# Tutorial - 1

Question 1: Let us design a DB for the following requirements:

- A highly popular '**McM Sangeet company**' plans to organize a competition for searching the right talents to train them for creating music albums. The company advertises their recruitment requirements on different channels (both print and digital media) for inviting talents by submitting their personal information, prior experience, and a 2-5 minutes media file, which is of either an audio file (for songs) or a video (for songs and/or playing music instruments), to a given URL in the advertisement. The company writes in the advertisement that the shortlisted candidate will be informed by Phone and/or Email.
- A candidate can submit more than one entry and has an option to provide more than one phone number in the submission, which is unique to him/her.
- After the closing date of entry submission, a panel evaluates all entries and recommends a set of candidates to invite them for the next round. The company maintains the information about the panellist (personal information, industry experience, association with the McM company). A candidate can see the outcome of their application online.
- These shortlisted candidates are then invited to perform a 'Live' show in Mumbai and finally top 'n' candidates are selected in each album category (i.e. audio and video). These 2n candidates are called as the members of McM-2020.
- Different music groups (pop, classic, leisure, evergreen, …) are formed to create the music albums (audio/video). Every member belongs to one or more music groups which is moderated by a director who himself/herself is a member of the group. Each member has a different role to play in each album.
- Once the album is created and approved by McM Director, its trailer is released online for limited time to the outsiders to give their comments (like and dislike), and all this recorded in the database (album name, date of release, number of visits, number of likes/dislikes, …).
- The McM decides the price of the album after analysing the data collected from its trailer release, and then the album is released to distributors who eventually will sell it online. Note, each distributor may be charged a different price per unit depending on the negotiation between the McM and distributor.
- When a distributor sells an album, the download request comes to the McM site. Thus, the McM company maintains the record for each download (Incoming URL – identifying the distributor, Album#, Date, Download Status – success/failure) so that they can track the number of downloads for raising the invoice.

Design and draw an ER diagram. List your assumptions, if any. We will accept all reasonable assumptions. Your E-R diagram should clearly –

- Identify all entities and their attributes
- Identify relationships between these entities and their attributes
- Identify primary key of each entity
- Identify entity relationship participation and constraints

Question 2: Convert your E-R diagram of Question 1 into relational schemas.

# <u>Tutorial-2</u>

Let us design a DB for the following requirements:

- A highly popular 'McM Sangeet company' plans to organize a competition for searching the right talents to train them for creating music albums. The company advertises their recruitment requirements on different channels (both print and digital media) for inviting talents by submitting their personal information, prior experience, and a 2-5 minutes media file, which is of either an audio file (for songs) or a video (for songs and/or playing music instruments), to a given URL in the advertisement. The company writes in the advertisement that the shortlisted candidate will be informed by Phone and/or Email.
- A candidate can submit more than one entry and has an option to provide more than one phone number in the submission, which is unique to him/her.
- After the closing date of entry submission, a panel evaluates all entries and recommends a set of candidates to invite them for the next round. The company maintains the information about the panellist (personal information, industry experience, association with the McM company). A candidate can see the outcome of their application online.
- These shortlisted candidates are then invited to perform a 'Live' show in Mumbai and finally top '*n*' candidates are selected in each album category (i.e. audio and video). These *2n* candidates are called as the members of McM-2020.
- Different music groups (pop, classic, leisure, evergreen, …) are formed to create the music albums (audio/video). Every member belongs to one or more music groups which is moderated by a director who himself/herself is a member of the group. Each member has a different role to play in each album.
- Once the album is created and approved by McM Director, its trailer is released online for limited time to the outsiders to give their comments (like and dislike), and all this recorded in the database (album name, date of release, number of visits, number of likes/dislikes, …).
- The McM decides the price of the album after analysing the data collected from its trailer release, and then the album is released to distributors who eventually will sell it online. Note, each distributor may be charged a different price per unit depending on the negotiation between the McM and distributor.
- When a distributor sells an album, the download request comes to the McM site. Thus, the McM company maintains the record for each download (Incoming URL – identifying the distributor, Album#, Date, Download Status – success/failure) so that they can track the number of downloads for raising the invoice.

Q1. Write the Relational Algebraic Expression (RAE) for each of the following query on relational schemas that was discussed in Tutorial-1.

A. List all 'Audio' albums released in 2020.

B. List all participants who have submitted both Audio and Video files.

C. List all members who have been the member of more than one group

D. List all members of ‘Pop’ music group who are not part of any other music group.

E. List all distributors who sold all types of albums.

Q2. Write the SQL statement on the same relational schemas for the following.

A. List all ‘Audio’ albums released in 2020.

B. List all members who have been the member of more than one group.

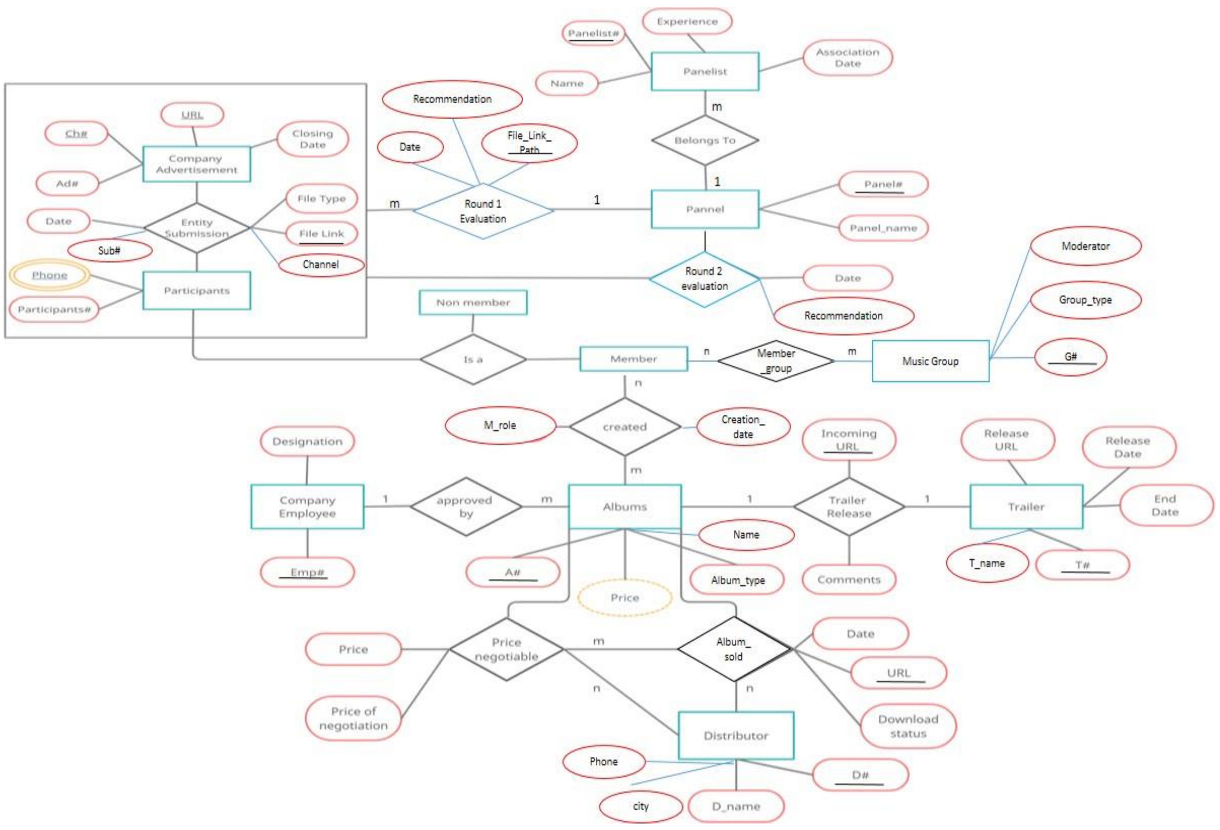
C. List all members of ‘Pop’ music group who are not part of any other music group.

D. List all participants who have submitted both Audio and Video files.

E. The McM company would like to analyze the data for “Which advertisement channel has been effective in attracting the maximum number of entry submissions?”

# Solution for Tutorial-1 and Tutorial-2

## Question1



**Design and draw an ER diagram.** List your assumptions, if any. We will accept all reasonable assumptions. Your E-R diagram should clearly --

- Identify all entities and their attributes
- Identify relationships between these entities and their attributes
- Identify and underline the primary key of each entity
- Identify entity relationship participation and constraints

**Entities-**

**Company_Advt** (Advt#, Channel, URL, Starting_Date, Closing_Date)

**Participants** (Participant#, Participant_Name, {Phone#}, City, State

Note: Phone is multi-valued

**Panellist** (Pannelist#, Name, Years_Experience, Joining_Date, Panel#)

**Panel** (Panel#, Panel_Name)

**Company_Employee** (E#, Desgination)

**Albums** (A#, Name, App_E#, Album_type, Price)

**Music_Group** (Group#, Name, Group_type, Moderator#)

**Trailer** (T#, T_Name, R-URL, R-Date, E-Date)

**Distributor** (D#, D_Name, Phone, City)

- Channel and media file can be weak entity.

**Relationships with attributes**

**Entry_Submission** (Participant#, URL, Channel, File_Type, Submission_Date, File_Link_Path, S#)

**Round1_evaluation** (Panel#, File_Link_Path, Date, Recommendation)

**Round2_evaluation** (Panel#, Member#, Recommendation, Date)

**Album_creation** (A#, M#, M_Role, Creation_Date)

**Tariler_Release** (A#, T#, Incoming_URL, Comments)

**Album_sold** (A#, D#, URL, Date)

**Member_Group** (M#, Group#)

**Price_negotiable**(A#, D#, Negotiated_Price, Date)

**Question 2:** Convert your E-R diagram of Question 1 into relational schemas.

**Company_Advt** (Advt#, Channel, URL, Starting_Date, Closing_Date)
**Participants** (Participant#, Participant_Name, {Phone#}, City, State)

**Entry_Submission** (Participant#, URL, Channel, File_Type, Submission_Date, File_Link_Path, S#)

**Panellist** (Pannelist#, Name, Years_Experience, Joining_Date, Panel#)

**Panel** (Panel#, Panel_Name)

**Round1_evaluation** (Panel#, File_Link_Path, Date, Recommendation)

**Round2_evaluation** (Panel#, Member#, Recommendation, Date)

**Company_Employee** (E#, Desgination)

**Albums** (A#, Name, App_E#, Album_type, Price)

**Album_creation** (A#, M#, M_Role, Creation_Date)

**Music_Group** (Group#, Name, Group_type, Moderator#)

**Member_Group** (M#, Group#)

**Trailer** (T#, T_Name, R-URL, R-Date, E-Date)

**Tariler_Release** (A#, T#, Incoming_URL, Comments)

**Distributor** (D#, D_Name, Phone, City)

**Album_sold** (A#, D#, URL, Date, Download status)

**Price_negotialble**(A#, D#, Negotiated_Price, Date)

**Question 3:** Write the Relational Algebraic Expression (RAE) for each of the following query on relational schemas of Question 2.

A. List all 'Audio' albums released in 2020.

$\sigma_{album_{type}={}^{F}Audio^{F}\text{A}\ creation\_dat(year)=2020}$(Album)

B. List all participants who have submitted both Audio and Video files.

$\pi_{Participants\#}(\sigma_{FileType=Audio\ \ \text{A}\ \ FileType=Video}$ (Entry_submission))

C. List all members who have been the member of more than one group

$\pi_{memberName}$ (Member_Group - ($\pi_{memberI}$(Member_Group)))

D. List all members of 'Pop' music group who are not part of any other music group.

$\pi_{memberID}$ ($\sigma_{Group\_Type="pop"}$(Member_Group $\bowtie$ Music_group)) $-$

$\pi_{memberID}$ ($\sigma_{Group\_Type\neq"pop"}$(Member_Group $\bowtie$ Music_group))

E. List all distributors who sold all types of albums

$\pi_{d\#}(\sigma_{AlbumType=Audio}$ (Album_sold $\bowtie$ Album)) A
$\pi_{d\#}(\sigma_{AlbumType=vedio}$ (Album_sold $\bowtie$Album))

**Question 4:** Write the SQL statement on relational schemas of Question 2.

A. List all ‘Audio’ albums released in 2020.

```
SELECT Album#
    FROM Album
    WHERE albumType =Audio AND creation_date(year) =2020;
```

B. List all members who have been the member of more than one group.

```
SELECT M#
    FROM Member_Group
    WHERE ( SELECT Distinct member as m
                    FROM Music_Group
                    WHERE Member_Group.m = m# AND
                            GROUP By group_name)>1;
```

C. List all members of ‘Pop’ music group who are not part of any other music group.

```
SELECT DISTINCT Member_Group.m#
    FROM Member_Group, Music_group
    WHERE Music_group.G# = Member_Group.G# AND
            Music_group.Group_type = ‘POP’ AND
            NOT EXIST IN
            (SELECT M#
            FROM Member_Group
            WHERE Music_group.Group_type != ‘POP’);
```

D. List all participants who have submitted both Audio and Video files.

```
SELECT participant#


    FROM Entry_submission
    WHERE fileType =Audio AND fileType =video;
```

E. The McM company would like to analyze the data for “Which advertisement channel has been effective that attracted maximum number of entry submissions?”

```
SELECT channel, COUNT(*)
    FROM Entry_submission
    GROUP BY channel;
```

# Tutorial-3

**Exercise 1:** Consider the following schema: the schema describes a database containing information about parts supplied by suppliers. The underlined attributes denote the primary keys of the relations. Domain of each field is listed after the field name. The Catalog relation lists the prices charged for parts by Suppliers.

Suppliers(sid: integer, sname: string, address: string)
Parts(pid: integer, pname: string, color: string)
Catalog(sid: integer, pid: integer, cost: real)

Write the following queries in SQL.

Q1. Find the names of suppliers who supply some red part.

```
A1. SELECT S.sname

        FROM Suppliers S, Parts P, Catalog C

        WHERE P.color='red' AND C.pid=P.pid AND C.sid=S.sid
```

Q2. Find the sids of suppliers who supply some red or green part.

```
A2.  SELECT C.sid

       FROM Catalog C, Parts P

        WHERE (P.color = 'red' OR P.color = 'green') AND P.pid = C.pid
```

Q3. Find the sids of suppliers who supply some red part or are at 221 Packer Street.

```
A3.      SELECT S.sid

        FROM Suppliers S

                WHERE S.address = '221
                Packer street' OR S.sid IN (
                SELECT C.sid
```

```
                          FROM Parts P, Catalog C

                          WHERE P.color='red' AND P.pid = C.pid )
```

Q4. Find the sids of suppliers who supply some red part and some green part.

A4.
```
    SELECT C.sid

     FROM Parts P, Catalog C

          WHERE P.color = 'red' AND
          P.pid = C.pid AND EXISTS (
          SELECT P2.pid

                          FROM Parts P2, Catalog C2

                              WHERE P2.color = 'green' AND C2.sid =
                              C.sid AND P2.pid = C2.pid)
```

Q5. Find the sids of suppliers who supply every part.

A5.
```
    SELECT C.sid

     FROM Catalog C

     WHERE NOT EXISTS (  SELECT P.pid
                        FROM Parts P
                          WHERE NOT EXISTS (   SELECT C1.sid
                                                FROM Catalog C1
                                                WHERE C1.sid =
                                                C.sid AND C1.pid =
                                                P.pid))
```

Q6. Find the sids of suppliers who supply every red part.

A6.
```
     SELECT C.sid

     FROM Catalog C

     WHERE NOT EXISTS (   SELECT P.pid
                                    FROM Parts P
                                    WHERE P.color =
                                    'red'

                                    AND (NOT EXISTS ( SELECT C1.sid

                                                     FROM Catalog C1
                                                     WHERE C1.sid =
                                                     C.sid AND C1.pid =
                                                     P.pid)))
```

Q7. Find the sids of suppliers who supply every red or green part.

```
A7.  SELECT C.sid

       FROM Catalog C

       WHERE NOT EXISTS (  SELECT P.pid
                          FROM Parts P
                            WHERE (P.color = 'red' OR P.color =
                            'green') AND (NOT EXISTS ( SELECT
                            C1.sid
                                                     FROM Catalog C1
                                                     WHERE C1.sid =
                                                     C.sid AND C1.pid =
                                                     P.pid)))
```

Q8. Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid.

```
A8.    SELECT C1.sid, C2.sid

       FROM Catalog C1, Catalog
       C2 WHERE C1.pid = C2.pid

              AND C1.cost > C2.cost
```

Q9. Find the pids of parts supplied by at least two different suppliers.

```
A9.   SELECT C.pid

      FROM Catalog C

      WHERE EXISTS ( SELECT C1.sid
                     FROM Catalog C1
                     WHERE C1.pid = C.pid AND C1.sid != C.sid )
```

****************************

**Exercise 2:** The following relations keep track of airline flight information:

Flights(flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: real)

Aircraft(aid: integer, aname: string, cruisingrange: integer) Certified(eid: integer, aid: integer)

Employees(eid: integer, ename: string, salary: integer)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly. Write each of the following queries in SQL.

Q1. Find the names of aircraft such that all pilots certified to operate them have salaries more than $80,000.

A1.

```
SELECT DISTINCT A.aname

FROM Aircraft A

WHERE A.Aid IN (SELECT C.aid
                FROM Certified C, Employees
                E WHERE C.eid = E.eid AND
                NOT EXISTS ( SELECT *

                        FROM Employees E1

                        WHERE E1.eid = E.eid AND E1.salary < 80000 ))
```

Q2. For each pilot who is certified for more than three aircraft, find the eid and the maximum *cruisingrange* of the aircraft for which she or he is certified.

A2.

```
 SELECT C.eid, MAX
 (A.cruisingrange) FROM Certified C,
 Aircraft A

WHERE C.aid =
A.aid GROUP BY
C.eid


HAVING COUNT (*) > 3
```

Q3. Find the names of pilots whose salary is less than the price of the cheapest route from Los Angeles to Honolulu.

A3.

```
SELECT DISTINCT E.ename

FROM Employees E

WHERE E.salary < ( SELECT MIN (F.price)

            FROM Flights F
```

```
                    WHERE F.from = 'Los Angeles' AND F.to = 'Honolulu' )
```

Q4. For all aircraft with *cruisingrange* over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

A4.

Observe that aid is the key for Aircraft, but the question asks for aircraft names; we deal with this complication by using an intermediate relation Temp:

```
SELECT Temp.name, Temp.AvgSalary

    FROM ( SELECT A.aid, A.aname AS name, AVG (E.salary) AS
    AvgSalary FROM Aircraft A, Certified C, Employees E
    WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruisingrange >
    1000 GROUP BY A.aid, A.aname ) AS Temp
```

Q4. Find the names of pilots certified for some Boeing

aircraft.

A5.

```
SELECT DISTINCT E.ename
FROM Employees E, Certified C, Aircraft A
WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname LIKE 'Boeing%'
```

Q6. Find the aids of all aircraft that can be used on routes from Los Angeles to Chicago.

A6.

```
SELECT A.aid

FROM Aircraft A

WHERE A.cruisingrange > ( SELECT MIN (F.distance)

                          FROM Flights F
                          WHERE F.from = 'Los Angeles' AND F.to = 'Chicago' )
```

Q7. Identify the routes that can be piloted by every pilot who makes more than

$100,000.

A7.

```
SELECT DISTINCT F.from, F.to

FROM Flights F

WHERE NOT EXISTS ( SELECT *

                    FROM Employees E
                    WHERE E.salary > 100000
                    AND NOT EXISTS (SELECT *

                                FROM Aircraft A, Certified C

                                WHERE A.cruisingrange > F.distance
                                AND E.eid = C.eid AND A.aid =
                                C.aid) )
```

Q8. Print the enames of pilots who can operate planes with *cruisingrange* greater than 3000 miles but are not certified on any Boeing aircraft.

A8.

```
SELECT DISTINCT E.ename

FROM Employees E

WHERE E.eid IN ( ( SELECT C.eid

                    FROM Certified C
                    WHERE EXISTS ( SELECT A.aid
                                     FROM Aircraft A
                                     WHERE A.aid =
                                     C.aid
AND A.cruisingrange > 3000 )

AND NOT EXISTS ( SELECT A1.aid
FROM Aircraft A1
WHERE A1.aid =
C.aid

AND A1.aname LIKE ‘Boeing%’ ))
```

Q9. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

A9.

```
SELECT
F.departs FROM
Flights F

      WHERE F.flno IN ( ( SELECT
                          F0.flno
                  FROM Flights F0
                    WHERE F0.from = ‘Madison’
                    AND F0.to = ‘New York’ AND F0.arrives <
                    ‘18:00’ ) UNION ( SELECT F0.flno
                                  FROM Flights F0, Flights
                                  F1 WHERE F0.from =
                                  ‘Madison’ AND F0.to <>
                                  ‘New York’ AND F0.to =
                                  F1.from
                                  AND F1.to = ‘New York’
                                  AND F1.departs >
                                  F0.arrives AND F1.arrives
                                  < ‘18:00’ ) UNION (
                                  SELECT F0.flno
FROM Flights F0, Flights F1, Flights
F2 WHERE F0.from = ‘Madison’
AND F0.to = F1.from
AND F1.to = F2.from
AND F2.to = ‘New
York’ AND F0.to <>
‘New York’ AND F1.to
<> ‘New York’
AND      F1.departs      >
F0.arrives               AND
F2.departs   >   F1.arrives
AND F2.arrives < ‘18:00’ ))
```

Q10. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).

A10.

```
SELECT Temp1.avg - Temp2.avg
FROM (SELECT AVG (E.salary) AS
avg

        FROM Employees E

        WHERE E.eid IN (SELECT DISTINCT C.eid
        FROM Certified C )) AS Temp1, (SELECT AVG
        (E1.salary) AS avg

        FROM Employees E1 ) AS Temp2
```

Q11. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.

A11.

```
 SELECT E.ename,
 E.salary FROM
 Employees E

WHERE E.eid NOT IN ( SELECT DISTINCT C.eid FROM Certified C )

AND E.salary > ( SELECT AVG (E1.salary)

               FROM Employees E1
               WHERE E1.eid IN ( SELECT DISTINCT C1.eid FROM Certified C1 ) )
```

Q12. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.

A12.

```
SELECT E.ename

 FROM Employees E, Certified C, Aircraft
 A WHERE C.aid = A.aid AND E.eid =
 C.eid GROUP BY E.eid, E.ename

HAVING EVERY (A.cruisingrange > 1000)
```

Q13. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircrafts.

A13.

```
SELECT E.ename

 FROM Employees E, Certified C, Aircraft
 A WHERE C.aid = A.aid AND E.eid =
 C.eid GROUP BY E.eid, E.ename

HAVING EVERY (A.cruisingrange > 1000) AND COUNT (*) > 1
```

Q14. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.

A14.

```
SELECT E.ename

 FROM Employees E, Certified C, Aircraft
 A WHERE C.aid = A.aid AND E.eid =
 C.eid GROUP BY E.eid, E.ename

HAVING EVERY (A.cruisingrange > 1000) AND ANY (A.aname = ’Boeing’)
```

**************************************************

**Exercise 3**:- Consider the following schema. The underlined attributes denote the primary keys of the relations.

EMP ( <u>eno: integer</u>, ename:string, sal:integer, dno:integer, mgr:integer )

DEPT ( <u>dno:integer</u>, dname:string )

Q1) Fetch all employees whose salary is greater than a’s salary.

```
A1) Select eno, ename, sal from

        EMP where sal > (Select sal from EMP

                              Where ename = ‘a’) ;
```

Q2) Fetch all employees who are drawing more salary than ANY ONE in dept 102.

A2) Select eno, ename, sal, dno from

EMP where dno != 102 AND

Sal > **ANY** (Select sal from

EMP where dno = 102 );

Q3) Fetch all employess who are drawing more salary than ALL employees in dept 102.

```
 A3) Select eno, ename, sal, dno from
          EMP where dno != 102 AND
          Sal > ALL (Select sal from
                        EMP where dno = 102 );
```

**(OR)**

```
          Select eno, ename, sal, dno from
          EMP where dno != 102 AND
          Sal >  (Select max(sal) from
                        EMP where dno = 102 );
```

Q4) Fetch all employees who are drawing the same salary as employees in dept 102.

```
A4) Select eno, ename, sal, dno from
          EMP where dno != 102 AND
          Sal  IN (Select sal from
                        EMP where dno = 102 );
```

Q6) Fetch the department that do not have any employee.

```
A6) Select * from
          DEPT where dno NOT IN ( Select
                    Distinct dno from EMP);
```

**********************************

**Exercise 4**:- Consider the following schema. The underlined attributes denote the primary keys of the relations.

EMP ( <u>eno: integer</u>, ename:string, sal:integer, dno:integer, mgr:integer )

DEPT ( <u>dno:integer</u>, dname:string )

JOB_GRADES ( grade:string, low_sal:integer, high_sal:integer )

Write down following SQL queries:

Q1) Fetch employee information along with department information.

A1) Select e.eno,e.ename,e.dno,d.dname

```
from EMP e,DEPT d

where e.eno=d.dno;
```

Q2) Fetch employee information along with their grades.

A2) Select e.eno, e.ename, e.sal, j.grade

```
        From EMP e,JOB_GRADES j

        Where e.sal BETWEEN j.low_sal  AND j.high_sal;
```

**(OR)**

```
Select e.eno, e.ename, e.sal, j.grade

        From EMP e,JOB_GRADES j

        Where e.sal >= j.low_sal  AND e.sal <= j.high_sal;
```

Q3) Fetch all employee information along with their managers information.

A3) Select e.eno, e.ename, m.mgr, m.ename as mgr_name

```
From EMP e, EMP m

Where e.mgr = m.eno;
```

**Constraints/Views:**

**Exercise 1:**
Table Name: Location
Table Schema:

```
+----------------+--------------+
 | Field | Type |
+----------------+--------------+
 | LOCATION_ID | decimal(4,0) |
 | STREET_ADDRESS | varchar(40) |
 | POSTAL_CODE | varchar(12) |
 | CITY | varchar(30) |
 | STATE_PROVINCE | varchar(25) |
 | COUNTRY_ID | varchar(2) |
+----------------+--------------+
```

Table Name: Jobs
Table Schema:

```
+------------+--------------+------+
 | Field | Type | Null |
+------------+--------------+------+
 | JOB_ID | int(11) | NO | PRIMARY KEY | JOB_TITLE | varchar(35)
 | NO |
 | SALARY | decimal(6,0) | YES |
 | Allowance | decimal(6,0) | YES |
+------------+--------------+------+
```

Q1: Add a primary key for the column location_id in the Locations table. A1: ALTER TABLE locations ADD PRIMARYKEY(location_id); SHOW COLUMNS FROM locations; //To verify

Q2: Add a primary key for a combination of columns location_id and country_id. A2: ALTER TABLE locations ADD PRIMARYKEY(location_id,country_id); SHOW COLUMNS FROM locations;

//To verify

Q3: Drop the existing primary from the table locations on a combination of columns location_id and country_id.

A3: ALTER TABLE locations DROP PRIMARY KEY;

SHOW COLUMNS FROM locations; //To verify
Q4: Add a foreign key on job_id column of job_history table referencing to the primary key job_id of jobs table.

A4: ALTER TABLE locations ADD CONSTRAINT fk_job_id ADD FOREIGN KEY(job_id)REFERENCES jobs(job_id)ON DELETE CASCADE;

SHOW COLUMNS FROM locations; //To verify

Q5: Drop the existing foreign key fk_job_id from locations table which is referencing to the job_id of jobs table.

A5: ALTER TABLE locations DROP FOREIGN KEY fk_job_id;

SHOW COLUMNS FROM locations; //To verify

Q6: Display all constraints on a table“locations”

A6: Method 1:

Select COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_COLUMN_NAME, REFERENCED_TABLE_NAME from information_schema KEY_COLUMN_USAGE where TABLE_NAME = ’locations';

Method2:

SHOW CREATE TABLE locations;

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

**Exercise 2:**

Table: Salesman

```
salesman_id | name | city | commission
-------------+------------+----------+------------  5001 | James Hoog | New
York | 0.15  5002 | Nail Knite | Paris | 0.13  5005 | Pit Alex | London |
```

0.11  5006 | Mc Lyon | Paris | 0.14  5007 | Paul Adam | Rome | 0.13
5003 | Lauson Hen | San Jose | 0.12 Table: customer

customer_id | cust_name | city | grade |salesman_id
-------------+----------------+------------+-------+----------  3002 | Nick Rimando | New York | 100 | 5001  3007 | Brad Davis | New York | 200 | 5001  3005 | Graham Zusi | California | 200 | 5002
 3008 | Julian Green | London | 300 | 5002  3004 | Fabian Johnson | Paris | 300 | 5006
3009 | Geoff Cameron | Berlin | 100 | 5003  3003 | Jozy Altidor | Moscow | 200 | 5007
3001 | Brad Guzan | London | | 5005 Table: Orders
ord_nopurch_amtord_datecustomer_idsalesman_id
---------- ---------- ---------- ----------- ----------- 70001 150.5 2012-10-05 3005 5002
70009 270.65 2012-09-10 3001 5005 70002 65.26 2012-10-05 3002 5001 70004
110.5 2012-08-17 3009 5003 70007 948.5 2012-09-10 3005 5002 70005 2400.6
2012-07-27 3007 5001 70008 5760 2012-09-10 3002 5001 70010 1983.43
2012-10-10 3004 5006 70003 2480.4 2012-10-10 3009 5003 70012 250.45
2012-06-27 3008 5002 70011 75.29 2012-08-17 3003 5007 70013 3045.6
2012-04-25 3002 5001

Q1. Write a query to create a view for those salesmen belongs to the city New

York. A1. CREATE VIEW newyorkstaff AS SELECT * FROM salesman WHERE

city ='New York';

Q2. Write a query to create a view for all salesmen with columns salesman_id, name, and city.

A2. CREATE VIEW salesown AS SELECT salesman_id, name,city FROM salesman;

Q3. Write a query to create a view to getting a count of how many customers we have at each level of a grade.

A3. CREATE VIEW gradecount(grade, number) AS SELECT grade,COUNT(*) FROM customer GROUP BY grade;

Q4. Write a query to create a view to keeping track the number of customers ordering, number of salesmen attached, average amount of orders and the total amount of orders in a day.

A4. CREATE VIEW totalforday AS SELECT ord_date,COUNT(DISTINCTcustomer_id),AVG(purch_amt),SUM(purch_amt) FROM orders GROUPBY ord_date;

Q5. Write a query to create a view that shows for each order the salesman and customer by name.

A5. CREATE VIEW nameorders AS
SELECTord_no,purch_amt,a.salesman_id, name,cust_name FROM
orders a, customer b, salesman c WHERE
a.customer_id=b.customer_id AND a.salesman_id=c.salesman_id;

Q6. Write a query to create a view that finds the salesman who has the customer with the highest order of a day.

A6. CREATE VIEW elitsalesman AS SELECT b.ord_date,a.salesman_id, a.nameFROM
salesman a, orders b WHERE a.salesman_id=b.salesman_id AND
b.purch_amt=(SELECTMAX(purch_amt) FROM orders cWHERE
c.ord_date=b.ord_date);

Q7. Write a query to create a view that finds the salesman who has the customer with the highest order at least 3 times on a day.

A7. CREATE VIEW incentive AS SELECT DISTINCT salesman_id, name FROM
elitsalesman a WHERE3<=(SELECTCOUNT(*) FROM elitsalesman b
WHEREa.salesman_id=b.salesman_id);

Q8. Write a query to create a view that shows all of the customers who have the highest grade.

A8. CREATE VIEW highgrade AS SELECT * FROM customer
WHERE grade =(SELECTMAX(grade)FROM customer);

Q9. Write a query to create a view that shows the number of the salesman in each city.

A9. CREATE VIEW citynum AS SELECT city,COUNT(DISTINCTsalesman_id)
FROM salesman GROUPBY city;

Q10. Write a query to create a view that shows the average and total orders for each salesman after his or her name. (Assume all names are unique).

A10. CREATE VIEW norders AS SELECT
name,AVG(purch_amt),SUM(purch_amt)FROM salesman, orders WHERE
salesman.salesman_id=orders.salesman_id GROUPBY name;

Q11. Write a query to create a view that shows all matches of customers with salesman such that at least one customer in the city of customer served by a salesman in the city of the salesman.
A11.CREATE VIEW citymatch(custcity,salescity) AS SELECT DISTINCT a.city,b.city FROM customer a, salesman b WHERE a.salesman_id=b.salesman_id;

Q12. Write a query to create a view that shows the number of orders in each day.

A12.CREATE VIEW dateord(ord_date,odcount) AS SELECT ord_date,COUNT(*) FROM orders GROUPBY ord_date;

Q13. Write a query to create a view that finds the salesmen who issued orders on October 10th, 2019.

A13. CREATE VIEW salesmanonoct AS SELECT * FROM salesman WHERE salesman_id IN (SELECTsalesman_id FROM orders WHERE ord_date='2019-10-10');

Q14. Write a query to create a view that finds the salesmen who issued orders on either August 17th, 2019 or October 10th, 2019.

A14.CREATE VIEW sorder AS SELECT salesman_id,ord_no,customer_id FROM orders WHERE ord_date IN ('2019-08-17','2019-10-10');

## Vaccine Distribution and inventory DB Example

The GoI plans to build a DB for managing the information about procurement, distribution of vaccines and the information about vaccinated citizens. The dept procures vaccines from various vendors and then distribute to local health center for vaccinations to citizens. A citizen books an appointment in the hospital for vaccination.

To make sure a citizen does not get vaccination twice, the health center checks the vaccination record for the citizen and update the DB once the citizen is vaccinated. Each health center CMO would like to get the status of vaccine inventory and how many folks in his/her health center are covered each day. If the inventory falls below certain threshold, the CMO office will send a fresh order for next supply.......

- Citizens are identified by an Aadhar, and their names, addresses, and ages must be recorded.
- Doctors are identified by an Aadhar. For each doctor, the name, associated health center, date of joining, and years of experience must be recorded.
- Each health center is identified by unique number, address, #of doctors, CMO, CMO-Aadhar, Phone
- Each distributor i.e. vaccine supplier (pharmaceutical company) is identified by name and has a phone number.
- For each vaccine procurement, it should have Order date, Company name, #of vaccines, Price per vaccine, Received date.
- For distribution, we should record Shipped date, Health center#, Vaccine name, #of vaccines,...
- ....

Vaccine Distribution and Inventory DB

Q1 What is the average years of experience of a doctor in a health center.

A1

```
SELECT AVG(years_of_experience)
FROM Doctors
GROUP BY health_center_ID;
```

Q2 Find the total number of vaccines procured (received) by GoI from a vaccine supplier in 2021.

A2

```
SELECT company_ID, company_name, SUM(num_of_vaccines)
FROM Vaccine_Procurement
GROUP BY company_ID
HAVING DATE(received_date) BETWEEN ‘2021-01-01’ AND ‘2021-12-31’;
```

Q3 CMO of a health care would like to know the name, date of joining and years of experience of all the doctors in his/her health center. Create a view table of doctors.

A3

```
CREATE VIEW doctor_view AS
SELECT d.health_care_ID, d.name, d.date_of_joining,d.years_of_experience
FROM Doctors d, Health_Care h
WHERE d.health_care_ID = h.health_care_ID;
```

Q4 Create a view/table of citizen information with age > 65

A4

```
CREATE VIEW new_citizen AS
SELECT aadhar_number, name, address, age
FROM Citizens
WHERE Citizens.age > 65;
```

Q5 Find the names of health centers where number of doctors are greater than 10.

A5

```
SELECT DISTINCT health_center_name
FROM Health_Care
WHERE num_of_ doctors> 10;
```