CHAPTER 7 REVIEW

Karla N.

CSE 3303-003

11/1/2023

Specify the following queries in SQL on the database schema of Figure 1.2.

a. Retrieve the names of all senior students majoring in ‘cs’ (computer science).

```
SELECT S.Name
FROM STUDENT AS S
WHERE S.MAJOR=’CS’ AND S.CLASS=4;
```

b. Retrieve the names of all courses taught by Professor King in 2007 and 2008.

```
SELECT C.Course_Name
FORM COURSE AS C
INNER JOIN SECTION ON C.Course_number=SECTION.Course_number
WHERE SECTION.Course_number=’King’ AND (SECTION.year = ’07 OR
SECTION.year =’08');
```

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

```
SELECT SC.course_number, SC.semester, SC.year,
COUNT(G.Student_number) As num_students

FROM SECTION AS SC
INNER JOIN GRADE_REPORT AS G ON SC.section_id = G.section_id
WHERE SC.instructors = ‘King’;
```

d. Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

```
SELECT S.name, C.course_name, C.course_number, C.credit_hours,
SEC.semester, SEC.year, GR.grade

FROM STUDENT AS S
JOIN GRADE_REPORT AS GR
ON S.student_number = GR.Student_number
```

```
JOIN SECTION SEC
ON GR.Section_identifier = SEC.section_identifier
JOIN COURSE C
ON SEC.course_number = C.course_number
WHERE S.class = 4 AND S.major = 'CS';
```

e. Retrieve the names and major departments of all straight-A students (students who have a grade of A in all their courses).

```
SELECT  S.Name, S.Major
FROM STUDENT AS S
WHERE NOT EXISTS ( SELECT *
FROM GRADE_REPORT AS GR
WHERE GR.Student_Number= S.Student_number AND
NOT(GR.Grade='A'));
```

f. Retrieve the names and major departments of all students who do not have a grade of A in any of their courses.

```
SELECT  S.Name, S.Major
FROM STUDENT AS S
WHERE NOT EXISTS ( SELECT *
FROM GRADE_REPORT AS GR
WHERE GR.Student_Number= S.Student_number AND (GR.Grade='A'));
```

2. Specify the following queries on the database in Figure 5.5 in SQL.
   a. For each department whose average employee salary is more than $30,000, retrieve the department name and the number of employees working for that department.

```
SELECT D.Dname, COUNT(E.Dno) AS NumOfEmployees
FROM DEPARTMENT AS D
JOIN EMPLOYEE AS E ON D.Dnumber=E.Dno
WHERE E.salary>30000;
```

   b. Suppose that we want the number of male employees in each department making more than $30,000, rather than all employees. Can we specify this query in SQL? Why or why not?

c. Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees

```
SELECT E.Fname, MAX(E.salary) AS highest_salary
FROM EMPLOYEE AS E
JOIN DEPARTMENT AS D ON E.Dno=D.Dnumber
GROUP BY E.Fname;
```

d. Retrieve the names of all employees whose supervisor's supervisor has '888665555' for Ssn.

```
SELECT E.Fname
FROM EMPLOYEE AS E
WHERE E.super_ssn= ‘888665555’;
```

e. Retrieve the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

```
SELECT E.Fname, MIN(E.Salary) AS min_salary
FROM EMPLOYEE AS E
WHERE E.salary = (min_salary+=10000);
```

f. Find the average salary for employees in each department.

```
SELECT AVG(E.salary)

FROM EMPLOYEE AS E

JOIN DEPARTMENT AS D ON E.Dno=D.Dnumber;
```

3. Specify the following queries in SQL on the database schema of Figure 6.6.
   a. Retrieve the most popular books in the library

```
SELECT B.title, COUNT(*) AS NumOfCheckouts

FROM BOOK AS B

JOIN BOOK_LOANS AS BL ON B.Book_id=BL.Book_id

GROUP BY B.Title

ORDER BY NumOfCheckouts DESC
```

b. List branch addresses that house the book titled ‘Don Quixote’

```
SELECT L.Address
FROM LIBRARY BRANCH L
JOIN BOOK COPIES BC ON L.Branch_id = BC.Branch_id
JOIN BOOK B ON BC.Book_id = B.Book_id
WHERE B.Title = 'Don Quixote';
```

c. Find all borrowers who have checked books authored by: 'JK Rowling'

```
SELECT DISTINCT BR.Name
FROM BORROWER BR
JOIN BOOK LOANS BL ON BR.Card_no = BL.Card_no
JOIN BOOK B ON BL.Book_id = B.Book_id
JOIN BOOK AUTHORS BA ON B.Book_id = BA.Book_id
WHERE BA.Author_name = 'JK Rowling';
```

d. Retrieve the number of books checked out by a particular borrower: 'Hughie Prim'.

```
SELECT BR.Name, COUNT(*) AS NumOfCheckouts
FROM BORROWER BR
JOIN BOOK LOANS BL ON BR.Card_no = BL.Card_no
WHERE BR.Name = 'Hughie Prim';
```

e. Retrieve the total number of checkouts for each borrower.

```
SELECT BR.Name, COUNT(*) AS NumOfCheckouts

FROM BORROWER BR

JOIN BOOK LOANS BL ON BR.Card_no = BL.Card_no;
```

f. Find the book that had the minimum number of checkouts.

```
SELECT B.Title, COUNT(BL.Book_id) AS CheckoutCount
FROM BOOK B
JOIN BOOK_LOANS BL ON B.Book_id = BL.Book_id
GROUP BY B.Book_id, B.Title
ORDER BY CheckoutCount
LIMIT 1;
```

4. Consider the following view, DEPT_SUMMARY, defined on the COMPANY database in Figure 5.6:

```
CREATE VIEW  DEPT_SUMMARY ( D, C, TOTAL_S, AVERAGE_S)
        AS SELECT       DNO, COUNT(*), SUM(SALARY), AVG(SALARY)
        FROM            EMPLOYEE
        GROUP BY        DNO;
```

State which of the following queries and updates would be allowed on the view. If a query or update would be allowed, show what the corresponding query or update on the base relations would look like, and give its result when applied to the database in Figure 5.6.

a. SELECT *

FROM DEPT_SUMMARY;

This query would be allowed and the result would be the all the data the DEPT_SUMMARY has.

5	4	133000	33250
4	3	93000	31000
1	1	55000	55000

b. SELECT D, C

FROM DEPT_SUMMARY

WHERE TOTAL_S > 100000;

This query would be allowed

5	4

c. SELECT D, AVG_S

FROM DEPT_SUMMARY

WHERE C > (SELECT C FROM DEPT_SUMMARY WHERE D = 4);

5	33250

d. UPDATE DEPT_SUMMARY

SET D=3

WHERE D= 4;

5	4	133000	33250
3	3	93000	31000
1	1	55000	55000

e. DELETE FROM DEPT_SUMMARY

WHERE C > 4;

4	3	93000	31000
1	1	55000	55000