to be composed of the Cust\_ID attribute and some combination of the product, quantity and date purchased to ensure uniqueness. Due to the key being a composite key over many attributes, it would be better to create a surrogate id called Purchase\_ID. This id along with the Cust\_ID will ensure uniqueness. The dimensional diagram for the purchase table in 1NF would therefore be:

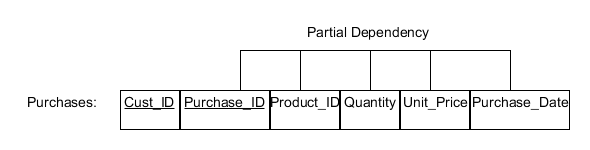


A partial dependency exists because the product, quantity, unit price and purchase date only depend on the Purchase\_ID. This issue will be handled when performing 2NF. There is a relationship between Product and Unit Price but not within the context of a purchase. While a product will have a given retail price and sales price associated with it, these prices can change and only the current prices are retained. Additionally, members of the reading club receive a 10% discount on books. Therefore, in this case the price needs to be captured based on the time of purchase as we would need to determine if the customer is an active club member.

We now need to visit the product entity at this time. A product will have general charateristics such as the product name, the type of product and the department the product is located in. For our purposes, a product can only be of one type and be located in one department. Additionally, the regular price of the item should be captured as well as any sales price. Prices that are stored are current in that we will not retain historical prices. Once a sale has ended, the sales price of the product will be deleted. As such, to provide additional reporting mechanisms we will additionally add a date representing the date that a sale price will revert to a regular price. We also need to capture the number of that item that is in stock for inventory control. Referring to the excel document, one can see an example of products on the Products worksheet. In this case there are no repeating groups so we just need to determine the primary key to ensure 1NF. As can be seen by the data, the Product name does not ensure uniqueness. We have two products that have the same name but have different type. A composite key of product and type would seem to be the answer, however, it is possible to have two products with the same name and be of the same type but not actually be the same product. For example, a two books could have the same name but be by different authors. Therefore, the utilization of a surrogate key to ensure uniquness is required. The following dependency diagram shows the product table in 1NF:

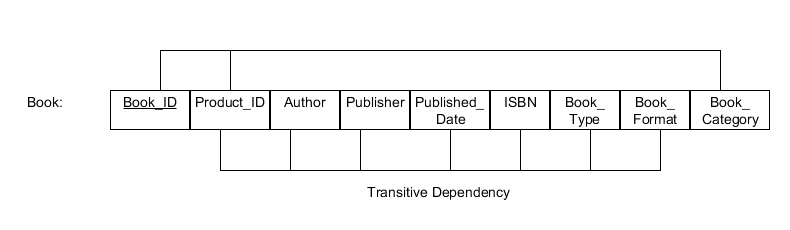


The assignment of the Product\_ID necessitates a change to the Purchases dependency diagram to replace the attribute Product with Product\_ID.



Now that the general product characteristics have been established we can look at the specific product type entities. Per the business requirements we have four product types those being books, music, movies and crafts. We will treat these entities as specializations of the Product entity. Starting with books, we need to capture the author, publisher, publisher address, publication date, title of book, ISBN number, type of book, format of the book and categories that the book is associated with. We are told that a book can be associated with multiple categories so we can establish that that attribute is multi varied. Additionally, a book can in fact have multiple authors so we will consider that a multi varies attribute as well. A book can have multiple publishers and multiple publishing dates. For example, the classic book “Alice in Wonderland” has been published by many publishers over the years. Since we are providing information for the specific product, we can therefore assign a different Product\_ID for each book based on publisher and date. Therefore, for our purposes we will only indicate the publisher for the actual book carried and the latest publication date found inside the book.

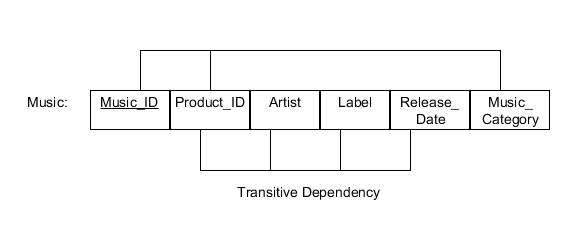
Looking at the excel worksheet Books, you can see that for a given Product there could be multiple formats. Initially, this would lead you to believe that format is also a multi variable attribute. In actuality, there is a different ISBN number for each format, however, while the ISBN can be used to determine the publisher it can’t be used to determine format so the two entities are not related. Due to this, each format of the book will actually be identified by a different Product\_ID. This is important since the Quantity\_On\_Hand applies not only to the product but to the format of the product. Based on these conditions the dependency diagram for the Books table would be:



The 1NF data can be found on the excel worksheet Books 1NF.

While the ISBN number would seem to be an ideal primary key since it is unique, the business requirements state that an ISBN number can be null. This is especially true on older books. Since a primary key cannot contain a null value, we will instead create as surrogate key, Book\_ID, as the primary key. Since there will eventually be only one row for every Product\_ID we could have just included this on the general Product table. This would be a bad thing to do, however, as this data only applies to books. Having all these fields in the Product table would result in a lot of wasted space. Instead, we will create the Book table, with the Product\_ID as a foreign key to the Product table, which would then have a relationship such that every Product\_ID in the Book table must exist only once in the Product table. The business requirements did not require any additional information about the publisher such as address so for this project we will consider that attribute as out of scope.

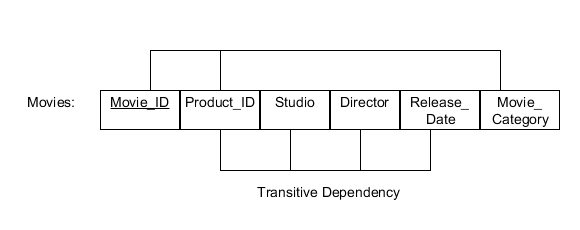
Following the same logic that we used in designing the Book table, we can then create the Music table. The business requirements state that title, artist, release date, label and category for a given music selection are required. It also states that category may be considered a multi value attribute. For our purposes, even though it’s possible to have multiple artists on a CD, when that occurs we will either list the primary artist or if a compilation of artists then the artist will be identified as “various”. These self-imposed constraints thus ensure that the artist attribute will be a single valued attribute. The excel worksheet Music contains the raw data and through it we can verify that the only multi-valued attribute that is acceptable is the category field. The worksheet Music 1NF contains the data with the null repopulated. The resulting dimensional diagram appears as follows;



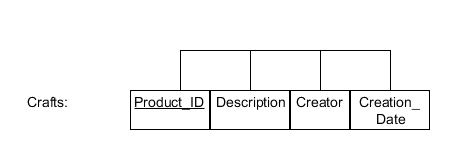
The title of the CD can be determined between the relationships between the Music table and the Product table via the foreign key Product\_ID. Like the Books table, for every Music\_ID in the Music table it will only appear once in the Product table as Product\_ID.

For movies, the business wished to capture data with regards to the title of the movie, the studio that produced the movie, the release date, director and format of the movie. There are two available formats, those being DVD and Blu-ray. This attribute, however, is related to the Product\_ID in the same way format is related in the Books table. In other words, for every different format for the same product there will be a uniquely identifiable Product\_ID. So while the field is technically multi valued its association forces a unique value for a given Product\_ID.

Like the music table, we can apply some constraints to the data. For example, there were no parameters with regards to capturing the actor’s in the movie so we will consider that attribute out of scope. We will assume that only one studio can release a film, therefore, each Product\_ID with a type of movie will have a singular studio data point. The director attribute would seem to be singular but films have had multiple directors in the past. I watched the 1967 version of Casino Royal last week and the movie listed five contributing directors. Though not common, we will therefore consider the director attribute as multi-valued as indicated on the attached excel worksheet Movies. For DVD’s of television episodes that typically have a different director for each week’s episode, either the primary director or an entry of “Various” will be entered for the director attribute. Finally, as indicated by the business requirements, the category attribute is also multi valued. The excel worksheet Movies contains the raw data while the worksheet Movies 1NF, shows the data will the nulls removed. Again, since we are using specialization, the foreign key for the Movies table, Product\_ID, will be the same as the primary key of the Products table creating a 1 to (0,1) relationship. As before, we will create a surrogate primary key on the Movie table as Movie\_ID. The dependency diagram for the Movie table is as follows:



We finally come to the last entity of Crafts. The only requirement for this entity is the description of the product as well as the creator and creation date. As define by the business requirements all attributes are single valued. The only way to guarantee uniqueness is to utilize all fields as the primary key as two products could have the same description and creator but they would have different creation dates. If they were the same creation date then then would be considered the same product and this would be shown in inventory as such. Instead of using a composite key that includes all fields in the table, we will instead utilize the Product\_ID to establish uniqueness. In this case the primary key, is also the foreign key for the Product table. As can been seen by the excel worksheet Craft, the table is already in 1NF. The dimensional diagram for the craft table is as follows:

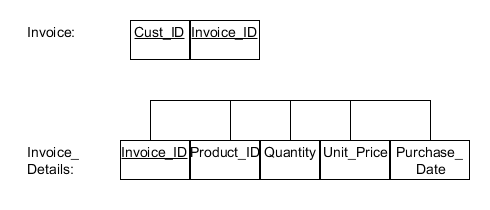


With all repeating groups eliminated and all primary keys assigned all table are now in 1NF and we can proceed with getting our table into 2NF.

2NF and Dependency Diagram:

For a table to be in 2NF, it must first be in 1NF and all partial dependencies must be eliminated. A partial dependency is where any attribute depends on only a portion of the primary key. As I have soon, all of our tables are currently in 1NF so that condition has been met. Reviewing our existing dependency diagrams, we can see that there is only one occasion where we have a partial dependency and that is on the Purchases table. On that table, the primary key is a composite key made up of the attributes Cust\_ID and Purchases\_ID. The attributes in this table, Product, Quantity, Unit\_Price and Purchase\_Dt depend solely on the Purchases\_ID. One can better understand this as an example of an invoice. A customer is directly related to an invoice and an invoice has specific details about the products purchased and the quantity purchased. To remove a partial dependency, one needs to create a new table and move a copy of the determinant portion of the primary key to this table. Along with this field, you then move all the attributes that were solely dependent on the determinant. So in our case, we would have two tables as a resultant. The first would contain the composite key while the second would contain the determinant and its dependents. Utilizing the concept of Invoice, I renamed my Purchase table as Invoice and renamed the attribute Purchase\_ID to Invoice\_ID. The second table was named Invoice\_Details as it provides the details for any particular Invoice.

The following is the resultant dependency diagrams for those two tables:

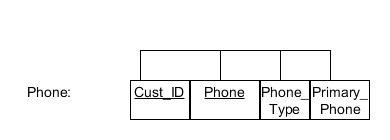


With no other partial dependencies to remove, we can now state that our tables are in 2NF and can proceed to establishing 3NF.

3NF and Dependency Diagram:

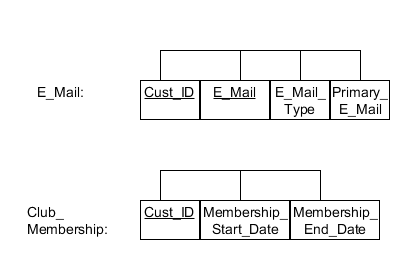
To reach 3NF, a table must be in 2NF and all transitive dependencies are eliminated. To eliminate a transient dependency, one must create a new table which will contain a copy of the determinate attribute as the primary key as well as the attributes dependent that are dependent on the determinant. All fields except the table’s primary key should be removed from the source table. Reviewing our existing 2NF dependency diagrams, we can see that four table have transitive dependencies on them – Customer, Books, Music and Movies. The Customer table has three different transitive dependencies so we will tackle that table first.

The first thing we want to look at is the field Phone, Phone\_Type and Primary\_Phone attributes. The Phone\_Type and Primary\_Phone depend solely on the non-key attribute Phone. Following the directions for normalization, we create a new table called Phone and move the necessary attributes. The end result is the following dependency diagram for the new table:



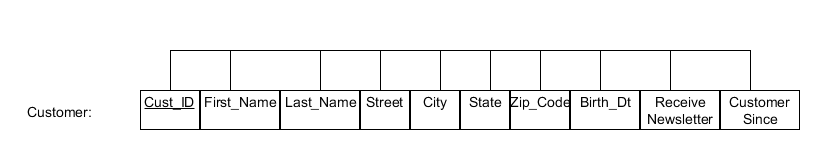
There is a possibility that you could have the same phone number assigned to another customer but that would be rare and the Phone\_type and Primary\_Phone data points may not match up. Therefore, the best approach is to create a composite key of Cust\_ID and Phone instead of creating a Phone\_ID attribute and a sepearte table that stores a single phone number with no duplicated.

Following the same process we eliminate the other two dependencies creating two new tables E\_Mail and Club\_Membership. The resulting dependency diagrams for those two tables are:

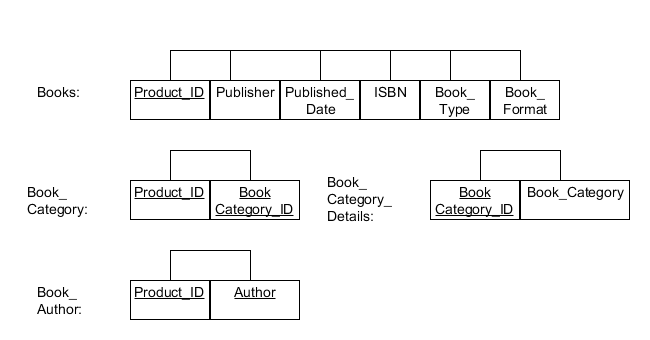


The Membership\_End\_Date field could be interpretted as a calculated field as the raw data shows that the experation date is a year from the Membership\_Start\_Dt. I decided to actually store the data point instead of calculating it for three reasons. First, the inclusion of the data point means that I don’t need to calculate it on the report, thus improving report performance. Second, unlike age which needs to be calculated every day, the data point will only change when the Membership\_Start\_Dt field changes so there isn’t a need to frequently recalculate the value. Finally, the inclusion of the field allows flexibility in changing database requirements. The business may wish to change the program from 12 months to 18 months. This would result in some records having an expiration set at 12 months and some at 18. If the calculation was in the report, the correct data point could still be retrieved but would require a complex CASE statement basing the analysis on the date that the change took place.

With thse fields removed from the Customer table, the result dependency diagram for the Customer table becomes:

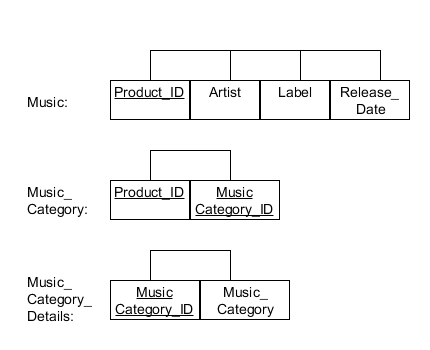


The approach is identitical for the remaining tables so I will not go into them in too much detail. The following dependency tables are produced through the removal of transitive dependencies in the Books table:

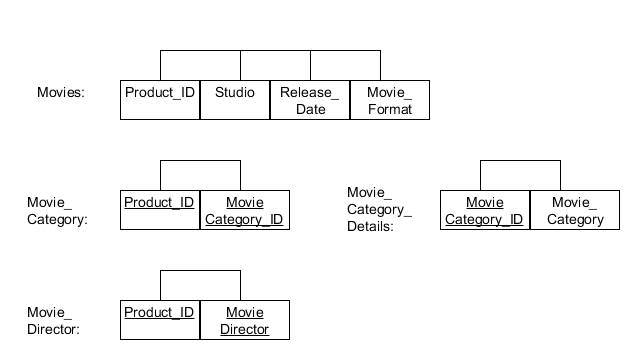


You will notice that we no longer have to utilize the surrogate primary key Book\_ID. With the removal of the transitive dependencies, there became a distinct Product\_ID for each Book\_ID. Utilizing the surrogate key would therefore be redundant and confusing. The result is that the Books tables has the primary key Product\_ID which also acts as a foreign key against the Products table create a 1 to (0,1) relationship. The creation of the Book\_Categories\_Details table is a design choice. I could have just left the Book\_Category attribute in the Book\_Category table, which with the attribute Product\_ID would form a composite primary key. I decided to create a separate Book\_Categories\_Details table so as to eliminate the potential of inconsistent data due to the input of data into a free form text field on the user interface. For example, for a computer book one employee might type “Sequel Server” in the category field when it’s actually SQL Server. By creating a separate table where a unique category is identified, we can apply a constraint on the user interface whereby the user can only choose categories listed in a list box. The list box is populated from the unique values in the Book\_Categories\_Details table. In addition to eliminating data anomalies this approach makes it easy to add new categories by just inserting new records into the Book\_Categories\_Details table.

The following dependency tables are produced through the removal of transitive dependencies in the Music table:



Construction and design choices for the three Music tables are based on the same logic used when I created the three Book tables. The following dependency tables are produced through the removal of transitive dependencies in the Movies table:

****

Once again, the same design characteristics used in developing the three Book tables was utilized here to create three Movie tables. One exception was the creation of the Movie\_Director table. We could have followed the same approach as the Movie\_Category but since there is not as much data duplication, I felt it was acceptable to just list the Director in the Movie\_Director.

Entity Relational Diagram

The next page contains the Entity Relation Diagram for the Chestnut Books and Collectibles database. The following diagram will help with the interpretation of the symbols on this page:





Activity #3

Data Dictionary for Logical Model

The corresponding data dictionary is located in the attached excel file under the Data Dictionary tab.

Activity #4

SQL Scripts to Implement Physical Design

The following script will create and populate all tables in the database Chestnut Books and Collectibles. Additionally, the script will define views on the tables so as to be able to do the required reporting.

/\*Check to see if the table we are about to create exists. If it does then drop the existing table. \*/

DECLARE

c int;

BEGIN

--Check if table BOOK\_AUTHOR exists

select count(\*) into c from user\_tables where table\_name = 'BOOK\_AUTHOR';

if c = 1 then

execute immediate 'DROP TABLE BOOK\_AUTHOR ';

end if;

--Check if table BOOK\_CATEGORY\_DETAILS exists

select count(\*) into c from user\_tables where table\_name = 'BOOK\_CATEGORY\_DETAILS';

if c = 1 then

execute immediate 'DROP TABLE BOOK\_CATEGORY\_DETAILS';

end if;

--Check if table BOOK\_CATEGORY exists

select count(\*) into c from user\_tables where table\_name = 'BOOK\_CATEGORY';

if c = 1 then

execute immediate 'DROP TABLE BOOK\_CATEGORY';

end if;

--Check if table BOOKS exists

select count(\*) into c from user\_tables where table\_name = 'BOOKS';

if c = 1 then

execute immediate 'DROP TABLE BOOKS';

end if;

--Check if table CLUB\_MEMBERSHIP exists

select count(\*) into c from user\_tables where table\_name = 'CLUB\_MEMBERSHIP';

if c = 1 then

execute immediate 'DROP TABLE CLUB\_MEMBERSHIP';

end if;

--Check if table CRAFTS exists

select count(\*) into c from user\_tables where table\_name = 'CRAFTS';

if c = 1 then

execute immediate 'DROP TABLE CRAFTS';

end if;

--Check if table CUSTOMERS exists

select count(\*) into c from user\_tables where table\_name = 'CUSTOMERS';

if c = 1 then

execute immediate 'DROP TABLE CUSTOMERS';

end if;

--Check if table E\_MAIL exists

select count(\*) into c from user\_tables where table\_name = 'E\_MAIL';

if c = 1 then

execute immediate 'DROP TABLE E\_MAIL';

end if;

--Check if table INVOICE exists

select count(\*) into c from user\_tables where table\_name = 'INVOICE';

if c = 1 then

execute immediate 'DROP TABLE INVOICE';

end if;

--Check if table INVOICE\_DETAILS exists

select count(\*) into c from user\_tables where table\_name = 'INVOICE\_DETAILS';

if c = 1 then

execute immediate 'DROP TABLE INVOICE\_DETAILS';

end if;

--Check if table MOVIE\_CATEGORY exists

select count(\*) into c from user\_tables where table\_name = 'MOVIE\_CATEGORY';

if c = 1 then

execute immediate 'DROP TABLE MOVIE\_CATEGORY';

end if;

--Check if table MOVIE\_CATEGORY\_DETAILS exists

select count(\*) into c from user\_tables where table\_name = 'MOVIE\_CATEGORY\_DETAILS';

if c = 1 then

execute immediate 'DROP TABLE MOVIE\_CATEGORY\_DETAILS';

end if;

--Check if table MOVIE\_DIRECTORS exists

select count(\*) into c from user\_tables where table\_name = 'MOVIE\_DIRECTORS';

if c = 1 then

execute immediate 'DROP TABLE MOVIE\_DIRECTORS';

end if;

--Check if table MOVIES exists

select count(\*) into c from user\_tables where table\_name = 'MOVIES';

if c = 1 then

execute immediate 'DROP TABLE MOVIES';

end if;

--Check if table MUSIC exists

select count(\*) into c from user\_tables where table\_name = 'MUSIC';

if c = 1 then

execute immediate 'DROP TABLE MUSIC';

end if;

--Check if table MUSIC\_CATEGORY exists

select count(\*) into c from user\_tables where table\_name = 'MUSIC\_CATEGORY';

if c = 1 then

execute immediate 'DROP TABLE MUSIC\_CATEGORY';

end if;

--Check if table MUSIC\_CATEGORY\_DETAILS exists

select count(\*) into c from user\_tables where table\_name = 'MUSIC\_CATEGORY\_DETAILS';

if c = 1 then

execute immediate 'DROP TABLE MUSIC\_CATEGORY\_DETAILS ';

end if;

--Check if table PHONE exists

select count(\*) into c from user\_tables where table\_name = 'PHONE';

if c = 1 then

execute immediate 'DROP TABLE PHONE';

end if;

--Check if table PRODUCT exists

select count(\*) into c from user\_tables where table\_name = 'PRODUCT';

if c = 1 then

execute immediate 'DROP TABLE PRODUCT';

end if;

--Check if table PRODUCT\_DEPARTMENT exists

select count(\*) into c from user\_tables where table\_name = 'PRODUCT\_DEPARTMENT';

if c = 1 then

execute immediate 'DROP TABLE PRODUCT\_DEPARTMENT';

end if;

END;

/\*Now that we are sure that the table does not exist, we can execute a DDL script to create each table that we need in our database. \*/

CREATE TABLE BOOK\_AUTHOR

(

PRODUCT\_ID NUMBER NOT NULL,

AUTHOR VARCHAR2 (60 BYTE) NOT NULL,

CONSTRAINT BOOK\_AUTHOR\_PK PRIMARY KEY

(PRODUCT\_ID, AUTHOR)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE BOOK\_CATEGORY

(

PRODUCT\_ID NUMBER NOT NULL,

BOOK\_CATEGORY\_ID NUMBER NOT NULL,

CONSTRAINT BOOK\_CATEGPRY\_PK PRIMARY KEY

(PRODUCT\_ID, BOOK\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE BOOK\_CATEGORY\_DETAILS

(

BOOK\_CATEGORY\_ID NUMBER NOT NULL,

BOOK\_CATEGORY VARCHAR2 (30 BYTE) NOT NULL,

CONSTRAINT BOOK\_CATEGORY\_DETAILS\_PK PRIMARY KEY

(BOOK\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE BOOKS

(

PRODUCT\_ID VARCHAR2 (20 BYTE) NOT NULL,

PUBLISHER VARCHAR2 (100 BYTE) NOT NULL,

PUBLISHED\_DT DATE NOT NULL,

ISBN VARCHAR2 (15 BYTE) NOT NULL,

BOOK\_TYPE VARCHAR2 (12 BYTE) NOT NULL,

BOOK\_FORMAT VARCHAR2 (9 BYTE) NOT NULL,

CONSTRAINT BOOKS\_PK PRIMARY KEY

(PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE CLUB\_MEMBERSHIP

(

CUST\_ID NUMBER NOT NULL,

MEMBERSHIP\_START\_DT DATE NOT NULL,

MEMBERSHIP\_END\_DT DATE NOT NULL,

CONSTRAINT CLUB\_MEMBERSHIP\_PK PRIMARY KEY

(CUST\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE CRAFTS

(

PRODUCT\_ID NUMBER NOT NULL,

DESCRIPTION VARCHAR2 (250 BYTE) NOT NULL,

CREATOR VARCHAR2 (60 BYTE) NOT NULL,

CREATION\_DT DATE NOT NULL,

CONSTRAINT CRAFTS\_PK PRIMARY KEY

(PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE CUSTOMERS

(

CUST\_ID NUMBER NOT NULL,

FIRST\_NAME VARCHAR2 (25 BYTE) NOT NULL,

LAST\_NAME VARCHAR2 (25 BYTE) NOT NULL,

STREET VARCHAR2 (30 BYTE) NOT NULL,

CITY VARCHAR2 (30 BYTE) NOT NULL,

STATE VARCHAR2 (2 BYTE) NOT NULL,

ZIP\_CODE VARCHAR2 (5 BYTE) NOT NULL,

BIRTH\_DT DATE,

RECEIVE\_NEWSLETTER VARCHAR2 (20 BYTE) NOT NULL,

CUSTOMER\_SINCE DATE NOT NULL,

CONSTRAINT CUSTOMERS\_PK PRIMARY KEY

(CUST\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE E\_MAIL

(

E\_MAIL VARCHAR2 (100 BYTE) NOT NULL,

CUST\_ID NUMBER NOT NULL,

E\_MAIL\_TYPE VARCHAR2 (10 BYTE) NOT NULL,

PRIMARY\_E\_MAIL VARCHAR2 (1 BYTE) NOT NULL,

CONSTRAINT E\_MAIL\_PK PRIMARY KEY

(E\_MAIL, CUST\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE INVOICE

(

CUST\_ID NUMBER NOT NULL,

INVOICE\_ID VARCHAR2 (20 BYTE) NOT NULL,

CONSTRAINT INVOICE\_PK PRIMARY KEY

(CUST\_ID, INVOICE\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE INVOICE\_DETAILS

(

INVOICE\_ID NUMBER NOT NULL,

PRODUCT\_ID NUMBER NOT NULL,

QUANTITY NUMBER NOT NULL,

UNIT\_PRICE NUMBER NOT NULL,

PURCHASE\_DT DATE NOT NULL,

CONSTRAINT INVOICE\_DETAILS\_PK PRIMARY KEY

(INVOICE\_ID, PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MOVIE\_CATEGORY

(

PRODUCT\_ID NUMBER NOT NULL,

MOVIE\_CATEGORY\_ID NUMBER NOT NULL,

CONSTRAINT MOVIE\_CATEHORY\_PK PRIMARY KEY

(PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MOVIE\_CATEGORY\_DETAILS

(

MOVIE\_CATEGORY\_ID NUMBER NOT NULL,

MOVIE\_CATEGORY VARCHAR2 (30 BYTE) NOT NULL,

CONSTRAINT MOVIE\_CATEGORY\_DETAILS\_PK PRIMARY KEY

(MOVIE\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT );

CREATE TABLE MOVIE\_DIRECTORS

(

PRODUCT\_ID NUMBER NOT NULL,

DIRECTOR VARCHAR2 (60 BYTE) NOT NULL,

CONSTRAINT MOVIE\_DIRECTORS\_PK PRIMARY KEY

(PRODUCT\_ID, DIRECTOR)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MOVIES

(

PRODUCT\_ID NUMBER NOT NULL,

STUDIO VARCHAR2 (60 BYTE) NOT NULL,

RELEASE\_DT DATE NOT NULL,

MOVIE\_FORMAT VARCHAR2 (10 BYTE),

CONSTRAINT MOVIES\_PK PRIMARY KEY

(PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MUSIC

(

PRODUCT\_ID NUMBER NOT NULL,

ARTIST VARCHAR2 (60 BYTE) NOT NULL,

LABEL VARCHAR2 (60 BYTE) NOT NULL,

RELEASE\_DT DATE NOT NULL,

CONSTRAINT MUSIC\_PK PRIMARY KEY

(PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MUSIC\_CATEGORY

(

PRODUCT\_ID NUMBER NOT NULL,

MUSIC\_CATEGORY\_ID NUMBER NOT NULL,

CONSTRAINT MUSIC\_CATEGORY\_PK PRIMARY KEY

(PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE MUSIC\_CATEGORY\_DETAILS

(

MUSIC\_CATEGORY\_ID NUMBER NOT NULL,

MUSIC\_CATEGORY VARCHAR2 (30 BYTE),

CONSTRAINT MUSIC\_CATEGORY\_DETAILS\_PK PRIMARY KEY

(MUSIC\_CATEGORY\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE PHONE

(

PHONE VARCHAR2 (12 BYTE) NOT NULL,

CUST\_ID NUMBER NOT NULL,

PHONE\_TYPE VARCHAR2 (10 BYTE) NOT NULL,

PRIMARY\_PHONE VARCHAR2 (1 BYTE) NOT NULL,

CONSTRAINT PHONE\_PK PRIMARY KEY

(PHONE, CUST\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE PRODUCT

(

PRODUCT\_ID NUMBER NOT NULL,

PRODUCT\_NAME VARCHAR2 (100 BYTE) NOT NULL,

PRODUCT\_TYPE VARCHAR2 (10 BYTE) NOT NULL,

DEPARTMENT\_ID NUMBER NOT NULL,

QUANTITY\_ON\_HAND NUMBER NOT NULL,

REG\_PRICE NUMBER NOT NULL,

SALE\_PRICE NUMBER,

SALE\_EXP\_DT DATE,

CONSTRAINT PRODUCT\_PK PRIMARY KEY

(PRODUCT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

CREATE TABLE PRODUCT\_DEPARTMENT

(

DEPARTMENT\_ID NUMBER NOT NULL,

DEPARTMENT VARCHAR2 (30 BYTE) NOT NULL,

CONSTRAINT PRODUCT\_DEPARTMENT\_PK PRIMARY KEY

(DEPARTMENT\_ID)

ENABLE

)

LOGGING

TABLESPACE "USERS"

PCTFREE 10

INITRANS 1

STORAGE

(

INITIAL 65536

NEXT 1048576

MINEXTENTS 1

MAXEXTENTS 2147483645

BUFFER\_POOL DEFAULT

);

Activity #5

SQL Scripts to Populate Relational Design

/\* With the tables now created we can populate the data into the tables. The tables on the 1 side of the 1:M relationship must be populated first.\*/

--Insert data into table PRODUCT\_DEPARTMENT

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (1, 'Animal Crafts');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (2, 'Audio Books');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (3, 'Biography');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (4, 'CDs');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (5, 'Childrens');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (6, 'Computers');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (7, 'History');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (8, 'Holiday Crafts');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (9, 'Home Crafts');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (10, 'Horror');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (11, 'Knitting Crafts');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (12, 'Local Artist');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (13, 'Movies');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (14, 'Painting Crafts');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (15, 'Romance');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (16, 'Science Fiction');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (17, 'Self Help');

INSERT into PRODUCT\_DEPARTMENT (DEPARTMENT\_ID, DEPARTMENT)

VALUES (18, 'Travel');

COMMIT;

--Insert data into table BOOK\_CATEGORY\_DETAILS

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (1, 'Action and Adventure');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (2, 'Blossom Street Series');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (3, 'Childrens');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (4, 'Computers');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (5, 'Contemporary Women');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (6, 'Diary Wimpy Kid Series');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (7, 'Fantasy and Magic');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (8, 'Forgotten Realms Series');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (9, 'Hardy Boys Series');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (10, 'History');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (11, 'Horror');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (12, 'Local Artist');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (13, 'Reference');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (14, 'Romance');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (15, 'Self Help');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (16, 'SQL Server');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (17, 'Stephen King');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (18, 'Travel');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (19, 'Vampires');

INSERT into BOOK\_CATEGORY\_DETAILS (BOOK\_CATEGORY\_ID, BOOK\_CATEGORY) VALUES (20, 'Young Adults');

COMMIT;

--Insert data into table MUSIC\_CATEGORY\_DETAILS

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (1, 'Alternative Rock');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (2, 'Best Seller');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (3, 'British Bands');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (4, 'Childrens');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (5, 'Classic Rock');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (6, 'Classical');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (7, 'Disney');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (8, 'Folk');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (9, 'Glam');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (10, 'Latin');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (11, 'Pop');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (12, 'Rhythm and Blues');

INSERT into MUSIC\_CATEGORY\_DETAILS (MUSIC\_CATEGORY\_ID, MUSIC\_CATEGORY) VALUES (13, 'Soundtrack');

COMMIT;

--Insert data into table MOVIE\_CATEGORY\_DETAILS

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (1, 'Action and Adventure');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (2, 'Animated');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (3, 'Childrens');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (4, 'Comedy');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (5, 'Disney');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (6, 'Documentary');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (7, 'Drama');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (8, 'Fantasy');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (9, 'Horror');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (10, 'Musicals');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (11, 'Pirates');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (12, 'Romance');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (13, 'Spy');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (14, 'Super Heroes');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (15, 'Suspense');

INSERT into MOVIE\_CATEGORY\_DETAILS (MOVIE\_CATEGORY\_ID, MOVIE\_CATEGORY) VALUES (16, 'TV Series');

COMMIT;

--Insert data into table CUSTOMERS

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (1, 'Guest', 'Guest', ' ', ' ', ' ', ' ', NULL , 'N', '1 JAN 1998');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (2, 'Frederick', 'McGovern', '299 South Street', 'Halifax', 'MA', '02338', '25 JUL 1953' , 'Y', '17 MAR 2013');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (3, 'Duke', 'Crocker', '156 Haven Yacht Club', 'Haven', 'ME', '04866', '20 JUN 1977' , 'N', '10 APR 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (4, 'Hester', 'Wilde', '23 Bonnet Rd', 'VinaHaven', 'ME', '04863', '12 AUG 1937' , 'N', '4 FEB 2013');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (5, 'Josephine', 'Baker', '7 St Bonavanture Rd', 'North Haven', 'ME', '04853', '8 NOV 1940' , 'N', '11 MAR 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (6, 'Nathan', 'Wuornos', '24 Temple Street', 'Haven', 'ME', '04866', '29 APR 1977' , 'Y', '5 NOV 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (7, 'Vince', 'Teagues', '77 Tremont Street', 'Haven', 'ME', '04866', '20 DEC 1957' , 'Y', '3 JAN 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (8, 'Sally', 'Redford', '12 Rolling Meadows Rd', 'Port Clyde', 'ME', '04855', '8 MAY 1944' , 'N', '8 MAY 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (9, 'Marion', 'Zoccolante', '176 Summer Street', 'Plymouth', 'MA', '02360', '1 JAN 1939', 'N', '14 FEB 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (10, 'Julie', 'Carr', '4 Orchard Park Circle', 'Haven', 'ME', '04866', '4 AUG 1982' , 'Y', '11 DEC 2012');

INSERT into CUSTOMERS (CUST\_ID, FIRST\_NAME, LAST\_NAME, STREET, CITY, STATE, ZIP\_CODE, BIRTH\_DT, RECEIVE\_NEWSLETTER, CUSTOMER\_SINCE)

VALUES (11, 'Audrey', 'Parker', '666 Nighmare Boulevard', 'Haven', 'ME', '04866', '6 JUN 1966' , 'Y', '31 OCT 2012');

COMMIT;

--Insert data into table PHONE

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('508-846-2857', 2, 'Cell', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('781-293-2235', 2, 'Home', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-8722', 3, 'Cell', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-438-9776', 4, 'Home', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-558-4802', 5, 'Home', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-2587', 6, 'Cell', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-2800', 6, 'Work', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-0477', 7, 'Cell', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-5160', 7, 'Work', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-763-8208', 8, 'Home', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('781-423-4253', 9, 'Home', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-7766', 10, 'Cell', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-1823', 10, 'Work', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-0066', 11, 'Home', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('207-650-5674', 11, 'Cell', 'N');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('800-470-2850', 11, 'Work', 'Y');

INSERT into PHONE (PHONE, CUST\_ID, PHONE\_TYPE, PRIMARY\_PHONE)

VALUES ('800-470-2855', 11, 'Fax', 'N');

COMMIT;

--Insert data into table E\_MAIL

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Frederick\_McGovern@msn.com', 2, 'Home', 'Y');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Dcrocker@havenboats.com', 3, 'Work', 'Y');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Nwuornos@havenpolice.com', 6, 'Work', 'N');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Nathan48@comcast.net', 6, 'Home', 'Y');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Vince22@comcast.net', 7, 'Home', 'N');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Vteagues@havenherald.com', 7, 'Work', 'Y');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('Jcarr@hpt.com', 10, 'School', 'Y');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('deathfriend@comcast.net', 11, 'Home', 'N');

INSERT into E\_MAIL (E\_MAIL, CUST\_ID, E\_MAIL\_TYPE, PRIMARY\_E\_MAIL)

VALUES ('parker@penguinbooks.net', 11, 'Work', 'Y');

COMMIT;

--Insert data into table CLUB\_MEMBERSHIP

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (2, ' 17 MAR 2013', '16 MAR 2014');

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (4, ' 4 FEB 2013', '3 FEB 2014');

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (5, ' 7 JUN 2013', '6 JUN 2014');

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (6, ' 5 NOV 2013', '4 NOV 2014');

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (8, ' 8 MAY 2013', '7 MAY 2014');

INSERT into CLUB\_MEMBERSHIP (CUST\_ID, MEMBERSHIP\_START\_DT, MEMBERSHIP\_END\_DT) VALUES (11, ' 4 JUL 2013', '6 JUL 2014');

COMMIT;

--Insert data into table PRODUCT

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (1, 'Bag of Bones', 'Book', 10, 2, 7.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (2, 'Dark Moon Rising', 'Book', 12, 4, 16.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (3, 'Dark Moon Rising', 'Book', 12, 4, 7.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (4, 'Desert Night Madness', 'Book', 12, 10, 17.99, 17.49, '1 JUN 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (5, 'Firearms through History', 'Book', 7, 1, 15.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (6, 'Hannahs List', 'Book', 15, 2, 16.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (7, 'Lifes a Beach', 'Book', 15, 1, 16.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (8, 'Lifes a Beach', 'Book', 15, 2, 8.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (9, 'Lifes a Beach', 'Book', 2, 2, 16.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (10, 'Promise of the Witch King', 'Book', 16, 3, 15.99, 14.49, '1 MAY 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (11, 'Mystery of Smugglers Cove', 'Book', 5, 1, 12.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (12, 'The Passage', 'Book', 10, 1, 18.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (13, 'The Regulators', 'Book', 10, 1, 17.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (14, 'The Regulators', 'Book', 10, 1, 8.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (15, 'The Third Wheel', 'Book', 5, 2, 8.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (16, 'The Wildwater Walking Club', 'Book', 15, 2, 15.49, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (17, 'The Wildwater Walking Club', 'Book', 15, 2, 7.49, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (18, 'Time Travelling for Dummies', 'Book', 17, 1, 12.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (19, 'Where the Wild Beasts Are', 'Book', 5, 3, 12.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (20, 'SQL Server 2008 Bible', 'Book', 6, 3, 24.99, 22.49, '1 APR 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (21, '10 in cross stich image of cat', 'Craft', 1, 0, 9.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (22, '6 '' Santa Clause', 'Craft', 8, 2, 6.99, 5.49, '25 DEC 2012');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (23, 'Decorate place mats', 'Craft', 9, 6, 5.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (24, 'Hand made shawl', 'Craft', 11, 1, 18.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (25, 'Ornate wooden shelve', 'Craft', 9, 1, 27.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (26, 'Painting of Edgar Allen Poe', 'Craft', 14, 0, 45.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (27, 'Painting of lighthouse', 'Craft', 14, 1, 32.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (28, 'Red Sweater XL', 'Craft', 11, 1, 17.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (29, 'small sheep w/ bow', 'Craft', 1, 0, 9.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (30, 'Witch Sign', 'Craft', 8, 0, 9.99, 7.49, '31 OCT 2012');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (31, 'Casino Royale (1967)', 'Movie', 13, 1, 14.99, 11.79, '1 APR 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (32, 'Groundhog Day', 'Movie', 13, 1, 15.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (33, 'Lois and Clark: Season 1', 'Movie', 13, 0, 49.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (34, 'Mission Impossible: Ghost Protocol', 'Movie', 13, 3, 24.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (35, 'Pirates of the Caribbean 4', 'Movie', 13, 4, 24.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (36, 'The Avengers', 'Movie', 13, 3, 24.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (37, 'The Avengers', 'Movie', 13, 3, 19.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (38, 'The Dark Knight', 'Movie', 13, 2, 24.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (39, 'The Dark Knight', 'Movie', 13, 5, 19.99, 14.99, '1 MAY 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (40, 'The Lion King', 'Movie', 13, 1, 15.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (41, 'The Matrix', 'Movie', 13, 5, 24.99, 19.99, '1 MAY 2013');

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (42, 'The Time Travelers Wife', 'Movie', 13, 1, 24.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (43, '21', 'Music', 4, 2, 12.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (44, 'Absolution', 'Music', 4, 1, 12.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (45, 'Eye in the Sky', 'Music', 4, 1, 11.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (46, 'Goodbye Yellow Brick Road', 'Music', 4, 1, 13.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (47, 'Let it Be', 'Music', 4, 2, 9.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (48, 'The Lion King', 'Music', 4, 1, 11.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (49, 'The Passage', 'Music', 4, 0, 17.99, NULL , NULL);

INSERT into PRODUCT (PRODUCT\_ID, PRODUCT\_NAME, PRODUCT\_TYPE, DEPARTMENT\_ID, QUANTITY\_ON\_HAND, REG\_PRICE, SALE\_PRICE, SALE\_EXP\_DT)

VALUES (50, 'Very Best of Peter, Paul and Mary', 'Music', 4, 1, 14.99, NULL , NULL);

COMMIT;

--Insert data into table BOOKS

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (1, 'Pocket Books', '1 JUN 1999', '0-671-02423-X', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (2, 'Signet Books', '31 OCT 2009', '978-0325694578', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (3, 'Signet Books', '31 OCT 2011', '978-0356489622', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (4, 'Signet Books', '31 OCT 2010', '978-7482342058', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (5, 'University of Nebraska Press', '1 OCT 1980', '978-0803289031', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (6, 'Mira', '27 APR 2010', '978-0778327806', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (7, 'Hyperion', '7 JUN 2007', '978-1401303242', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (8, 'Hyperion', '6 MAY 2008', '978-1401340784', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (9, 'AudioGO', '29 JUN 2008', '978-1602839915', 'Fiction', 'Audio');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (10, 'Wizards of the Coast', '12 OCT 2006', '978-0786940738', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (11, 'Grosset and Dunlap', '21 APR 2005', '978-0448437019', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (12, 'Ballentine ', '8 JUN 2010', '978-0345504968', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (13, 'Dutton Books', '1 OCT 1996', '978-0445874569', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (14, 'Signet Books', '1 SEP 1997', '978-0451191014', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (15, 'Amulet', '13 NOV 2012', '978-1419705847', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (16, 'Voice', '5 MAY 2009', '978-1401340896', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (17, 'Voice', '10 APR 2010', '978-1401341233', 'Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (18, 'Einstein Books', '29 FEB 2012', '978-0569845065', 'Non Fiction', 'Paperback');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (19, 'Harper Collins', '9 NOV 1988', '978-0060254920', 'Fiction', 'Hardcover');

INSERT into BOOKS (PRODUCT\_ID, PUBLISHER, PUBLISHED\_DT, ISBN, BOOK\_TYPE, BOOK\_FORMAT)

VALUES (20, 'Wiley', '31 AUG 2009', '978-0470257043', 'Non Fiction', 'Hardcover');

COMMIT;

--Insert data into table BOOK\_AUTHOR

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (1, 'Stephen King');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (2, 'Audrey Parker');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (3, 'Audrey Parker');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (4, 'Audrey Parker');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (5, 'Carl Russell');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR)VALUES (6, 'Debbie MaComber');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (7, 'Claire Cook');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (8, 'Claire Cook');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (9, 'Claire Cook');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (10, 'R.A. Salvatore');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (11, 'Franklin W. Dixon');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (12, 'Justin Cronin');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (13, 'Richard Bachman');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (14, 'Richard Bachman');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (15, 'Jeff Kinney');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (16, 'Claire Cook');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (17, 'Claire Cook');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (18, 'Draco McGookin');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (18, 'Terence Howard');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (19, 'Maurice Sendak');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (20, 'Paul Nielsen');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (20, 'Uttam Parui');

INSERT into BOOK\_AUTHOR (PRODUCT\_ID, AUTHOR) VALUES (20, 'Mike White');

COMMIT;

--Insert data into table BOOK\_CATEGORY

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (1, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (1, 17);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (2, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (2, 12);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (3, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (3, 12);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (4, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (4, 12);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (5, 10);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (6, 2);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (6, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (7, 5);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (7, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (8, 5);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (8, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (9, 5);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (9, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (10, 7);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (10, 8);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (11, 1);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (11, 9);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (12, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (12, 19);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (13, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (13, 17);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (14, 11);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (14, 17);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (15, 6);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (15, 20);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (16, 5);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (16, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (17, 5);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (17, 14);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (18, 13);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (18, 15);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (19, 3);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (19, 7);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (20, 4);

INSERT into BOOK\_CATEGORY (PRODUCT\_ID, BOOK\_CATEGORY\_ID)

VALUES (20, 16);

COMMIT;

--Insert data into table MUSIC

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (43, 'Adele', 'Columbia', '22 FEB 2011');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (44, 'Muse', 'Warners Brothers', '23 MAR 2004');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (45, 'Alan Parsons Project', 'Sony Legacy', '20 MAR 2007');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (46, 'Elton John', 'Island Records', '20 FEB 1996');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (47, 'Beatles', 'EMI', '9 SEP 2009');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (48, 'Elton John', 'Disney', '1 JAN 1996');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (49, 'Al Stewart', 'Rhino Records', '27 APR 2005');

INSERT into MUSIC (PRODUCT\_ID, ARTIST, LABEL, RELEASE\_DT)

VALUES (50, 'Peter, Paul and Mary', 'Rhino Records', '29 JUL 2008');

COMMIT;

--Insert data into table MUSIC\_CATEGORY

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (43, 2);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (43, 11);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (44, 1);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (44, 3);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (45, 11);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (46, 9);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (46, 11);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (47, 3);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (47, 5);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (48, 4);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (48, 7);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (48, 13);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (49, 8);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (49, 11);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (50, 4);

INSERT into MUSIC\_CATEGORY (PRODUCT\_ID, MUSIC\_CATEGORY\_ID)

VALUES (50, 8);

COMMIT;

--Insert data into table MOVIES

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (31, 'MGM', '15 OCT 2002', 'DVD');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (32, 'Sony Pictures', '27 JAN 2009', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (33, 'Warnes Home Video', '7 JUN 2005', 'DVD');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (34, 'Paramount Pictures', '17 APR 2012', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (35, 'Walt Disney Studios', '15 NOV 2011', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (36, 'Walt Disney Studios', '25 SEP 2012', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (37, 'Walt Disney Studios', '25 SEP 2012', 'DVD');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (38, 'Warnes Home Video', '9 DEC 2008', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (39, 'Warnes Home Video', '9 DEC 2008', 'DVD');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (40, 'Walt Disney Studios', '15 NOV 2011', 'DVD');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (41, 'Warnes Home Video', '15 MAY 2007', 'Blu-ray');

INSERT into MOVIES (PRODUCT\_ID, STUDIO, RELEASE\_DT, MOVIE\_FORMAT)

VALUES (42, 'New Line', '9 FEB 2010', 'Blu-ray');

COMMIT;

--Insert data into table MOVIE\_CATEGORY

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (31, 4);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (31, 13);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (32, 4);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (33, 14);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (33, 16);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (34, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (34, 13);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (35, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (35, 11);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (36, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (36, 14);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (37, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (37, 14);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (38, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (38, 14);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (39, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (39, 14);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (40, 2);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (40, 3);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (40, 5);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (41, 1);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (41, 8);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (42, 8);

INSERT into MOVIE\_CATEGORY (PRODUCT\_ID, MOVIE\_CATEGORY\_ID)

VALUES (42, 12);

COMMIT;

--Insert data into table MOVIE\_DIRECTOR

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (31, 'Don Medford');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (31, 'John Huston');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (31, 'Joseph McGrath');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (31, 'Ken Hughes');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (31, 'Richard Talmadge');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (32, 'Harold Ramis');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (33, 'Bill D elia');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (34, 'Brad Bird');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (35, 'Rob Marshall');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (36, 'Joss Whedon');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (37, 'Joss Whedon');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (38, 'Christopher Nolan');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (39, 'Christopher Nolan');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (40, 'Rob Minkoff');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (41, 'The Wachowski Brothers');

INSERT into MOVIE\_DIRECTORS (PRODUCT\_ID, DIRECTOR)

VALUES (42, 'Robert Schwentke');

COMMIT;

--Insert data into table CRAFTS

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (21, '10 in cross stich image of black cat sitting on a fireplace', 'Jacqueline Davies', '1 FEB 2013');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (22, '6 '' Santa Clause made with croquet and felt', 'Norma Koeppel', '25 NOV 2012');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (23, 'Decorate place mats with Italian scenery', 'Brenda Thomas', '15 JUL 2012');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (24, 'Hand made blue shawl', 'Jacqueline Davies', '14 MAR 2013');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (25, 'Ornate wooden shelve with carved heart on sides', 'Peter Clausson', '27 NOV 2011');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (26, 'Painting of Edgar Allen Poe on canvas', 'Joan Archer', '19 JAN 2012');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (27, 'Painting of lighthouse using oils', 'Joan Archer', '20 SEP 2012');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (28, 'Red Cashmere Sweater Size XL', 'Norma Koeppel', '13FEB 2013');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (29, 'Knitted small sheep w/ blue bow around neck', 'Norma Koeppel', '16 DEC 2012');

INSERT into CRAFTS (PRODUCT\_ID, DESCRIPTION, CREATOR, CREATION\_DT)

VALUES (30, 'Witch Sign - Various Halloween expressions', 'David Peers', '3 OCT 2012');

COMMIT;

--Insert data into table INVOICE\_DETAILS

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (1, 4, 1, 16.19, '17 MAR 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (1, 45, 1, 11.99, '17 MAR 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (2, 35, 1, 24.99, '10 APR 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (3, 11, 1, 12.99, '20 DEC 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (4, 2, 1, 16.99, '28 FEB 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (5, 21, 1, 9.99, '4 FEB 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (6, 5, 1, 14.39, '5 NOV 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (7, 18, 1, 11.69, '5 JAN 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (8, 42, 1, 24.99, '14 MAR 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (9, 25, 1, 27.99, '3 JAN 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (10, 50, 1, 14.99, '8 JUL 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (11, 23, 4, 5.99, '21 JUL 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (12, 1, 1, 7.99, '8 OCT 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (12, 4, 1, 17.49, '8 OCT 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (13, 6, 1, 15.29, '8 MAY 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (13, 9, 1, 15.29, '8 MAY 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (14, 16, 1, 13.94, '10 SEP 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (15, 15, 1, 8.99, '20 DEC 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (15, 20, 1, 24.99, '20 DEC 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (15, 29, 1, 9.99, '20 DEC 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (16, 36, 1, 24.99, '12 FEB 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (17, 12, 1, 17.09, '31 OCT 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (17, 30, 2, 7.49, '31 OCT 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (18, 13, 1, 16.19, '11 NOV 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (19, 26, 1, 45.99, '20 DEC 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (20, 19, 1, 11.69, '7 JAN 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (21, 10, 1, 14.49, '11 MAR 2012');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (22, 14, 1, 8.99, '20 FEB 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (22, 24, 1, 24, '20 FEB 2013');

INSERT into INVOICE\_DETAILS (INVOICE\_ID, PRODUCT\_ID, QUANTITY, UNIT\_PRICE, PURCHASE\_DT) VALUES (22, 39, 1, 14.99, '20 FEB 2013');

COMMIT;

--Insert data into table INVOICE

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (2, 1);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (3, 2);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (3, 3);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (3, 4);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (4, 5);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (6, 6);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (6, 7);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (6, 8);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (7, 9);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (7, 10);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (7, 11);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (7, 12);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (8, 13);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (8, 14);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (10, 15);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (10, 16);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (11, 17);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (11, 18);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (11, 19);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (11, 20);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (11, 21);

INSERT into INVOICE (CUST\_ID, INVOICE\_ID) VALUES (1, 22);

COMMIT;

Activity #6

SQL Scripts to Meet Reporting Requirements

/\* We can now create views to show the data in the various reports.\*/

/\*Report Basic Product Detail - Basic product details including the quantity on hand, the price both regular and if on sale the sales price, the type of product and the department the product is located in must all be available by product id.\*/

SELECT A.PRODUCT\_ID, A.PRODUCT\_NAME, A.PRODUCT\_TYPE, B.DEPARTMENT, A.QUANTITY\_ON\_HAND,

A.REG\_PRICE, A.SALE\_PRICE, A.SALE\_EXP\_DT

FROM

PRODUCT A

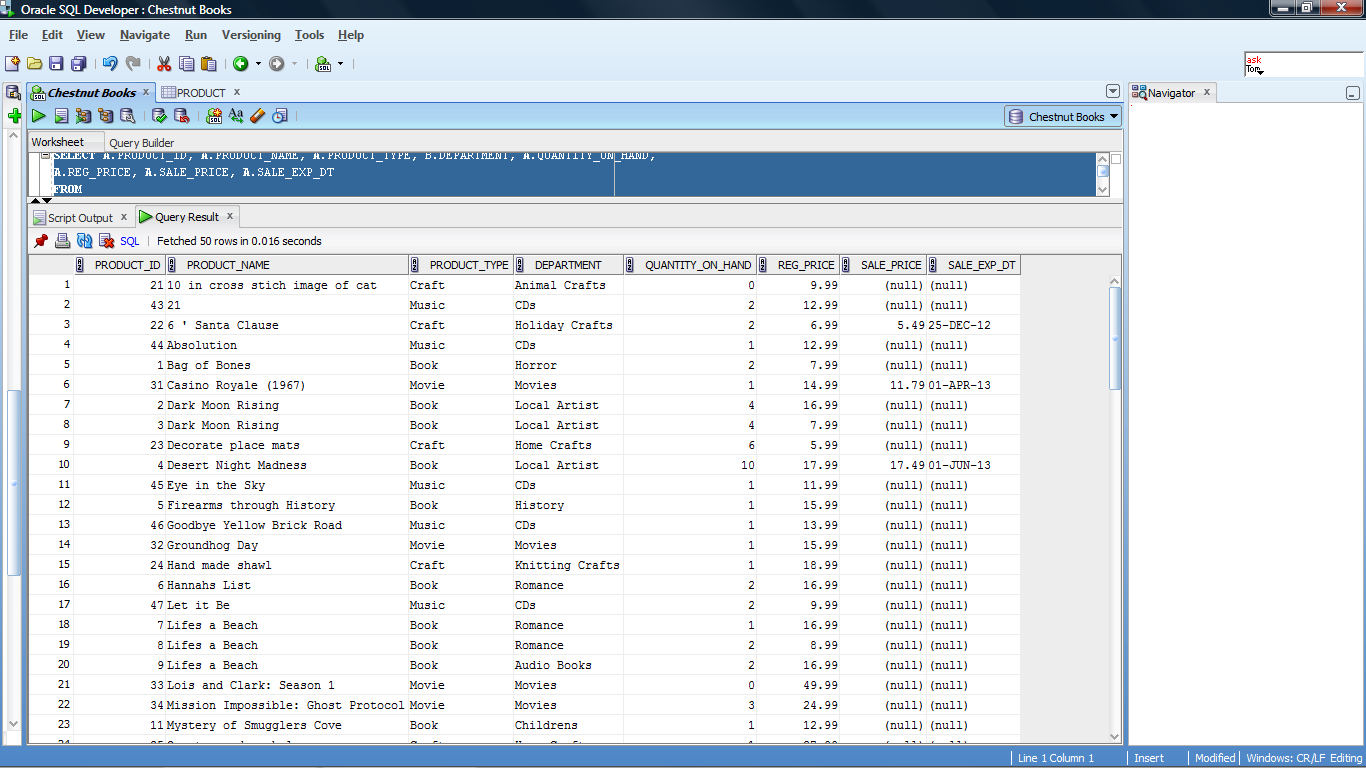
INNER JOIN

PRODUCT\_DEPARTMENT B

ON A.DEPARTMENT\_ID = B.DEPARTMENT\_ID

ORDER BY A.PRODUCT\_NAME, A.PRODUCT\_ID

--Output for Report Basic Product Detail--



/\*Report Book Detail -For a given book, a query must be developed that will return all detailed information including the author, publisher, publication date, title of book, ISBN number if available, the type and format of the book as well as any categories that the book is associated with and its price. \*/

SELECT A.PRODUCT\_ID, A.PRODUCT\_NAME, C.AUTHOR, B.PUBLISHER, B.PUBLISHED\_DT, B.ISBN, B.BOOK\_TYPE, B.BOOK\_FORMAT,

E.BOOK\_CATEGORY, A.REG\_PRICE, A.SALE\_PRICE

FROM

PRODUCT A

INNER JOIN

BOOKS B

ON A.PRODUCT\_ID = B.PRODUCT\_ID

INNER JOIN

BOOK\_AUTHOR C

ON B.PRODUCT\_ID = C.PRODUCT\_ID

INNER JOIN

BOOK\_CATEGORY D

ON B.PRODUCT\_ID = D.PRODUCT\_ID

INNER JOIN

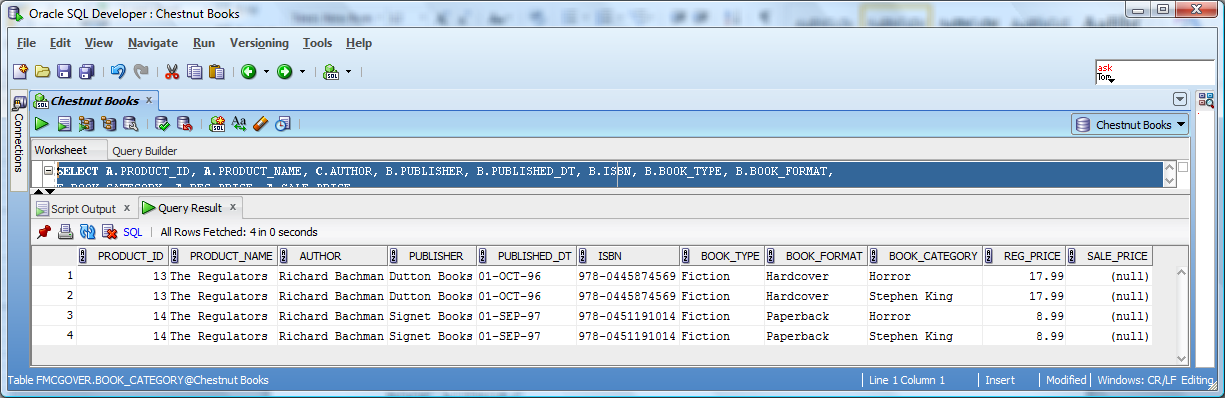
BOOK\_CATEGORY\_DETAILS E

ON D.BOOK\_CATEGORY\_ID = E.BOOK\_CATEGORY\_ID

WHERE A.PRODUCT\_NAME = 'The Regulators'

ORDER BY A.PRODUCT\_ID, E.BOOK\_CATEGORY

--Output for Report Book Detail—



/\*Report Music Detail - For music CD’s a query must return details such as title, release date, label, artist, category and price for a given music selection.\*/

SELECT A.PRODUCT\_ID, A.PRODUCT\_NAME, B.ARTIST, B.LABEL, B.RELEASE\_DT, D.MUSIC\_CATEGORY, A.REG\_PRICE, A.SALE\_PRICE

FROM

PRODUCT A

INNER JOIN

MUSIC B

ON A.PRODUCT\_ID = B.PRODUCT\_ID

INNER JOIN

MUSIC\_CATEGORY C

ON B.PRODUCT\_ID = C.PRODUCT\_ID

INNER JOIN

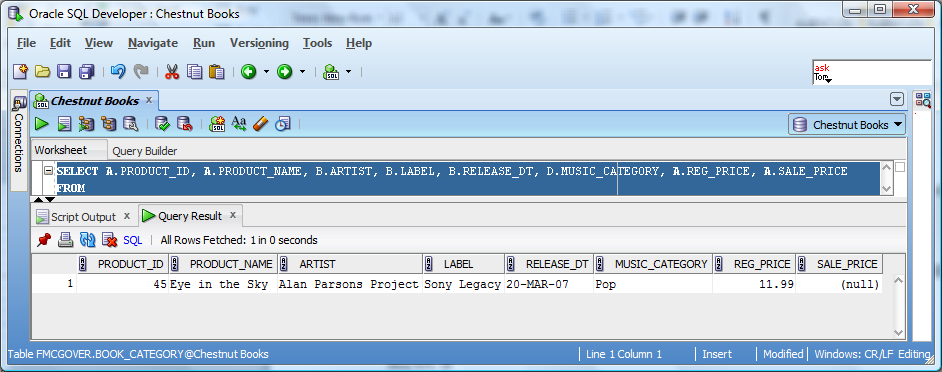
MUSIC\_CATEGORY\_DETAILS D

ON C.MUSIC\_CATEGORY\_ID = D.MUSIC\_CATEGORY\_ID

WHERE A.PRODUCT\_NAME = 'Eye in the Sky'

ORDER BY A.PRODUCT\_ID, D.MUSIC\_CATEGORY

--Output for Report Music Detail—



/\*Report Movie Detail - For DVD’s and Blu-ray’s, a report must be developed that will return to the user details such as title, studio, release date, director and format and price. .\*/

SELECT A.PRODUCT\_ID, A.PRODUCT\_NAME, B.STUDIO, E.DIRECTOR, B.RELEASE\_DT, B.MOVIE\_FORMAT, D.MOVIE\_CATEGORY, A.REG\_PRICE, A.SALE\_PRICE

FROM

PRODUCT A

INNER JOIN

MOVIES B

ON A.PRODUCT\_ID = B.PRODUCT\_ID

INNER JOIN

MOVIE\_CATEGORY C

ON B.PRODUCT\_ID = C.PRODUCT\_ID

INNER JOIN

MOVIE\_CATEGORY\_DETAILS D

ON C.MOVIE\_CATEGORY\_ID = D.MOVIE\_CATEGORY\_ID

INNER JOIN

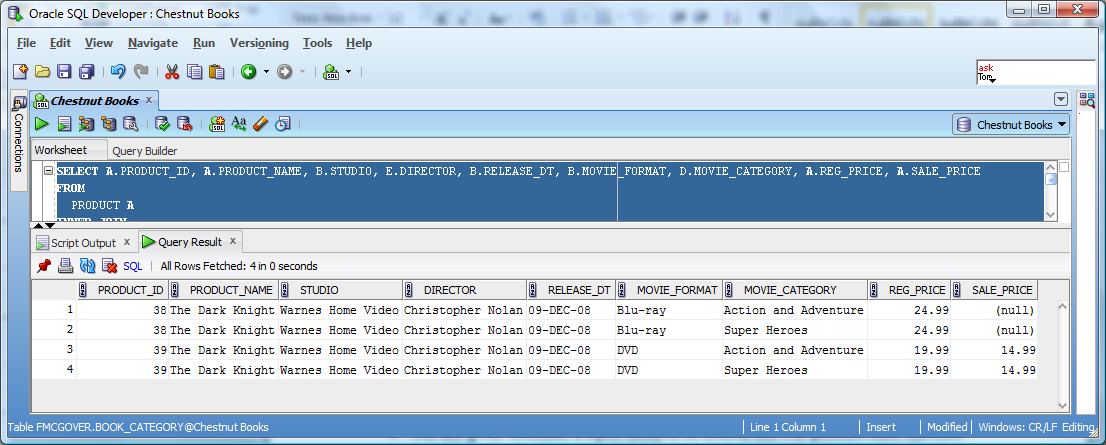
MOVIE\_DIRECTORS E

ON B.PRODUCT\_ID = E.PRODUCT\_ID

WHERE A.PRODUCT\_NAME = 'The Dark Knight'

ORDER BY A.PRODUCT\_ID, D.MOVIE\_CATEGORY

--Output for Report Movie Detail—



/\*Report Craft Detail - A report listing all handmade crafts available for sale which includes a description of the item, the category assigned to the item and the price of the item.\*/

SELECT A.PRODUCT\_ID, A.PRODUCT\_NAME, B.DESCRIPTION, B.CREATOR, B.CREATION\_DT, A.REG\_PRICE, A.SALE\_PRICE

FROM

PRODUCT A

INNER JOIN

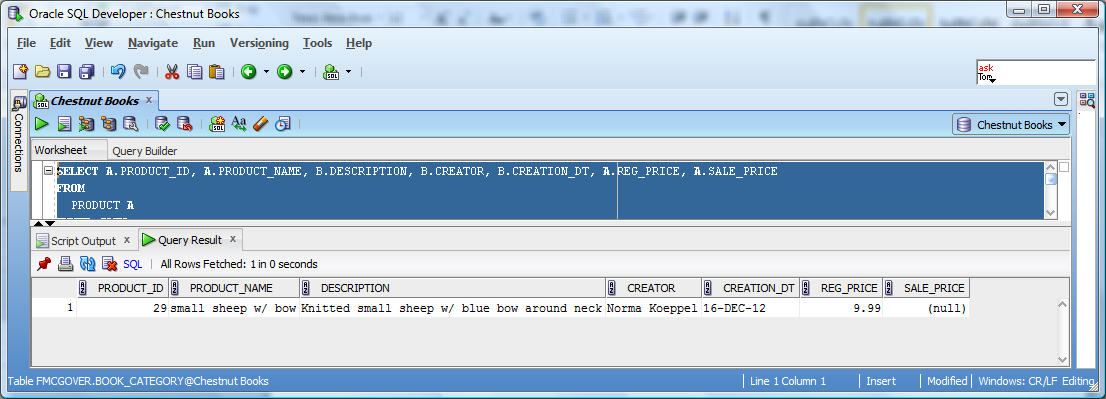
CRAFTS B

ON A.PRODUCT\_ID = B.PRODUCT\_ID

WHERE A.PRODUCT\_NAME = 'small sheep w/ bow'

ORDER BY A.PRODUCT\_ID

--Output for Report Craft Detail—



/\*Report Customer Detail - For any given customer, a report needs to be created that will generate basic customer information such as name, address, phone and email as well as whether or not the customer has signed up for the monthly newsletter and whether or not they are part of the reader’s club discount program.\*/

SELECT A.FIRST\_NAME, A.LAST\_NAME, A.STREET, A.CITY, A.STATE, A.ZIP\_CODE, A.BIRTH\_DT, A.RECEIVE\_NEWSLETTER, A.CUSTOMER\_SINCE,

B.PHONE, B.PHONE\_TYPE, B.PRIMARY\_PHONE, C.E\_MAIL, C.E\_MAIL\_TYPE, C.PRIMARY\_E\_MAIL,

(CASE WHEN D.CUST\_ID IS NULL THEN 'N' ELSE 'Y' END) AS CLUB\_MEMBER, D.MEMBERSHIP\_START\_DT, D.MEMBERSHIP\_END\_DT

FROM

CUSTOMERS A

LEFT OUTER JOIN

PHONE B

ON A.CUST\_ID = B.CUST\_ID

LEFT OUTER JOIN

E\_MAIL C

ON A.CUST\_ID = C.CUST\_ID

LEFT OUTER JOIN

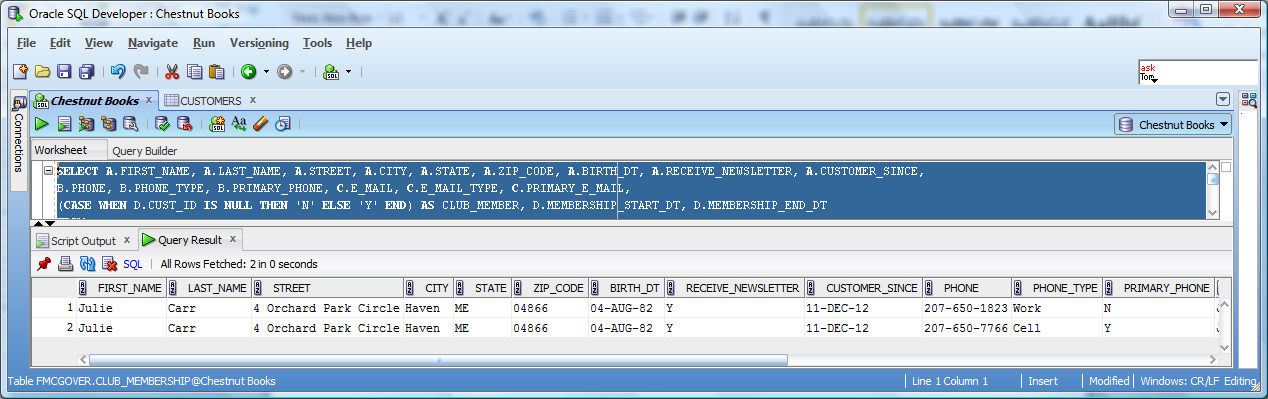
CLUB\_MEMBERSHIP D

ON A.CUST\_ID = D.CUST\_ID

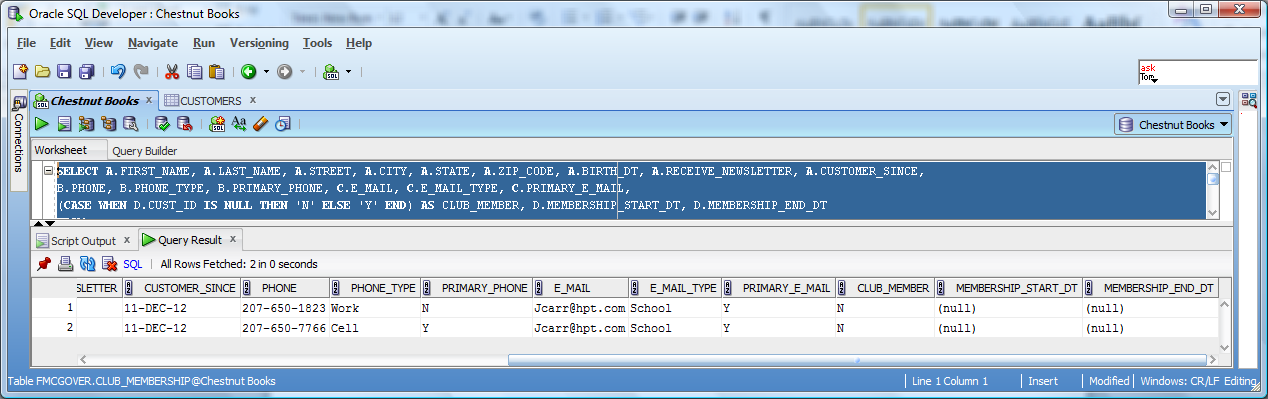
WHERE A.CUST\_ID = 10

--Output for Report Customer Detail—

--Part 1--



--Part 2—



/\*Report Book Club Customer Invoice - For any given customer signed up for the reader’s discount club, a report that details the customers purchases either on a specific date or over a given date range should be created. This report would provide the customer number, invoice number, the product id and the price paid for that product.\*/

SELECT A.CUST\_ID, B.INVOICE\_ID, C.PRODUCT\_ID, D.PRODUCT\_NAME, D.PRODUCT\_TYPE, C.QUANTITY, C.UNIT\_PRICE AS PURCHASE\_PRICE, C.PURCHASE\_DT

FROM

CUSTOMERS A

INNER JOIN

INVOICE B

ON A.CUST\_ID = B.CUST\_ID

INNER JOIN

INVOICE\_DETAILS C

ON B.INVOICE\_ID = C.INVOICE\_ID

INNER JOIN

PRODUCT D

ON C.PRODUCT\_ID = D.PRODUCT\_ID

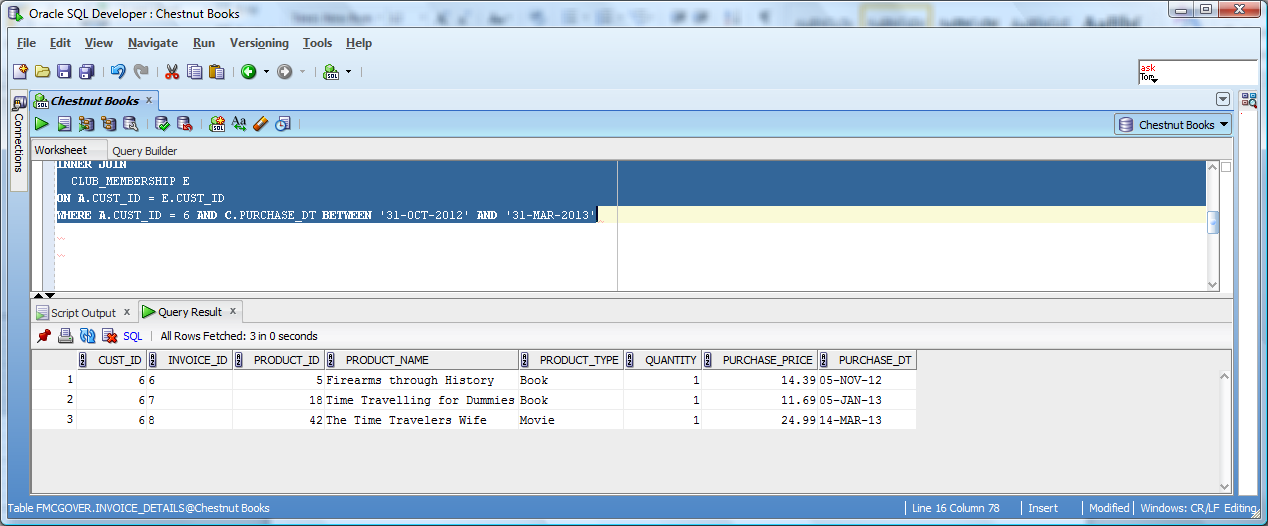
INNER JOIN

CLUB\_MEMBERSHIP E

ON A.CUST\_ID = E.CUST\_ID

WHERE A.CUST\_ID = 6 AND C.PURCHASE\_DT BETWEEN '31-OCT-2012' AND '31-MAR-2013'

--Output for Report Book Club Customer Invoice –



/\*Report Local Author Sales - Since the company likes to emphasize the works of local horror writer Audrey Parker, a report should be created that provide the number of copies sold and total sales by title for a given month. \*/

SELECT B.PRODUCT\_NAME, SUM(A.QUANTITY) AS UNITS\_SOLD, SUM(A.QUANTITY \* A.UNIT\_PRICE) AS TTL\_SALES

FROM

INVOICE\_DETAILS A

INNER JOIN

PRODUCT B

ON A.PRODUCT\_ID = B.PRODUCT\_ID

INNER JOIN

BOOKS C

ON B.PRODUCT\_ID = C.PRODUCT\_ID

INNER JOIN

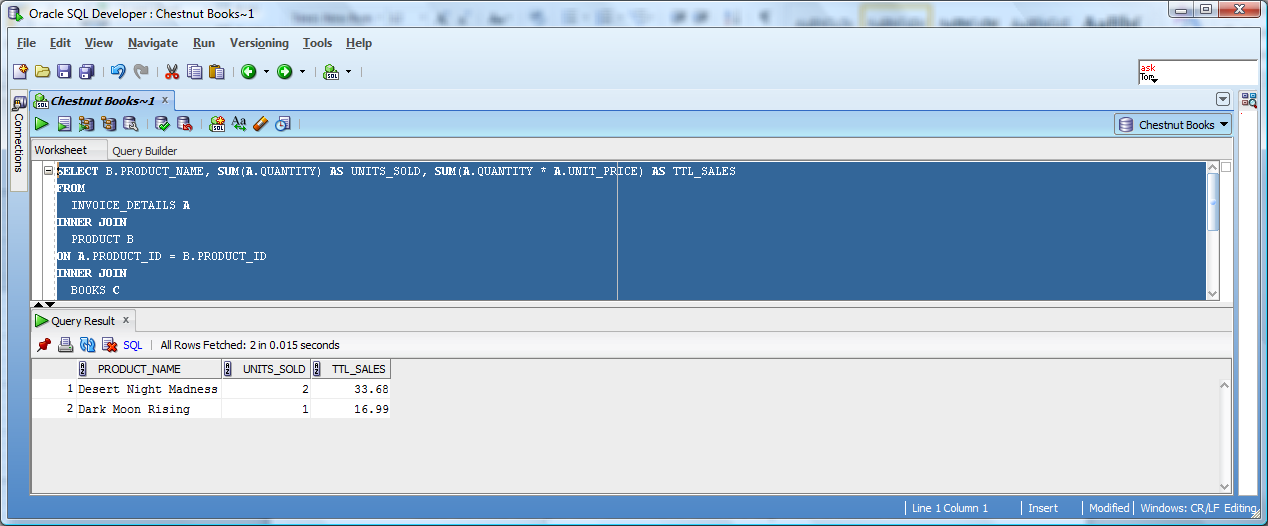
BOOK\_AUTHOR D

ON C.PRODUCT\_ID = D.PRODUCT\_ID

WHERE D.AUTHOR = 'Audrey Parker' AND A.PURCHASE\_DT >'31 DEC 2011'

GROUP BY B.PRODUCT\_NAME

--Output for Report Local Author Sales –



/\*Report Book Club Renewals - Since the book club provides additional revenue it is important to remind customers when their membership is about to expire so that they can renew it. A report must therefore be generated to list all customers with expiring memberships within the next 90 days.\*/

SELECT A.FIRST\_NAME, A.LAST\_NAME, A.STREET, A.CITY, A.STATE, A.ZIP\_CODE, C.PHONE, C.PRIMARY\_PHONE, B.MEMBERSHIP\_END\_DT

FROM

CUSTOMERS A

INNER JOIN

CLUB\_MEMBERSHIP B

ON A.CUST\_ID = B.CUST\_ID

LEFT OUTER JOIN

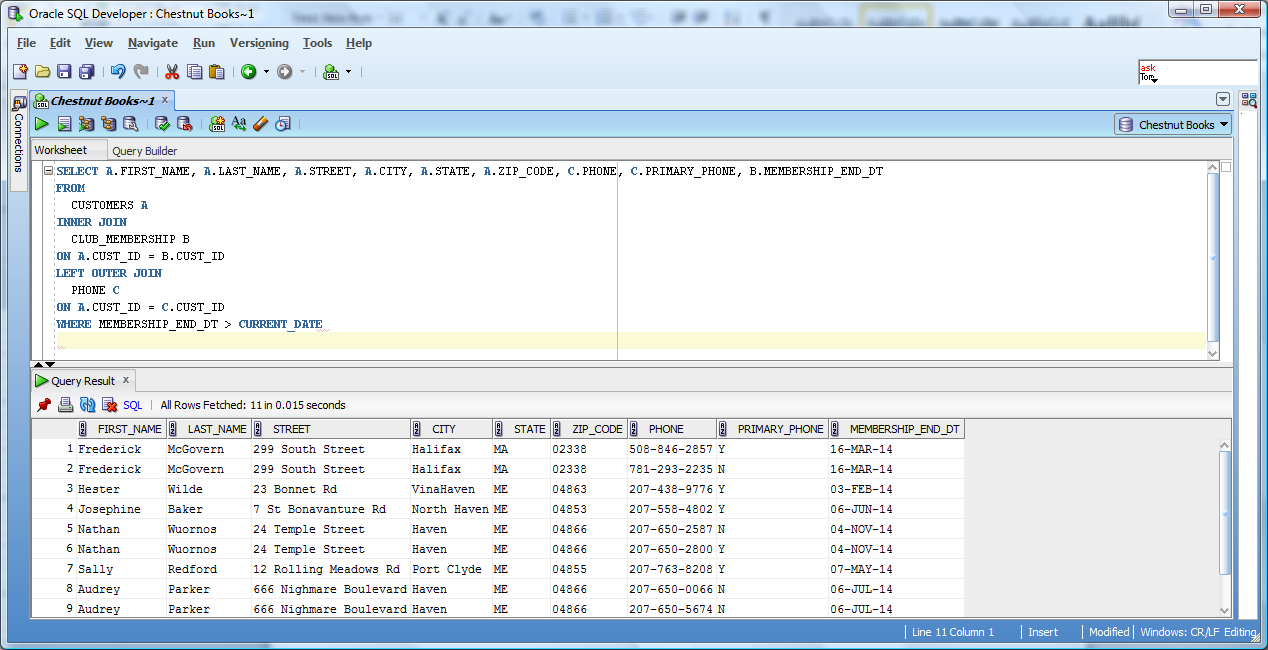
PHONE C

ON A.CUST\_ID = C.CUST\_ID

WHERE MEMBERSHIP\_END\_DT > CURRENT\_DATE

--Note no record exist with 90 day parameter so changed query to produce results--

--Output for Report Book Club Renewals

 –

Activity #7

User Interface Requirements

The user interface that would be required would probably be built utilizing VB.Net or C#.Net. The interface will have a general login screen that employees will utilize to get access to the system. The user id assigned to each employee will establish the rights they have to the database as each user would be assigned to a particular security group.

Once logged in, the main screen will be a switchboard to will list all the options that are available. Clicking on a particular link on the switchboard screen will result in the user being shown the screen necessary for a particular task. The most commonly used screen would be the transaction screen. On this screen, the user can start a new sale or look up a prior sale. A prior Invoice\_ID can be entered or the employee can search by either the customer’s name or reader membership id to retrieve the information on a prior invoice. Only customers where the company has contact information and/or are members of the reading club would have a stored invoice. Assuming the customer is starting a new transaction, the employee can do one of three things:

* Enter in the customer Book Club ID so as to return a Cust\_ID.
* Look up Cust\_ID by entering the customers Last\_Name and Phone number
* Use the guest account for the sale

Once one of these is completed, a new invoice id can be generated which is related to the Cust\_ID being used. The employee will need to enter in a Product\_ID into the form, which would then return either the regular price or sales price if one exists and the sale has not expired. This application does not currently utilize Universal Pricing Codes, so the item being purchased should have a tag on it with the assigned Product\_ID. Once the product detail is returned to the screen, the employee can verify that the item being purchased matches the item description in the system. If it does, the employee will enter in a quantity if greater than one (system would default to one). The employee would continue the process of entering Product\_id’s into the interface until all items have been entered. A query would then be executed which would return the total amount for all items ie Sum(Quantity \* Price being used). Once payment has been made, the data is committed to first the Invoice\_Detail table and then to the Invoice table. The Product Quantity\_On\_Hand field is updated to be the current value – the quantity for that product just sold.

Since the company collects customer data and offers a book club, there would also be a separate form to fill out for new customers. All basic customer data that is contained in the Customer table would be entered first including name and address. Once the employee clicks the “Have you verified all items” button, two things will occur. Behind the scenes, the data is added to the customer table and a Cust\_ID value is returned. Secondly, a screen will be displayed that allows the employee to enter in the customer phone information. The form has multiple entry areas so that an employee can add multiple phone number, select the type of phone the number corresponds to and what number will be assigned as the primary number. Once this data is entered the data will be stored in the Phone table along with the newly created Cust\_ID. Another form will then be displayed so that the employee can add in the customer’s email address. Similar to the phone display, there would be multiple entry areas, the ability to assign a type to an email and to set a specific email to the primary email. Once the enter button is selected, the data is committed to the E\_Mail table with the associated Cust\_ID. If the customer has elected to sign up for the book club and paid the fee, the employee will select Yes/No from a dropdown control. If the employee selected “Yes”, the Cust\_ID will be added to the Club\_Membership table along with the current date, which is used to set the Membership\_Start\_Dt and through a calculation the Membership\_End\_Dt.

Additional screens, which would be restricted to manager’s only or have restricted functionality would be used for inventory control and product informational queries. A product screen would need to exist so that any employee can look up an item for a customer to given them information such as what department the product is located in , the quantity on hand, the name of the product etc. Based on the type of product additional information can be returned such as the book author, the name of a CD for a particular artist, or the release date for a Blu-ray disk. The manager only would have some additional capabilities including:

* Selecting the option to add a product. This would open a secondary screen for that task.
* Select an option to change the price. Again this would open a secondary screen for that task
* Update the inventory either because of a return or because new stock has arrived

Assuming the manager is adding a new product, a new screen would be displayed. The manager would enter the basic items found on the Product table such as the Product Name, Department assigned , the Product\_Type, quantity and price. Once all data has been supplied, the data is committed to the Product table and a Product\_ID is returned. Based on the type of product entered, one of four specialty forms will be displayed so as to enter in the details for that product. For example, if the new item was a book, a screen would be displayed that would require information including the author(s), publisher, published\_dt, ISBN number, format and the category(s) that the book should be assigned to. The manager would select one or more categories from a drop down display that shows all the entries in the associated category\_details table, in this example that being the Book\_Category\_Detail table. If the manager wants to create a new category, they can click a button which would navigate them to a form to do so. If a new category is entered, the data is committed to the Category\_Detail table, a category\_id is returned and control returns to the prior form. Once all the detail data is entered, the manager will click submit. This will commit the date first to the specific product table based on product type (ie Books, Music, Movies, Crafts) . The product\_id and category\_id that have been being stored in the form are then commiteed to the bridge table (ie Books\_Category). The process is identical for the other product type with the only differences being the detail attributes that need to be entered.