Assignment 3

Textbook Exercises 6.7, 6.8, 6.10, 6.12, 6.13

**6.7**

Consider the following Hotel, Room, Booking and Guest schemas in a DBMS. The hotelNo is the primary key for Hotel table and roomNo is the primary key for the Room relation. Booking stores the details of room reservations and bookingNo is the primary key. Guest stores the guests details and guestNo is the primary key.

Hotel (hotelNo, hotelName, hotelType, hotelAddress, hotelCity, numRoom)

Room (roomNo, hotelNo, roomPrice)

Booking (bookingNo, hotelNo, guestNo, checkIn, checkout, totelGuest, roomNo)

Guest (guestNo, firstName, lastName, guestAddress)

1. Write the SQL to list full details of all the hotels.
2. Write the SQL to list full details of all the hotels in New York.
3. Write the SQL to list the guests in New York in descending order by last name.

Please use the exercise\_67.sql DDL to complete this exercise. You will need to write a join between booking, hotels, and guests. Your book specifies how to do on page 222, however, this does not work in PostgreSQL. Below is the format you will need to use, please replace all <> to answer the question. Please submit the query and the query results.

SELECT a.\*

FROM <relation> a

JOIN <relation> b

ON <alias.column> = <alias.column>

JOIN < alias.column > c

ON b.< alias.column > = c.< alias.column >

WHERE c.< alias.column > = '<value>';

**ANSWERS:**

-- Question 1

SELECT \* FROM hotels.hotel;

A screenshot of a computer

Description automatically generated

-- Question 2

SELECT \* FROM hotels.hotel WHERE hotel.hotelCity='New York';

A screenshot of a computer

Description automatically generated

-- Question 3

SELECT \*

FROM hotels.hotel

JOIN hotels.booking

ON hotel.hotelNo = booking.hotelNo

JOIN hotels.guest

ON booking.guestNo = guest.guestNo

WHERE hotel.hotelCity = 'New York'

ORDER BY lastname DESC;

A screenshot of a computer

Description automatically generated

**6.8**

Write appropriate SQL DDL statements for declaring the LIBRARY relational database schema of Figure 6.6. Specify the keys and referential triggered actions.A diagram of a library

Description automatically generated

Write the schema create statement along with the relation create statements. You may insert data, but this is optional. Please submit your SQL DDL (schema & tables).

**ANSWER:**

CREATE SCHEMA IF NOT EXISTS lib;

CREATE TABLE IF NOT EXISTS lib.publisher (

Publisher\_name VARCHAR NOT NULL,

Address VARCHAR,

Phone VARCHAR,

PRIMARY KEY (Publisher\_name)

);

CREATE TABLE IF NOT EXISTS lib.borrower (

Card\_no INT NOT NULL,

Borrower\_name VARCHAR NOT NULL,

Address VARCHAR NOT NULL,

Phone VARCHAR NOT NULL,

PRIMARY KEY (Card\_no)

);

CREATE TABLE IF NOT EXISTS lib.library\_branch (

Branch\_id INT NOT NULL,

Branch\_name VARCHAR,

Address VARCHAR,

PRIMARY KEY (Branch\_id)

);

CREATE TABLE IF NOT EXISTS lib.book (

Book\_id INT NOT NULL,

Title VARCHAR,

Publisher\_name VARCHAR,

PRIMARY KEY (Book\_id),

FOREIGN KEY (Publisher\_name) REFERENCES lib.publisher(Publisher\_name) ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS lib.book\_authors (

Book\_id INT NOT NULL,

Author\_name VARCHAR NOT NULL,

FOREIGN KEY (Book\_id) REFERENCES lib.book(Book\_id) ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS lib.book\_copies (

Book\_id INT NOT NULL,

Branch\_id INT NOT NULL,

No\_of\_copies INT,

PRIMARY KEY (Book\_id, Branch\_id),

FOREIGN KEY (Book\_id) REFERENCES lib.book(Book\_id) ON UPDATE CASCADE,

FOREIGN KEY (Branch\_id) REFERENCES lib.library\_branch(Branch\_id) ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS lib.book\_loans (

Book\_id INT NOT NULL,

Branch\_id INT NOT NULL,

Card\_no INT NOT NULL,

Date\_out DATE NOT NULL,

Due\_date DATE NOT NULL,

FOREIGN KEY (Book\_id) REFERENCES lib.book(Book\_id) ON UPDATE CASCADE,

FOREIGN KEY (Branch\_id) REFERENCES lib.library\_branch(Branch\_id) ON UPDATE CASCADE,

FOREIGN KEY (Card\_no) REFERENCES lib.borrower(Card\_no)

);

**6.10**

Specify the following queries in SQL on the COMPANY relational database schema shown in Figure 5.5. Show the result of each query if it is applied to the COMPANY database in Figure 5.6. You will need to create the INSERT statements to match the data in figure 5.5 (page 191 & 192)

A screenshot of a computer

Description automatically generated

A close-up of a document

Description automatically generated

1. Retrieve the names of all employees in department 5 who earns more than 3000 and works on ProductZ project.

**ANSWER:**

---Part A

SELECT

employee.Fname

,employee.Minit

,employee.Lname

FROM company.employee

JOIN company.works\_on ON employee.Ssn=works\_on.Essn

JOIN company.project ON works\_on.Pno=project.Pnumber

WHERE employee.Salary>=3000 AND employee.Dno=5 AND project.Pname='ProductX'

;

A screenshot of a computer

Description automatically generated

1. List the names of all employees who are from Houston, Texas and works under manager 333445555.

**ANSWER:**

---Part B

SELECT

employee.Fname

,employee.Minit

,employee.Lname

FROM company.employee

WHERE employee.Address LIKE '%Houston%' AND employee.Super\_ssn=333445555

;

1. Find the names of all employees who are working in the project Computerization.

**ANSWER:**

---Part C

SELECT

employee.Fname

,employee.Minit

,employee.Lname

FROM company.employee

JOIN company.works\_on ON employee.Ssn=works\_on.Essn

JOIN company.project ON works\_on.Pno=project.Pnumber

WHERE project.Pname='Computerization'

;

A screenshot of a computer

Description automatically generated

Please submit your DDL (schema and tables), queries, and query results.

**DDL Statements:**

**6.12**

Specify the following queries in SQL on the database schema of Figure 1.2. (page 38)

A close-up of a document

Description automatically generated

1. Retrieve the course names of all the courses that comes under the department of ‘cs’ (computer science).

**ANSWER:**

--- Part A

SELECT Course\_name

FROM university.course

WHERE Department='CS';

A screenshot of a computer

Description automatically generated

1. Retrieve the names of all courses along with the name of the instructor taught during the fall of 2008.

**ANSWER:**

---Part B

SELECT course.Course\_name, section.Instructor

FROM university.course

JOIN university.section

ON course.Course\_number=section.Course\_number

WHERE section.Semester='Fall' AND section.Section\_year='08'

ORDER BY Course\_name;

A screenshot of a computer

Description automatically generated

1. For each section taught by Professor Anderson, retrieve the course number, semester, year, and number of students who took the section.

**ANSWER:**

---Part C

SELECT

section.Course\_number

,section.Semester

,section.Section\_year

,COUNT(\*)

FROM university.section

JOIN university.grade\_report

ON section.Section\_identifier=grade\_report.Section\_identifier

WHERE section.Instructor='Anderson'

GROUP BY section.Course\_number,section.Semester,section.Section\_year

;

A screenshot of a computer

Description automatically generated

1. Retrieve the name and transcript of each junior student (Class = 1) majoring in mathematics (MATH). A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

**ANSWER:**

---Part D

SELECT

student.Student\_name

,course.Course\_name

,section.Course\_number

,section.Semester

,section.Section\_year

,grade\_report.Grade

FROM university.student

JOIN university.grade\_report ON student.Student\_number=grade\_report.Student\_number

JOIN university.section ON section.Section\_identifier=grade\_report.Section\_identifier

JOIN university.course ON course.Course\_number=section.Course\_number

WHERE student.Student\_class=1 AND student.Major='MATH';

No students in the database major in MATH, so query results in an empty table.

**DDL Statements:**

You MUST WRITE THE DDL and INSERT statements to create this schema and tables. Please submit your queries AND results. The DDL is needed for the final question.

ALTER TABLE IF EXISTS university.section DROP CONSTRAINT IF EXISTS section\_course\_number\_fkey;

ALTER TABLE IF EXISTS university.grade\_report DROP CONSTRAINT IF EXISTS grade\_report\_student\_number\_fkey;

ALTER TABLE IF EXISTS university.grade\_report DROP CONSTRAINT IF EXISTS grade\_report\_section\_identifier\_fkey;

ALTER TABLE IF EXISTS university.prerequisite DROP CONSTRAINT IF EXISTS grade\_report\_course\_number\_fkey;

ALTER TABLE IF EXISTS university.prerequisite DROP CONSTRAINT IF EXISTS grade\_report\_prequisite\_number\_fkey;

DROP TABLE IF EXISTS university.student;

DROP TABLE IF EXISTS university.course;

DROP TABLE IF EXISTS university.section;

DROP TABLE IF EXISTS university.grade\_report;

DROP TABLE IF EXISTS university.prerequisite;

DROP SCHEMA IF EXISTS university;

CREATE SCHEMA IF NOT EXISTS university;

CREATE TABLE IF NOT EXISTS university.course (

Course\_name VARCHAR NOT NULL UNIQUE,

Course\_number VARCHAR NOT NULL,

Credit\_hours INT NOT NULL,

Department VARCHAR NOT NULL,

PRIMARY KEY (COurse\_number)

);

CREATE TABLE IF NOT EXISTS university.student (

Student\_name VARCHAR NOT NULL,

Student\_number INT NOT NULL,

Student\_class INT,

Major VARCHAR,

PRIMARY KEY (Student\_number)

);

CREATE TABLE IF NOT EXISTS university.section (

Section\_identifier INT NOT NULL,

Course\_number VARCHAR NOT NULL,

Semester VARCHAR NOT NULL,

Section\_year VARCHAR NOT NULL,

Instructor VARCHAR,

PRIMARY KEY (Section\_identifier),

FOREIGN KEY (Course\_number) REFERENCES university.course(Course\_number) ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS university.grade\_report (

Student\_number INT NOT NULL,

Section\_identifier INT NOT NULL,

Grade VARCHAR,

FOREIGN KEY (Student\_number) REFERENCES university.student(Student\_number) ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY (Section\_identifier) REFERENCES university.section(Section\_identifier) ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS university.prerequisite (

Course\_number VARCHAR NOT NULL,

Prequisite\_number VARCHAR,

FOREIGN KEY (Course\_number) REFERENCES university.course(Course\_number) ON UPDATE CASCADE ON DELETE CASCADE,

FOREIGN KEY (Prequisite\_number) REFERENCES university.course(Course\_number) ON UPDATE CASCADE ON DELETE CASCADE

);

INSERT INTO university.student

VALUES

('Smith',17,1,'CS'),

('Brown',8,2,'CS');

INSERT INTO university.course

VALUES

('Intro to Computer Science','CS1310',4,'CS'),

('Data Structures','CS3320',4,'CS'),

('Discrete Mathematics','MATH2410',3,'MATH'),

('Database','CS3380',3,'CS');

INSERT INTO university.section

VALUES

(85,'MATH2410','Fall','07','King'),

(92,'CS1310','Fall','07','Anderson'),

(102,'CS3320','Spring','08','Knuth'),

(112,'MATH2410','Fall','08','Chang'),

(119,'CS1310','Fall','08','Anderson'),

(135,'CS3380','Fall','08','Stone');

INSERT INTO university.grade\_report

VALUES

(17,112,'B'),

(17,119,'C'),

(8,85,'A'),

(8,92,'A'),

(8,102,'B'),

(8,135,'A');

INSERT INTO university.prerequisite

VALUES

('CS3380','CS3320'),

('CS3380','MATH2410'),

('CS3320','CS1310');

**6.13**

Write SQL update statements to do the following on the database schema shown in Figure 1.2.

1. Insert a new course, <’Financial Accounting’, ‘fac4390’,5,’BUSINESS’>

Insert into course (course\_name,course\_number,credit\_hours,department)

1. Insert a new section, <145, ‘fac4390’, ‘Fall’, ‘17’, ‘Hanif’>
2. Insert a new student, <’Robin’, 34, 2, ‘BUSINESS’>.
3. Update the record for the student whose student number is 17 and change his class from 1 to 3.

You only need to submit the SQL for creating the INSERT and UPDATE statements.

**ANSWERS:**

---Part A

INSERT INTO university.course (Course\_name,Course\_number,Credit\_hours,Department)

VALUES

('Financial Accounting','FAC4390',5,'BUSINESS');

---Part B

INSERT INTO university.section

VALUES

(145,'FAC4390','Fall','17','Hanif');

---Part C

INSERT INTO university.student

VALUES

('Robin',34,2,'BUSINESS');

---Part D

UPDATE university.student

SET Student\_class = 3

WHERE Student\_number = 17;