EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

1. ***Specify the following queries based on the Nested Queries concepts on the database above in SQL. Show the query results.***
    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.***

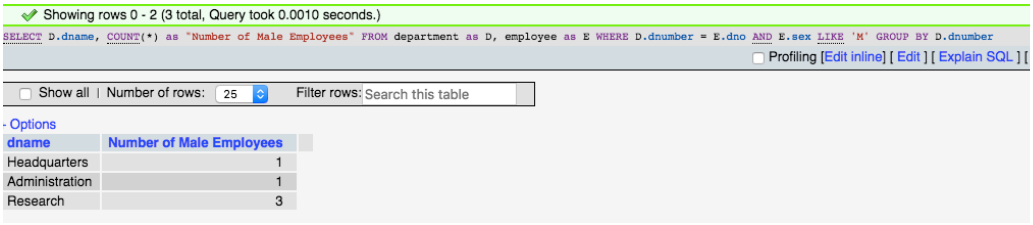
```
SELECT D.dname, COUNT(*) as "Number of Employees"
FROM department as D, employee as E
WHERE D.dnumber = E.dno
GROUP BY D.dnumber
HAVING AVG(E.salary) > 30000
```

    b. ***Suppose we want the number of male employees in each department rather than all employees. Can we specify this query in SQL? Why or why not? Show the***

query results if applied to the database.

Yes, we can specify such query in database as we have employees' gender stored in 'sex' attribute of 'employee' table.

```
SELECT D.dname, COUNT(*) as "Number of Male Employees"
FROM department as D, employee as E
WHERE D.dnumber = E.dno
  AND E.sex LIKE 'M'
GROUP BY D.dnumber
```



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

```
SELECT CONCAT(fname, " ", minit, " ", lname) AS "Name"
FROM employee
WHERE dno IN (
    SELECT dno
    FROM employee
    WHERE salary = (
          SELECT MAX(salary)
          FROM employee
    )
)
```

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

```
SELECT CONCAT(fname, " ", minit, " ", lname) AS "Name"
FROM employee
WHERE super_ssn IN (
    SELECT ssn
    FROM employee
    WHERE 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 CONCAT(fname, " ", minit, " ", lname) AS "Name"
FROM employee
WHERE salary > 10000 + (
    SELECT MIN(salary) FROM employee
    )
```

2. ***Specify the following views in SQL on the COMPANY database schema shown in the Figure above.***
   a. ***A view that has the department name, manager name, and manager salary for every department.***

```
CREATE VIEW department_info(Dept_Name, Mgr_Name, Mgr_Salary) AS
SELECT D.dname,
       concat(E.fname, " ", E.minit, " ", E.lname),
       E.salary
FROM department AS D, employee as E
WHERE D.dnumber = E.dno
  AND D.mgr_ssn = E.ssn
GROUP BY D.dnumber
```

   b. ***A view that has the employee name, supervisor name, and employee salary for each employee who works in the 'Research' department.***

```
CREATE VIEW research_dept_info(Emp_Name, Mgr_Name, Emp_Salary) AS
SELECT CONCAT(E.fname, " ", E.minit, " ", E.lname),
       CONCAT(M.fname, " ", M.minit, " ", M.lname),
       E.salary
FROM employee E
LEFT OUTER JOIN employee AS M ON M.ssn = E.super_ssn
LEFT OUTER JOIN department AS D ON D.dnumber = E.dno
WHERE D.dname = "Research"
```

   c. ***A view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project.***

```
CREATE VIEW timesheet(Proj_Name, Dept, Employees, hours_per_week) AS
SELECT P.project_name, D.dname, COUNT(*), SUM(W.hours)
FROM works_on AS W
LEFT OUTER JOIN employee AS E ON W.w_ssn = E.ssn
LEFT OUTER JOIN project AS P ON W.w_project_id = P.project_id
LEFT OUTER JOIN department AS D ON P.department_id = D.dnumber
GROUP BY P.project_id
```

   d. ***A view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it.***

```
CREATE VIEW timesheet_2(Proj_Name, Dept, Employees, hours_per_week) AS
SELECT P.project_name, D.dname, COUNT(*), SUM(W.hours)
FROM works_on AS W
LEFT OUTER JOIN employee AS E ON W.w_ssn = E.ssn
LEFT OUTER JOIN project AS P ON W.w_project_id = P.project_id
LEFT OUTER JOIN department AS D ON P.department_id = D.dnumber
GROUP BY P.project_id HAVING COUNT(*) > 1
```

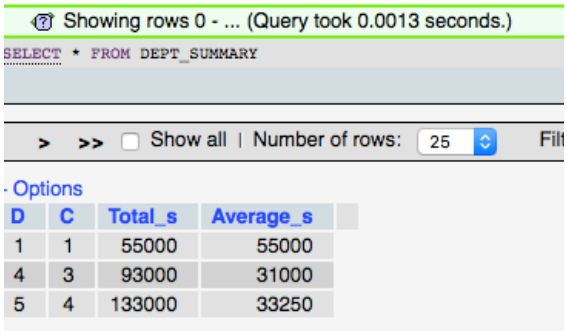
3. ***Consider the following view, DEPT_SUMMARY, defined on the COMPANY database:***

```
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 and give its result.

a. SELECT *
FROM DEPT_SUMMARY;

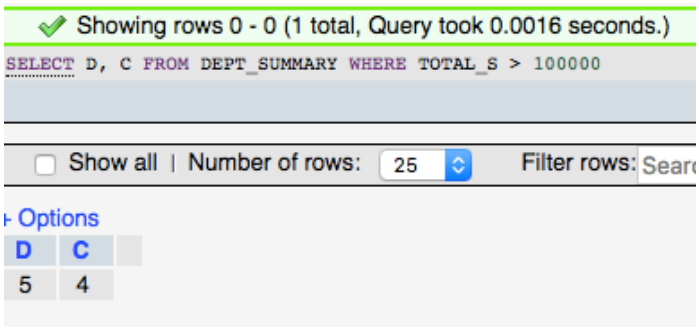
The following query is allowed on the view. The result is given below:



D	C	Total_s	Average_s
1	1	55000	55000
4	3	93000	31000
5	4	133000	33250

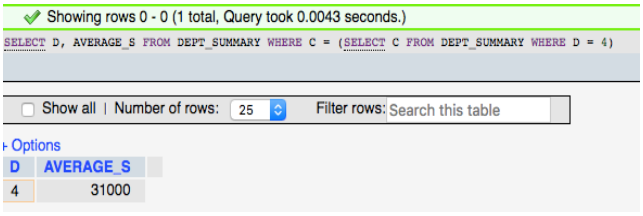
b. SELECT D, C
FROM DEPT_SUMMARY
WHERE TOTAL_S > 100000

The following query is allowed on the view. The result is given below:



D	C
5	4

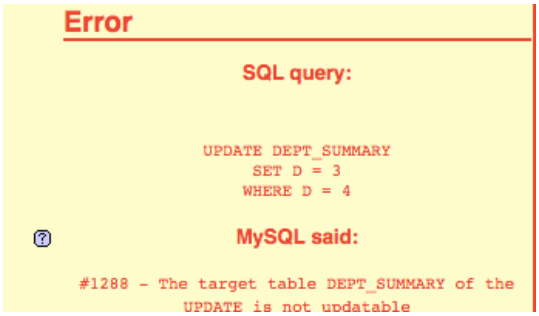
c. SELECT D, AVERAGE_S
FROM DEPT_SUMMARY
WHERE C > (SELECT C FROM DEPT_SUMMARY WHERE D = 4);
The following query is allowed on the view. The result is given below:



D	AVERAGE_S
4	31000

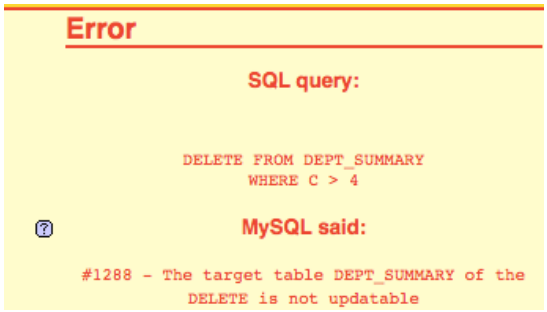
d. UPDATE DEPT_SUMMARY
SET D = 3
WHERE D = 4;

The following UPDATE query is not allowed on the view.



e. DELETE FROM DEPT_SUMMARY
WHERE C > 4;

The following DELETE query is not allowed on the view.



4. Specify the following queries in SQL on the (University) database schema in Figure below

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

a. 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, GRADE_REPORT AS GR
WHERE S.Student_number = GR.Student_number
```

```
  AND GR.Grade LIKE 'A'
GROUP BY S.Student_number
HAVING COUNT(*) = (
  SELECT COUNT(*)
  FROM GRADE_REPORT AS GR2
  WHERE GR2.Student_number = S.Student_number
)
```

b. 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, GRADE_REPORT AS GR
WHERE S.Student_number = GR.Student_number
  AND GR.Grade NOT LIKE 'A'
GROUP BY S.Student_number
HAVING COUNT(*) = (
  SELECT COUNT(*)
  FROM GRADE_REPORT AS GR2
  WHERE GR2.Student_number = S.Student_number
)
```

5. What are Triggers? Give an example of situation where triggers can be used?

It can defined as rules set in the database which are activated based on an update to the database. They result in performing some defined operation on same or other tables or even performing of notifying by sending messages and so on. It can also be termed as a active listener which performs actions based on specific inputs given.

For example, if we use a server management database where real time server performance is being recorded for every interval of time. If the input entry reads that a specific server is running at a CPU Utilization of 80%, we could use TRIGGER to alert the owner based on the same. We could also push this data along with current applications running on the server to an ALERT table.