Prac 1

* **ERD Exercises**

You are required to create ERDs for the following exercises and submit (or show to your prac supervisor) them to get marked off for this week’s prac activity

1. Given the following business rules, create the appropriate Crow’s Foot ERD.
   * A company operates many departments
   * Each department employs one or more employees
   * Each of the employees may or may not have one or more dependents.
   * Each employee may or may not have an employment history.
2. Use the following business rules to create a Crow’s Foot ERD. Present all appropriate connectivities and cardinalities in the ERD.

* A department employs many employees, but each employee is employed by one department.
* Some employees are not assigned to any department.
* A division operates many departments, but each department is operated by one division.
* An employee may be assigned many projects, and a project may have many employees assigned to it.
* A project must have at least one employee assigned to it.
* One of the employees manages each department, and each department is managed by only one employee.
* One of the employees runs each division, and each division is run by only one employee.

Hint: More than one relationship may be existing between two entities.

Prac 2

* **ERD Exercises**

**Note: For the following exercises you are required to create a completed version of models by creating necessary attributes for each entity (while the last prac exercises required to list a couple of essential attributes only).**

**As each exercise is for you to design a full version of ERD (though the business rules here are a lot simpler than real-world situations), you will need to carefully design all components of your model considering the business rules provided. Such consideration may include deciding about “what attributes are included in each entity?”, “Should the relationship be 1:1, 1:M or M:N?”, “Should the relationship be identifying or non-identifying?” “Should the relationship on one entity side be optional or mandatory?” etc.**

**You are recommended to start drawing your draft ERD on the paper firstly before creating the more confirmed ERD using MySQL Workbench or other drawing tools.**

1. Use the following business rules to create a Crow’s Foot ERD. Present all appropriate connectivities and cardinalities in the ERD.

* A department employs many employees, but each employee is employed by one department.
* Some employees are not assigned to any department.
* A division operates many departments, but each department is operated by one division.
* An employee may be assigned many projects, and a project may have many employees assigned to it.
* A project must have at least one employee assigned to it.
* One of the employees manages each department, and each department is managed by only one employee.
* One of the employees runs each division, and each division is run by only one employee.

Hint: More than one relationship may be existing between two entities.

1. Create an ERD based on the Crow’s Foot model, using the following requirements:

* An INVOICE is written by a SALESREP. Each sales representative can write many invoices, but each invoice is written by a single sales representative.
* The INVOICE is written for a single CUSTOMER. However, each customer can have many invoices.
* An INVOICE can include many detail lines (LINE), which describe the products bought by the customer.
* The product information is stored in a PRODUCT entity.
* The product’s vendor information is found in a VENDOR entity.

1. Create a Crow’s Foot ERD, using the following business Rules.

* A Team may or may not have a Player
* A Player must belong to a Team
* A Team may have many Players
* A Player has only one Team
* A Team may or may not have a Coach
* A Coach must have a Team
* A Team may have many Coaches
* A Coach has only one Team
* A Player must have a Parent
* A Parent must have a Player
* A Player may have up to 2 Parents
* A Parent may have many Players

Prac 5

**Note:**

* **Exercises 1**

Given the dependency diagram shown below, answer items a ~ c:

(Present your answers for a~c on paper or WORD document)



* 1. Identify and discuss each of the indicated dependencies.
  2. Create a database whose tables are at least in 2NF, showing the dependency diagrams for each table.

1. Create a database whose tables are at least in 3NF, showing the dependency diagrams for each table.

* **Exercise 2**

Using the STUDENT table structure shown in the table as below, do the following (a ~ c):

Sample STUDENT Records

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Sample Value** | **Sample Value** | **Sample Value** | **Sample Value** | **Sample Value** |
| STU\_NUM | 211343 | 200128 | 199876 | 199877 | 223456 |
| STU\_LNAME | Stephanos | Smith | Jones | Ortiz | McKulski |
| STU\_MAJOR | Accounting | Accounting | Marketing | Marketing | Statistics |
| DEPT\_CODE | ACCT | ACCT | MKTG | MKTG | MATH |
| DEPT\_NAME | Accounting | Accounting | Marketing | Marketing | Mathematics |
| DEPT\_PHONE | 4356 | 4356 | 4378 | 4378 | 3420 |
| COLLEGE\_NAME | Business Admin | Business Admin | Business Admin | Business Admin | Arts & Sciences |
| STU\_GPA | 3.87 | 2.78 | 2.31 | 3.45 | 3.58 |
| STU\_HOURS | 75 | 45 | 117 | 113 | 87 |
| STU\_CLASS | Junior | Sophomore | Senior | Senior | Junior |

(Present your answers for a and b on paper or WORD document. For c, you are required to draw the ERD using MySQL Workbench or other drawing tool)

a. Draw its dependency diagram and identify all dependencies, including all transitive dependencies.

b. Write the relational schema and draw the dependency diagram to meet the 3NF requirements to the greatest extent possible. If you believe that practical considerations dictate using a 2NF structure, explain why your decision to retain 2NF is appropriate. If necessary, add or modify attributes to create appropriate determinants and to adhere to the naming conventions.

c. Draw the Crow’s Foot ERD.

* **Exercise 3**

The manager of a consulting firm has asked you to evaluate a database that contains the table structure shown below.

Sample CLIENT Records

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute Name | **Sample Value** | **Sample value** | **Sample Value** |
| CLIENT\_NUM | 298 | 289 | 289 |
| CLIENT\_NAME | Marianne R. Brown | James D. Smith | James D. Smith |
| CLIENT\_REGION | Midwest | Southeast | Southeast |
| CONTRACT\_DATE | 10-Feb-2010 | 15-Feb-2010 | 12-Mar-2010 |
| CONTRACT\_NUMBER | 5841 | 5842 | 5843 |
| CONTRACT\_AMOUNT | $2,985,00.00 | $670,300.00 | $1,250,000.00 |
| CONSULT\_CLASS\_1 | Database Administration | Internet Services | Database Design |
| CONSULT\_CLASS\_2 | Web Applications |  | Database Administration |
| CONSULT\_CLASS\_3 |  |  | Network Installation |
| CONSULT\_CLASS\_4 |  |  |  |
| CONSULTANT\_NUM\_1 | 29 | 34 | 25 |
| CONSULTANT\_NAME\_1 | Rachel G. Carson | Gerald K. Ricardo | Angela M. Jamison |
| CONSULTANT\_REGION\_1 | Midwest | Southeast | Southeast |
| CONSULTANT\_NUM\_2 | 56 | 38 | 34 |
| CONSULTANT\_NAME\_2 | Karl M. Spenser | Anne T. Dimarco | Gerald K. Ricardo |
| CONSULTANT\_REGION\_2 | Midwest | Southeast | Southeast |
| CONSULTANT\_NUM\_3 | 22 | 45 |  |
| CONSULTANT\_NAME\_3 | Julian H. Donatello | Geraldo J. Rivera |  |
| CONSULTANT\_REGION\_3 | Midwest | Southeast |  |

This table was created to enable the manager to match clients with consultants. The objective is to match a client within a given region with a consultant in that region, and to make sure that the client’s need for specific consulting services is properly matched to the consultant’s expertise. For example, if the client need help with database design and is located in the Southeast, the objective is to make a match with a consultant who is located in the Southeast and whose expertise is in database design. (Although the consulting company manage tries to match consultant and client locations to minimize travel expense, it is not always possible to do so.) The following basic business rules are maintained:

* Each client is located in one region
* A region can contain many clients.
* Each consultant can work on many contracts
* Each contract might require the services of many consultants.
* A client can sign more than one contract, but each contract is signed by only one client.
* Each contract might cover multiple consulting classifications. (For example, a contract may list consulting services in database and networking.)
* Each consultant is located in one region.
* A region can contain many consultants.
* Each consultant has one or more areas of expertise (class). For example, a consultant might be classified as an expert in both database design and networking.
* Each area of expertise (class) can have many consultants in it. For example, the consulting company might employ many consultants who are networking experts.

1. Given that brief description of the requirements and the business rules, write the relational schema and draw the dependency diagram for the preceding (and very poor) table structure. Label all transitive and/or partial dependencies.
2. Break up the dependency diagram you drew in the previous problem segment (a) to produce dependency diagrams that are in 3NF. (*Hint*: You might have to create a few new attributes. Also make sure that the new dependency diagrams contain attributes that meet proper design criteria; that is, make sure that there are no multivalued attributes, that the naming conventions are met, and so on.)
3. Using the results of the previous problem segment (b), draw the Crow’s Foot ERD.

**Prac 6**

**[Task 1]**

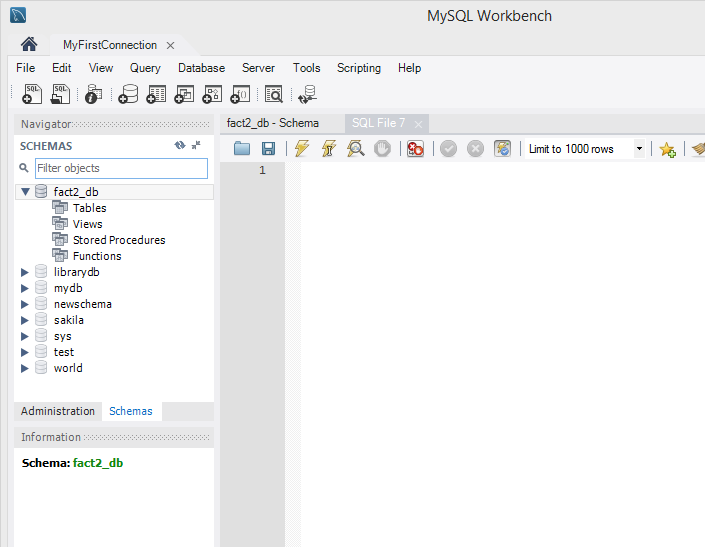
**Create a database using SQL commands on MySQL Workbench**

We can use SQL to create database and table structures and to perform basic data management chores (insert, update, and delete). Follow the process as instructed below to create the library database using SQL commands.

1. Start the new connection on MySQL Workbench
2. On the SCHEMAS Navigator panel, put the mouse over the list of schemas and right-click on the mouse, then select “Create Schema ….” menu.
3. Rename the schema as jcxxxxxx\_prac6\_db, for example, jc111222\_prac6\_db, and click “Apply”. This will process to create a new (empty) database.
4. Check through SCHEMAS Navigator panel to find that the current list of schemas includes your new created database. (If needed, refresh the list by clicking the refresh button)
5. Double click on the new database and you will find that the database currently contains no components like tables.

Now,it is time to create table structures. Each table structure will be created using one SQL query each.

1. Open a new SQL query editor by clicking the SQL+ icon on the overhead menu bar.



1. Write the following SQL query to create the author table. After writing the query, click ‘execute’ button to execute the SQL code. (Note: you need to make sure to select the database (jcxxxxxx\_prac6\_db) by double-clicking on the database name before executing the query)

CREATE TABLE AUTHOR (

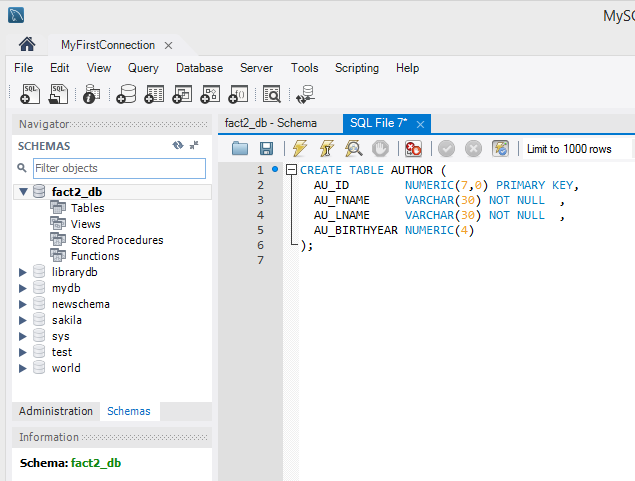
AU\_ID NUMERIC(7,0) PRIMARY KEY,

AU\_FNAME VARCHAR(30) NOT NULL ,

AU\_LNAME VARCHAR(30) NOT NULL ,

AU\_BIRTHYEAR NUMERIC(4)

);



After running this query, you will find (through the navigator panel) that the jcxxxxxx\_prac6\_db contains one new table called author. (refresh the list if needed)

(You can close the current query editor for your convenience, but it is optional to save the individual query or not. At this stage, you do not need to save, but you will need to save the query if needed for your assignment☺)

1. Create the second table patron by running the following SQL query.

CREATE TABLE PATRON (

PAT\_ID NUMERIC(10,0) PRIMARY KEY,

PAT\_FNAME VARCHAR(20) NOT NULL ,

PAT\_LNAME VARCHAR(20) NOT NULL ,

PAT\_TYPE VARCHAR(10) NOT NULL

);

1. You can apply the same way to create the other three tables by running each SQL query. However, you run a series of queries by running once. For this, you need to add BEGIN; command at the start and add COMMIT; at the end. The whole code to create three tables (book, patron, writes) at one time is shown below.

BEGIN;

CREATE TABLE PATRON (

PAT\_ID NUMERIC(10,0) PRIMARY KEY,

PAT\_FNAME VARCHAR(20) NOT NULL ,

PAT\_LNAME VARCHAR(20) NOT NULL ,

PAT\_TYPE VARCHAR(10) NOT NULL

);

CREATE TABLE BOOK (

BOOK\_NUM NUMERIC(10,0) PRIMARY KEY,

BOOK\_TITLE VARCHAR(120) NOT NULL ,

BOOK\_YEAR NUMERIC(4) ,

BOOK\_COST NUMERIC(8,2) ,

BOOK\_SUBJECT VARCHAR(120) ,

PAT\_ID NUMERIC(10),

FOREIGN KEY(PAT\_ID) REFERENCES PATRON(PAT\_ID)

);

CREATE TABLE CHECKOUT (

CHECK\_NUM NUMERIC(15) PRIMARY KEY,

BOOK\_NUM NUMERIC(10),

PAT\_ID NUMERIC(10),

CHECK\_OUT\_DATE DATE,

CHECK\_DUE\_DATE DATE,

CHECK\_IN\_DATE DATE,

FOREIGN KEY (BOOK\_NUM) REFERENCES BOOK(BOOK\_NUM),

FOREIGN KEY (PAT\_ID) REFERENCES PATRON(PAT\_ID)

);

CREATE TABLE WRITES (

BOOK\_NUM NUMERIC(10),

AU\_ID NUMERIC(7),

CONSTRAINT WRITES\_BOOK\_AU\_PK PRIMARY KEY (BOOK\_NUM, AU\_ID),

CONSTRAINT WRITES\_BOOK\_NUM\_FK FOREIGN KEY(BOOK\_NUM) REFERENCES BOOK(BOOK\_NUM),

CONSTRAINT WRITES\_AU\_ID\_FK FOREIGN KEY(AU\_ID) REFERENCES AUTHOR(AU\_ID)

);

COMMIT;

Now your database (jcxxxxxx\_prac6\_db) currently has table structures with no actual record (data). At this stage, you may import table data directly from external files as learned in the Prac 4. For this task, let’s try to use SQL code to insert each data into the existing table one by one (or by series of SQL codes as a whole).

1. Create a new query to insert the first record of author table.

INSERT INTO AUTHOR VALUES (185, 'Benson', 'Reeves', 1990);

1. To insert all other records, you need to create and run multiple number of SQL queries. For your convenience, MySQL codes for creating this database are provided. Please download the file ‘Prac6\_Mysql.txt’ and use the code appropriately for your usage. (Hint: you can run the multiple queries at one time using ‘begin;’ and ‘commit;’ command)
2. Fully check (through SCHEMAS navigator) that all tables and data records are correctly imported.
3. Save the database as a self-contained file (jcxxxxxx\_prac6.sql) so that you can open the database anywhere when needed in the future. (Refer to previous pracs (Prac 3 or 4) to learn how to back up your database).
4. Submit (or show your prac tutor) this exported dump file (jcxxxxxx\_prac6.sql) to be marked off this week’s lab activity

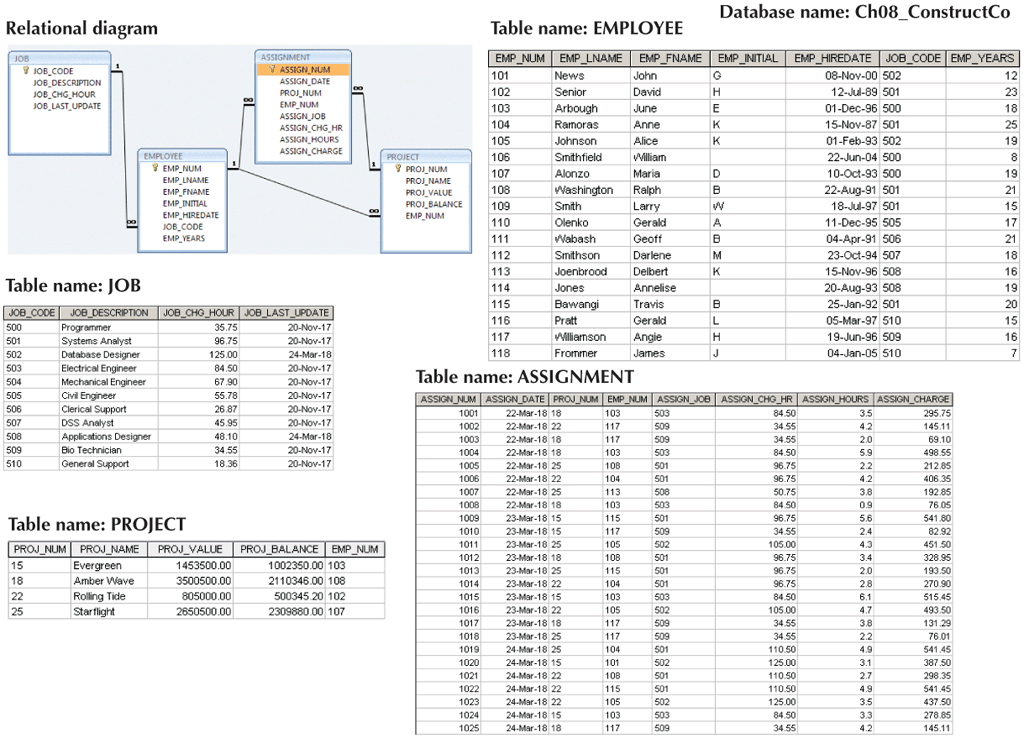
-------------------------------------------------------------------------------------------------------------------------------

**[Task 2]**

Source: Coronel-Morris textbook (13th edition) **Chapter 8 Problems Q1 ~ Q15**

1. Create a database named jcxxxxxx\_ConstructCo\_db on MySQL Workbench. Use the SQL command scripts provided (Ch08\_ConstructCo\_MySQL.txt).
2. Apply ‘Reverse Engineering’ process to create an ERD for this database. Save the ERD as a file named jcxxxxxx\_ConstructCo.mwb.

If you created the database successfully, you will see that the structure and contents of the ConstructCo database are as shown in the following figure.



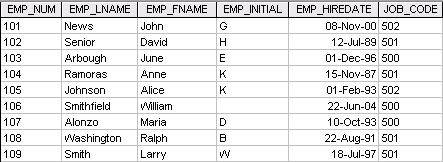
The Ch07\_ConstructCo database stores data for a consulting company that tracks all charges to projects. The charges are based on the hours each employee works on each project.

Note that the ASSIGNMENT table stores the JOB\_CHG\_HOUR values as an attribute (ASSIGN\_CHG\_HR) to maintain historical accuracy of the data. The JOB\_CHG\_HOUR values are likely to change over time. In fact, a JOB\_CHG\_HOUR change will be reflected in the ASSIGNMENT table. And, naturally, the employee primary job assignment might change, so the ASSIGN\_JOB is also stored. Because those attributes are required to maintain the historical accuracy of the data, they are not redundant.

Given the structure and contents of the ConstructCo database, use SQL commands for the following problems (1)~(15). (Save each SQL command as a separate file or copy/paste to one WORD document)

1. Write the SQL code that will create the table structure for a table named EMP\_1. This table is a subset of the EMPLOYEE table. The basic EMP\_1 table structure is summarized in the table below. Use EMP\_NUM as the primary key. Note that the JOB\_CODE is the FK to JOB so be certain to enforce referential integrity. Your code should also prevent null entries in EMP\_LNAME and EMP\_FNAME.
2. Having created the table structure in Problem (1), write the SQL code to enter the first two rows for the table shown as below. Each row should be inserted individually, without using a subquery. Insert the rows in the order that they are listed in the table as shown.

[The contents of the EMP\_1 table]



1. Using the EMPLOYEE table that already exists, use a subquery to insert the remaining rows from the EMPLOYEE table into the EMP\_1 table. Remember, your subquery should only retrieve the columns needed for the EMP\_1 table and only the employees shown in the figure.
2. Write the SQL code that will save the changes made to the EMP\_1 table.
3. Write the SQL code to change the job code to 501 for the person whose employee number (EMP\_NUM) is 107.
4. Write the SQL code to delete the row for the person named William Smithfield, who was hired on June 22, 2004, and whose job code classification is 500. (Hint: Use logical operators to include all of the information given in this problem. Remember, if you are using MySQL, will have to first disable “safe mode”. You can set up this under Edit>Preferences>SQL Editor>unclick “Safe Updates”. Please be cautious that this setting requires restart of your server and reconnection thus you will need to save your database firstly in particular if you use Workbench in a lab computer.)
5. Write the SQL code to create a copy of EMP\_1, including all of its data, and naming the copy EMP\_2.
6. Using the EMP\_2 table, write the SQL code that will add the attributes EMP\_PCT and PROJ\_NUM to EMP\_2. The EMP\_PCT is the bonus percentage to be paid to each employee. The new attribute characteristics are:

EMP\_PCTNUMBER(4,2)

PROJ\_NUMCHAR(3)

(Note: If your SQL implementation allows it, you may use DECIMAL(4,2) or NUMERIC(4,2) rather than NUMBER(4,2). Use DECIMAL(4,2) for MySQL Workbench.)

1. Using the EMP\_2 table, write the SQL code to change the EMP\_PCT value to 3.85 for the person whose employee number (EMP\_NUM) is 103.
2. Using the EMP\_2 table, write a single SQL command to change the EMP\_PCT values to 5.00 for the people with employee numbers 101, 105, and 107.
3. Using the EMP\_2 table, write a single SQL command to change the EMP\_PCT values to 10.00 for all employees who do not currently have a value for EMP\_PCT.
4. Using the EMP\_2 table, write a single SQL command to add 0.15 to the EMP\_PCT of the employee whose name is Maria D. Alonzo. (Use the employee name in your command to determine the correct employee.)
5. Using a single command sequence with the EMP\_2 table, write the SQL code that will change the project number (PROJ\_NUM) to 18 for all employees whose job classification (JOB\_CODE) is 500.
6. Using a single command sequence with the EMP\_2 table, write the SQL code that will change the project number (PROJ\_NUM) to 25 for all employees whose job classification (JOB\_CODE) is 502 or higher.
7. Write the SQL code that will change the PROJ\_NUM to 14 for employees who were hired before January 1, 1994, and whose job code is at least 501.

When you finish Problems 7 ~ 15, the EMP\_2 table will contain the data shown in the following table.

****

**Prac 7**

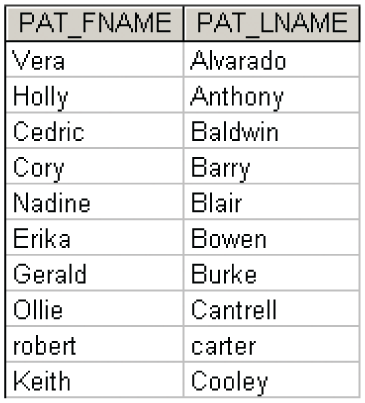
**Exercises:**

For each of these exercises, a figure of the correct output is provided. If the output of the query is very large, only the first several rows of the output are shown.

Save a query for each question as the name of “Q1.sql”, “Q2.sql”, … You are required to submit these files (as a zipped folder) to be marked off.

Note that some questions are provided with the accompanied solution to help your learning.

1. Write a query that displays the first and last name of every patron sorted by last name and then first name. Ensure the sort is case insensitive. (See the figure below for first part of the output. The actual result will have 50 rows)



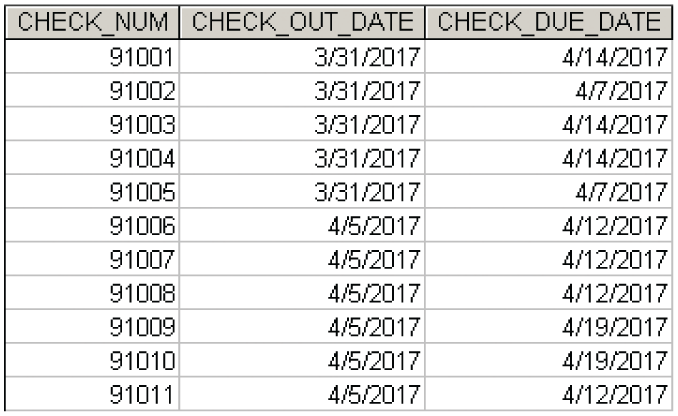
Answer provided:

SELECT BOOK\_TITLE, BOOK\_COST, BOOK\_YEAR

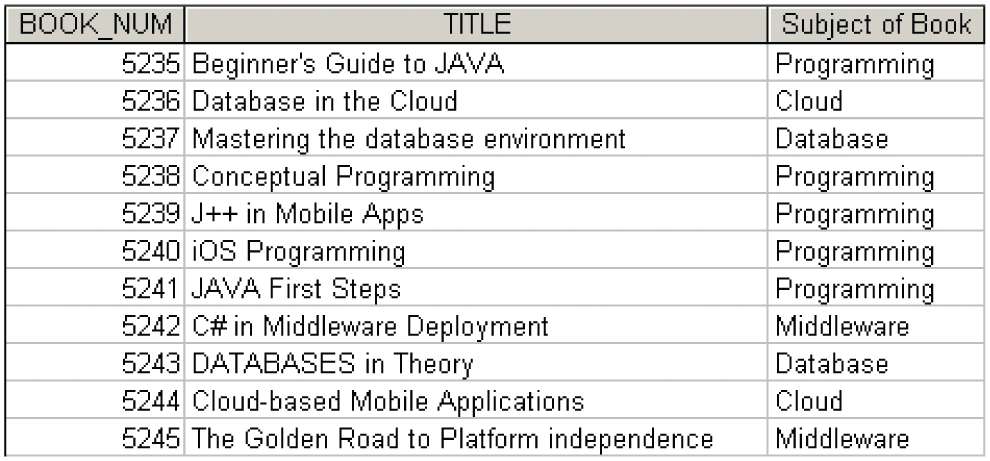
FROM BOOK

ORDER BY BOOK\_TITLE;

1. Write a query to display the checkout number, check out date, and due date for every book that has been checked out sorted by checkout number. (See the figure below for first part of the output. The actual result will have 68 rows)



1. Write a query to display the book number, book title, and subject for every book sorted by book number (See the figure below for the output. The actual result will have 20 rows)



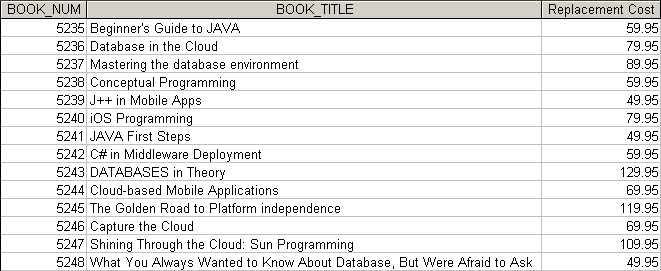
Answer provided:

SELECT BOOK\_NUM, BOOK\_TITLE AS TITLE, BOOK\_SUBJECT AS "Subject of Book"

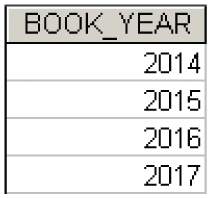
FROM BOOK

ORDER BY BOOK\_NUM;

1. Write a query to display the book number, title, and cost of each book sorted by book number (See the figure below for the output).



1. Write a query to display the different years in which books have been published in. Include each year only once and sort the results by year (See the figure below for the output).



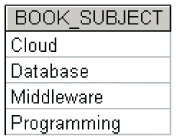
Answer provided:

SELECT DISTINCT BOOK\_YEAR

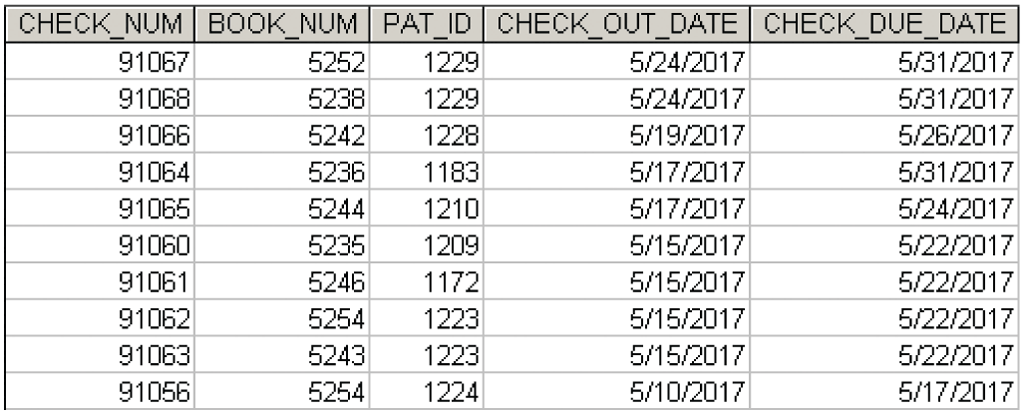
FROM BOOK

ORDER BY BOOK\_YEAR;

1. Write a query to display the different subjects on which this library has books. Include each subject only once and sort the results by subject (See the figure below for the output).



1. Write a query to display the checkout number, book number, patron ID, checkout date, and due date for every checkout that has ever occurred in the system. Sort the results by checkout date in descending order and then by checkout number in ascending order (See the figure below for first part of the output. The actual result will have 68 rows)



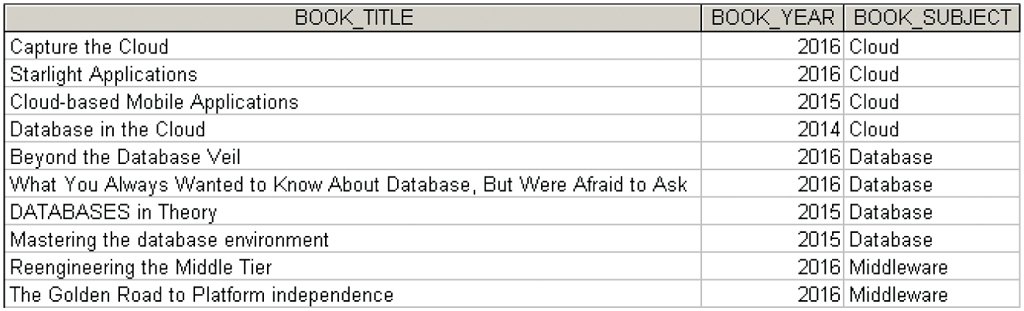
Answer provided:

SELECT CHECK\_NUM, BOOK\_NUM, PAT\_ID, CHECK\_OUT\_DATE, CHECK\_DUE\_DATE

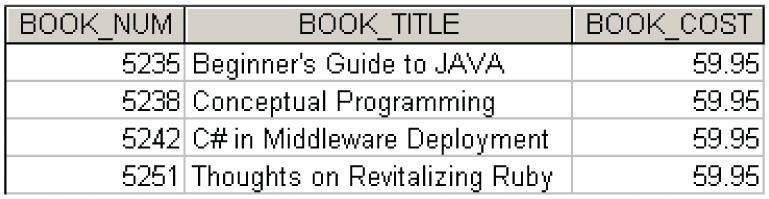
FROM CHECKOUT

ORDER BY CHECK\_OUT\_DATE DESC, CHECK\_NUM;

1. Write a query to display the book title, year, and subject for every book. Sort the results by book subject in ascending order, year in descending order, and then title in ascending order (See the figure below for the output. The actual result will have 20 rows).



1. Write a query to display the book number, title, and cost for all books that cost $59.95 sorted by book number (See the figure below for the output).



Answer provided:

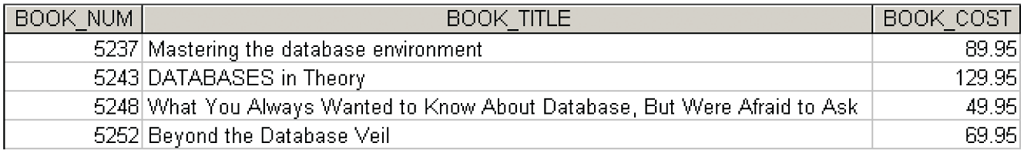
SELECT BOOK\_NUM, BOOK\_TITLE, BOOK\_COST

FROM BOOK

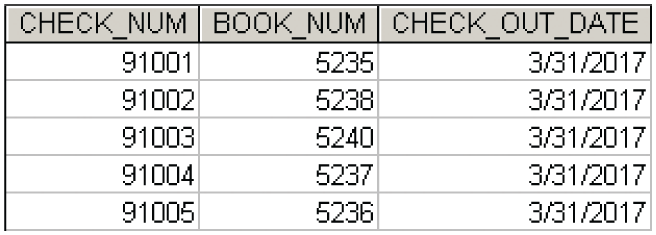
WHERE BOOK\_COST = 59.95

ORDER BY BOOK\_NUM;

1. Write a query to display the book number, title, and replacement cost for all books in the “Database” subject sorted by book number (See the figure below for the output).



1. Write a query to display the checkout number, book number, and checkout date of all books checked out before April 5, 2017 sorted by checkout number (See the figure below for the output).



Answer provided:

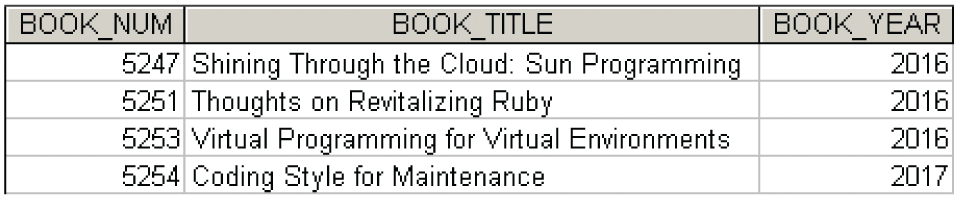
SELECT CHECK\_NUM, BOOK\_NUM, CHECK\_OUT\_DATE

FROM checkout

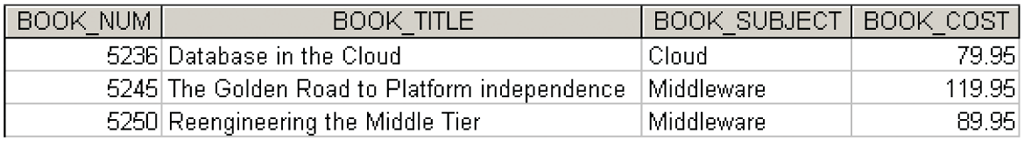
WHERE CHECK\_OUT\_DATE < '2017-04-05'

ORDER BY CHECK\_NUM;

1. Write a query to display the book number, title, and year of all books published after 2015 and on the “Programming” subject sorted by book number (See the figure below for the output).



1. Write a query to display the book number, title, subject, and cost for all books that are on the subjects of “Middleware” or “Cloud,” and that cost more than $70 sorted by book number (See the figure below for the output).



Answer provided:

SELECT BOOK\_NUM, BOOK\_TITLE, BOOK\_SUBJECT, BOOK\_COST

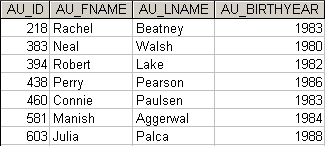
FROM BOOK

WHERE (BOOK\_SUBJECT = 'Middleware' OR BOOK\_SUBJECT = 'Cloud')

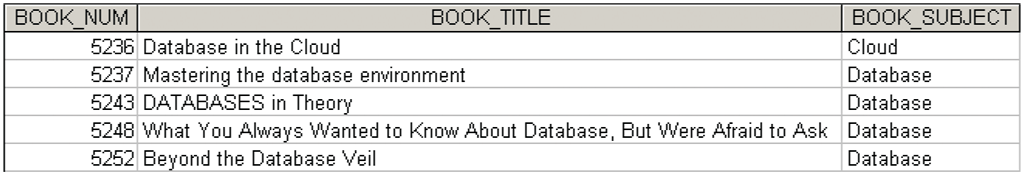
AND BOOK\_COST > 70

ORDER BY BOOK\_NUM;

1. Write a query to display the author ID, first name, last name, and year of birth for all authors born in the decade of the 1980s sorted by author ID (See the figure below for the output).



1. Write a query to display the book number, title, and subject for all books that contain the word “Database” in the title, regardless of how it is capitalized. Sort the results by book number (See the figure below for the output).



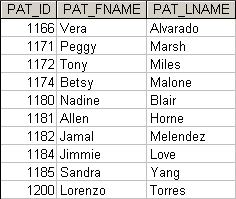
Answer provided:

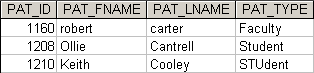
SELECT BOOK\_NUM, BOOK\_TITLE, BOOK\_SUBJECT

FROM BOOK

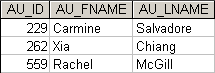
WHERE Upper(BOOK\_TITLE) LIKE '%DATABASE%'

ORDER BY BOOK\_NUM;

1. Write a query to display the patron ID, first and last name of all patrons who are students, sorted by patron ID (See the figure below for first part of the output. The actual result will have 44 rows)
2. Write a query to display the patron ID, first and last name, and patron type for all patrons whose last name begins with the letter “C”, sorted by patron ID (See the figure below for the output).



1. Write a query to display the author ID, first and last name of all authors whose year of birth is unknown. Sort the results by author ID (See the figure below for the output).



Answer provided:

SELECT AU\_ID, AU\_FNAME, AU\_LNAME

FROM AUTHOR

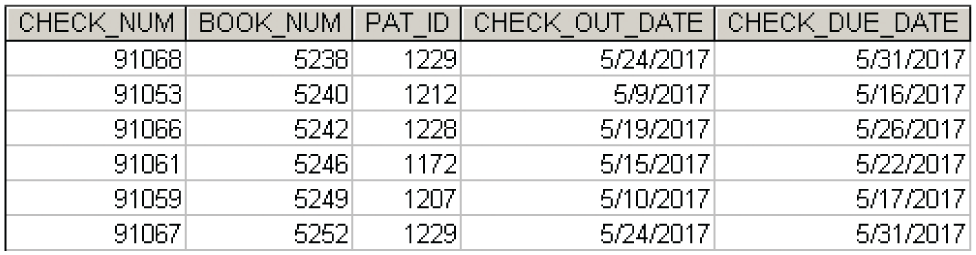
WHERE AU\_BIRTHYEAR IS NULL

ORDER BY AU\_ID;

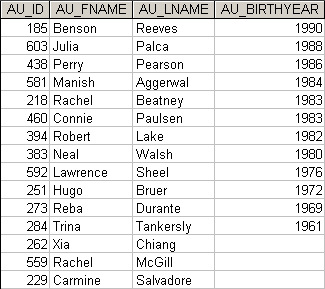
1. Write a query to display the author ID, first and last name of all authors whose year of birth is known. Ensure the results are sorted by author ID (See the figure below for the output).



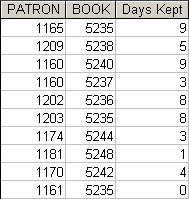
1. Write a query to display the checkout number, book number, patron ID, check out date, and due date for all checkouts that have not yet been returned. Sort the results by book number (See the figure below for the output).



1. Write a query to display the author ID, first name, last name, and year of birth for all authors. Sort the results in descending order by year of birth, and then in ascending order by last name (See the figure below for the output). (Note that some DBMS sort NULLs as being large and some DBMS sort NULLs as being small.)



1. Write a query to display the patron ID, book number, and days kept for each checkout. “Days Kept” is the difference from the date on which the book is returned to the date it was checked out. Sort the results by days kept in descending order, then by patron ID, and then by book number. (See the figure below for the output. The actual result will have 68 rows)

****

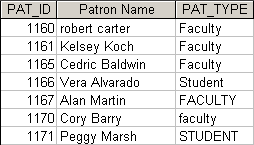
Answer provided:

SELECT PAT\_ID AS PATRON, BOOK\_NUM AS BOOK, datediff(CHECK\_IN\_DATE, CHECK\_OUT\_DATE) AS "Days Kept"

FROM CHECKOUT

ORDER BY datediff(CHECK\_IN\_DATE, CHECK\_OUT\_DATE) DESC, PAT\_ID, BOOK\_NUM;

1. Write a query to display the patron ID, patron full name, and patron type for each patron, sorted by patron ID (See the figure below for the output. The actual result will have 50 rows)

****

Answer provided:

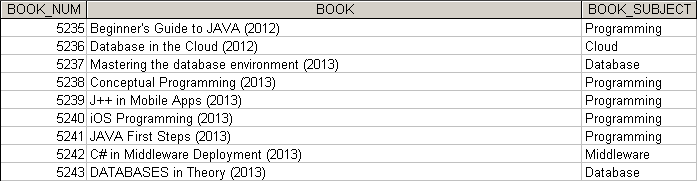
SELECT PAT\_ID, CONCAT(PAT\_FNAME, ' ', PAT\_LNAME) AS "Patron Name", PAT\_TYPE

FROM PATRON

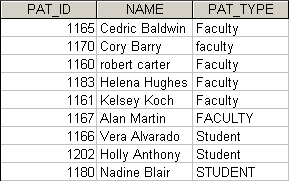
ORDER BY PAT\_ID;

Note: CONCAT is a special function compatible with MySQL which can be used to concatenate multiple columns data into one. If you use any other DBMS, you need to refer to the manual of the DBMS for further information about the specific functions they provide for concatenating two columns data into one.

1. Write a query to display the book number, title with year, and subject for each book. Sort the results by the book number (See the figure below for the output. The actual result will have 20 rows)

****

1. Write a query to display the patron ID, full name (first and last), and patron type for all patrons. Sort the results by patron type and then by last name and first name. Ensure that all sorting is case insensitive. (See the figure below for the output. The actual result will have 50 rows)

****