JAMIA MILLIA ISLAMIA, NEW DELHI

DATABASE MANAGEMENT LAB

NAME: FAIZAN CHOUDHARY

ROLL NO: 20BCS021

SUBJECT CODE: CEN 491

SEMESTER: 4th

COURSE: B.TECH.(COMPUTER ENGG.)

DEPT: DEPT OF COMPUTER ENGG.

ASSIGNMENT 1

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
```

Figure 6.1 SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

31st January 2022

Creation:

- ```
  mysql> create database 20bcs021_faizan;
  ```
- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table Employee
      -> (
      -> f_name varchar(15) not null,
      -> m_init char,
      -> l_name varchar(15) not null,
      -> ssn char(10) not null,
      -> dob date,
      -> address varchar(30),
      -> gender char,
      -> salary decimal(10,2),
      -> super_ssn char(10),
      -> d_no int not null,
      -> primary key(ssn)
      -> );
  ```
- ```
  mysql> insert into Employee
      -> values ('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);
  ```

- mysql> select * from Employee;

```
mysql> select * from Employee;
+---------+--------+---------+-----------+------------+-------------------------+--------+----------+-----------+------+
| f_name  | m_init | l_name  | ssn       | dob        | address                 | gender | salary   | super_ssn | d_no |
+---------+--------+---------+-----------+------------+-------------------------+--------+----------+-----------+------+
| John    | B      | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M      | 30000.00 | 333445555 |    5 |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M      | 40000.00 | 888665555 |    5 |
| Joyce   | A      | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F      | 25000.00 | 333445555 |    5 |
| Ramesh  | K      | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M      | 38000.00 | 333445555 |    5 |
| James   | E      | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M      | 55000.00 | NULL      |    1 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F      | 43000.00 | 888665555 |    4 |
| Ahmad   | V      | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX  | M      | 25000.00 | 987654321 |    4 |
| Alicia  | J      | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX  | F      | 25000.00 | 987654321 |    4 |
+---------+--------+---------+-----------+------------+-------------------------+--------+----------+-----------+------+
8 rows in set (0.00 sec)
```

- mysql> create table Department
  -> (
  -> d_name varchar(15) not null,
  -> d_number int not null,
  -> mgr_ssn char(10) not null,
  -> mgr_start_date date,
  -> primary key(d_number),
  -> unique(d_name),
  -> foreign key (mgr_ssn) references employee(ssn)
  ->);
- mysql> insert into Department values
  -> ('Research', 5, '333445555', '1988-05-22'),
  -> ('Administration', 4, '987654321', '1995-01-01'),
  -> ('Headquarters', 1, '888665555', '1981-06-19');
- mysql> select * from Department;

```
mysql> select * from Department;
+----------------+----------+-----------+-----------------+
| d_name         | d_number | mgr_ssn   | mgr_start_date  |
+----------------+----------+-----------+-----------------+
| Headquarters   |        1 | 888665555 | 1981-06-19      |
| Administration |        4 | 987654321 | 1995-01-01      |
| Research       |        5 | 333445555 | 1988-05-22      |
+----------------+----------+-----------+-----------------+
3 rows in set (0.00 sec)
```

- mysql> create table Project
  -> (
  -> p_name varchar(15) not null,
  -> p_number int not null,
  -> p_location varchar(15),
  -> d_num int not null,
  -> primary key (p_number),
  -> unique (p_name),
  -> foreign key (d_num) references Department (d_number)
  ->);

- mysql> insert into Project values
  -> ('ProductX', 1, 'Bellaire', 5),
  -> ('ProductY', 2, 'Sugarland', 5),

```
    -> ('ProductZ', 3, 'Houston', 5),
    -> ('Computerization', 10, 'Strafford', 4),
    -> ('Reorganization', 20, 'Houston', 1),
    -> ('Newbenefits', 30, 'Strafford', 4);
```

- `mysql> select * from Project;`

```
mysql> select * from Project;
+-----------------+----------+------------+-------+
| p_name          | p_number | p_location | d_num |
+-----------------+----------+------------+-------+
| ProductX        |        1 | Bellaire   |     5 |
| ProductY        |        2 | Sugarland  |     5 |
| ProductZ        |        3 | Houston    |     5 |
| Computerization |       10 | Strafford  |     4 |
| Reorganization  |       20 | Houston    |     1 |
| Newbenefits     |       30 | Strafford  |     4 |
+-----------------+----------+------------+-------+
6 rows in set (0.00 sec)
```

Queries:

1. **Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.**

```
mysql> select dob, address
    -> from Employee
    -> where f_name = 'John' and m_init = 'B' and l_name = 'Smith';
```

OUTPUT:

```
+------------+--------------------------+
| dob        | address                  |
+------------+--------------------------+
| 1965-01-09 | 731 Fondren, Houston, TX |
+------------+--------------------------+
1 row in set (0.01 sec)
```

2. **Retrieve the name and address of all employees who work for the 'Research' department.**

```
mysql> select f_name, l_name, address
    -> from Employee, Department
    -> where d_name = 'Research' and d_number = d_no;
```

OUTPUT:

```
+----------+---------+--------------------------+
| f_name   | l_name  | address                  |
+----------+---------+--------------------------+
| John     | Smith   | 731 Fondren, Houston, TX |
| Franklin | Wong    | 638 Voss, Houston, TX    |
| Joyce    | English | 5631 Rice, Houston, TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble, TX |
+----------+---------+--------------------------+
4 rows in set (0.00 sec)
```

3. **For every project located in 'Strafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.**

```
mysql> select p_number, d_num, l_name, address, dob
    -> from Project, Department, Employee
    -> where d_num = d_number and mgr_ssn = ssn and p_location =
'Strafford';
```

OUTPUT:

```
+----------+-------+---------+--------------------------+------------+
| p_number | d_num | l_name  | address                  | dob        |
+----------+-------+---------+--------------------------+------------+
|       10 |     4 | Wallace | 291 Berry, Bellaire, TX | 1941-06-20 |
|       30 |     4 | Wallace | 291 Berry, Bellaire, TX | 1941-06-20 |
+----------+-------+---------+--------------------------+------------+
2 rows in set (0.01 sec)
```

4. **Retrieve the names and addresses of the employees working in the 'Research' Department.**

```
mysql> select Employee.f_name, Employee.l_name, Employee.address
    -> from Employee, Department
    -> where Department.d_name = 'Research' and d_no = d_number;
```

OUTPUT:

```
+----------+---------+--------------------------+
| f_name   | l_name  | address                  |
+----------+---------+--------------------------+
| John     | Smith   | 731 Fondren, Houston, TX |
| Franklin | Wong    | 638 Voss, Houston, TX    |
| Joyce    | English | 5631 Rice, Houston, TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble, TX |
+----------+---------+--------------------------+
4 rows in set (0.00 sec)
```

5. **For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.**

```
mysql> select E.f_name, E.l_name, S.f_name, S.l_name
    -> from Employee as E, Employee as S
    -> where E.super_ssn = S.ssn;
```

OUTPUT:

```
+----------+---------+----------+---------+
| f_name   | l_name  | f_name   | l_name  |
+----------+---------+----------+---------+
| John     | Smith   | Franklin | Wong    |
| Franklin | Wong    | James    | Borg    |
| Joyce    | English | Franklin | Wong    |
| Ramesh   | Narayan | Franklin | Wong    |
| Jennifer | Wallace | James    | Borg    |
| Ahmad    | Jabbar  | Jennifer | Wallace |
| Alicia   | Zelaya  | Jennifer | Wallace |
+----------+---------+----------+---------+
```

6. **Select all EMPLOYEE ssn's .**

```
mysql> select ssn from Employee;
```

OUTPUT:

```
+-----------+
| ssn       |
+-----------+
| 123456789 |
| 333445555 |
| 453453453 |
| 666884444 |
| 888665555 |
| 987654321 |
| 987987987 |
| 999887777 |
+-----------+
8 rows in set (0.00 sec)
```

7. **Select all combinations of EMPLOYEE ssn and DEPARTMENT d_name in the database.**

```
mysql> select ssn, d_name
    -> from Employee, Department;
```

OUTPUT:

```
+-----------+----------------+
| ssn       | d_name         |
+-----------+----------------+
| 123456789 | Research       |
| 123456789 | Headquarters   |
| 123456789 | Administration |
| 333445555 | Research       |
| 333445555 | Headquarters   |
| 333445555 | Administration |
| 453453453 | Research       |
| 453453453 | Headquarters   |
| 453453453 | Administration |
| 666884444 | Research       |
| 666884444 | Headquarters   |
| 666884444 | Administration |
| 888665555 | Research       |
| 888665555 | Headquarters   |
| 888665555 | Administration |
| 987654321 | Research       |
| 987654321 | Headquarters   |
| 987654321 | Administration |
| 987987987 | Research       |
| 987987987 | Headquarters   |
| 987987987 | Administration |
| 999887777 | Research       |
| 999887777 | Headquarters   |
| 999887777 | Administration |
+-----------+----------------+
24 rows in set (0.00 sec)
```

8. **Retrieve all the attribute values of any EMPLOYEE who works in DEPARTMENT number 5.**

```
mysql> select *
    -> from Employee
    -> where d_no = 5;
```

OUTPUT:

```
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+
| f_name  | m_init | l_name  | ssn       | dob        | address                  | gender | salary   | super_ssn | d_no |
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+
| John    | B      | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M      | 30000.00 | 333445555 |    5 |
| Franklin| T      | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M      | 40000.00 | 888665555 |    5 |
| Joyce   | A      | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F      | 25000.00 | 333445555 |    5 |
| Ramesh  | K      | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M      | 38000.00 | 333445555 |    5 |
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+
4 rows in set (0.00 sec)
```

9. **Retrieve all the attributes of an EMPLOYEE and the attributes of the DEPARTMENT in which he or she works for every employee of the 'Research' department.**

```
mysql> select ssn, d_name
    -> from Employee, Department
    -> where d_name = 'Research' and d_no = d_number;
```

OUTPUT:

```
+-----------+----------+
| ssn       | d_name   |
+-----------+----------+
| 123456789 | Research |
| 333445555 | Research |
| 453453453 | Research |
| 666884444 | Research |
+-----------+----------+
4 rows in set (0.00 sec)
```

10. **Display the CROSS PRODUCT of the EMPLOYEE and DEPARTMENT relations.**

```
mysql> select * from Employee, Department;
```

OUTPUT:

```
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+----------------+----------+-----------+----------------+
| f_name  | m_init | l_name  | ssn       | dob        | address                  | gender | salary   | super_ssn | d_no | d_name         | d_number | mgr_ssn   | mgr_start_date |
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+----------------+----------+-----------+----------------+
| John    | B      | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M      | 30000.00 | 333445555 |    5 | Research       |        5 | 333445555 | 1988-05-22     |
| John    | B      | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M      | 30000.00 | 333445555 |    5 | Administration |        4 | 987654321 | 1995-01-01     |
| John    | B      | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M      | 30000.00 | 333445555 |    5 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M      | 40000.00 | 888665555 |    5 | Research       |        5 | 333445555 | 1988-05-22     |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M      | 40000.00 | 888665555 |    5 | Administration |        4 | 987654321 | 1995-01-01     |
| Franklin | T     | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M      | 40000.00 | 888665555 |    5 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Joyce   | A      | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F      | 25000.00 | 333445555 |    5 | Research       |        5 | 333445555 | 1988-05-22     |
| Joyce   | A      | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F      | 25000.00 | 333445555 |    5 | Administration |        4 | 987654321 | 1995-01-01     |
| Joyce   | A      | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F      | 25000.00 | 333445555 |    5 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Ramesh  | K      | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M      | 38000.00 | 333445555 |    5 | Research       |        5 | 333445555 | 1988-05-22     |
| Ramesh  | K      | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M      | 38000.00 | 333445555 |    5 | Administration |        4 | 987654321 | 1995-01-01     |
| Ramesh  | K      | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble, TX | M      | 38000.00 | 333445555 |    5 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| James   | E      | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M      | 55000.00 | NULL      |    1 | Research       |        5 | 333445555 | 1988-05-22     |
| James   | E      | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M      | 55000.00 | NULL      |    1 | Administration |        4 | 987654321 | 1995-01-01     |
| James   | E      | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M      | 55000.00 | NULL      |    1 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F      | 43000.00 | 888665555 |    4 | Research       |        5 | 333445555 | 1988-05-22     |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F      | 43000.00 | 888665555 |    4 | Administration |        4 | 987654321 | 1995-01-01     |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire, TX  | F      | 43000.00 | 888665555 |    4 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Ahmad   | V      | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX  | M      | 25000.00 | 987654321 |    4 | Research       |        5 | 333445555 | 1988-05-22     |
| Ahmad   | V      | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX  | M      | 25000.00 | 987654321 |    4 | Administration |        4 | 987654321 | 1995-01-01     |
| Ahmad   | V      | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston, TX  | M      | 25000.00 | 987654321 |    4 | Headquarters   |        1 | 888665555 | 1981-06-19     |
| Alicia  | J      | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX  | F      | 25000.00 | 987654321 |    4 | Research       |        5 | 333445555 | 1988-05-22     |
| Alicia  | J      | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX  | F      | 25000.00 | 987654321 |    4 | Administration |        4 | 987654321 | 1995-01-01     |
| Alicia  | J      | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring, TX  | F      | 25000.00 | 987654321 |    4 | Headquarters   |        1 | 888665555 | 1981-06-19     |
+---------+--------+---------+-----------+------------+--------------------------+--------+----------+-----------+------+----------------+----------+-----------+----------------+
24 rows in set (0.00 sec)
```

ASSIGNMENT 2

Lab Assignment-1

Exercise 1: Create the tables described below

a) Table Name: Client_Master
Description: used to store information about clients.

Column_Name	Data_Type	Size
ClientNo	varchar2	6
Name	varchar2	20
Address1	varchar2	30
Address2	varchar2	30
City	varchar2	15
PinCode	Number	8
State	varchar2	15
BalDue	Number	10

b) Table Name: Product_Master
Description: Used to store information about products.

Column_Name	Data_Type	Size
ProductNo	varchar2	6
Description	varchar2	15
ProfitPercent	varchar2	4
UnitMeasure	varchar2	10
QtyOnHand	Number	8
RecorderLvi	Number	8
SellPrice	Number	8
CostPrice	Number	8

Exercise 2: Insert the following data into the tables:

a) Data for Client_Master Table.

ClientNo	Name	Address1	Address2	City	PinCode	State	BalDue
C00001	Ivan Bayross	Address1	Address2	Mumbai	400001	Maharashtra	15000
C00002	Mamta Muzumdar	Address1	Address2	Madras	780001	Tamil Nadu	0
C00003	Chhaya Bankar	Address1	Address2	Mumbai	400057	Maharashtra	5000
C00004	Ashwini Joshi	Address1	Address2	Bangalore	560001	Karnataka	0
C00005	Hansel Colaco	Address1	Address2	Mumbai	400060	Maharashtra	2000
C00006	Deepak Sharma	Address1	Address2	Mangalore	560050	Karnataka	0

b) Data for Product_Master Table.

ProductNo	ProductName	Quantity	Piece	QtyOnHand	Tax	CostPrice	SalePrice
P00001	T-Shirts	5.00	Piece	200	50	350	250
P0345	Shirts	6.00	Piece	150	50	500	350
P06734	Cotton Jeans	5.00	Piece	100	20	600	450
P07865	Jeans	5.00	Piece	100	20	750	500
P07868	Trousers	2.00	Piece	150	50	850	550
P07885	Pull Overs	2.50	Piece	150	50	850	550
P07868	Denim Shirts	4.00	Piece	100	40	350	250
P07868	Lycra Tops	5.00	Piece	70	30	300	175
P07868	Skirts	5.00	Piece	75	30	450	300

c) Data for Salesman_Master Table.

SalesmanNo	SalesmanName	Address1	Address2	City	PinCode	State	SalAmt	TgtToGet	YtdSales	Remarks
S00001	Aman	A/14	Worli	Mumbai	400002	Maharashtra	3000	100	50	Good
S00002	Omkar	65	Nariman	Mumbai	400001	Maharashtra	3000	200	100	Good
S00003	Raj	P-7	Bandra	Mumbai	400032	Maharashtra	3000	200	100	Good
S00004	Ashish	A/5	Juhu	Mumbai	400044	Maharashtra	3500	200	150	Good

Exercise 3: Exercise on retreiving the data.

a) Find out the name of all the clients.

b) Retreive the entire contents of the Client_Master table.

c) Retreive the list of names, city and the state of al the clients.

d) List the various products available from the Product_Master teble.

e) List all the clients who live in Mumbai.

f) Find the names of salesman who have a salary equal to Rs.3000.

Exercise 4: Exercise on updating the records in the table.

a) Change the city of ClientNo 'C00005' to 'Bangalore'.

b) Change the BalDue of ClientNo 'C00001' to Rs.1000.

c) Change the cost price of 'Trousers' to Rs.950.00.

d) Change the city of the salesman to Pune.

c) Data for Salesman_Master Table.

SalesmanNo	SalesmanName	Address1	Address2	City	PinCode	State	SalAmt	TgtToGet	YtdSales	Remarks
S00001	Aman	A/14	Worli	Mumbai	400002	Maharashtra	3000	100	50	Good
S00002	Omkar	65	Nariman	Mumbai	400001	Maharashtra	3000	200	100	Good
S00003	Raj	P-7	Bandra	Mumbai	400032	Maharashtra	3000	200	100	Good
S00004	Ashish	A/5	Juhu	Mumbai	400044	Maharashtra	3500	200	150	Good

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

7th February 2022

Creation:

- ```
  mysql> create database 20bcs021_faizan;
  ```
- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table client_master
      -> (
      -> client_no        varchar(6),
      -> name             varchar(20),
      -> address1         varchar(30),
      -> address2         varchar(30),
      -> city             varchar(15),
      -> pin_code         int(8),
      -> state            varchar(15),
      -> bal_due          int(10)
      -> );
  ```
- ```
  mysql> insert into client_master
      -> values ('C00001', 'Ivan Bayross', 'B/5', 'Mira Rd',
  'Mumbai', 400001, 'Maharashtra', 15000),
      -> ('C00002', 'Mamta Mazumdar', 'P-4', 'Chennai',
  'Madras', 780001, 'Tamil Nadu', 0),
      -> ('C00003', 'Chhaya Bankar', 'A/10', 'Juhu', 'Mumbai',
  400057, 'Maharashtra', 5000),
      -> ('C00004', 'Ashwini Joshi', 'J-5', 'Alur', 'Bangalore',
  560001, 'Karnataka', 0),
       -> ('C00005', 'Hansei Colaco', 'D-8', 'Vile Parle',
  'Mumbai', 400060, 'Maharashtra', 2000),
       -> ('C00006', 'Deepak Sharma', 'E/2', 'Attur',
  'Mangalore', 560050, 'Karnataka', 0);
  ```

- mysql> select * from client_master;

```
mysql> select * from client_master;
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
| client_no | name          | address1 | address2   | city      | pin_code | state       | bal_due |
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
| C00001    | Ivan Bayross  | B/5      | Mira Rd    | Mumbai    |   400001 | Maharashtra |   15000 |
| C00002    | Mamta Mazumdar| P-4      | Chennai    | Madras    |   780001 | Tamil Nadu  |       0 |
| C00003    | Chhaya Bankar | A/10     | Juhu       | Mumbai    |   400057 | Maharashtra |    5000 |
| C00004    | Ashwini Joshi | J-5      | Alur       | Bangalore |   560001 | Karnataka   |       0 |
| C00005    | Hansei Colaco | D-8      | Vile Parle | Mumbai    |   400060 | Maharashtra |    2000 |
| C00006    | Deepak Sharma | E/2      | Attur      | Mangalore |   560050 | Karnataka   |       0 |
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
6 rows in set (0.00 sec)
```

- mysql> create table product_master

```
    -> (
    -> product_no       varchar(6),
    -> product_name     varchar(15),
    -> quantity         float (5,2),
    -> piece            varchar(10),
    -> qty_on_hand      int(8),
    -> tax              int(5),
    -> cost_price       int(8),
    -> sale_price       int(8)
    -> );
```

- mysql> insert into product_master values

```
    -> ('P00001', 'T-Shirts', 5.00, 'Piece', 200, 50, 350,
250),
    -> ('P0345', 'Shirts', 6.00, 'Piece', 150, 50, 500, 350),
    -> ('P06734', 'Cotton Jeans', 5.00, 'Piece', 100, 20, 600,
450),
    -> ('P07865', 'Jeans', 5.00, 'Piece', 100, 20, 750, 500),
    -> ('P07868', 'Trousers', 2.00, 'Piece', 150, 50, 850,
550),
    -> ('P07885', 'Pull Overs', 2.50, 'Piece', 150, 50, 850,
550),
    -> ('P07868', 'Denim Shirts', 4.00, 'Piece', 100, 40, 350,
250),
    -> ('P07868', 'Lycra Tops', 5.00, 'Piece', 70, 30, 300,
175),
    -> ('P07868', 'Skirts', 5.00, 'Piece', 75, 30, 450, 300);
```

- `mysql> select * from product_master;`

```
mysql> select * from product_master;
+------------+--------------+----------+-------+-------------+------+------------+------------+
| product_no | product_name | quantity | piece | qty_on_hand | tax  | cost_price | sale_price |
+------------+--------------+----------+-------+-------------+------+------------+------------+
| P00001     | T-Shirts     |     5.00 | Piece |         200 |   50 |        350 |        250 |
| P0345      | Shirts       |     6.00 | Piece |         150 |   50 |        500 |        350 |
| P06734     | Cotton Jeans |     5.00 | Piece |         100 |   20 |        600 |        450 |
| P07865     | Jeans        |     5.00 | Piece |         100 |   20 |        750 |        500 |
| P07868     | Trousers     |     2.00 | Piece |         150 |   50 |        850 |        550 |
| P07885     | Pull Overs   |     2.50 | Piece |         150 |   50 |        850 |        550 |
| P07868     | Denim Shirts |     4.00 | Piece |         100 |   40 |        350 |        250 |
| P07868     | Lycra Tops   |     5.00 | Piece |          70 |   30 |        300 |        175 |
| P07868     | Skirts       |     5.00 | Piece |          75 |   30 |        450 |        300 |
+------------+--------------+----------+-------+-------------+------+------------+------------+
9 rows in set (0.00 sec)
```

Queries:

1. **Find out the name of all the clients.**

```
mysql> select name
    -> from client_master;
```

OUTPUT:

```
+-----------------+
| name            |
+-----------------+
| Ivan Bayross    |
| Mamta Mazumdar  |
| Chhaya Bankar   |
| Ashwini Joshi   |
| Hansei Colaco   |
| Deepak Sharma   |
+-----------------+
6 rows in set (0.00 sec)
```

2. **Retrieve the entire contents of the client_master table.**

```
mysql> select * from client_master;
```

OUTPUT:

```
mysql> select * from client_master;
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
| client_no | name          | address1 | address2   | city      | pin_code | state       | bal_due |
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
| C00001    | Ivan Bayross  | B/5      | Mira Rd    | Mumbai    |   400001 | Maharashtra |   15000 |
| C00002    | Mamta Mazumdar| P-4      | Chennai    | Madras    |   780001 | Tamil Nadu  |       0 |
| C00003    | Chhaya Bankar | A/10     | Juhu       | Mumbai    |   400057 | Maharashtra |    5000 |
| C00004    | Ashwini Joshi | J-5      | Alur       | Bangalore |   560001 | Karnataka   |       0 |
| C00005    | Hansei Colaco | D-8      | Vile Parle | Mumbai    |   400060 | Maharashtra |    2000 |
| C00006    | Deepak Sharma | E/2      | Attur      | Mangalore |   560050 | Karnataka   |       0 |
+-----------+---------------+----------+------------+-----------+----------+-------------+---------+
6 rows in set (0.00 sec)
```

3. **Retrieve the list of names, city and the state of all the clients.**

```
mysql> select name, city, state
    -> from client_master;
```

OUTPUT:

```
+----------------+-----------+-------------+
| name           | city      | state       |
+----------------+-----------+-------------+
| Ivan Bayross   | Mumbai    | Maharashtra |
| Mamta Mazumdar | Madras    | Tamil Nadu  |
| Chhaya Bankar  | Mumbai    | Maharashtra |
| Ashwini Joshi  | Bangalore | Karnataka   |
| Hansei Colaco  | Mumbai    | Maharashtra |
| Deepak Sharma  | Mangalore | Karnataka   |
+----------------+-----------+-------------+
6 rows in set (0.00 sec)
```

4. List the various products available from the product_master table.

```
mysql> select product_name
    -> from product_master;
```

OUTPUT:

```
+--------------+
| product_name |
+--------------+
| T-Shirts     |
| Shirts       |
| Cotton Jeans |
| Jeans        |
| Trousers     |
| Pull Overs   |
| Denim Shirts |
| Lycra Tops   |
| Skirts       |
+--------------+
9 rows in set (0.00 sec)
```

5. List all the clients who live in Mumbai.

```
mysql> select name
    -> from client_master
    -> where city = 'Mumbai';
```

OUTPUT:

```
+---------------+
| name          |
+---------------+
| Ivan Bayross  |
| Chhaya Bankar |
| Hansei Colaco |
+---------------+
3 rows in set (0.01 sec)
```

6. Change the city of client_no 'C00005' to 'Bangalore'

```
mysql> update client_master
    -> set city = 'Bangalore'
    -> where client_no = 'C00005';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from client_master;
```

OUTPUT:

```
mysql> select * from client_master;
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
| client_no | name           | address1 | address2   | city      | pin_code | state       | bal_due |
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
| C00001    | Ivan Bayross   | B/5      | Mira Rd    | Mumbai    |   400001 | Maharashtra |   15000 |
| C00002    | Mamta Mazumdar | P-4      | Chennai    | Madras    |   780001 | Tamil Nadu  |       0 |
| C00003    | Chhaya Bankar  | A/10     | Juhu       | Mumbai    |   400057 | Maharashtra |    5000 |
| C00004    | Ashwini Joshi  | J-5      | Alur       | Bangalore |   560001 | Karnataka   |       0 |
| C00005    | Hansei Colaco  | D-8      | Vile Parle | Bangalore |   400060 | Maharashtra |    2000 |
| C00006    | Deepak Sharma  | E/2      | Attur      | Mangalore |   560050 | Karnataka   |       0 |
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
6 rows in set (0.00 sec)
```

7. Change the bal_due of client_no 'C00001' to Rs. 1000.

```
mysql> update client_master
    -> set bal_due = 1000
    -> where client_no = 'C00001';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from client_master;
```

OUTPUT:

```
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
| client_no | name           | address1 | address2   | city      | pin_code | state       | bal_due |
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
| C00001    | Ivan Bayross   | B/5      | Mira Rd    | Mumbai    |   400001 | Maharashtra |    1000 |
| C00002    | Mamta Mazumdar | P-4      | Chennai    | Madras    |   780001 | Tamil Nadu  |       0 |
| C00003    | Chhaya Bankar  | A/10     | Juhu       | Mumbai    |   400057 | Maharashtra |    5000 |
| C00004    | Ashwini Joshi  | J-5      | Alur       | Bangalore |   560001 | Karnataka   |       0 |
| C00005    | Hansei Colaco  | D-8      | Vile Parle | Bangalore |   400060 | Maharashtra |    2000 |
| C00006    | Deepak Sharma  | E/2      | Attur      | Mangalore |   560050 | Karnataka   |       0 |
+-----------+----------------+----------+------------+-----------+----------+-------------+---------+
6 rows in set (0.00 sec)
```

8. Change the cost_price of 'Trousers' to Rs. 950.

```
mysql> update product_master
    -> set cost_price = 950
    -> where product_name = 'Trousers';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from product_master;
```

OUTPUT:

product_no	product_name	quantity	piece	qty_on_hand	tax	cost_price	sale_price
P00001	T-Shirts	5.00	Piece	200	50	350	250
P0345	Shirts	6.00	Piece	150	50	500	350
P06734	Cotton Jeans	5.00	Piece	100	20	600	450
P07865	Jeans	5.00	Piece	100	20	750	500
P07868	Trousers	2.00	Piece	150	50	950	550
P07885	Pull Overs	2.50	Piece	150	50	850	550
P07868	Denim Shirts	4.00	Piece	100	40	350	250
P07868	Lycra Tops	5.00	Piece	70	30	300	175
P07868	Skirts	5.00	Piece	75	30	450	300

9 rows in set (0.00 sec)

9. Delete all products from product_master where the qty_on_hand is equal to 100.

```
mysql> delete from product_master
    -> where qty_on_hand = 100;
```

Query OK, 3 rows affected (0.01 sec)

```
mysql> select * from product_master;
```

OUTPUT:

product_no	product_name	quantity	piece	qty_on_hand	tax	cost_price	sale_price
P00001	T-Shirts	5.00	Piece	200	50	350	250
P0345	Shirts	6.00	Piece	150	50	500	350
P07868	Trousers	2.00	Piece	150	50	950	550
P07885	Pull Overs	2.50	Piece	150	50	850	550
P07868	Lycra Tops	5.00	Piece	70	30	300	175
P07868	Skirts	5.00	Piece	75	30	450	300

6 rows in set (0.00 sec)

ASSIGNMENT 3

Questions for Lab Assignment 1

Exercise 1: Create the tables described below

a) Table Name: Client_Master
Description: used to store information about clients.

Column_Name	Data_Type	Size
ClientNo	varchar2	6
Name	varchar2	20
Address1	varchar2	30
Address2	varchar2	30
City	varchar2	15
PinCode	Number	8
State	varchar2	15
BalDue	Number	10

b) Table Name: Product_Master
Description: Used to store information about products.

Column_Name	Data_Type	Size
ProductNo	varchar2	6
Description	varchar2	15
ProfitPercent	varchar2	4
UnitMeasure	varchar2	10
QtyOnHand	Number	8
RecorderLvi	Number	8
SellPrice	Number	8
CostPrice	Number	8

a) Table Name: Salesman_Master
Description: Used to store information about salesman working in the company.

Column_Name	Data_Type	Size
SalesmanNo	varchar2	6
SalesmanName	varchar2	20
Address1	varchar2	30
Address2	varchar2	30
City	varchar2	20
PinCode	Number	8
State	varchar2	20
SalAmt	Number	8
TgttoGet	Number	6
YtdSales	Number	6
Remarks	varchar2	60

Exercise 2: Insert the following data into the tables:

a) Data for Client_Master Table.

ClientNo	Name	Address1	Address2	City	PinCode	State	BailDue
C00001	Ivan Bayross	Address1	Address2	Mumbai	400001	Maharashtra	15000
C00002	Mamta Muzumdar	Address1	Address2	Madras	780001	Tamil Nadu	0
C00003	Chhaya Bankar	Address1	Address2	Mumbai	400057	Maharashtra	5000
C00004	Ashwini Joshi	Address1	Address2	Bangalore	560001	Karnataka	0
C00005	Hansei Colaco	Address1	Address2	Mumbai	400060	Maharashtra	2000
C00006	Deepak Sharma	Address1	Address2	Mangalore	560050	Karnataka	0

b) Data for Product_Master Table.

ProductNo	ProductName	Quantity	Piece	QtyOnHand	Tax	CostPrice	SalePrice
P00001	T-Shirts	5.00	Piece	200	50	350	250
P0345	Shirts	6.00	Piece	150	50	500	350
P06734	Cotton Jeans	5.00	Piece	100	20	600	450
P07865	Jeans	5.00	Piece	100	20	750	500
P07868	Trousers	2.00	Piece	150	50	850	550
P07885	Pull Overs	2.50	Piece	150	50	850	550
P07868	Denim Shirts	4.00	Piece	100	40	350	250
P07868	Lycra Tops	5.00	Piece	70	30	300	175
P07868	Skirts	5.00	Piece	75	30	450	300

b) Data for Salesman_Master Table.

SalesmanNo	SalesmanName	Address1	Address2	City	PinCode	State	SalAmt	TgtToGet	YtdSales	Remarks
S00001	Aman	A/14	Worli	Mumbai	400002	Maharashtra	3000	100	50	Good
S00002	Omkar	65	Nariman	Mumbai	400001	Maharashtra	3000	200	100	Good
S00003	Raj	P-7	Bandra	Mumbai	400032	Maharashtra	3000	200	100	Good
S00004	Ashish	A/5	Juhu	Mumbai	400044	Maharashtra	3500	200	150	Good

Exercise 3: Exercise on retreiving the data.

a) Find out the name of all the clients.

b) Retreive the entire contents of the Client_Master table.

c) Retreive the list of names,city and the state of al the clients.

d) List the various products available from the Product_Master teble.

e) List all the clients who live in Mumbai.

f) Find the names of salesman who have a salary equal to Rs.3000.

Exercise 4: Exercise on updating the records in the table.

a) Change the city of ClientNo ‘C00005’ to ‘Bangalore’.

b) Change the BalDue of ClientNo ‘C00001’ to Rs.1000.

c) Change the cost price of ‘Trousers’ to Rs.950.00.

d) Change the city of the salesman to Pune.

Exercise 5: Exercise on deleting the records in the table.

a) Delete all the salesman from the Salesman_Master whose salaries are equal
to 3500.

b) Delete all products from Product_Master where the QtyOnHand is equal to

100.

a) Delete from Client where the column state =’Maharashtra’.

Exercise 6: Exercise on altering the table structure.

b) Add a column called ‘Telephone’ of data type ‘number’ and size ‘10’ to the
Client_Master table.

c) Change the size ODF SellPrice column Product_Master to 10,2.

Exercise 7: Exercise on deleting the table structure.

d)Destroy table Client_Master along with its data.

Exercise 8: Exercise on renaming the table.

e) Change the name of the Salesman_Master to sman_mast.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

14th February 2022

Creation:

- mysql> use 20bcs021_faizan;
- mysql> create table salesman_master

```
    -> (
    -> salesman_no      varchar(6),
    -> salesman_name    varchar(20),
    -> address1         varchar(30),
    -> address2         varchar(30),
    -> city             varchar(20),
    -> pincode          int(8),
    -> state            varchar(20),
    -> sal_amt          int(8),
    -> tgt_to_get       int(6),
    -> ytd_sales        int(6),
    -> remarks          varchar(60)
    -> );
```

- mysql> insert into salesman_master

```
    -> values ('S00001', 'Aman', 'A/14', 'Worli', 'Mumbai',
400002, 'Maharashtra', 3000, 100, 50, 'Good'),
    -> ('S00002', 'Omkar', '65', 'Nariman', 'Mumbai', 400001,
'Maharashtra', 3000, 200, 100, 'Good'),
    -> ('S00003', 'Raj', 'P-7', 'Bandra', 'Mumbai', 400032,
'Maharashtra', 3000, 200, 100, 'Good'),
    -> ('S00004', 'Ashish', 'A/5', 'Juhu', 'Mumbai', 400044,
'Maharashtra', 3500, 200, 150, 'Good');
```

- mysql> select * from salesman_master;

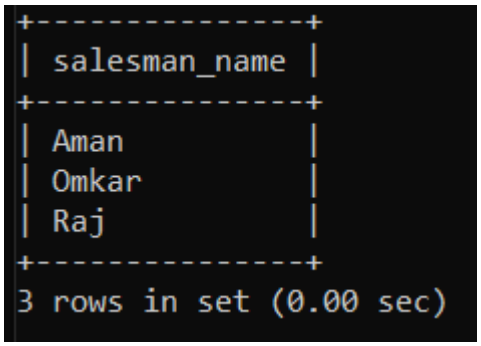
```
mysql> select * from salesman_master;
+------------+--------------+----------+----------+--------+---------+-------------+---------+------------+-----------+---------+
| salesman_no | salesman_name | address1 | address2 | city   | pincode | state       | sal_amt | tgt_to_get | ytd_sales | remarks |
+------------+--------------+----------+----------+--------+---------+-------------+---------+------------+-----------+---------+
| S00001      | Aman         | A/14     | Worli    | Mumbai |  400002 | Maharashtra |    3000 |        100 |        50 | Good    |
| S00002      | Omkar        | 65       | Nariman  | Mumbai |  400001 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00003      | Raj          | P-7      | Bandra   | Mumbai |  400032 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00004      | Ashish       | A/5      | Juhu     | Mumbai |  400044 | Maharashtra |    3500 |        200 |       150 | Good    |
+------------+--------------+----------+----------+--------+---------+-------------+---------+------------+-----------+---------+
4 rows in set (0.00 sec)
```

Queries:

1. **Find the names of salesman who have a salary equal to Rs.3000.**

```
mysql> select salesman_name
    -> from salesman_master
    -> where sal_amt = 3000;
```

OUTPUT:

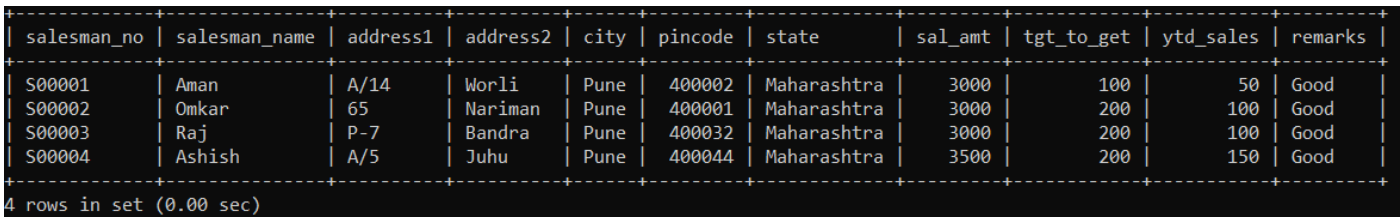


```
+---------------+
| salesman_name |
+---------------+
| Aman          |
| Omkar         |
| Raj           |
+---------------+
3 rows in set (0.00 sec)
```

2. **Change the city of the salesman to Pune.**

```
mysql> update salesman_master
    -> set city = 'Pune';
```

OUTPUT:

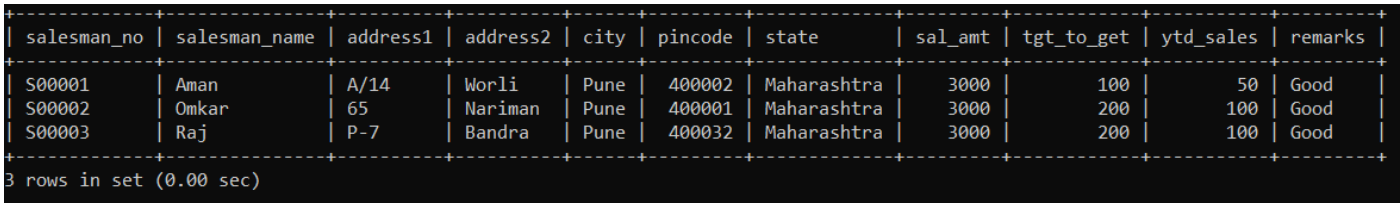


```
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
| salesman_no | salesman_name | address1 | address2 | city | pincode | state       | sal_amt | tgt_to_get | ytd_sales | remarks |
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
| S00001      | Aman         | A/14     | Worli    | Pune |  400002 | Maharashtra |    3000 |        100 |        50 | Good    |
| S00002      | Omkar        | 65       | Nariman  | Pune |  400001 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00003      | Raj          | P-7      | Bandra   | Pune |  400032 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00004      | Ashish       | A/5      | Juhu     | Pune |  400044 | Maharashtra |    3500 |        200 |       150 | Good    |
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
4 rows in set (0.00 sec)
```

3. **Delete all the salesman from the salesman_master whose salaries are equal to 3500.**

```
mysql> delete from salesman_master
    -> where sal_amt = 3500;
```

OUTPUT:

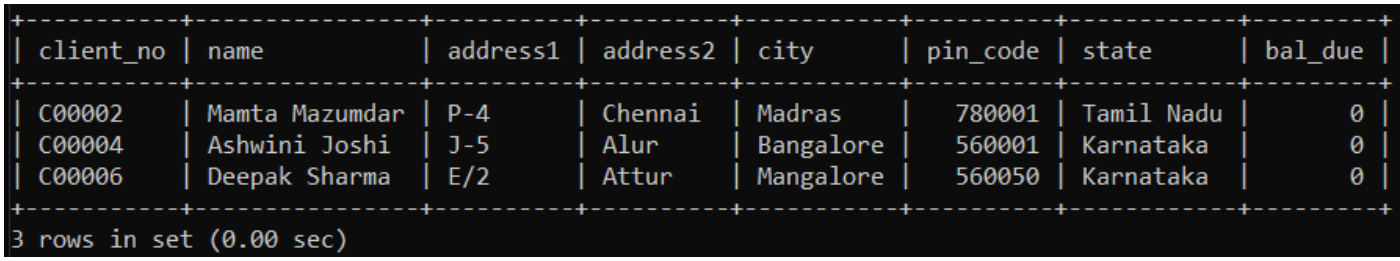


```
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
| salesman_no | salesman_name | address1 | address2 | city | pincode | state       | sal_amt | tgt_to_get | ytd_sales | remarks |
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
| S00001      | Aman         | A/14     | Worli    | Pune |  400002 | Maharashtra |    3000 |        100 |        50 | Good    |
| S00002      | Omkar        | 65       | Nariman  | Pune |  400001 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00003      | Raj          | P-7      | Bandra   | Pune |  400032 | Maharashtra |    3000 |        200 |       100 | Good    |
+------------+--------------+----------+----------+------+---------+-------------+---------+------------+-----------+---------+
3 rows in set (0.00 sec)
```

4. **Delete from Client where the column state =’Maharashtra’.**

```
mysql> delete from client_master
    -> where state = 'Maharashtra';
```

OUTPUT:



```
+-----------+-----------------+----------+----------+-----------+----------+-------------+----------+
| client_no | name            | address1 | address2 | city      | pin_code | state       | bal_due  |
+-----------+-----------------+----------+----------+-----------+----------+-------------+----------+
| C00002    | Mamta Mazumdar  | P-4      | Chennai  | Madras    |   780001 | Tamil Nadu  |        0 |
| C00004    | Ashwini Joshi   | J-5      | Alur     | Bangalore |   560001 | Karnataka   |        0 |
| C00006    | Deepak Sharma   | E/2      | Attur    | Mangalore |   560050 | Karnataka   |        0 |
+-----------+-----------------+----------+----------+-----------+----------+-------------+----------+
3 rows in set (0.00 sec)
```

5. **Add a column called 'Telephone' of data type 'number' and size '10' to the client_master table.**

```
mysql> alter table client_master
    -> add telephone int(10);
```

OUTPUT:

```
+-----------+----------------+----------+----------+-----------+----------+------------+---------+-----------+
| client_no | name           | address1 | address2 | city      | pin_code | state      | bal_due | telephone |
+-----------+----------------+----------+----------+-----------+----------+------------+---------+-----------+
| C00002    | Mamta Mazumdar | P-4      | Chennai  | Madras    |   780001 | Tamil Nadu |       0 |      NULL |
| C00004    | Ashwini Joshi  | J-5      | Alur     | Bangalore |   560001 | Karnataka  |       0 |      NULL |
| C00006    | Deepak Sharma  | E/2      | Attur    | Mangalore |   560050 | Karnataka  |       0 |      NULL |
+-----------+----------------+----------+----------+-----------+----------+------------+---------+-----------+
3 rows in set (0.00 sec)
```

6. **Change the size of sale_price column product_master to 10,2.**

```
mysql> alter table product_master
    -> modify sale_price decimal(10,2);
```

OUTPUT:

```
+------------+--------------+----------+-------+-------------+------+------------+------------+
| product_no | product_name | quantity | piece | qty_on_hand | tax  | cost_price | sale_price |
+------------+--------------+----------+-------+-------------+------+------------+------------+
| P00001     | T-Shirts     |     5.00 | Piece |         200 |   50 |        350 |     250.00 |
| P0345      | Shirts       |     6.00 | Piece |         150 |   50 |        500 |     350.00 |
| P07868     | Trousers     |     2.00 | Piece |         150 |   50 |        950 |     550.00 |
| P07885     | Pull Overs   |     2.50 | Piece |         150 |   50 |        850 |     550.00 |
| P07868     | Lycra Tops   |     5.00 | Piece |          70 |   30 |        300 |     175.00 |
| P07868     | Skirts       |     5.00 | Piece |          75 |   30 |        450 |     300.00 |
+------------+--------------+----------+-------+-------------+------+------------+------------+
6 rows in set (0.00 sec)
```

7. **Destroy table client_master along with its data.**

```
mysql> alter table product_master
    -> modify sale_price decimal(10,2);
```

OUTPUT:

```
ERROR 1146 (42S02): Table '20bcs021_faizan.client_master' doesn't exist
```

8. **Change the name of the Salesman_Master to sman_mast.**

```
mysql> alter table salesman_master
    -> rename to sman_mast;
```

OUTPUT:

```
ERROR 1146 (42S02): Table '20bcs021_faizan.salesman_master' doesn't exist
```

```
mysql> select * from sman_mast;
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
| salesman_no | salesman_name | address1 | address2 | city | pincode | state       | sal_amt | tgt_to_get | ytd_sales | remarks |
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
| S00001     | Aman         | A/14     | Worli    | Pune |  400002 | Maharashtra |    3000 |        100 |        50 | Good    |
| S00002     | Omkar        | 65       | Nariman  | Pune |  400001 | Maharashtra |    3000 |        200 |       100 | Good    |
| S00003     | Raj          | P-7      | Bandra   | Pune |  400032 | Maharashtra |    3000 |        200 |       100 | Good    |
+------------+--------------+----------+----------+------+---------+-------------+---------+-----------+-----------+---------+
3 rows in set (0.01 sec)
```

ASSIGNMENT 4

Lab Assignment-2

Create a relational database that contains the following tables and insert the following data into these tables.

department:

DeptID	DeptName
1	Information Technology
2	Electrical
3	Civil
4	Mechanical
5	Chemical

stud_member:

RollNo	FName	MName	SName	DeptID	Semester	ContactNo	Gender
1	Ankur	Samir	Kahar	1	1	272121	M
2	Dhaval	Dhiren	Joshi	1	1	232122	M
3	Ankita	Biren	Shah	1	1	112121	F
10	Komal	MaheshKumar	Pandya	2	3	123123	F
13	Amit	Jitenkumar	Mehta	3	3	453667	M
23	Jinal	Ashish	Gandhi	2	1	323232	M
22	Ganesh	Asha	Patel	2	3	124244	M
4	Shweta	Mihir	Patel	3	1	646341	F
7	Pooja	Mayaank	Desai	3	3	328656	F
8	Komal	Krishnaraj	Bhatia	2	3	257422	F
43	Kiran	Viraj	Shah	1	1	754124	F

Now, solve the following SQL Queries:

1. Display the names and contact numbers of all student members.
2. Give the names and roll number of all students of Information Technology who are the members.
3. Display names of Departments whose students are members.

4. Display names of Departments for which no student are members.

5. Display names of all Departments.

6. Find the number of students of Electrical Department who are members.

7. Display information of student members whose name begins with the letter 'A'.

8. Display all details of Male members only.

9. Display data of student members who are currently in semester 3.

10. Display data of student female members in alphabetical order.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

21st February 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table department2
      -> (
      -> dept_id  int(5),
      -> dept_name        varchar(30)
      -> );
  ```
- ```
  mysql> insert into department2 values
      -> (1, 'Information Technology'),
      -> (2, 'Electrical'),
      -> (3, 'Civil'),
      -> (4, 'Mechanical'),
      -> (5, 'Chemical');
  ```
- ```
  mysql> select * from department2;
  ```

```
mysql> select * from department2;
+---------+-----------------------+
| dept_id | dept_name             |
+---------+-----------------------+
|       1 | Information Technology |
|       2 | Electrical            |
|       3 | Civil                 |
|       4 | Mechanical            |
|       5 | Chemical              |
+---------+-----------------------+
5 rows in set (0.00 sec)
```

- ```
  mysql> create table stud_member
      -> (
      -> roll_no          int(5),
      -> f_name           varchar(15),
      -> m_name           varchar(15),
      -> s_name           varchar(15),
      -> dept_id          int(5),
      -> semester         int(1),
  ```
  ```

```
        -> contact_no       int(11),
        -> gender           char(2)
        -> );
```

- ```
  mysql> insert into stud_member values
        -> (1, 'Ankur', 'Samir', 'Kahar', 1, 1, 272121, 'M'),
        -> (2, 'Dhaval', 'Dhiren', 'Joshi', 1, 1, 232122, 'M'),
        -> (3, 'Ankita', 'Biren', 'Shah', 1, 1, 112121, 'F'),
        -> (10, 'Komal', 'MaheshKumar', 'Pandya', 2, 3, 123123,
  'F'),
        -> (13, 'Amit', 'Jitenkumar', 'Mehta', 3, 3, 453667, 'M'),
        -> (23, 'Jinal', 'Ashish', 'Gandhi', 2, 1, 323232, 'M'),
        -> (22, 'Ganesh', 'Asha', 'Patel', 2, 3, 124244, 'M'),
        -> (4, 'Shweta', 'Mihir', 'Patel', 3, 1, 646341, 'F'),
        -> (7, 'Pooja', 'Mayaank', 'Desai', 3, 3, 328656, 'F'),
        -> (8, 'Komal', 'Krishnaraj', 'Bhatia', 2, 3, 257422,
  'F'),
        -> (43, 'Kiran', 'Viraj', 'Shah', 1, 1, 754124, 'F');
  ```
- ```
  mysql> select * from stud_member;
  ```

```
mysql> select * from stud_member;
+---------+--------+-------------+--------+---------+----------+------------+--------+
| roll_no | f_name | m_name      | s_name | dept_id | semester | contact_no | gender |
+---------+--------+-------------+--------+---------+----------+------------+--------+
|       1 | Ankur  | Samir       | Kahar  |       1 |        1 |     272121 | M      |
|       2 | Dhaval | Dhiren      | Joshi  |       1 |        1 |     232122 | M      |
|       3 | Ankita | Biren       | Shah   |       1 |        1 |     112121 | F      |
|      10 | Komal  | MaheshKumar | Pandya |       2 |        3 |     123123 | F      |
|      13 | Amit   | Jitenkumar  | Mehta  |       3 |        3 |     453667 | M      |
|      23 | Jinal  | Ashish      | Gandhi |       2 |        1 |     323232 | M      |
|      22 | Ganesh | Asha        | Patel  |       2 |        3 |     124244 | M      |
|       4 | Shweta | Mihir       | Patel  |       3 |        1 |     646341 | F      |
|       7 | Pooja  | Mayaank     | Desai  |       3 |        3 |     328656 | F      |
|       8 | Komal  | Krishnaraj  | Bhatia |       2 |        3 |     257422 | F      |
|      43 | Kiran  | Viraj       | Shah   |       1 |        1 |     754124 | F      |
+---------+--------+-------------+--------+---------+----------+------------+--------+
11 rows in set (0.00 sec)
```

Queries:

1. **Display the names and contact numbers of all student members.**

```
mysql> select f_name, m_name, s_name, contact_no
    -> from stud_member;
```

OUTPUT:

```
+--------+-------------+--------+------------+
| f_name | m_name      | s_name | contact_no |
+--------+-------------+--------+------------+
| Ankur  | Samir       | Kahar  |     272121 |
| Dhaval | Dhiren      | Joshi  |     232122 |
| Ankita | Biren       | Shah   |     112121 |
| Komal  | MaheshKumar | Pandya |     123123 |
| Amit   | Jitenkumar  | Mehta  |     453667 |
| Jinal  | Ashish      | Gandhi |     323232 |
| Ganesh | Asha        | Patel  |     124244 |
| Shweta | Mihir       | Patel  |     646341 |
| Pooja  | Mayaank     | Desai  |     328656 |
| Komal  | Krishnaraj  | Bhatia |     257422 |
| Kiran  | Viraj       | Shah   |     754124 |
+--------+-------------+--------+------------+
11 rows in set (0.00 sec)
```

2. **Give the names and roll number of all students of Information Technology who are the members.**

```
mysql> select roll_no, f_name, m_name, s_name
    -> from stud_member, department2
    -> where stud_member.dept_id = department2.dept_id and
stud_member.dept_id = 1;
```

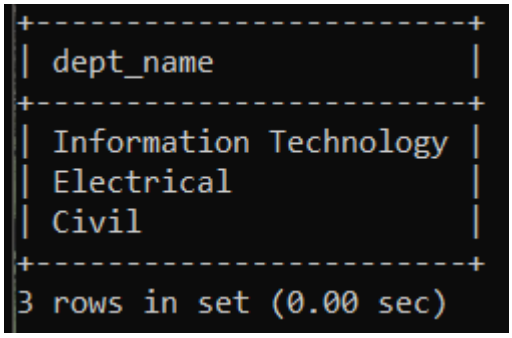
OUTPUT:

```
+---------+--------+--------+--------+
| roll_no | f_name | m_name | s_name |
+---------+--------+--------+--------+
|       1 | Ankur  | Samir  | Kahar  |
|       2 | Dhaval | Dhiren | Joshi  |
|       3 | Ankita | Biren  | Shah   |
|      43 | Kiran  | Viraj  | Shah   |
+---------+--------+--------+--------+
4 rows in set (0.00 sec)
```

3. **Display names of Departments whose students are members.**

```
mysql> select dept_name
    -> from department2
    -> where department2.dept_id in (select dept_id from
stud_member);
```

OUTPUT:

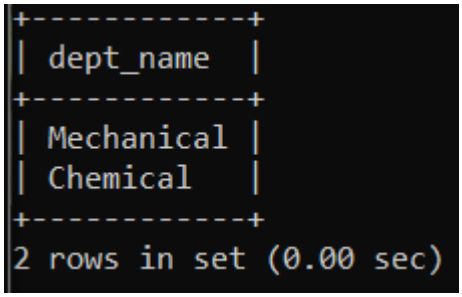


```
+------------------------+
| dept_name              |
+------------------------+
| Information Technology |
| Electrical             |
| Civil                  |
+------------------------+
3 rows in set (0.00 sec)
```

4. **Display names of Departments for which no student are members.**

```
mysql> select dept_name
    -> from department2
    -> where department2.dept_id not in (select dept_id from
stud_member);
```

OUTPUT:

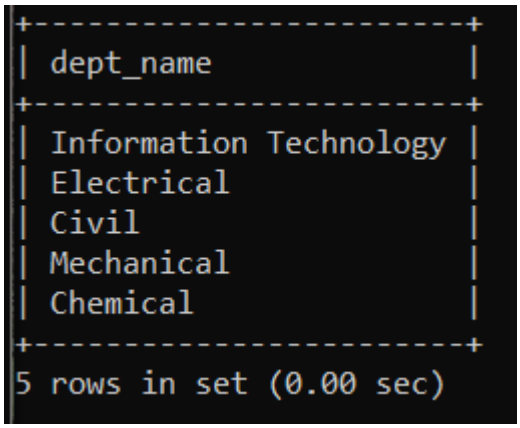


```
+------------+
| dept_name  |
+------------+
| Mechanical |
| Chemical   |
+------------+
2 rows in set (0.00 sec)
```

5. **Display names of all Departments.**

```
mysql> select dept_name
    -> from department2;
```

OUTPUT:



```
+------------------------+
| dept_name              |
+------------------------+
| Information Technology |
| Electrical             |
| Civil                  |
| Mechanical             |
| Chemical               |
+------------------------+
5 rows in set (0.00 sec)
```

6. **Find the number of students of Electrical Department who are members.**

```
mysql> select count(*)
    -> from stud_member, department2
    -> where stud_member.dept_id = department2.dept_id and
department2.dept_name = "Electrical";
```

OUTPUT:

```
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.01 sec)
```

7. **Display information of student members whose name begins with the letter 'A'.**

```
mysql> select *
    -> from stud_member
    -> where f_name like 'A%';
```

OUTPUT:

```
+---------+--------+------------+--------+---------+----------+------------+--------+
| roll_no | f_name | m_name     | s_name | dept_id | semester | contact_no | gender |
+---------+--------+------------+--------+---------+----------+------------+--------+
|       1 | Ankur  | Samir      | Kahar  |       1 |        1 |     272121 | M      |
|       3 | Ankita | Biren      | Shah   |       1 |        1 |     112121 | F      |
|      13 | Amit   | Jitenkumar | Mehta  |       3 |        3 |     453667 | M      |
+---------+--------+------------+--------+---------+----------+------------+--------+
3 rows in set (0.01 sec)
```

8. **Display all details of Male members only.**

```
mysql> select *
    -> from stud_member
    -> where gender = 'M';
```

OUTPUT:

```
+---------+--------+------------+--------+---------+----------+------------+--------+
| roll_no | f_name | m_name     | s_name | dept_id | semester | contact_no | gender |
+---------+--------+------------+--------+---------+----------+------------+--------+
|       1 | Ankur  | Samir      | Kahar  |       1 |        1 |     272121 | M      |
|       2 | Dhaval | Dhiren     | Joshi  |       1 |        1 |     232122 | M      |
|      13 | Amit   | Jitenkumar | Mehta  |       3 |        3 |     453667 | M      |
|      23 | Jinal  | Ashish     | Gandhi |       2 |        1 |     323232 | M      |
|      22 | Ganesh | Asha       | Patel  |       2 |        3 |     124244 | M      |
+---------+--------+------------+--------+---------+----------+------------+--------+
5 rows in set (0.00 sec)
```

9. **Display data of student members who are currently in semester 3.**

```
mysql> select *
    -> from stud_member
    -> where semester = 3;
```

OUTPUT:

```
+---------+--------+-------------+--------+---------+----------+------------+--------+
| roll_no | f_name | m_name      | s_name | dept_id | semester | contact_no | gender |
+---------+--------+-------------+--------+---------+----------+------------+--------+
|      10 | Komal  | MaheshKumar | Pandya |       2 |        3 |     123123 | F      |
|      13 | Amit   | Jitenkumar  | Mehta  |       3 |        3 |     453667 | M      |
|      22 | Ganesh | Asha        | Patel  |       2 |        3 |     124244 | M      |
|       7 | Pooja  | Mayaank     | Desai  |       3 |        3 |     328656 | F      |
|       8 | Komal  | Krishnaraj  | Bhatia |       2 |        3 |     257422 | F      |
+---------+--------+-------------+--------+---------+----------+------------+--------+
5 rows in set (0.00 sec)
```

10. Display data of student female members in alphabetical order.

```
mysql> select *
    -> from stud_member
    -> where gender = 'F'
    -> order by f_name;
```

OUTPUT:

```
+---------+--------+-------------+--------+---------+----------+------------+--------+
| roll_no | f_name | m_name      | s_name | dept_id | semester | contact_no | gender |
+---------+--------+-------------+--------+---------+----------+------------+--------+
|       3 | Ankita | Biren       | Shah   |       1 |        1 |     112121 | F      |
|      43 | Kiran  | Viraj       | Shah   |       1 |        1 |     754124 | F      |
|      10 | Komal  | MaheshKumar | Pandya |       2 |        3 |     123123 | F      |
|       8 | Komal  | Krishnaraj  | Bhatia |       2 |        3 |     257422 | F      |
|       7 | Pooja  | Mayaank     | Desai  |       3 |        3 |     328656 | F      |
|       4 | Shweta | Mihir       | Patel  |       3 |        1 |     646341 | F      |
+---------+--------+-------------+--------+---------+----------+------------+--------+
6 rows in set (0.00 sec)
```

ASSIGNMENT 5

Questions for Lab Assignment 4

Create the tables and solve the following queries:

Table 1: Employee

Emp_id	Emp_name	Salary	DNo
101	Amit	25000	D1001
102	Sunil	20000	D1002
103	Rakesh	18000	D1003
104	Ajay	16000	D1001
105	Suhail	20000	D1002
106	Arif	18000	D1004
107	Suresh	24000	D1002
108	Vijay	22000	D1003

Table 2: Department

DNo	Dept_name
D1001	IT
D1002	Sales
D1003	Marketing
D1004	HR

1. Display total sum required to pay the salary of all employees.

2. Display the average salary, minimum salary and maximum salary of the Company.

3. Display the sum of salary department-wise.

1. Display the maximum salary department-wise.

5 Display the details of the employee who earns the maximum salary.

6. Display details of every employee having maximum salary in his Department.

7. Display the details of the employee who earns more salary than the average salary of his department.

8. Display total number of employees in each department along with the department name.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

7th March 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table employee2
      -> (
      -> emp_id   int(4),
      -> emp_name varchar(15),
      -> salary   int(8),
      -> d_no     varchar(10)     not null,
      -> primary key(d_no)
      -> );
  ```
- ```
  mysql> insert into employee2 values
      -> (101, 'Amit', 25000, 'D1001'),
      -> (102, 'Sunil', 20000, 'D1002'),
      -> (103, 'Rakesh', 18000, 'D1003'),
      -> (104, 'Ajay', 16000, 'D1001'),
      -> (105, 'Suhail', 20000, 'D1002'),
      -> (106, 'Arif', 18000, 'D1004'),
      -> (107, 'Suresh', 24000, 'D1002'),
      -> (108, 'Vijay', 22000, 'D1003');
  ```

- ```
  mysql> select * from employee2;
  ```

```
mysql> select * from employee2;
+--------+----------+--------+-------+
| emp_id | emp_name | salary | d_no  |
+--------+----------+--------+-------+
|    101 | Amit     |  25000 | D1001 |
|    102 | Sunil    |  20000 | D1002 |
|    103 | Rakesh   |  18000 | D1003 |
|    104 | Ajay     |  16000 | D1001 |
|    105 | Suhail   |  20000 | D1002 |
|    106 | Arif     |  18000 | D1004 |
|    107 | Suresh   |  24000 | D1002 |
|    108 | Vijay    |  22000 | D1003 |
+--------+----------+--------+-------+
8 rows in set (0.00 sec)
```

- ```
  mysql> create table department3
      -> (
      -> d_no      varchar(10)      not null,
      -> dept_name varchar(15)
      -> );
  ```
- ```
  mysql> insert into department3 values
      -> ('D1001', 'IT'),
      -> ('D1002', 'Sales'),
      -> ('D1003', 'Marketing'),
      -> ('D1004', 'HR');
  ```
- ```
  mysql> select * from department3;
  ```

```
mysql> select * from department3;
+-------+-----------+
| d_no  | dept_name |
+-------+-----------+
| D1001 | IT        |
| D1002 | Sales     |
| D1003 | Marketing |
| D1004 | HR        |
+-------+-----------+
4 rows in set (0.00 sec)
```

Queries:

1. **Display total sum required to pay the salary of all employees.**

```
mysql> select sum(salary) as tot_sal from employee2;
```

OUTPUT:

```
+---------+
| tot_sal |
+---------+
|  163000 |
+---------+
1 row in set (0.00 sec)
```

2. **Display the average salary, minimum salary and maximum salary of the Company.**

```
mysql> select avg(salary) as avg_sal, min(salary) as min_sal,
max(salary) as max_sal from employee2;
```

OUTPUT:
```

```
+------------+---------+---------+
| avg_sal    | min_sal | max_sal |
+------------+---------+---------+
| 20375.0000 |   16000 |   25000 |
+------------+---------+---------+
1 row in set (0.00 sec)
```

3. **Display the sum of salary department-wise.**

```
mysql> select employee2.d_no, sum(salary), dept_name
    -> from employee2, department3
    -> where employee2.d_no = department3.d_no
    -> group by dept_name;
```

OUTPUT:

```
+-------+-------------+-----------+
| d_no  | sum(salary) | dept_name |
+-------+-------------+-----------+
| D1001 |       41000 | IT        |
| D1002 |       64000 | Sales     |
| D1003 |       40000 | Marketing |
| D1004 |       18000 | HR        |
+-------+-------------+-----------+
4 rows in set (0.00 sec)
```

4. **Display the maximum salary department-wise.**

```
mysql> select employee2.d_no, max(salary), dept_name
    -> from employee2, department3
    -> where employee2.d_no = department3.d_no
    -> group by dept_name;
```

OUTPUT:

```
+-------+-------------+-----------+
| d_no  | max(salary) | dept_name |
+-------+-------------+-----------+
| D1001 |       25000 | IT        |
| D1002 |       24000 | Sales     |
| D1003 |       22000 | Marketing |
| D1004 |       18000 | HR        |
+-------+-------------+-----------+
4 rows in set (0.00 sec)
```

5. **Display the details of the employee who earns the maximum salary.**

```
mysql> select *
    -> from employee2
    -> where salary = (select max(salary) from employee2);
```

OUTPUT:

```
+--------+----------+--------+-------+
| emp_id | emp_name | salary | d_no  |
+--------+----------+--------+-------+
|    101 | Amit     |  25000 | D1001 |
+--------+----------+--------+-------+
1 row in set (0.00 sec)
```

6. **Display details of every employee having maximum salary in his Department.**

```
mysql> select *
    -> from employee2 natural join department3
    -> where (salary, dept_name) in
          (select max(salary), dept_name
           from employee2 natural join department3
           group by dept_name);
```

OUTPUT:

```
+-------+--------+----------+--------+-----------+
| d_no  | emp_id | emp_name | salary | dept_name |
+-------+--------+----------+--------+-----------+
| D1001 |    101 | Amit     |  25000 | IT        |
| D1004 |    106 | Arif     |  18000 | HR        |
| D1002 |    107 | Suresh   |  24000 | Sales     |
| D1003 |    108 | Vijay    |  22000 | Marketing |
+-------+--------+----------+--------+-----------+
4 rows in set (0.00 sec)
```

7. **Display the details of the employee who earns more salary than the average salary of his department.**

```
mysql> select *
    -> from employee2 natural join
          (select dept_name, d_no, avg(salary) as avg_sal
           from department3 natural join employee2
           group by dept_name) temp
    -> where employee2.d_no = temp.d_no and salary > avg_sal;
```

OUTPUT:

```
+-------+--------+----------+--------+-----------+------------+
| d_no  | emp_id | emp_name | salary | dept_name | avg_sal    |
+-------+--------+----------+--------+-----------+------------+
| D1001 |    101 | Amit     |  25000 | IT        | 20500.0000 |
| D1002 |    107 | Suresh   |  24000 | Sales     | 21333.3333 |
| D1003 |    108 | Vijay    |  22000 | Marketing | 20000.0000 |
+-------+--------+----------+--------+-----------+------------+
3 rows in set (0.00 sec)
```

8. **Display total number of employees in each department along with the department name.**

```
mysql> select dept_name, count(*) as no_employees
    -> from employee2 natural join department3
    -> group by dept_name;
```

OUTPUT:

```
+-----------+--------------+
| dept_name | no_employees |
+-----------+--------------+
| IT        |            2 |
| Sales     |            3 |
| Marketing |            2 |
| HR        |            1 |
+-----------+--------------+
4 rows in set (0.00 sec)
```

ASSIGNMENT 6

Questions for Lab Assignment 5

Table: Sales

OrderID	Date	Price	Quantity	CustomerName
1	2005/12/22	160	2	Smith
2	2005/08/10	190	3	Johnson
3	2005/07/13	500	5	Baldwin
4	2005/07/15	420	2	Smith
5	2005/12/22	1000	4	Wood
6	2005/10/02	820	4	Smith
7	2005/11/03	2000	2	Baldwin

Create the table Sales and Write SQL queries for the following:

1. Count how many orders have made a customer with CustomerName of Smith.

2. Find number of unique customers that have ordered from the store.

3. Find out total no. of items ordered by all the customers.

4. Find out average number of items per order.

5. Find out the average Quantity for all orders with Price greater than 200.

6. Find out what was the minimum price paid for any of the orders.

7. Find out the highest Price form the given sales table.

8. List out unique customers name only from the table.

9. List out name of the customers who have given order in the month of December.

10. Find out the total amount of money spent for each of the customers.

1. Select all unique customers who have spent more than 1200 in the store.

2. Select all customers that have ordered more than 5 items in total from all their orders.

3. Select all customers who have spent more than 1000, after 10/01/2005.

4. Select orders in increasing order of order price.

5. Select orders in decreasing order of order price.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

7th March 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table Sales
      -> (
      -> order_id int,
      -> date     date,
      -> price    int(5),
      -> qty      int,
      -> cust     varchar(15)
      -> );
  ```
- ```
  mysql> insert into Sales values
      -> (1, '2005/12/22', 160, 2, 'Smith'),
      -> (2, '2005/08/10', 190, 3, 'Johnson'),
      -> (3, '2005/07/13', 500, 5, 'Baldwin'),
      -> (4, '2005/07/15', 420, 2, 'Smith'),
      -> (5, '2005/12/22', 1000, 4, 'Wood'),
      -> (6, '2005/10/02', 820, 4, 'Smith'),
      -> (7, '2005/11/03', 2000, 2, 'Baldwin');
  ```
- ```
  mysql> select * from Sales;
  ```

```
mysql> select * from Sales;
+----------+------------+-------+------+---------+
| order_id | date       | price | qty  | cust    |
+----------+------------+-------+------+---------+
|        1 | 2005-12-22 |   160 |    2 | Smith   |
|        2 | 2005-08-10 |   190 |    3 | Johnson |
|        3 | 2005-07-13 |   500 |    5 | Baldwin |
|        4 | 2005-07-15 |   420 |    2 | Smith   |
|        5 | 2005-12-22 |  1000 |    4 | Wood    |
|        6 | 2005-10-02 |   820 |    4 | Smith   |
|        7 | 2005-11-03 |  2000 |    2 | Baldwin |
+----------+------------+-------+------+---------+
7 rows in set (0.00 sec)
```

Queries:

1. **Count how many orders have made a customer with CustomerName of Smith.**

```
mysql> select cust, count(*) as no_of_orders
    -> from Sales
    -> where cust = 'Smith';
```

OUTPUT:

```
+-------+--------------+
| cust  | no_of_orders |
+-------+--------------+
| Smith |            3 |
+-------+--------------+
1 row in set (0.00 sec)
```

2. **Find number of unique customers that have ordered from the store.**

```
mysql> select count(distinct cust) as no_of_cust
    -> from Sales;
```

OUTPUT:

```
+------------+
| no_of_cust |
+------------+
|          4 |
+------------+
1 row in set (0.00 sec)
```

3. **Find out total no. of items ordered by all the customers.**

```
mysql> select sum(qty) as tot_items
    -> from Sales;
```

OUTPUT:

```
+-----------+
| tot_items |
+-----------+
|        22 |
+-----------+
1 row in set (0.00 sec)
```

4. **Find out average number of items per order.**

```
mysql> select avg(qty) as avg_items
    -> from Sales;
```

OUTPUT:

```
+-----------+
| avg_items |
+-----------+
|    3.1429 |
+-----------+
1 row in set (0.00 sec)
```

5. **Find out the average Quantity for all orders with Price greater than 200.**

```
mysql> select avg(qty) as avg_items
    -> from Sales
    -> where price > 200;
```

OUTPUT:

```
+-----------+
| avg_items |
+-----------+
|    3.4000 |
+-----------+
1 row in set (0.00 sec)
```

6. **Find out what was the minimum price paid for any of the orders.**

```
mysql> select min(price) as min_price
    -> from Sales;
```

OUTPUT:

```
+-----------+
| min_price |
+-----------+
|       160 |
+-----------+
1 row in set (0.00 sec)
```

7. **Find out the highest Price from the given sales table.**

```
mysql> select max(price) as max_price
    -> from Sales;
```

OUTPUT:

```
+-----------+
| max_price |
+-----------+
|      2000 |
+-----------+
1 row in set (0.00 sec)
```

8. **List out unique customers name only from the table.**

```
mysql> select distinct cust
    -> from Sales;
```

OUTPUT:

```
+---------+
| cust    |
+---------+
| Smith   |
| Johnson |
| Baldwin |
| Wood    |
+---------+
4 rows in set (0.00 sec)
```

9. List out name of the customers who have given order in the month of December.

```
mysql> select cust
    -> from Sales
    -> where date like '____-12-__';
```

OUTPUT:

```
+-------+
| cust  |
+-------+
| Smith |
| Wood  |
+-------+
2 rows in set (0.00 sec)
```

10. Find out the total amount of money spent for each of the customers.

```
mysql> select cust, sum(price*qty) as tot_amt
    -> from Sales
    -> group by cust;
```

OUTPUT:

```
+---------+---------+
| cust    | tot_amt |
+---------+---------+
| Smith   |    4440 |
| Johnson |     570 |
| Baldwin |    6500 |
| Wood    |    4000 |
+---------+---------+
4 rows in set (0.00 sec)
```

11. Select all unique customers who have spent more than 1200 in the store.

```
mysql> select distinct cust, sum(price*qty) as amt
    -> from Sales
    -> group by cust
    -> having amt > 1200;
```

OUTPUT:

```
+---------+------+
| cust    | amt  |
+---------+------+
| Smith   | 4440 |
| Baldwin | 6500 |
| Wood    | 4000 |
+---------+------+
3 rows in set (0.00 sec)
```

12. Select all customers that have ordered more than 5 items in total from all their orders.

```
mysql> select cust, sum(qty) as tot_items
    -> from Sales
    -> group by cust
    -> having tot_items > 5;
```

OUTPUT:

```
+---------+-----------+
| cust    | tot_items |
+---------+-----------+
| Smith   |         8 |
| Baldwin |         7 |
+---------+-----------+
2 rows in set (0.00 sec)
```

13. Select all customers who have spent more than 1000, after 10/01/2005.

```
mysql> select cust, sum(price*qty) as amt, date
    -> from Sales
    -> group by cust
    -> having amt > 1000 and date > '2005-01-10';
```

OUTPUT:

```
+---------+------+------------+
| cust    | amt  | date       |
+---------+------+------------+
| Smith   | 4440 | 2005-12-22 |
| Baldwin | 6500 | 2005-07-13 |
| Wood    | 4000 | 2005-12-22 |
+---------+------+------------+
3 rows in set (0.00 sec)
```

14. Select orders in increasing order of order price.

```
mysql> select *
    -> from Sales
    -> order by price asc;
```

OUTPUT:

order_id	date	price	qty	cust
1	2005-12-22	160	2	Smith
2	2005-08-10	190	3	Johnson
4	2005-07-15	420	2	Smith
3	2005-07-13	500	5	Baldwin
6	2005-10-02	820	4	Smith
5	2005-12-22	1000	4	Wood
7	2005-11-03	2000	2	Baldwin

7 rows in set (0.00 sec)

15. Select orders in decreasing order of order price.

```
mysql> select *
    -> from Sales
    -> order by price desc;
```

OUTPUT:

order_id	date	price	qty	cust
7	2005-11-03	2000	2	Baldwin
5	2005-12-22	1000	4	Wood
6	2005-10-02	820	4	Smith
3	2005-07-13	500	5	Baldwin
4	2005-07-15	420	2	Smith
2	2005-08-10	190	3	Johnson
1	2005-12-22	160	2	Smith

7 rows in set (0.00 sec)

ASSIGNMENT 7

Questions for Lab Assignment 6

Table sales:

OrderId	OrderDate	OrderPrice	OrderQuantity	CustomerName
1	2005-12-22	160	2	Smith
2	2005-08-10	190	2	Johnson
3	2005-07-13	500	5	Baldwin
4	2005-07-15	420	2	Smith
5	2005-12-22	1000	4	Wood
6	2005-10-02	820	4	Smith
7	2005-11-03	2000	2	Baldwin
8	2002-12-22	1000	4	Wood
9	2004-12-29	5000	4	Smith

Table products:

Product_id	OrderId	Manufacture_Date	Raw_Material	Vender_id
AZ145	2	2005-12-23	Steel	1
AZ147	6	2002-08-15	Steel	3
CS435	5	2001-11-04	Steel	1
CS783	1	2004-11-03	Plastic	2
CS784	4	2005-11-28	Plastic	2
FD123	2	2005-10-03	Milk	2
FD267	5	2002-21-03	Bread	4
FD333	9	2001-12-12	Milk	1
FD344	3	2005-11-03	Milk	1
GR233	3	2005-11-30	Pulses	2
GR567	6	2005-09-03	Pulses	2

Table vender_info:

Vender_id	Vender_name
1	Smith
2	Wills
3	Johnson
4	Roger

Table venders:

Raw_Material	Venders	Vender_id
Steel	Smith	1
Plastic	Wills	2
Steel	Johnson	3
Milk	Smith	1
Pulses	Wills	2
Bread	Roger	4
Bread	Wills	2
Milk	Wills	3

Solve the following queries for above tables:

1. Display product information which are ordered in the same year of its manufacturing year.

2. Display product information which are ordered in the same year of its manufacturing year where vender is ‘Smith’.

3. Display total number of orders placed in each year.

4. Display total number of orders placed in each year by vender Wills.

5. Display the name of all those persons who are venders and customers both.

6. Display total number of food items ordered every year.

7. Display total number of food items ordered every year made from bread.

8. Display list of product_id whose vender and customer is different.

9. Display all those customers who are ordering products of milk by smith.

10. Display total number of orders by each vender every year.

11. Display name of those venders whose products are sold more than 2000 Rs.Every year.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

21st March 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> create table sales2 as select * from sales;
  ```
  // since this table is similar to table given in A6 with slight modifications
- ```
  mysql> insert into sales2 values
      -> (8, '2002/12/22', 1000, 4, 'Wood'),
      -> (9, '2004/12/29', 5000, 4, 'Smith');
  ```
- ```
  mysql> update sales2
      -> set qty=2
      -> where order_id=2;
  ```
- ```
  mysql> alter table sales2
      -> rename column price to order_price;
  ```
- ```
  mysql> alter table sales2
      -> rename column date to order_date;
  ```
- ```
  mysql> select * from sales2;
  ```

```
mysql> select * from sales2;
+----------+------------+-------------+------+---------+
| order_id | order_date | order_price | qty  | cust    |
+----------+------------+-------------+------+---------+
|        1 | 2005-12-22 |         160 |    2 | Smith   |
|        2 | 2005-08-10 |         190 |    2 | Johnson |
|        3 | 2005-07-13 |         500 |    5 | Baldwin |
|        4 | 2005-07-15 |         420 |    2 | Smith   |
|        5 | 2005-12-22 |        1000 |    4 | Wood    |
|        6 | 2005-10-02 |         820 |    4 | Smith   |
|        7 | 2005-11-03 |        2000 |    2 | Baldwin |
|        8 | 2002-12-22 |        1000 |    4 | Wood    |
|        9 | 2004-12-29 |        5000 |    4 | Smith   |
+----------+------------+-------------+------+---------+
9 rows in set (0.00 sec)
```

- ```
  mysql> create table products
      -> (
      -> product_id       varchar(10),
      -> order_id         int,
      -> mfg_date         date,
      -> raw_mtrl         varchar(10),
      -> vendor_id        int
      -> );
  ```
- ```
  mysql> insert into products values
      -> ('AZ145', 2, '2005-12-23', 'Steel', 1),
      -> ('AZ147', 6, '2002-08-15', 'Steel', 3),
      -> ('CS435', 5, '2001-11-04', 'Steel', 1),
      -> ('CS783', 1, '2004-11-03', 'Plastic', 2),
  ```
  ```

```
    -> ('CS784', 4, '2005-11-28', 'Plastic', 2),
    -> ('FD123', 2, '2005-10-03', 'Milk', 2),
    -> ('FD267', 5, '2002-03-21', 'Bread', 4),
    -> ('FD333', 9, '2001-12-12', 'Milk', 1),
    -> ('FD344', 3, '2005-11-03', 'Milk', 1),
    -> ('GR233', 3, '2005-11-30', 'Pulses', 2),
    -> ('GR567', 6, '2005-09-03', 'Pulses', 2);
```

- `mysql> select * from products;`

```
mysql> select * from products;
+------------+----------+------------+----------+-----------+
| product_id | order_id | mfg_date   | raw_mtrl | vendor_id |
+------------+----------+------------+----------+-----------+
| AZ145      |        2 | 2005-12-23 | Steel    |         1 |
| AZ147      |        6 | 2002-08-15 | Steel    |         3 |
| CS435      |        5 | 2001-11-04 | Steel    |         1 |
| CS783      |        1 | 2004-11-03 | Plastic  |         2 |
| CS784      |        4 | 2005-11-28 | Plastic  |         2 |
| FD123      |        2 | 2005-10-03 | Milk     |         2 |
| FD267      |        5 | 2002-03-21 | Bread    |         4 |
| FD333      |        9 | 2001-12-12 | Milk     |         1 |
| FD344      |        3 | 2005-11-03 | Milk     |         1 |
| GR233      |        3 | 2005-11-30 | Pulses   |         2 |
| GR567      |        6 | 2005-09-03 | Pulses   |         2 |
+------------+----------+------------+----------+-----------+
11 rows in set (0.00 sec)
```

-
```
mysql> create table vendor_info
    -> (
    -> vendor_id        int,
    -> vendor_name      varchar(10)
    -> );
```

-
```
mysql> insert into vendor_info values
    -> (1, 'Smith'),
    -> (2, 'Wills'),
    -> (3, 'Johnson'),
    -> (4, 'Roger');
```

- `mysql> select * from vendor_info;`

```
mysql> select * from vendor_info;
+-----------+-------------+
| vendor_id | vendor_name |
+-----------+-------------+
|         1 | Smith       |
|         2 | Wills       |
|         3 | Johnson     |
|         4 | Roger       |
+-----------+-------------+
4 rows in set (0.00 sec)
```

-
```
mysql> create table vendors
    -> (
    -> raw_mtrl         varchar(10),
    -> vendors          varchar(10),
    -> vendor_id        int
    -> );
```

- ```
  mysql> insert into vendors values
      -> ('Steel', 'Smith', 1),
      -> ('Plastic', 'Wills', 2),
      -> ('Steel', 'Johnson', 3),
      -> ('Milk', 'Smith', 1),
      -> ('Pulses', 'Wills', 2),
      -> ('Bread', 'Roger', 4),
      -> ('Bread', 'Wills', 2),
      -> ('Milk', 'Wills', 3);
  ```
- ```
  mysql> select * from vendors;
  ```

```
mysql> select * from vendors;
+----------+---------+-----------+
| raw_mtrl | vendors | vendor_id |
+----------+---------+-----------+
| Steel    | Smith   |         1 |
| Plastic  | Wills   |         2 |
| Steel    | Johnson |         3 |
| Milk     | Smith   |         1 |
| Pulses   | Wills   |         2 |
| Bread    | Roger   |         4 |
| Bread    | Wills   |         2 |
| Milk     | Wills   |         3 |
+----------+---------+-----------+
8 rows in set (0.00 sec)
```

Queries:

1. **Display product information which are ordered in the same year of its manufacturing year.**

```
mysql> select product_id, products.order_id, mfg_date, raw_mtrl,
vendor_id
    -> from products inner join sales2 on sales2.order_id =
products.order_id
    -> where year(sales2.order_date) = year(products.mfg_date);
```

OUTPUT:

```
+------------+----------+------------+----------+-----------+
| product_id | order_id | mfg_date   | raw_mtrl | vendor_id |
+------------+----------+------------+----------+-----------+
| AZ145      |        2 | 2005-12-23 | Steel    |         1 |
| CS784      |        4 | 2005-11-28 | Plastic  |         2 |
| FD123      |        2 | 2005-10-03 | Milk     |         2 |
| FD344      |        3 | 2005-11-03 | Milk     |         1 |
| GR233      |        3 | 2005-11-30 | Pulses   |         2 |
| GR567      |        6 | 2005-09-03 | Pulses   |         2 |
+------------+----------+------------+----------+-----------+
6 rows in set (0.00 sec)
```

2. **Display product information which are ordered in the same year of its manufacturing year where vender is 'Smith'.**

```
mysql> select product_id, products.order_id, mfg_date, raw_mtrl,
products.vendor_id, vendor_info.vendor_name
    -> from sales2 inner join (products inner join vendor_info on
products.vendor_id = vendor_info.vendor_id) on sales2.order_id =
products.order_id
    -> where year(sales2.order_date) = year(products.mfg_date) and
vendor_info.vendor_name='Smith';
```

OUTPUT:

```
+------------+----------+------------+----------+-----------+-------------+
| product_id | order_id | mfg_date   | raw_mtrl | vendor_id | vendor_name |
+------------+----------+------------+----------+-----------+-------------+
| AZ145      |        2 | 2005-12-23 | Steel    |         1 | Smith       |
| FD344      |        3 | 2005-11-03 | Milk     |         1 | Smith       |
+------------+----------+------------+----------+-----------+-------------+
2 rows in set (0.00 sec)
```

3. **Display total number of orders placed in each year.**

```
mysql> select year(order_date) as year, sum(qty) as no_of_orders
    -> from sales2
    -> group by year;
```

OUTPUT:

```
+------+--------------+
| year | no_of_orders |
+------+--------------+
| 2005 |           21 |
| 2002 |            4 |
| 2004 |            4 |
+------+--------------+
3 rows in set (0.00 sec)
```

4. **Display total number of orders placed in each year by vender Wills.**

```
mysql> select year(order_date) as year, sum(qty) as no_of_orders,
v.vendor_name as name, v.vendor_id as id
    -> from sales2 inner join (products inner join vendor_info v on
products.vendor_id = v.vendor_id) on sales2.order_id =
products.order_id
    -> where v.vendor_name='Wills'
    -> group by year;
```

OUTPUT:

```
+------+--------------+-------+------+
| year | no_of_orders | name  | id   |
+------+--------------+-------+------+
| 2005 |           15 | Wills |    2 |
+------+--------------+-------+------+
1 row in set (0.00 sec)
```

5. **Display the name of all those persons who are venders and customers both.**

```
mysql> select vendor_name as name
    -> from vendor_info
    -> where vendor_name in (select cust from sales2);
```

OUTPUT:

```
+---------+
| name    |
+---------+
| Smith   |
| Johnson |
+---------+
2 rows in set (0.00 sec)
```

6. **Display total number of food items ordered every year.**

```
mysql> select year(order_date) as year, sum(qty) as no_of_orders,
raw_mtrl as item
    -> from sales2 s inner join products p on s.order_id =
p.order_id
    -> where raw_mtrl in ('Milk', 'Bread', 'Pulses')
    -> group by raw_mtrl, year;
```

OUTPUT:

```
+------+--------------+--------+
| year | no_of_orders | item   |
+------+--------------+--------+
| 2005 |            7 | Milk   |
| 2005 |            9 | Pulses |
| 2005 |            4 | Bread  |
| 2004 |            4 | Milk   |
+------+--------------+--------+
4 rows in set (0.00 sec)
```

7. **Display total number of food items ordered every year made from bread.**

```
mysql> select year(order_date) as year, sum(qty) as no_of_orders,
raw_mtrl as item
    -> from sales2 s inner join products p on s.order_id =
p.order_id
    -> where raw_mtrl in ('Milk', 'Bread', 'Pulses')
    -> group by raw_mtrl, year
    -> having raw_mtrl='Bread';
```

OUTPUT:

```
+------+--------------+-------+
| year | no_of_orders | item  |
+------+--------------+-------+
| 2005 |            4 | Bread |
+------+--------------+-------+
1 row in set (0.00 sec)
```

8. **Display list of product_id whose vender and customer is different.**

```
mysql> select product_id
    -> from sales2 s inner join (products p inner join vendor_info
v on p.vendor_id = v.vendor_id) on s.order_id = p.order_id
    -> where s.cust != v.vendor_name;
```

OUTPUT:

```
+------------+
| product_id |
+------------+
| CS783      |
| FD123      |
| AZ145      |
| GR233      |
| FD344      |
| CS784      |
| FD267      |
| CS435      |
| GR567      |
| AZ147      |
+------------+
10 rows in set (0.00 sec)
```

9. **Display all those customers who are ordering products of milk by Smith.**

```
mysql> select cust
    -> from sales2 s inner join (products p inner join vendor_info
v on p.vendor_id = v.vendor_id) on s.order_id = p.order_id
    -> where raw_mtrl='Milk' and vendor_name='Smith';
```

OUTPUT:

```
+---------+
| cust    |
+---------+
| Baldwin |
| Smith   |
+---------+
2 rows in set (0.00 sec)
```

10. **Display total number of orders by each vender every year.**

```
mysql> select sum(qty) as no_of_orders, vendor_name as vendor,
year(order_date) as year
    -> from sales2 s inner join (products p inner join vendor_info
v on p.vendor_id = v.vendor_id) on s.order_id = p.order_id
    -> group by vendor, year;
```

OUTPUT:

```
+--------------+---------+------+
| no_of_orders | vendor  | year |
+--------------+---------+------+
|           15 | Wills   | 2005 |
|           11 | Smith   | 2005 |
|            4 | Roger   | 2005 |
|            4 | Johnson | 2005 |
|            4 | Smith   | 2004 |
+--------------+---------+------+
5 rows in set (0.00 sec)
```

11. Display name of those venders whose products are sold more than 2000 Rs. every year.

```
mysql> select vendor_name, sum(order_price * qty) as tot_price,
year(order_date) as year
    -> from sales2 s inner join (products p inner join vendor_info
v on p.vendor_id = v.vendor_id) on s.order_id = p.order_id
    -> group by year(order_date), vendor_name
    -> having tot_price > 2000;
```

OUTPUT:

```
+-------------+-----------+------+
| vendor_name | tot_price | year |
+-------------+-----------+------+
| Wills       |      7320 | 2005 |
| Smith       |      6880 | 2005 |
| Roger       |      4000 | 2005 |
| Johnson     |      3280 | 2005 |
| Smith       |     20000 | 2004 |
+-------------+-----------+------+
5 rows in set (0.00 sec)
```

ASSIGNMENT 8

DBMS LAB Assignment No. 11
Department of Computer
Engineering Jamia Millia Islamia

- **Select the right option for each of the question by creating the required tables and inserting the required data only.**
- **Suitable data can be assumed if necessary.**

GATE-2010

1. A relational schema for a train reservation database is given below.

Passenger (pid, pname, age)
Reservation (pid, class, tid)

Table: Passenger

pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Table : Reservation

pid	class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

What pids are returned by the following SQL query for the above instance of the tables?

```
SELECT pid

FROM Reservation

WHERE class ‘AC’ AND

    EXISTS (SELECT *

       FROM Passenger

       WHERE age > 65 AND

       Passenger. pid = Reservation.pid)
```

(a) 1, 0 (b) 1, 2 (c) 1, 5 (d) 1, 3

GATE-2009
Common Data for Questions 2 and 3

1. Consider the following relational schema:
Suppliers(sid:integer, sname:string, city:string, street:string) Parts(pid:integer, pname:string, color:string) Catalog(sid:integer, pid:integer, cost:real)
Consider the following relational query on the above database:

```
SELECT S.sname

    FROM Suppliers S

        WHERE S.sid NOT IN (SELECT C.sid FROM Catalog C

                            WHERE C.pid NOT IN (SELECT P.pid FROM Parts P
WHERE P.color<> 'blue'))
```

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?
(a) Find the names of all suppliers who have not supplied a non-blue part.
(b) Find the names of all suppliers who have supplied only blue parts.
(c) Find the names of all suppliers who have not supplied only blue parts.
(d) Find the names of all suppliers who have supplied a non-blue part.

GATE-2004

2. The employee information in a company is stored in the relation Employee (name, sex, salary, deptName)
Consider the following SQL query

```
   Select deptName

         From Employee

         Where sex = 'M'

    Group by deptName

    Having avg(salary) >

         (select avg (salary) from Employee)
```

It returns the names of the department in which

(a) the average salary of male employees is more than the average salary in the company
(b) the average salary is more than the average salary in the company
(c) the average salary of male employees is more than the average salary of all male employees in the company
(d) the average salary of male employees is more than the average salary of employees in the same department.

GATE-2005

1. The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list?

```
Select title

From book as B

Where (Select count(*)

   from book as T

   Where T.price > B.price) < 5
```

(a) Titles of the four most expensive books (b) Title of the fifth most inexpensive book
(c) Title of the fifth most expensive book
(d) Titles of the five most expensive books

GATE – 2006

2. Consider the relation enrolled (student, course) in which (student, course) is the primary key, and the relation paid (student, amount) where student is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:

Query1: select student from enrolled where student in (select student from paid) Query2: select student from paid where student in (select student from enrolled) Query3: select E.student from enrolled E, paid P where E.student = P.student Query4: select student from paid where exists
(select * from enrolled where enrolled.student = paid.student)

Which one of the following statements is correct?

(a) All queries return identical row sets for any database

(b) Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

(c) There exist databases for which Query3 returns strictly fewer rows than Query2.

(d) There exist databases for which Query4 will encounter an integrity violation at runtime.

GATE-2006

1. Consider the relation account (customer, balance) where customer is a primary key and there are no null values. We would like to rank customers according to decreasing balance. The customer with the largest balance gets rank 1. Ties are not broke but ranks are skipped: if exactly two customers have the largest balance they each get rank 1 and rank 2 is not assigned.

Query1:
select A.customer,
count(B.customer) from account
A, account B
where A.balance <=B.balance
group by A.customer

Query2:
select A.customer, 1+count(B.customer)
from account A, account B
where A.balance < B.balance
group by A.customer

Consider these statements about Query1 and Query2.
1. Query1 will produce the same row set as Query2 for some but not all databases.
2. Both Query1 and Query2 are correct implementation of the specification
3. Query1 is a correct implementation of the specification but Query2 is not
4. Neither Query1 nor Query2 is a correct implementation of the specification
5. Assigning rank with a pure relational query takes less time than scanning in decreasing balance order assigning ranks using ODBC.

Which two of the above statements are correct?
(a) 2 and 5
(b) 1 and 3
(c) 1 and 4
(d) 3 and 5

GATE-2011

7. Database table by name Loan_Records is given below.

```
---------------------------------------------------------------------------------------------
BorrowerBank_Manager    Loan_Amount
---------------------------------------------------------------------------------------------
RameshSunderajan10000.0
0 SureshRamgopal5000.00
MaheshSunderajan7000.00
---------------------------------------------------------------------------------------------
```

What is the output of the following SQL query?

```
SELEC  Count(*)

FROM  ( (SELECT Borrower, Bank_Manager

       FROM   Loan_Records) AS S

        NATURAL JOIN (SELECT Bank_Manager,

                             Loan_Amount

                      FROM   Loan_Records) AS T );
```

(a) 3 (b) 5 (c) 9 (d) 6

GATE-2007

7. Consider the table employee(empId, name, department, salary) and the two queries Q1, Q2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?

Q1 : Select e.empId
From employee e
Where not exists
(Select * From employee s where s.department = "5" and
s.salary >=e.salary)
Q2 : Select e.empId
From employee e
Where e.salary >
Any
(Select distinct salary From employee s Where s.department = "5")

(a) Q1 is the correct query
(b) Q2 is the correct query
(c) Both Q1 and Q2 produce the same answer.
(d) Neither Q1 nor Q2 is the correct query.

GATE-2000

8. Given relations r(w, x) and s(y, z), the result of

```
select distinct w, x
from r, s
```

is guaranteed to be same as r, provided
(a) r has no duplicates and s is non-empty
(b) r and s have no duplicates
(c) s has no duplicates and r is non-empty
(d) r and s have the same number of tuples

GATE – 2015

7. Consider the following relations:

Student	
Roll_No	Student_Name
1	Raj
2	Rohit
3	Raj

Performance		
Roll_No	Course	Marks
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Consider the following SQL query.

```
SELECT S. Student_Name, sum (P.Marks)

FROM Student S, Performance P

WHERE S. Roll_No =P.Roll_No

GROUP BY S.Student_Name
```

The number of rows that will be returned by the SQL query is _____.

GATE – 2015

8. Consider the following relation

```
Cinema (theater, address, capacity)
```

Which of the following options will be needed at the end of the SQL query

```
SELECT P1.address FROM Cinema P1
```

such that it always finds the addresses of theaters of theaters with maximum capacity?

(a) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)
(b) WHERE P1.capacity > All (select max (P2. capacity) from Cinema P2)
(c) WHERE P1.capacity >Any (select max (P2. capacity) from Cinema P2)
(d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

28th March 2022

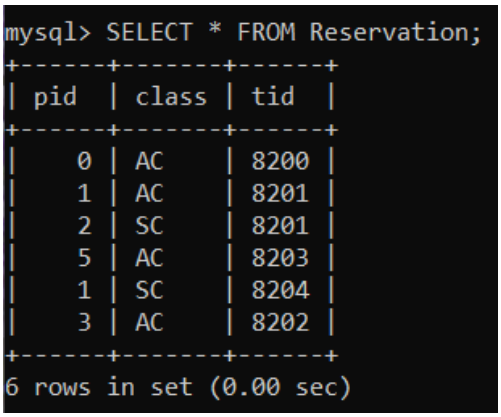
1. GATE 2010

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> CREATE TABLE Passenger
      -> (
      -> pid      INT,
      -> pname    VARCHAR(25),
      -> age      INT
      -> );
  ```
- ```
  mysql> INSERT INTO Passenger VALUES
      -> (0, 'Sachin', 65),
      -> (1, 'Rahul', 66),
      -> (2, 'Sourav', 67),
      -> (3, 'Anil', 69);
  ```
- ```
  mysql> SELECT * FROM Passenger;
  ```

```
mysql> SELECT * FROM Passenger;
+------+--------+------+
| pid  | pname  | age  |
+------+--------+------+
|    0 | Sachin |   65 |
|    1 | Rahul  |   66 |
|    2 | Sourav |   67 |
|    3 | Anil   |   69 |
+------+--------+------+
4 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Reservation
      -> (
      -> pid      INT,
      -> class    VARCHAR(3),
      -> tid      INT
      -> );
  ```
- ```
  mysql> INSERT INTO Reservation VALUES
      -> (0,'AC', 8200),
      -> (1,'AC', 8201),
      -> (2,'SC', 8201),
      -> (5,'AC', 8203),
      -> (1,'SC', 8204),
      -> (3,'AC', 8202);
  ```
- ```
  mysql> SELECT * FROM Reservation;
  ```



```
mysql> SELECT * FROM Reservation;
+------+-------+------+
| pid  | class | tid  |
+------+-------+------+
|    0 | AC    | 8200 |
|    1 | AC    | 8201 |
|    2 | SC    | 8201 |
|    5 | AC    | 8203 |
|    1 | SC    | 8204 |
|    3 | AC    | 8202 |
+------+-------+------+
6 rows in set (0.00 sec)
```

Query:

1. **What pids are returned by the following SQL query for the above instance of the tables?**

```
SELECT pid
FROM Reservation
WHERE class 'AC' AND
     EXISTS (SELECT *
          FROM Passenger
          WHERE age > 65 AND
          Passenger. pid = Reservation.pid);
```

(a) 1, 0 (b) 1, 2 (c) 1, 5 (d) 1, 3
```

OUTPUT:

```
+------+
| pid  |
+------+
|    1 |
|    3 |
+------+
2 rows in set (0.00 sec)
```

ANSWER: Option (d) 1, 3

2.GATE 2010

Creation:

- ```
  mysql> CREATE TABLE Suppliers
      -> (
      -> sid      INT,
      -> sname    VARCHAR(25),
      -> city     VARCHAR(25),
      -> street   VARCHAR(25)
      -> );
  ```

- ```
  mysql> INSERT INTO Suppliers VALUES
      -> (3, 'Dinesh', 'Mumbai', 'Lal Chowk'),
      -> (1, 'Yash', 'Mumbai', 'Greek Rd'),
      -> (5, 'Ayush', 'New Delhi', 'Chawri Bazaar'),
      -> (4, 'Abdullah', 'Bhopal', 'Jalal Rd'),
      -> (2, 'Tarvinder', 'Punjab', 'Bathinda Rd');
  ```

- ```
  mysql> SELECT * FROM Suppliers;
  ```

```
mysql> SELECT * FROM Suppliers;
+------+-----------+-----------+---------------+
| sid  | sname     | city      | street        |
+------+-----------+-----------+---------------+
|    3 | Dinesh    | Mumbai    | Lal Chowk     |
|    1 | Yash      | Mumbai    | Greek Rd      |
|    5 | Ayush     | New Delhi | Chawri Bazaar |
|    4 | Abdullah  | Bhopal    | Jalal Rd      |
|    2 | Tarvinder | Punjab    | Bathinda Rd   |
+------+-----------+-----------+---------------+
5 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Parts
      -> (
      -> pid      INT,
      -> pname    VARCHAR(10),
  ```

```
    -> color    VARCHAR(10)
    -> );
```

- ```
  mysql> INSERT INTO Parts VALUES
      -> (8, 'Pulley', 'Black'),
      -> (3, 'Hand Rail', 'Blue'),
      -> (2, 'Shaper', 'Blue'),
      -> (1, 'Bolts', 'White'),
      -> (6, 'Wheel Rail', 'Red'),
      -> (4, 'Blower', 'Blue');
  ```
- ```
  mysql> SELECT * FROM Parts;
  ```

```
mysql> SELECT * FROM Parts;
+------+------------+-------+
| pid  | pname      | color |
+------+------------+-------+
|    8 | Pulley     | Black |
|    3 | Hand Rail  | Blue  |
|    2 | Shaper     | Blue  |
|    1 | Bolts      | White |
|    6 | Wheel Rail | Red   |
|    4 | Blower     | Blue  |
+------+------------+-------+
6 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Catalogue
      -> (
      -> sid      INT,
      -> pid      INT,
      -> cost     DECIMAL(10,2)
      -> );
  ```
- ```
  mysql> INSERT INTO Catalogue VALUES
      -> (5, 3, 1200.00),
      -> (3, 2, 2000.00),
      -> (2, 4, 1000.00),
      -> (1, 6, 1500.00),
      -> (4, 1, 500.00),
      -> (4, 8, 800.00);
  ```
- ```
  mysql> SELECT * FROM Catalogue;
  ```

```
mysql> SELECT * FROM Catalogue;
+------+------+---------+
| sid  | pid  | cost    |
+------+------+---------+
|    5 |    3 | 1200.00 |
|    3 |    2 | 2000.00 |
|    2 |    4 | 1000.00 |
|    1 |    6 | 1500.00 |
|    4 |    1 |  500.00 |
|    4 |    8 |  800.00 |
+------+------+---------+
6 rows in set (0.00 sec)
```

Query:

1. **Consider the following relational schema:**
```

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

Consider the following relational query on the above database:

```
SELECT S.sname
FROM Suppliers S
WHERE S.sid NOT IN (SELECT C.sid FROM Catalog C
   WHERE C.pid NOT IN (SELECT P.pid FROM Parts P
         WHERE P.color<> 'blue'));
```

Assume that relations corresponding to the above schema are not empty. Which one of the following is the correct interpretation of the above query?

(a) Find the names of all suppliers who have not supplied a non-blue part.
(b) Find the names of all suppliers who have supplied only blue parts.
(c) Find the names of all suppliers who have not supplied only blue parts.
(d) Find the names of all suppliers who have supplied a non-blue part

OUTPUT:

```
+----------+
| sname    |
+----------+
| Yash     |
| Abdullah |
+----------+
2 rows in set (0.01 sec)
```

ANSWER: Option (c) Find the names of all suppliers who have not supplied only blue parts.
EXPLANATION: The innermost subquery returns the pids of the non-blue parts, and since the middle subquery uses NOT IN, therefore this query searches for pids having blue parts and returns the sids of the suppliers. Now since the outermost query also uses a NOT IN statement, it returns the names of suppliers who have NOT supplied only the blue parts. Here, Yash only supplies a Red part and Abdullah supplies both a Blue part and a Black part, so both of them have not supplied only blue parts.

3.GATE 2004

Creation:

- ```
  mysql> CREATE TABLE Employee3
      -> (
      -> name     VARCHAR(25),
      -> sex      CHAR,
      -> salary   DECIMAL(10,2),
      -> deptName VARCHAR(20)
      -> );
  ```
- ```
  mysql> INSERT INTO Employee3 VALUES
      -> ('Amit', 'M', 30000.00, 'Management'),
      -> ('Rita', 'F', 60000.00, 'Headquarters'),
      -> ('Jitendra', 'M', 80000.00, 'Headquarters'),
      -> ('Riya', 'F', 40000.00, 'Research'),
      -> ('Ravish', 'M', 20000.00, 'Outreach'),
      -> ('Prachi', 'F', 25000.00, 'Management'),
      -> ('Sita', 'F', 65000.00, 'Research'),
  ```

```
        -> ('Utkarsh', 'M', 30000.00, 'Research');
```

- `mysql> SELECT * FROM Employee3;`

```
mysql> SELECT * FROM Employee3;
+----------+------+----------+--------------+
| name     | sex  | salary   | deptName     |
+----------+------+----------+--------------+
| Amit     | M    | 30000.00 | Management   |
| Rita     | F    | 60000.00 | Headquarters |
| Jitendra | M    | 80000.00 | Headquarters |
| Riya     | F    | 40000.00 | Research     |
| Ravish   | M    | 20000.00 | Outreach     |
| Prachi   | F    | 25000.00 | Management   |
| Sita     | F    | 65000.00 | Research     |
| Utkarsh  | M    | 30000.00 | Research     |
+----------+------+----------+--------------+
8 rows in set (0.00 sec)
```

Query:

1. **The employee information in a company is stored in the relation**
   `Employee (name, sex, salary, deptName)`
   Consider the following SQL query

```
Select deptName
From Employee
   Where sex = 'M'
Group by deptName
Having avg(salary) >
   (select avg (salary) from Employee);
```

It returns the names of the department in which

(a) the average salary of male employees is more than the average salary in the company
(b) the average salary is more than the average salary in the company
(c) the average salary of male employees is more than the average salary of all male employees in the company
(d) the average salary of male employees is more than the average salary of employees in the same department.

OUTPUT:

```
+--------------+
| deptName     |
+--------------+
| Headquarters |
+--------------+
1 row in set (0.01 sec)
```

ANSWER: Option (a) the average salary of male employees is more than the average salary in the company
EXPLANATION: The subquery has no conditional and therefore returns the average salary of ALL employees in the company, be it male or female. The outer query compares this avg sal by the avg sal of male employees and returns those who have a greater avg sal than those in the company. Here in the table, we can see that the avg(salary) for ALL employees is 43750, while for Headquarters the average salary for male employees is 80000, therefore the result of the query is 'Headquarters'.

4.GATE 2005

Creation:

- ```
  mysql> CREATE TABLE book
      -> (
      -> title    VARCHAR(100),
      -> price    DECIMAL(8,2)
      -> );
  ```
- ```
  mysql> INSERT INTO book VALUES
      -> ('The Invisible Life of Addie Larue', 750.00),
      -> ('Maybe You should Talk to Someone', 550.00),
      -> ('Verity', 600.00),
      -> ('The Giver Of Stars', 700.00),
      -> ('The Love Hypothesis', 800.00),
      -> ('Malibu Rising', 525.00),
      -> ('People we Meet on Vacation', 625.00),
      -> ('The Defining Decade', 650.00);
  ```
- ```
  mysql> SELECT * FROM book;
  ```

```
mysql> SELECT * FROM book;
+-----------------------------------+--------+
| title                             | price  |
+-----------------------------------+--------+
| The Invisible Life of Addie Larue | 750.00 |
| Maybe You should Talk to Someone  | 550.00 |
| Verity                            | 600.00 |
| The Giver Of Stars                | 700.00 |
| The Love Hypothesis               | 800.00 |
| Malibu Rising                     | 525.00 |
| People we Meet on Vacation        | 625.00 |
| The Defining Decade               | 650.00 |
+-----------------------------------+--------+
8 rows in set (0.00 sec)
```

Query:

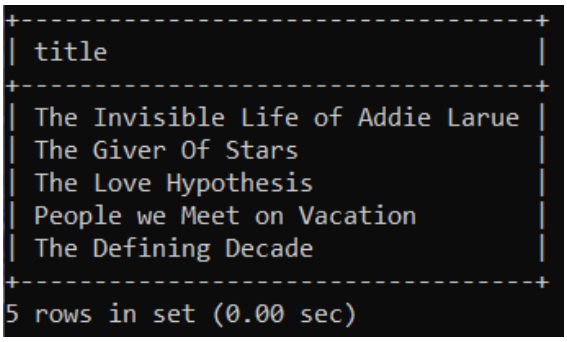
1. **The relation** `book (title, price)` **contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list?**

```
Select title
From book as B
Where (Select count(*)
   from book as T
   Where T.price > B.price) < 5;
```

(a) Titles of the four most expensive books
```

(b) Title of the fifth most inexpensive book
(c) Title of the fifth most expensive book
(d) Titles of the five most expensive books

OUTPUT:



```
+---------------------------------+
| title                           |
+---------------------------------+
| The Invisible Life of Addie Larue |
| The Giver Of Stars              |
| The Love Hypothesis             |
| People we Meet on Vacation      |
| The Defining Decade             |
+---------------------------------+
5 rows in set (0.00 sec)
```

ANSWER: Option (d) Titles of the five most expensive books

EXPLANATION: The inner subquery returns the count of books which have a greater price (expensive) than a specific book taken up from the outer query. The range of values for this subquery is 0,1,2,… and so on. So, for the comparison (< 5), it has the following options 0,1,2,3,4 which adds up to a total of five books. Therefore this query returns the titles of the five most expensive books.

5.GATE 2006

Creation:

- ```
  mysql> CREATE TABLE enrolled
      -> (
      -> student  VARCHAR(30) NOT NULL,
      -> course   VARCHAR(15) NOT NULL,
      -> PRIMARY KEY (student, course)
      -> );
  ```
- ```
  mysql> INSERT INTO enrolled VALUES
      -> ('Arjun', 'Frontend'),
      -> ('Divaker', 'WebD'),
      -> ('Fatima', 'App Dev'),
      -> ('Yash', 'AI'),
      -> ('Pallavi', 'IoT'),
      -> ('Pallavi', 'Linux'),
      -> ('Arjun', 'Game Dev');
  ```
- ```
  mysql> SELECT * FROM enrolled;
  ```
  ```

```
mysql> SELECT * FROM enrolled;
+---------+----------+
| student | course   |
+---------+----------+
| Arjun   | Frontend |
| Arjun   | Game Dev |
| Divaker | WebD     |
| Fatima  | App Dev  |
| Pallavi | IoT      |
| Pallavi | Linux    |
| Yash    | AI       |
+---------+----------+
7 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE paid
      -> (
      -> student  VARCHAR(30) NOT NULL,
      -> amount   DECIMAL(8,2) NOT NULL,
      -> PRIMARY KEY (student)
      -> );
  ```
- ```
  mysql> INSERT INTO paid VALUES
      -> ('Arjun', 1000.00),
      -> ('Fatima', 800.00),
      -> ('Divaker', 450.00),
      -> ('Yash', 650.00),
      -> ('Pallavi', 600.00);
  ```
- ```
  mysql> SELECT * FROM paid;
  ```

```
mysql> SELECT * FROM paid;
+---------+---------+
| student | amount  |
+---------+---------+
| Arjun   | 1000.00 |
| Divaker |  450.00 |
| Fatima  |  800.00 |
| Pallavi |  600.00 |
| Yash    |  650.00 |
+---------+---------+
5 rows in set (0.00 sec)
```

Query:

1. **Consider the relation** `enrolled (student, course)` **in which** `(student, course)` **is the primary key, and the relation** `paid (student, amount)` **where** `student` **is the primary key. Assume no null values and no foreign keys or integrity constraints. Given the following four queries:**

   Query1:
   ```
   select student from enrolled where student in (select student from
   paid)
   ```

   OUTPUT:

```
+---------+
| student |
+---------+
| Arjun   |
| Arjun   |
| Divaker |
| Fatima  |
| Pallavi |
| Pallavi |
| Yash    |
+---------+
```
```

Query2:
```
select student from paid where student in (select student from
    enrolled)
```

OUTPUT:

student
Arjun
Divaker
Fatima
Pallavi
Yash

Query3:
```
select E.student from enrolled E, paid P where E.student = P.student
```

OUTPUT:

student
Arjun
Arjun
Divaker
Fatima
Pallavi
Pallavi
Yash

Query4:
```
select student from paid where exists
(select * from enrolled where enrolled.student = paid.student
```

OUTPUT:

student
Arjun
Divaker
Fatima
Pallavi
Yash

Which one of the following statements is correct?

(a) All queries return identical row sets for any database

(b) Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

(c) There exist databases for which Query3 returns strictly fewer rows than Query2.

(d) There exist databases for which Query4 will encounter an integrity violation at runtime.

ANSWER: Option (b) Query2 and Query4 return identical row sets for all databases but there exist databases for which Query1 and Query2 return different row sets.

EXPLANATION: Q1 and Q3 return the same row sets while Q2 and Q4 return the exact same row sets. Therefore, Option (a) is ruled out. Q3 returns a greater number of rows than Q2, since the primary key in enrolled table is a unique pair of student and course, which reduces down the number of matches. Option (c) is ruled out in this case. Q4 works satisfactorily for all databases. Therefore the answer is option (b).

6.GATE 2006

Creation:

- ```
  mysql> CREATE TABLE account
      -> (
      -> customer VARCHAR(25) NOT NULL,
      -> balance  DECIMAL(10,2) NOT NULL,
      -> PRIMARY KEY (customer)
      -> );
  ```
- ```
  mysql> INSERT INTO account VALUES
      -> ('Younus', 160000.00),
      -> ('Hadiya', 120000.00),
      -> ('Chandler', 170500.00),
      -> ('Joey', 340000.00),
      -> ('Phoebe', 300000.00),
      -> ('Rachel', 200000.00);
  ```
- ```
  mysql> SELECT * FROM account;
  ```

```
mysql> SELECT * FROM account;
+----------+-----------+
| customer | balance   |
+----------+-----------+
| Chandler | 170500.00 |
| Hadiya   | 120000.00 |
| Joey     | 340000.00 |
| Phoebe   | 300000.00 |
| Rachel   | 200000.00 |
| Younus   | 160000.00 |
+----------+-----------+
6 rows in set (0.00 sec)
```

Query:

1. **Consider the relation** `account (customer, balance)` **where** `customer` **is a primary key and there are no null values. We would like to rank customers according to decreasing balance. The customer with the largest balance gets rank 1. Ties are not broke but ranks are skipped: if exactly two customers have the largest balance they each get rank 1 and rank 2 is not assigned**

   Query1:
   ```
   select A.customer, count(B.customer)
       from account A, account B
           where A.balance <=B.balance
           group by A.customer
   ```
   ```

OUTPUT:

customer	count(B.customer)
Younus	5
Hadiya	6
Chandler	4
Rachel	3
Phoebe	2
Joey	1

Query2:
```
select A.customer, 1+count(B.customer)
    from account A, account B
        where A.balance < B.balance
        group by A.customer
```

OUTPUT:

customer	1+count(B.customer)
Younus	5
Hadiya	6
Rachel	3
Phoebe	2
Chandler	4

Consider these statements about Query1 and Query2.

1. Query1 will produce the same row set as Query2 for some but not all databases.

2. Both Query1 and Query2 are correct implementation of the specification

3. Query1 is a correct implementation of the specification but Query2 is not

4. Neither Query1 nor Query2 is a correct implementation of the specification

5. Assigning rank with a pure relational query takes less time than scanning in decreasing balance order assigning ranks using ODBC.

Which two of the above statements are correct?

(a) 2 and 5

(b) 1 and 3

(c) 1 and 4

(d) 3 and 5

ANSWER: Option (c) 1 and 4

EXPLANATION: Q1 returns the ranks as according to number of duplicate values of balance, for 'k' customers holding the same balance, the rank given is 'k' and not rank 1,2,… etc. Q2 can never return a rank 1 unless there's an empty set (`count(B.customer)` = 0 implies `1+count(B.customer) = 1`). Statement 4 is therefore correct, so the correct option is option (c). Statement 1 is true for empty set returned by both queries.

7.GATE 2011

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> CREATE TABLE Loan_Records
      -> (
      -> borrower varchar(15),
      -> bank_manager     varchar(15),
      -> loan_amt         decimal(10,2)
      -> );
  ```
- ```
  mysql> INSERT INTO Loan_Records values
      -> ('Ramesh', 'Sunderajan', 10000.00),
      -> ('Suresh', 'Ramgopal', 15000.00),
      -> ('Mahesh', 'Sunderajan', 7000.00);
  ```
- ```
  mysql> SELECT * FROM Loan_Records;
  ```

```
mysql> SELECT * FROM Loan_Records;
+----------+-------------+----------+
| borrower | bank_manager | loan_amt |
+----------+-------------+----------+
| Ramesh   | Sunderajan   | 10000.00 |
| Suresh   | Ramgopal     | 15000.00 |
| Mahesh   | Sunderajan   |  7000.00 |
+----------+-------------+----------+
3 rows in set (0.00 sec)
```

Query:

1. **What is the output of the following SQL query?**

```
SELECT Count(*)
FROM ((SELECT Borrower, Bank_Manager
      FROM Loan_Records) AS S
      NATURAL JOIN (SELECT Bank_Manager, Loan_Amount
            FROM Loan_Records) AS T);
```

(a) 3 (b) 5 (c) 9 (d) 6

OUTPUT:

```
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.01 sec)
```

ANSWER: Option (b) 5

EXPLANATION: As a result of natural join from first subquery, two entries of bank_manager as 'Sunderajan' are created in the temporary table S, which is joined to temporary table T by a common attribute values of bank_manager. Since there are two duplicate 'Sunderajan', there will be four entries in the result table after joining, each one from S matched on to each one from T (2x2), and one entry as a common entry ('Ramgopal').

8.GATE 2007

Creation:

- `mysql> create table employee4 as select * from employee2;`

  // since this table is similar to table employee2 with slight modifications

- `mysql> SELECT * FROM employee4;`

```
mysql> SELECT * FROM employee4;
+-------+--------+--------+------------+
| empId | name   | salary | department |
+-------+--------+--------+------------+
|   101 | Amit   |  25000 |          1 |
|   102 | Sunil  |  20000 |          1 |
|   103 | Rakesh |  18000 |          5 |
|   104 | Ajay   |  16000 |          5 |
|   105 | Suhail |  20000 |          1 |
|   106 | Arif   |  18000 |          1 |
|   107 | Suresh |  24000 |          5 |
|   108 | Vijay  |  22000 |          5 |
+-------+--------+--------+------------+
8 rows in set (0.00 sec)
```

Query:

1. **Consider the table** `employee(empId, name, department, salary)` **and the two queries Q1, Q2 below. Assuming that department 5 has more than one employee, and we want to find the employees who get higher salary than anyone in the department 5, which one of the statements is TRUE for any arbitrary employee table?**

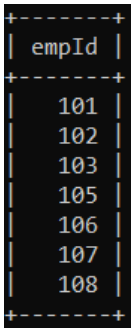
   Q1:
   ```
   Select e.empId
   From employee e
   Where not exists
      (Select * From employee s where s.department = "5" and s.salary
   >=e.salary)
   ```

   OUTPUT:

```
+-------+
| empId |
+-------+
|   101 |
+-------+
```

   Query2:
   ```
   Select e.empId
   From employee e
   Where e.salary > Any
      (Select distinct salary From employee s Where s.department = "5")
   ```

OUTPUT:



empId
101
102
103
105
106
107
108

(a) Q1 is the correct query
(b) Q2 is the correct query
(c) Both Q1 and Q2 produce the same answer.
(d) Neither Q1 nor Q2 is the correct query.

ANSWER: Option (a) Q1 is the correct query
EXPLANATION: ANY keyword matches for any conditional returned by the subquery, while EXISTS ensures for only one match. Therefore, Q1 compares the query to all results from the subquery and returns the correct answer of the employee having higher salary than ANYONE ELSE in department 5. Here, Q1 returns the empId as 101, which is the highest paid employee, but Q2 returns all paid employees except the lowest paid.

9.GATE 2000

Creation:

- ```
  mysql> CREATE TABLE r
      -> (
      -> w        INT,
      -> X        INT
      -> );
  ```
- ```
  mysql> INSERT INTO r VALUES
      -> (4, 8),
      -> (7, 5),
      -> (3, 6),
      -> (2, 1),
      -> (0, 4),
      -> (5, 3);
  ```
- ```
  mysql> SELECT * FROM r;
  ```
  ```

```
mysql> SELECT * FROM r;
+------+------+
| w    | X    |
+------+------+
|    4 |    8 |
|    7 |    5 |
|    3 |    6 |
|    2 |    1 |
|    0 |    4 |
|    5 |    3 |
+------+------+
6 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE s
      -> (
      -> y         INT,
      -> z         INT
      -> );
  ```
- ```
  mysql> INSERT INTO s VALUES
      -> (5, 2),
      -> (9, 0),
      -> (4, 1);
  ```
- ```
  mysql> SELECT * FROM s;
  ```

```
mysql> SELECT * FROM s;
+------+------+
| y    | z    |
+------+------+
|    5 |    2 |
|    9 |    0 |
|    4 |    1 |
+------+------+
3 rows in set (0.00 sec)
```

Query:

1. **Given relations** `r(w, x)` **and** `s(y,z)` **the result of**

   ```
   select distinct w, x
   From r, s
   ```

   OUTPUT:

```
+------+------+
| w    | X    |
+------+------+
|    4 |    8 |
|    7 |    5 |
|    3 |    6 |
|    2 |    1 |
|    0 |    4 |
|    5 |    3 |
+------+------+
```

   (which is the same as the output for ‘r’)

   is guaranteed to be same as r, provided

   (a) r has no duplicates and s is non-empty
   (b) r and s have no duplicates
   (c) s has no duplicates and r is non-empty
   (d) r and s have the same number of tuples

   ANSWER: Option (a) r has no duplicates and s is non-empty
```

10. GATE 2015

Creation:

- ```
  mysql> CREATE TABLE student
      -> (
      -> Roll_No  INT,
      -> Student_Name     VARCHAR(25)
      -> );
  ```

- ```
  mysql> INSERT INTO student VALUES
      -> (1, 'Raj'),
      -> (2, 'Rohit'),
      -> (3, 'Raj');
  ```

- ```
  mysql> SELECT * FROM student;
  ```

```
mysql> SELECT * FROM student;
+---------+--------------+
| Roll_No | Student_Name |
+---------+--------------+
|       1 | Raj          |
|       2 | Rohit        |
|       3 | Raj          |
+---------+--------------+
3 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Performance
      -> (
      -> Roll_No  INT,
      -> Course   VARCHAR(10),
      -> Marks    INT
      -> );
  ```

- ```
  mysql> INSERT INTO Performance VALUES
      -> (1, 'Math', 80),
      -> (1, 'English', 70),
      -> (2, 'Math', 75),
      -> (3, 'English', 80),
      -> (2, 'Physics', 65),
      -> (3, 'Math', 80);
  ```

- ```
  mysql> SELECT * FROM Performance;
  ```

```
mysql> SELECT * FROM Performance;
+---------+---------+-------+
| Roll_No | Course  | Marks |
+---------+---------+-------+
|       1 | Math    |    80 |
|       1 | English |    70 |
|       2 | Math    |    75 |
|       3 | English |    80 |
|       2 | Physics |    65 |
|       3 | Math    |    80 |
+---------+---------+-------+
6 rows in set (0.00 sec)
```

Query:

1. **Consider the following SQL query.**

```
SELECT S.Student_Name, sum (P.Marks)
FROM Student S, Performance P
WHERE S.Roll_No =P.Roll_No
GROUP BY S.Student_Name
```

OUTPUT:

```
+--------------+--------------+
| Student_Name | sum(P.Marks) |
+--------------+--------------+
| Raj          |          310 |
| Rohit        |          140 |
+--------------+--------------+
```

The number of rows that will be returned by the SQL query is _____.

ANSWER: 2

11. GATE 2015

Creation:

- ```
  mysql> CREATE TABLE Cinema
      -> (
      -> theater  VARCHAR(20),
      -> address  VARCHAR(50),
      -> capacity  INT
      -> );
  ```

- ```
  mysql> INSERT INTO Cinema VALUES
       -> ('IMAX Phoenix', 'Vile Parle, Mumbai', 7000),
       -> ('IMAX SGC', 'SGC, Navi Mumbai', 9000),
       -> ('Rathore Cinema', 'Indira Mall, Delhi', 5000),
       -> ('Ujala Cinema', 'Shantiniketan Mall, Ranchi', 2000);
  ```

- ```
  mysql> SELECT * FROM Cinema;
  +---------------+---------------------------+----------+
  | theater       | address                   | capacity |
  +---------------+---------------------------+----------+
  | IMAX Phoenix  | Vile Parle, Mumbai        |     7000 |
  | IMAX SGC      | SGC, Navi Mumbai          |     9000 |
  | Rathore Cinema| Indira Mall, Delhi        |     5000 |
  | Ujala Cinema  | Shantiniketan Mall, Ranchi|     2000 |
  +---------------+---------------------------+----------+
  4 rows in set (0.00 sec)
  ```
  ```

Query:

1. Consider the following relation `Cinema (theater, address, capacity)`
Which of the following options will be needed at the end of the SQL query

```
SELECT P1.address
FROM Cinema P1
```

such that it always finds the addresses of theaters of theaters with maximum capacity?

(a) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)
(b) WHERE P1.capacity > All (select max (P2. capacity) from Cinema P2)
(c) WHERE P1.capacity >Any (select max (P2. capacity) from Cinema P2)
(d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)

OUTPUT:

Query (a) –

```
+---------------------------+
| address                   |
+---------------------------+
| Vile Parle, Mumbai        |
| SGC, Navi Mumbai          |
| Indira Mall, Delhi        |
| Shantiniketan Mall, Ranchi |
+---------------------------+
```

Query (b) –

```
Empty set (0.00 sec)
```

Query (c) –

```
Empty set (0.00 sec)
```

Query (d) –

```
+------------------+
| address          |
+------------------+
| SGC, Navi Mumbai |
+------------------+
```

ANSWER: Option (d) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)

ASSIGNMENT 9

Lab
Assignment-9

Consider the following tables:

Table Student:

snum	sname	major	level	age
101	Jhon	CS	SR	19
102	Smith	CS	JR	20
103	Jacob	ECE	SR	20
104	Tom	CS	JR	20
105	Sid	CS	JR	20
106	Harry	History	SR	21
107	Hellen	CS	JR	21
108	Bob	English	SR	22
109	Andy	ECE	JR	21
110	Charles	History	SR	23

Table Class:

cname	meets_at	room	fid
CSC342	Morning	R128	201
CSC343	Noon	R128	203
CSC345	Night	R154	204
ECE300	Morning	R111	202
ECE301	Noon	R111	203
ENG366	Morning	R154	203
ENG367	Evening	R111	205
HIS320	Evening	R128	205

Table Enrolled:

snum	cname
101	CSC342
101	CSC343
101	CSC345
101	ECE300
101	ENG366
102	CSC343
102	CSC345
102	ECE301
103	ECE300
103	ECE301
104	CSC342
104	ECE301
105	CSC345
105	ECE300
106	ENG366
106	HIS320
107	CSC342
107	ENG366
108	ENG367
108	HIS320
109	ECE300
109	ECE301
110	ENG366
110	HIS320

Table Faculty:

fid	fname	deptid
201	S. Jackson	301
202	M. Shanks	302
203	I. Teach	302
204	A. Zobrah	303
205	M. Jensen	303

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.

Write the SQL statements required to create these relations, including appropriate versions of all primary and foreign key integrity constraints.

Read all questions first and insert values accordingly.

No duplicates should be printed in any of the

answers. Write the following queries in SQL:

1. Find the names of all Juniors(Level = JR) who are enrolled in a class taught by I. Teach.

2. Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach.

3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.

4. Find the names of all students who are enrolled in two class that meet atthe same time.

5. Find the names of faculty members who teach in every room in which some class is taught.

6. Find the names of faculty members for whom the combined enrollment ofthe course that they teach is less than five.

7. For each level, print the level and the average age of students for that level.

8. For all levels except JR, print the level and the average age of studentsfor that level.

9. For each faculty member that has taught class only in room R128 print the faculty member's name and the total number of classes he or she has taught.

10. Find the names of students enrolled in the maximum number of classes.

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

4th April 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> CREATE TABLE Student2
      -> (
      -> snum     INT     NOT NULL,
      -> sname    VARCHAR(15),
      -> major    VARCHAR(10),
      -> level    VARCHAR(3),
      -> age      INT,
      -> PRIMARY KEY (snum)
      -> );
  ```
- ```
  mysql> INSERT INTO Student2 VALUES
      -> (101, 'Jhon', 'CS', 'SR', 19),
      -> (102, 'Smith', 'CS', 'JR', 20),
      -> (103, 'Jacob', 'ECE', 'SR', 20),
      -> (104, 'Tom', 'CS', 'JR', 20),
      -> (105, 'Sid', 'CS', 'JR', 20),
      -> (106, 'Harry', 'History', 'SR', 21),
      -> (107, 'Hellen', 'CS', 'JR', 21),
      -> (108, 'Bob', 'English', 'SR', 22),
      -> (109, 'Andy', 'ECE', 'JR', 21),
      -> (110, 'Charles', 'History', 'SR', 23);
  ```
- ```
  mysql> SELECT * FROM Student2;
  ```

```
mysql> SELECT * FROM Student2;
+------+---------+---------+-------+------+
| snum | sname   | major   | level | age  |
+------+---------+---------+-------+------+
|  101 | Jhon    | CS      | SR    |   19 |
|  102 | Smith   | CS      | JR    |   20 |
|  103 | Jacob   | ECE     | SR    |   20 |
|  104 | Tom     | CS      | JR    |   20 |
|  105 | Sid     | CS      | JR    |   20 |
|  106 | Harry   | History | SR    |   21 |
|  107 | Hellen  | CS      | JR    |   21 |
|  108 | Bob     | English | SR    |   22 |
|  109 | Andy    | ECE     | JR    |   21 |
|  110 | Charles | History | SR    |   23 |
+------+---------+---------+-------+------+
10 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Faculty
      -> (
      -> fid       INT     NOT NULL,
      -> fname     VARCHAR(15),
      -> deptid    INT,
      -> PRIMARY KEY (fid)
      -> );
  ```
- ```
  mysql> INSERT INTO Faculty VALUES
      -> (201, 'S. Jackson', 301),
      -> (202, 'M. Shanks', 302),
      -> (203, 'I. Teach', 302),
      -> (204, 'A. Zobrah', 303),
      -> (205, 'M. Jensen', 303);
  ```
- ```
  mysql> SELECT * FROM Faculty;
  ```

```
mysql> SELECT * FROM Faculty;
+-----+------------+--------+
| fid | fname      | deptid |
+-----+------------+--------+
| 201 | S. Jackson |    301 |
| 202 | M. Shanks  |    302 |
| 203 | I. Teach   |    302 |
| 204 | A. Zobrah  |    303 |
| 205 | M. Jensen  |    303 |
+-----+------------+--------+
5 rows in set (0.00 sec)
```

- ```
  mysql> CREATE TABLE Class
      -> (
      -> cname     VARCHAR(10)      NOT NULL,
      -> meets_at  VARCHAR(10),
      -> room      VARCHAR(5),
      -> fid       INT     NOT NULL,
      -> PRIMARY KEY (cname),
      -> FOREIGN KEY (fid) REFERENCES Faculty(fid)
      -> );
  ```
- ```
  mysql> INSERT INTO Class VALUES
      -> ('CSC342', 'Morning', 'R128', 201),
      -> ('CSC343', 'Noon', 'R128', 203),
      -> ('CSC345', 'Night', 'R154', 204),
  ```
  ```

```
    -> ('ECE300', 'Morning', 'R111', 202),
    -> ('ECE301', 'Noon', 'R111', 203),
    -> ('ENG366', 'Morning', 'R154', 203),
    -> ('ENG367', 'Evening', 'R111', 205),
    -> ('HIS320', 'Evening', 'R128', 205);
```

- `mysql> SELECT * FROM Class;`

```
mysql> SELECT * FROM Class;
+--------+----------+------+-----+
| cname  | meets_at | room | fid |
+--------+----------+------+-----+
| CSC342 | Morning  | R128 | 201 |
| CSC343 | Noon     | R128 | 203 |
| CSC345 | Night    | R154 | 204 |
| ECE300 | Morning  | R111 | 202 |
| ECE301 | Noon     | R111 | 203 |
| ENG366 | Morning  | R154 | 203 |
| ENG367 | Evening  | R111 | 205 |
| HIS320 | Evening  | R128 | 205 |
+--------+----------+------+-----+
8 rows in set (0.00 sec)
```

-
```
mysql> CREATE TABLE Enrolled2
    -> (
    -> snum      INT,
    -> cname     VARCHAR(10),
    -> FOREIGN KEY (snum) REFERENCES Student2(snum),
    -> FOREIGN KEY (cname) REFERENCES Class (cname)
    -> );
```

-
```
mysql> INSERT INTO Enrolled2 VALUES
    -> (101, 'CSC342'),
    -> (101, 'CSC343'),
    -> (101, 'CSC345'),
    -> (101, 'ECE300'),
    -> (101, 'ENG366'),
    -> (102, 'CSC343'),
    -> (102, 'CSC345'),
    -> (102, 'ECE301'),
    -> (103, 'ECE300'),
    -> (103, 'ECE301'),
    -> (104, 'CSC342'),
    -> (104, 'ECE301'),
    -> (105, 'CSC345'),
    -> (105, 'ECE300'),
    -> (106, 'ENG366'),
    -> (106, 'HIS320'),
    -> (107, 'CSC342'),
    -> (107, 'ENG366'),
    -> (108, 'ENG367'),
    -> (108, 'HIS320'),
    -> (109, 'ECE300'),
    -> (109, 'ECE301'),
    -> (110, 'ENG366'),
    -> (110, 'HIS320');
```

- `mysql> SELECT * FROM Enrolled2;`

```
mysql> SELECT * FROM Enrolled2;
+------+--------+
| snum | cname  |
+------+--------+
|  101 | CSC342 |
|  101 | CSC343 |
|  101 | CSC345 |
|  101 | ECE300 |
|  101 | ENG366 |
|  102 | CSC343 |
|  102 | CSC345 |
|  102 | ECE301 |
|  103 | ECE300 |
|  103 | ECE301 |
|  104 | CSC342 |
|  104 | ECE301 |
|  105 | CSC345 |
|  105 | ECE300 |
|  106 | ENG366 |
|  106 | HIS320 |
|  107 | CSC342 |
|  107 | ENG366 |
|  108 | ENG367 |
|  108 | HIS320 |
|  109 | ECE300 |
|  109 | ECE301 |
|  110 | ENG366 |
|  110 | HIS320 |
+------+--------+
24 rows in set (0.00 sec)
```

Queries:

1. **Find the names of all Juniors (Level = JR) who are enrolled in a class taught by I. Teach.**

```
mysql> SELECT sname, cname
    -> FROM Student2 S NATURAL JOIN Enrolled2 E NATURAL JOIN Class
C NATURAL JOIN Faculty F
    -> WHERE S.level='JR' AND F.fname='I. Teach';
```

OUTPUT:

```
+--------+--------+
| sname  | cname  |
+--------+--------+
| Smith  | CSC343 |
| Smith  | ECE301 |
| Tom    | ECE301 |
| Hellen | ENG366 |
| Andy   | ECE301 |
+--------+--------+
5 rows in set (0.00 sec)
```

2. **Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach.**

```
mysql> SELECT S.sname, S.age
    -> FROM Student2 S NATURAL JOIN Enrolled2 E NATURAL JOIN Class
C NATURAL JOIN Faculty F
    -> WHERE major='History' AND fname='I. Teach' AND S.age =
(SELECT MAX(age) FROM Student2 WHERE major='History' OR fname='I.
Teach');
```

OUTPUT:

```
+---------+------+
| sname   | age  |
+---------+------+
| Charles |   23 |
+---------+------+
1 row in set (0.00 sec)
```

3. **Find the names of all classes that either meet in room R128 or have five or more students enrolled.**

```
mysql> SELECT C.cname, COUNT(*) AS count
    -> FROM Student2 S NATURAL JOIN Enrolled2 E NATURAL JOIN Class
C NATURAL JOIN Faculty F
    -> WHERE room='R128' OR C.cname IN (SELECT cname FROM Enrolled2
GROUP BY cname HAVING COUNT(*) >= 5) GROUP BY C.cname;
```

OUTPUT:

```
+--------+-------+
| cname  | count |
+--------+-------+
| CSC342 |     3 |
| CSC343 |     2 |
| HIS320 |     3 |
+--------+-------+
3 rows in set (0.00 sec)
```

4. **Find the names of all students who are enrolled in two class that meet at the same time.**

```
mysql> SELECT sname
    -> FROM Student2 S NATURAL JOIN Enrolled2 E NATURAL JOIN Class
C NATURAL JOIN Faculty F
    -> GROUP BY sname, meets_at
    -> HAVING COUNT(*) >= 2;
```

OUTPUT:

```
+--------+
| sname  |
+--------+
| Jhon   |
| Hellen |
| Smith  |
| Bob    |
+--------+
4 rows in set (0.00 sec)
```

5. **Find the names of faculty members who teach in every room in which some class is taught.**

```
mysql> SELECT fname
    -> FROM Class C NATURAL JOIN Faculty F
    -> GROUP BY fname
    -> HAVING COUNT(*) = (SELECT COUNT(DISTINCT room) FROM Class);
```

OUTPUT:

```
+----------+
| fname    |
+----------+
| I. Teach |
+----------+
1 row in set (0.00 sec)
```

6. **Find the names of faculty members for whom the combined enrollment of the course that they teach is less than five.**

```
mysql> SELECT fname, COUNT(*) AS count
    -> FROM Class C NATURAL JOIN Enrolled2 E NATURAL JOIN Faculty F
    -> GROUP BY fname
    -> HAVING COUNT(*) < 5;
```

OUTPUT:

```
+------------+-------+
| fname      | count |
+------------+-------+
| S. Jackson |     3 |
| M. Shanks  |     4 |
| A. Zobrah  |     3 |
| M. Jensen  |     4 |
+------------+-------+
4 rows in set (0.00 sec)
```

7. **For each level, print the level and the average age of students for that level.**

```
mysql> SELECT level, AVG(age) as avg_age
    -> FROM Student2
    -> GROUP BY level;
```

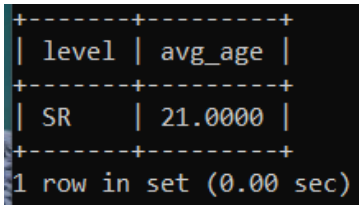
OUTPUT:

```
+-------+---------+
| level | avg_age |
+-------+---------+
| SR    | 21.0000 |
| JR    | 20.4000 |
+-------+---------+
2 rows in set (0.00 sec)
```

8. **For all levels except JR, print the level and the average age of students for that level.**

```
mysql> SELECT level, AVG(age) as avg_age
    -> FROM Student2
    -> WHERE level <> 'JR'
    -> GROUP BY level;
```

OUTPUT:

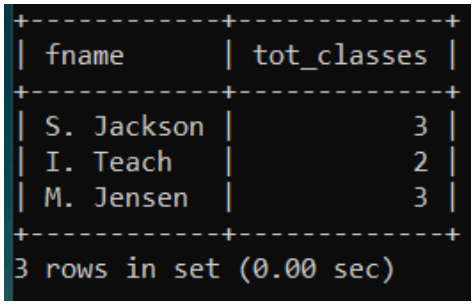


```
+-------+---------+
| level | avg_age |
+-------+---------+
| SR    | 21.0000 |
+-------+---------+
1 row in set (0.00 sec)
```

9. **For each faculty member that has taught class only in room R128 print the faculty member's name and the total number of classes he or she has taught.**

```
mysql> SELECT fname, COUNT(*) as tot_classes
    -> FROM Faculty NATURAL JOIN Class NATURAL JOIN Enrolled2
    -> WHERE room = 'R128'
    -> GROUP BY fname;
```

OUTPUT:

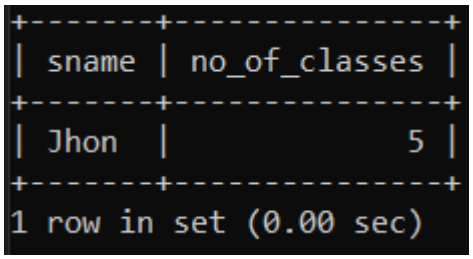


```
+------------+-------------+
| fname      | tot_classes |
+------------+-------------+
| S. Jackson |           3 |
| I. Teach   |           2 |
| M. Jensen  |           3 |
+------------+-------------+
3 rows in set (0.00 sec)
```

10. **Find the names of students enrolled in the maximum number of classes.**

```
mysql> SELECT DISTINCT S.sname, COUNT(snum) AS no_of_classes
    -> FROM Student2 S NATURAL JOIN Enrolled2
    -> WHERE S.snum IN (SELECT E.snum FROM Enrolled2 E GROUP BY
E.snum HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM Enrolled2 E2
GROUP BY E2.snum));
```

OUTPUT:



```
+-------+---------------+
| sname | no_of_classes |
+-------+---------------+
| Jhon  |             5 |
+-------+---------------+
1 row in set (0.00 sec)
```

ASSIGNMENT 10

Questions

1. The following relations keep track of airline flight information:

Flights(*flno:* **integer**, *from:* **string**, *to:* **string**, *distance:* **integer**, *departs:* **time**,
arrives: **time**, *price:* **integer**)
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 aswell; every pilot is certified for some aircraft, and only pilots are certified to fly.Write each of the following queries in SQL.

a. Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.
b. 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.
c. Find the names of pilots whose *salary* is less than the price of the cheapest route from Los Angeles to Honolulu.
d. 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.
e. Find the names of pilots certified for some Boeing aircraft.
f. Find the *aids* of all aircraft that can be used on routes from Los Angeles to Chicago.
g. Identify the routes that can be piloted by every pilot who makes more than $100,000.
h. Print the *enames* of pilots who can operate planes with *cruisingrange* greater than 3000 miles but are not certified on any Boeing aircraft.
i. 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.
j. Compute the difference between the average salary of a pilot and the average salary of all employees (including pilots).
k. Print the name and salary of every nonpilot whose salary is more than the average salary for pilots.

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

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

18th April 2022

Creation:

- ```
  mysql> use 20bcs021_faizan;
  ```
- ```
  mysql> CREATE TABLE Flights
      -> (
      -> flno      INT,
      -> source    VARCHAR(10),
      -> dest      VARCHAR(10),
      -> distance  INT,
      -> departs   TIME,
      -> arrives   TIME,
      -> price     INT
      -> );
  ```
- ```
  mysql> INSERT INTO Flights VALUES
  -> (1,'Bangalore','Mangalore',360,'10:45:00','12:00:00',10000),
  -> (2,'Bangalore','Delhi',5000,'12:15:00','04:30:00',25000),
  -> (3,'Bangalore','Mumbai',3500,'02:15:00','05:25:00',30000),
  -> (4,'Delhi','Mumbai',4500,'10:15:00','12:05:00',35000),
  -> (5,'Delhi','Frankfurt',18000,'07:15:00','05:30:00',90000),
  -> (6,'Bangalore','Frankfurt',19500,'10:00:00','07:45:00',95000),
  -> (7,'Bangalore','Frankfurt',17000,'12:00:00','06:30:00',99000);
  ```
- ```
  mysql> SELECT * FROM Flights;
  ```

```
mysql> SELECT * FROM Flights;
+------+-----------+-----------+----------+----------+----------+-------+
| flno | source    | dest      | distance | departs  | arrives  | price |
+------+-----------+-----------+----------+----------+----------+-------+
|    1 | Bangalore | Mangalore |      360 | 10:45:00 | 12:00:00 | 10000 |
|    2 | Bangalore | Delhi     |     5000 | 12:15:00 | 04:30:00 | 25000 |
|    3 | Bangalore | Mumbai    |     3500 | 02:15:00 | 05:25:00 | 30000 |
|    4 | Delhi     | Mumbai    |     4500 | 10:15:00 | 12:05:00 | 35000 |
|    5 | Delhi     | Frankfurt |    18000 | 07:15:00 | 05:30:00 | 90000 |
|    6 | Bangalore | Frankfurt |    19500 | 10:00:00 | 07:45:00 | 95000 |
|    7 | Bangalore | Frankfurt |    17000 | 12:00:00 | 06:30:00 | 99000 |
+------+-----------+-----------+----------+----------+----------+-------+
```

- ```
  mysql> CREATE TABLE Aircraft
      -> (
      -> aid       INT,
      -> aname     VARCHAR(15),
      -> cruisingrange     INT
      -> );
  ```
- ```
  mysql> INSERT INTO Aircraft VALUES
      -> (123,'Airbus',1000),
      -> (302,'Boeing',5000),
      -> (306,'Jet01',5000),
      -> (378,'Airbus380',8000),
      -> (456,'Aircraft',500),
      -> (789,'Aircraft02',800),
      -> (951,'Aircraft03',1000);
  ```

- ```
  mysql> SELECT * FROM Aircraft;
  ```

```
mysql> SELECT * FROM Aircraft;
+------+------------+---------------+
| aid  | aname      | cruisingrange |
+------+------------+---------------+
|  123 | Airbus     |          1000 |
|  302 | Boeing     |          5000 |
|  306 | Jet01      |          5000 |
|  378 | Airbus380  |          8000 |
|  456 | Aircraft   |           500 |
|  789 | Aircraft02 |           800 |
|  951 | Aircraft03 |          1000 |
+------+------------+---------------+
```

- ```
  mysql> CREATE TABLE Employees
      -> (
      -> eid       INT,
      -> ename     VARCHAR(20),
      -> salary    INT
      -> );
  ```
- ```
  mysql> INSERT INTO Employees VALUES
      -> (1,'Ajay',30000),
      -> (2,'Ajith',85000),
      -> (3,'Arnab',50000),
      -> (4,'Harry',45000),
      -> (5,'Ron',90000),
      -> (6,'Josh',75000),
      -> (7,'Ram',100000),
      -> (8, 'John', 75000),
      -> (9, 'Uttam', 25000);
  ```
- ```
  mysql> SELECT * FROM Employees;
  ```

```
+------+-------+--------+
| eid  | ename | salary |
+------+-------+--------+
|    1 | Ajay  |  30000 |
|    2 | Ajith |  85000 |
|    3 | Arnab |  50000 |
|    4 | Harry |  45000 |
|    5 | Ron   |  90000 |
|    6 | Josh  |  75000 |
|    7 | Ram   | 100000 |
|    8 | John  |  75000 |
|    9 | Uttam |  25000 |
+------+-------+--------+
```

- ```
  mysql> CREATE TABLE Certified
      -> (
      -> eid      INT,
      -> aid      INT
      -> );
  ```
- ```
  mysql> INSERT INTO Certified VALUES
      -> (1,123),
      -> (2,123),
      -> (1,302),
      -> (5,302),
      -> (7,302),
      -> (1,306),
      -> (2,306),
      -> (1,378),
      -> (2,378),
      -> (4,378),
      -> (6,456),
      -> (3,456),
      -> (5,789),
      -> (6,789),
      -> (3,951),
      -> (1,951),
      -> (1,789);
  ```
- ```
  mysql> SELECT * FROM Certified;
  ```

```
mysql> SELECT * FROM Certified;
+------+------+
| eid  | aid  |
+------+------+
|    1 |  123 |
|    2 |  123 |
|    1 |  302 |
|    5 |  302 |
|    7 |  302 |
|    1 |  306 |
|    2 |  306 |
|    1 |  378 |
|    2 |  378 |
|    4 |  378 |
|    6 |  456 |
|    3 |  456 |
|    5 |  789 |
|    6 |  789 |
|    3 |  951 |
|    1 |  951 |
|    1 |  789 |
+------+------+
```
```

Queries:

1. **Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.**

```
mysql> SELECT A.aname, E.ename, E.salary
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE E.salary > 80000;
```

OUTPUT:

```
+------------+-------+--------+
| aname      | ename | salary |
+------------+-------+--------+
| Airbus     | Ajith |  85000 |
| Boeing     | Ram   | 100000 |
| Boeing     | Ron   |  90000 |
| Jet01      | Ajith |  85000 |
| Airbus380  | Ajith |  85000 |
| Aircraft02 | Ron   |  90000 |
+------------+-------+--------+
```

2. **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.**

```
mysql> SELECT E.eid, MAX(A.cruisingrange) AS max_cruiserange,
E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> GROUP BY E.eid
    -> HAVING COUNT(*) > 3;
```

OUTPUT:

```
+------+-----------------+-------+
| eid  | max_cruiserange | ename |
+------+-----------------+-------+
|    1 |            8000 | Ajay  |
+------+-----------------+-------+
```

3. **Find the names of pilots whose salary is less than the price of the cheapest route from Bangalore to Frankfurt.**

```
mysql> SELECT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid INNER JOIN Flights F
    -> WHERE E.salary < (SELECT MIN(price) FROM Flights WHERE
source='Bangalore' AND dest='Frankfurt')
    -> GROUP BY E.ename;
```

OUTPUT:

```
+-------+
| ename |
+-------+
| Arnab |
| Ajay  |
| Ron   |
| Josh  |
| Ajith |
| Harry |
+-------+
```

4. **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.**

```
mysql> SELECT A.aname, AVG(E.salary) AS avg_sal
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.cruisingrange > 1000
    -> GROUP BY A.aid;
```

OUTPUT:

```
+----------+------------+
| aname    | avg_sal    |
+----------+------------+
| Airbus380 | 53333.3333 |
| Jet01    | 57500.0000 |
| Boeing   | 73333.3333 |
+----------+------------+
```

5. **Find the names of pilots certified for some Boeing aircraft.**

```
mysql> SELECT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.aname LIKE 'Boeing%';
```

OUTPUT:

```
+-------+
| ename |
+-------+
| Ajay  |
| Ron   |
| Ram   |
+-------+
```

6. **Find the aid's of all aircraft that can be used on routes from Bangalore to Delhi.**

```
mysql> SELECT A.aid
    -> FROM Aircraft A INNER JOIN Flights F
    -> WHERE A.cruisingrange > (SELECT MIN(distance) FROM Flights
WHERE source='Bangalore' AND dest='Delhi')
    -> GROUP BY A.aid;
```

OUTPUT:

```
+------+
| aid  |
+------+
|  378 |
+------+
```

7. **Identify the routes that can be piloted by every pilot who makes more than $80,000.**

```
mysql> SELECT F.source, F.dest
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid INNER JOIN Flights F
    -> WHERE E.salary > 90000 AND A.cruisingrange > F.distance;
```

OUTPUT:

```
+-----------+-----------+
| source    | dest      |
+-----------+-----------+
| Bangalore | Mangalore |
| Bangalore | Mumbai    |
| Delhi     | Mumbai    |
+-----------+-----------+
```

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

```
mysql> SELECT DISTINCT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.cruisingrange > 3000 AND A.aname NOT LIKE 'Boeing%';
```

OUTPUT:

```
+-------+
| ename |
+-------+
| Ajay  |
| Ajith |
| Harry |
+-------+
```

9. **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.**

```
mysql> (SELECT F.departs
    -> FROM Flights F
    -> WHERE F.source='Madison' AND F.dest='New York' AND
TIME(F.arrives) < '18:00:00')
    -> UNION
    -> (SELECT F1.departs
    -> FROM Flights F1 INNER JOIN Flights F2 ON F1.dest = F2.source
    -> WHERE F1.source='Madison' AND F2.dest='New York'
    -> AND TIME(F1.arrives) < TIME(F2.departs) AND TIME(F2.arrives)
< '18:00:00')
    -> UNION
    -> (SELECT F1.departs
    -> FROM Flights F1 INNER JOIN Flights F2 ON F1.dest = F2.source
INNER JOIN Flights F3 ON F2.dest = F3.source
    -> WHERE F1.source='Madison' AND F3.dest='New York'
    -> AND TIME(F1.arrives) < TIME(F2.departs) AND TIME(F2.arrives)
< TIME(F3.departs) AND TIME(F3.arrives) < '18:00:00');
```

OUTPUT:

```
Empty set (0.00 sec)
```

(No such entry was added, due to complexity of this query)

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

```
mysql> SELECT
    -> ((SELECT AVG(salary) FROM employees) -
    -> (SELECT AVG(salary) FROM employees INNER JOIN
certified ON employees.eid=certified.eid))
    -> AS avg_diff;
```

OUTPUT:

```
+-----------+
| avg_diff  |
+-----------+
| 4477.1242 |
+-----------+
```

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

```
mysql> SELECT E.ename, E.salary
    -> FROM Employees E
    -> WHERE E.eid NOT IN (SELECT DISTINCT eid FROM Certified)
    -> AND E.salary > (SELECT AVG(salary) FROM Employees E1 INNER
JOIN Certified C1 ON E1.eid=C1.eid);
```

OUTPUT:

```
+-------+--------+
| ename | salary |
+-------+--------+
| John  |  75000 |
+-------+--------+
```

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

```
mysql> SELECT DISTINCT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.cruisingrange > 1000;
```

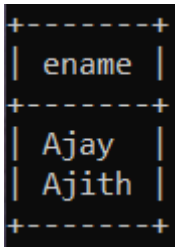
OUTPUT:

```
+-------+
| ename |
+-------+
| Ajay  |
| Ajith |
| Harry |
| Ron   |
| Ram   |
+-------+
```

13. **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.**

```
mysql> SELECT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.cruisingrange > 1000
    -> GROUP BY E.ename
    -> HAVING COUNT(*) >= 2;
```

OUTPUT:

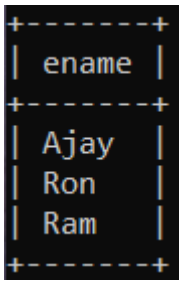


```
+-------+
| ename |
+-------+
| Ajay  |
| Ajith |
+-------+
```

14. **Print the names of employees who are certified only on aircrafts with cruisingrange longer than 1000 miles and who are certified on some Boeing aircraft.**

```
mysql> SELECT E.ename
    -> FROM Aircraft A INNER JOIN (Certified C INNER JOIN Employees
E ON C.eid=E.eid) ON A.aid=C.aid
    -> WHERE A.cruisingrange > 1000 AND A.aname LIKE 'Boeing%';
```

OUTPUT:



```
+-------+
| ename |
+-------+
| Ajay  |
| Ron   |
| Ram   |
+-------+
```

ASSIGNMENT 11

2. Stored Procedures and Functions

1) Write a SQL function and stored procedure for average of three numbers.
Function:

```
create function 19MTavg3no(a int,b int,c int) returns int
begin
        declare sum,avg int;
        set sum = a + b + c;
        set avg = sum/3;
        return avg;
end]
```

```
mysql> create function 19MTavg3no(a int,b int,c int) returns int
    -> begin
    -> declare sum,avg int;
    -> set sum = a + b + c;
    -> set avg = sum/3;
    -> return avg;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> select 19MTavg3no(4,5,6)]
+-------------------+
| 19MTavg3no(4,5,6) |
+-------------------+
|                 5 |
+-------------------+
1 row in set (0.01 sec)
```

Stored Procedure:

```
create procedure 19MTavg3no(In a int,In b int,In c int,Out t int)
begin
        declare sum int;
        set sum = a + b + c;
        set t = sum/3;
end]
```

```
mysql> create procedure 19MTavg3no(In a int,In b int,In c int,Out t int)
    -> begin
    -> declare sum int;
    -> set sum = a + b + c;
    -> set t = sum/3;
    -> end]
Query OK, 0 rows affected (0.01 sec)

mysql> call 19MTavg3no(4,5,6,@avg)]
Query OK, 0 rows affected (0.00 sec)

mysql> select @avg]
+------+
| @avg |
+------+
|    5 |
+------+
1 row in set (0.00 sec)
```

2) Write a SQL function and stored procedure to calculate factorial.

Function:

```
create function 19MTfactorial(n int) returns int
begin
        declare f,i int default 1;
        myloop:loop
        if i > n then
        leave myloop;
        else
        set f = f * i;
        set i = i + 1;
        iterate myloop;
        end if;
        end loop;
        return f;
end]
```

```
mysql> create function 19MTfactorial(n int) returns int
    -> begin
    -> declare f,i int default 1;
    -> myloop:loop
    -> if i > n then
    -> leave myloop;
    -> else
    -> set f = f * i;
    -> set i = i + 1;
    -> iterate myloop;
    -> end if;
    -> end loop;
    -> return f;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> select 19MTfactorial(5)]
+------------------+
| 19MTfactorial(5) |
+------------------+
|              120 |
+------------------+
1 row in set (0.00 sec)
```

Stored Procedure:

```
create procedure 19MTfactorial(In n int,Out fact int)
begin
        declare f,i int default 1;
        myloop:loop
        if i > n then
        leave myloop;
        else
        set f = f * i;
        set i = i + 1;
        iterate myloop;
        end if;
        end loop;
        set fact = f;
end]
```

```
mysql> create procedure 19MTfactorial(In n int,Out fact int)
    -> begin
    -> declare f,i int default 1;
    -> myloop:loop
    -> if i > n then
    -> leave myloop;
    -> else
    -> set f = f * i;
    -> set i = i + 1;
    -> iterate myloop;
    -> end if;
    -> end loop;
    -> set fact = f;
    -> end]
Query OK, 0 rows affected (0.01 sec)

mysql> call 19MTfactorial(5, @factorial)]
Query OK, 0 rows affected (0.01 sec)

mysql> select @factorial]
+------------+
| @factorial |
+------------+
|        120 |
+------------+
1 row in set (0.00 sec)
```

3) Write a SQL function and stored procedure to print fibonacci series upto n terms and its sum.

Function:

```
create function 19MTfibonacci(n int) returns varchar(1000)
begin
        declare i int default 3;
        declare a,temp int default 0;
        declare b,sum int default 1;
        declare str varchar(1000);
        set str = CAST(a as char(2));
        set str = CONCAT(str, " ");
        myloop:loop
        if i > n then
        leave myloop;
        else
        set temp = a + b;
        set a = b;
        set b = temp;
        set i = i+1;
        set sum = sum + temp;
        set str = CONCAT(str, CAST(a as char(2)));
        set str = CONCAT(str, " ");
        end if;
        end loop;
        set str = CONCAT(str, CAST(b as char(2)));
        set str = CONCAT(str, " and sum = ");
        set str = CONCAT(str, CAST(sum as char(3)));
        return str;
end]
```

```
mysql> create function 19MTfibonacci(n int) returns varchar(1000)
    -> begin
    -> declare i int default 3;
    -> declare a,temp int default 0;
    -> declare b,sum int default 1;
    -> declare str varchar(1000);
    -> set str = CAST(a as char(2));
    -> set str = CONCAT(str, " ");
    -> myloop:loop
    -> if i > n then
    -> leave myloop;
    -> else
    -> set temp = a + b;
    -> set a = b;
    -> set b = temp;
    -> set i = i+1;
    -> set sum = sum + temp;
    -> set str = CONCAT(str, CAST(a as char(2)));
    -> set str = CONCAT(str, " ");
    -> end if;
    -> end loop;
    -> set str = CONCAT(str, CAST(b as char(2)));
    -> set str = CONCAT(str, " and sum = ");
    -> set str = CONCAT(str, CAST(sum as char(3)));
    -> return str;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> select 19MTfibonacci(6)]
+--------------------------+
| 19MTfibonacci(6)         |
+--------------------------+
| 0 1 1 2 3 5 and sum = 12 |
+--------------------------+
1 row in set (0.01 sec)
```

Stored Procedure:

```
create procedure 19MTsprofibonacci(In n int, Out retStr varchar(1000))
begin
        declare i int default 3;
        declare a,temp int default 0;
        declare b,sum int default 1;
        declare str varchar(1000);
        set str = CAST(a as char(2));
        set str = CONCAT(str, " ");
        myloop:loop
        if i > n then
        leave myloop;
        else
        set temp = a + b;
        set a = b;
        set b = temp;
        set i = i+1;
        set sum = sum + temp;
        set str = CONCAT(str, CAST(a as char(2)));
        set str = CONCAT(str, " ");
        end if;
        end loop;
        set str = CONCAT(str, CAST(b as char(2)));
        set str = CONCAT(str, " and sum = ");
        set str = CONCAT(str, CAST(sum as char(3)));
        set retStr = str;
end]
```

```
mysql> create procedure 19MTsprofibonacci(In n int, Out retStr varchar(1000))
    -> begin
    -> declare i int default 3;
    -> declare a,temp int default 0;
    -> declare b,sum int default 1;
    -> declare str varchar(1000);
    -> set str = CAST(a as char(2));
    -> set str = CONCAT(str, " ");
    -> myloop:loop
    -> if i > n then
    -> leave myloop;
    -> else
    -> set temp = a + b;
    -> set a = b;
    -> set b = temp;
    -> set i = i+1;
    -> set sum = sum + temp;
    -> set str = CONCAT(str, CAST(a as char(2)));
    -> set str = CONCAT(str, " ");
    -> end if;
    -> end loop;
    -> set str = CONCAT(str, CAST(b as char(2)));
    -> set str = CONCAT(str, " and sum = ");
    -> set str = CONCAT(str, CAST(sum as char(3)));
    -> set retStr = str;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> call 19MTsprofibonacci(6, @str)]
Query OK, 0 rows affected (0.00 sec)

mysql> select @str]
+-------------------------+
| @str                    |
+-------------------------+
| 0 1 1 2 3 5 and sum = 12 |
+-------------------------+
1 row in set (0.00 sec)
```

4) Write a SQL function and stored procedure to calculate age.

Function:

```
create function 19MTcalcAge(dat date) returns varchar(25)
begin
        declare curDate date default CURRENT_DATE();
        declare tempDate date;
        declare year,month,date int default 0;
        declare str varchar(25) default "";
        set year = TIMESTAMPDIFF(YEAR, dat, curDate);
        set month = TIMESTAMPDIFF(MONTH, dat, curDate);
        set month = month - (year * 12);
        set tempDate = DATE_ADD(dat, INTERVAL year YEAR);
        set tempDate = DATE_ADD(tempDate, INTERVAL month MONTH);
        set date = DATEDIFF(curDate, tempDate) + 1;
        set str = CONCAT(str, CAST(year as char(2)));
        set str = CONCAT(str, "Y ");
        set str = CONCAT(str, CAST(month as char(2)));
        set str = CONCAT(str, "M ");
        set str = CONCAT(str, CAST(date as char(2)));
        set str = CONCAT(str, "D");
        return str;
end]
```

```
mysql> create function 19MTcalcAge(dat date) returns varchar(25)
    -> begin
    -> declare curDate date default CURRENT_DATE();
    -> declare tempDate date;
    -> declare year,month,date int default 0;
    -> declare str varchar(25) default "";
    -> set year = TIMESTAMPDIFF(YEAR, dat, curDate);
    -> set month = TIMESTAMPDIFF(MONTH, dat, curDate);
    -> set month = month - (year * 12);
    -> set tempDate = DATE_ADD(dat, INTERVAL year YEAR);
    -> set tempDate = DATE_ADD(tempDate, INTERVAL month MONTH);
    -> set date = DATEDIFF(curDate, tempDate) + 1;
    -> set str = CONCAT(str, CAST(year as char(2)));
    -> set str = CONCAT(str, "Y ");
    -> set str = CONCAT(str, CAST(month as char(2)));
    -> set str = CONCAT(str, "M ");
    -> set str = CONCAT(str, CAST(date as char(2)));
    -> set str = CONCAT(str, "D");
    -> return str;
    -> end]
Query OK, 0 rows affected (0.02 sec)

mysql> select 19MTcalcAge('1992-05-11')]
+--------------------------+
| 19MTcalcAge('1992-05-11') |
+--------------------------+
| 27Y 4M 26D               |
+--------------------------+
1 row in set (0.00 sec)
```

Stored Procedure:

```
create procedure 19MTsprocalcAge(In dat date, Out retStr varchar(25))
begin
        declare curDate date default CURRENT_DATE();
        declare tempDate date;
        declare year,month,date int default 0;
        declare str varchar(25) default "";
        set year = TIMESTAMPDIFF(YEAR, dat, curDate);
        set month = TIMESTAMPDIFF(MONTH, dat, curDate);
        set month = month - (year * 12);
        set tempDate = DATE_ADD(dat, INTERVAL year YEAR);
        set tempDate = DATE_ADD(tempDate, INTERVAL month MONTH);
        set date = DATEDIFF(curDate, tempDate) + 1;
        set str = CONCAT(str, CAST(year as char(2)));
        set str = CONCAT(str, "Y ");
        set str = CONCAT(str, CAST(month as char(2)));
        set str = CONCAT(str, "M ");
        set str = CONCAT(str, CAST(date as char(2)));
        set str = CONCAT(str, "D");
        set retStr = str;
end]
```

```
mysql> create procedure 19MTsprocalcAge(In dat date, Out retStr varchar(25))
    -> begin
    -> declare curDate date default CURRENT_DATE();
    -> declare tempDate date;
    -> declare year,month,date int default 0;
    -> declare str varchar(25) default "";
    -> set year = TIMESTAMPDIFF(YEAR, dat, curDate);
    -> set month = TIMESTAMPDIFF(MONTH, dat, curDate);
    -> set month = month - (year * 12);
    -> set tempDate = DATE_ADD(dat, INTERVAL year YEAR);
    -> set tempDate = DATE_ADD(tempDate, INTERVAL month MONTH);
    -> set date = DATEDIFF(curDate, tempDate) + 1;
    -> set str = CONCAT(str, CAST(year as char(2)));
    -> set str = CONCAT(str, "Y ");
    -> set str = CONCAT(str, CAST(month as char(2)));
    -> set str = CONCAT(str, "M ");
    -> set str = CONCAT(str, CAST(date as char(2)));
    -> set str = CONCAT(str, "D");
    -> set retStr = str;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> call 19MTsprocalcAge('1992-05-11',@age)]
Query OK, 0 rows affected (0.00 sec)

mysql> select @age]
+------------+
| @age       |
+------------+
| 27Y 4M 26D |
+------------+
1 row in set (0.00 sec)
```

5) Write a SQL function and stored procedure to count the total number of employees present in employee table.

Function:

```
create function 19MTtotalNoEmployees() returns int
begin
        declare s int;
        select count(*) from EMPLOYEE into s;
        return s;
end]
```

```
mysql> create function 19MTtotalNoEmployees() returns int
    -> begin
    -> declare s int;
    -> select count(*) from EMPLOYEE into s;
    -> return s;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> select 19MTtotalNoEmployees()]
+-------------------------+
| 19MTtotalNoEmployees() |
+-------------------------+
|                      12 |
+-------------------------+
1 row in set (0.00 sec)
```

Stored Procedure:

```
create procedure 19MTtotalNoEmployees(Out count int)
begin
        declare s int;
        select count(*) from EMPLOYEE into s;
        set count = s;
end]
```

```
mysql> create procedure 19MTtotalNoEmployees(Out count int)
    -> begin
    -> declare s int;
    -> select count(*) from EMPLOYEE into s;
    -> set count = s;
    -> end]
Query OK, 0 rows affected (0.01 sec)

mysql> call 19MTtotalNoEmployees(@res)]
Query OK, 1 row affected (0.00 sec)

mysql> select @res]
+------+
| @res |
+------+
|   12 |
+------+
1 row in set (0.00 sec)
```

6) Write a SQL function and stored procedure to calculate the budget of the department.

Function:

```
create function 19MTcalcBudget(dept varchar(30)) returns int
begin
        declare deptnumber varchar(5);
        declare budget int default 0;
        select deptno from DEPARTMENT where dname = dept into deptnumber;
        select sum(sal) from EMPLOYEE where deptno = deptnumber into budget;
        return budget;
end]
```

```
mysql> create function 19MTcalcBudget(dept varchar(30)) returns int
    -> begin
    -> declare deptnumber varchar(5);
    -> declare budget int default 0;
    -> select deptno from DEPARTMENT where dname = dept into deptnumber;
    -> select sum(sal) from EMPLOYEE where deptno = deptnumber into budget;
    -> return budget;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> select 19MTcalcBudget('Accounting')]
+------------------------------+
| 19MTcalcBudget('Accounting') |
+------------------------------+
|                        67000 |
+------------------------------+
1 row in set (0.02 sec)
```

Stored Procedure:

```
create procedure 19MTcalcBudget(In dept varchar(30), Out budget int)
begin
        declare deptnumber varchar(5);
        declare sumSal int default 0;
        select deptno from DEPARTMENT where dname = dept into deptnumber;
        select sum(sal) from EMPLOYEE where deptno = deptnumber into sumSal;
        set budget = sumSal;
end]
```

```
mysql> create procedure 19MTcalcBudget(In dept varchar(30), Out budget int)
    -> begin
    -> declare deptnumber varchar(5);
    -> declare sumSal int default 0;
    -> select deptno from DEPARTMENT where dname = dept into deptnumber;
    -> select sum(sal) from EMPLOYEE where deptno = deptnumber into sumSal;
    -> set budget = sumSal;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> call 19MTcalcBudget('Accounting',@res)]
Query OK, 1 row affected (0.14 sec)

mysql> select @res]
+--------+
| @res   |
+--------+
| 67000 |
+--------+
1 row in set (0.01 sec)
```

7) Write a SQL function and stored procedure to print the following message:

"Hello <name> How are you".

Function:

```
create function 19MTprintMsg(name varchar(50)) returns varchar(100)
begin
        declare msg varchar(100) default "Hello ";
        set msg = CONCAT(msg, name);
        set msg = CONCAT(msg, " How are you?");
        return msg;
end]
```

```
mysql> create function 19MTprintMsg(name varchar(50)) returns varchar(100)
    -> begin
    -> declare msg varchar(100) default "Hello ";
    -> set msg = CONCAT(msg, name);
    -> set msg = CONCAT(msg, " How are you?");
    -> return msg;
    -> end]
Query OK, 0 rows affected (0.03 sec)

mysql> select 19MTprintMsg('Salman')]
+--------------------------+
| 19MTprintMsg('Salman')   |
+--------------------------+
| Hello Salman How are you? |
+--------------------------+
1 row in set (0.00 sec)
```

Stored Procedure:

```
create procedure 19MTprintMsg(In name varchar(50),Out message varchar(100))
begin
        declare msg varchar(100) default "Hello ";
        set msg = CONCAT(msg, name);
        set msg = CONCAT(msg, " How are you?");
        set message = msg;
end]
```

```
mysql> create procedure 19MTprintMsg(In name varchar(50),Out message varchar(100))
    -> begin
    -> declare msg varchar(100) default "Hello ";
    -> set msg = CONCAT(msg, name);
    -> set msg = CONCAT(msg, " How are you?");
    -> set message = msg;
    -> end]
Query OK, 0 rows affected (0.01 sec)

mysql> call 19MTprintMsg('Salman', @message)]
Query OK, 0 rows affected (0.00 sec)

mysql> select @message]
+--------------------------+
| @message                 |
+--------------------------+
| Hello Salman How are you? |
+--------------------------+
1 row in set (0.00 sec)
```

3. Triggers

Creating tables:

1. Employee

```
create table Employee (
Eid varchar(5) primary key,
Ename varchar(50),
Esal varchar(6)
);
```

```
mysql> create table Employee (
    -> Eid varchar(5) primary key,
    -> Ename varchar(50),
    -> Esal varchar(6)
    -> );
Query OK, 0 rows affected (0.87 sec)
```

2. LogTable

```
create table LogTable (
User varchar(50),
Operation varchar(20),
Time varchar(20),
Peid varchar(5),
Pename varchar(50),
Pesal varchar(6),
Neid varchar(5),
Nename varchar(50),
Nesal varchar(6)
);
```

```
mysql> create table LogTable (
    -> User varchar(50),
    -> Operation varchar(20),
    -> Time varchar(20),
    -> Peid varchar(5),
    -> Pename varchar(50),
    -> Pesal varchar(6),
    -> Neid varchar(5),
    -> Nename varchar(50),
    -> Nesal varchar(6)
    -> );
Query OK, 0 rows affected (0.29 sec)
```

1) Insert Trigger

```
create trigger insertTrig after insert on Employee for each row
begin
insert into LogTable values (user(),'Insert',now(),'-','-','-',new.Eid,new.Ename,new.Esal);
end]
```

```
mysql> create trigger insertTrig after insert on Employee for each row
    -> begin
    -> insert into LogTable values (user(),'Insert',now(),'-','-','-',new.Eid,new.Ename,new.Esal);
    -> end]
Query OK, 0 rows affected (0.19 sec)

mysql> insert into Employee values ('E0001','Salman Zafar','79350')]
Query OK, 1 row affected (0.40 sec)

mysql> select * from LogTable]
+------------------+-----------+---------------------+------+--------+-------+-------+--------------+-------+
| User             | Operation | Time                | Peid | Pename | Pesal | Neid  | Nename       | Nesal |
+------------------+-----------+---------------------+------+--------+-------+-------+--------------+-------+
| Salman@localhost | Insert    | 2019-10-16 22:04:01 | -    | -      | -     | E0001 | Salman Zafar | 79350 |
+------------------+-----------+---------------------+------+--------+-------+-------+--------------+-------+
1 row in set (0.02 sec)
```

2) Update Trigger

```
create trigger updateTrig after update on Employee for each row
begin
insert into LogTable values
(user(),'Update',now(),old.Eid,old.Ename,old.Esal,new.Eid,new.Ename,new.Esal);
end]
```

```
mysql> create trigger updateTrig after update on Employee for each row
    -> begin
    -> insert into LogTable values (user(),'Update',now(),old.Eid,old.Ename,old.Esal,new.Eid,new.Ename,new.Esal);
    -> end]
Query OK, 0 rows affected (0.13 sec)

mysql> update Employee set Esal = '100000' where Eid = 'E0001']
Query OK, 1 row affected (0.28 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from LogTable]
+------------------+-----------+---------------------+-------+--------------+-------+-------+--------------+--------+
| User             | Operation | Time                | Peid  | Pename       | Pesal | Neid  | Nename       | Nesal  |
+------------------+-----------+---------------------+-------+--------------+-------+-------+--------------+--------+
| Salman@localhost | Insert    | 2019-10-16 22:04:01 | -     | -            | -     | E0001 | Salman Zafar | 79350  |
| Salman@localhost | Update    | 2019-10-16 22:36:16 | E0001 | Salman Zafar | 79350 | E0001 | Salman Zafar | 100000 |
+------------------+-----------+---------------------+-------+--------------+-------+-------+--------------+--------+
2 rows in set (0.00 sec)
```

3) Delete Trigger

```
create trigger deleteTrig after delete on Employee for each row
begin
insert into LogTable values (user(),'Delete',now(),old.Eid,old.Ename,old.Esal,'-','-','-');
end]
```

```
mysql> create trigger deleteTrig after delete on Employee for each row
    -> begin
    -> insert into LogTable values (user(),'Delete',now(),old.Eid,old.Ename,old.Esal,'-','-','-');
    -> end]
Query OK, 0 rows affected (0.07 sec)

mysql> delete from Employee where Eid = 'E0001']
Query OK, 1 row affected (0.08 sec)

mysql> select * from LogTable]
+------------------+-----------+---------------------+-------+--------------+--------+-------+--------------+--------+
| User             | Operation | Time                | Peid  | Penane       | Pesal  | Neid  | Nename       | Nesal  |
+------------------+-----------+---------------------+-------+--------------+--------+-------+--------------+--------+
| Salman@localhost | Insert    | 2019-10-16 22:04:01 | -     | -            | -      | E0001 | Salman Zafar | 79350  |
| Salman@localhost | Update    | 2019-10-16 22:36:16 | E0001 | Salman Zafar | 79350  | E0001 | Salman Zafar | 100000 |
| Salman@localhost | Delete    | 2019-10-16 22:41:42 | E0001 | Salman Zafar | 100000 | -     | -            | -      |
+------------------+-----------+---------------------+-------+--------------+--------+-------+--------------+--------+
3 rows in set (0.00 sec)
```

4. Cursor

i) Write a cursor to output salary of all employees in a string.

```
create procedure mypro(out s varchar(6))
begin
declare f int default 1;
declare str longtext default "";
declare cur cursor for select Esal from Employee;
declare continue handler for not found set f=0;
open cur;
myloop:ioop
fetch cur into s:
if f=0 then
leave myloop:
else
set str = CONCAT(str," ",s);
end if:
end loop;
close cur:
select str:
end]
```

```
mysql> select * from Employee]
+-------+-------+-------+
| Eid   | Ename | Esal  |
+-------+-------+-------+
| E0002 | ABC   | 30000 |
| E0003 | XYZ   | 40000 |
| E0004 | DEF   | 50000 |
+-------+-------+-------+
3 rows in set (0.00 sec)

mysql> create procedure mypro(out s varchar(6))
    -> begin
    -> declare f int default 1;
    -> declare str longtext default "";
    -> declare cur cursor for select Esal from Employee;
    -> declare continue handler for not found set f=0;
    -> open cur;
    -> myloop:loop
    -> fetch cur into s;
    -> if f=0 then
    -> leave myloop;
    -> else
    -> set str = CONCAT(str," ",s);
    -> end if;
    -> end loop;
    -> close cur;
    -> select str;
    -> end]
Query OK, 0 rows affected (0.00 sec)

mysql> call mypro(@s)]
+--------------------+
| str                |
+--------------------+
|  30000 40000 50000 |
+--------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

FAIZAN CHOUDHARY

20BCS021

DBMS LAB

25th April 2022

1. Write a SQL function and stored procedure for average of three numbers.

Function:

```
mysql> use 20bcs021_faizan;
mysql> DELIMITER ]          // to change the default delimiter to ]
mysql> CREATE FUNCTION 21CSavg3no (a INT, b INT, c INT)
    -> RETURNS INTEGER
    -> BEGIN
    ->     DECLARE sum, avg INT;
    ->     SET sum = a+b+c;
    ->     SET avg = sum/3;
    ->     RETURN avg;
    -> END]
mysql> DELIMITER ;          // to change back the default delimiter to ;
```

Function Query:

```
mysql> SELECT 21CSavg3no (4,5,6);
```

OUTPUT:

```
+-------------------+
| 21CSavg3no(4,5,6) |
+-------------------+
|                 5 |
+-------------------+
```

Stored Procedure:

```
mysql> DELIMITER ]
mysql> CREATE PROCEDURE 21CSavg3no (IN a INT, IN b INT, IN c
INT, OUT t INT)
    -> BEGIN
    ->     DECLARE sum INT;
    ->     SET sum = a+b+c;
    ->     SET t = sum/3;
    -> END]
```

Stored Procedure Query:

```
mysql> CALL 21CSavg3no (4,5,5,@avg)]
mysql> SELECT @avg]                    // since it returns an INT
```

OUTPUT:

```
+------+
| @avg |
+------+
|    5 |
+------+
```

2. Write a SQL function and stored procedure to calculate factorial.

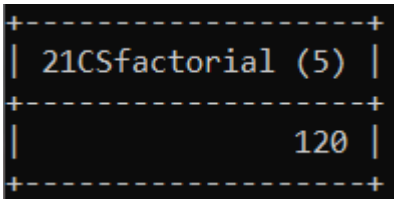
Function:

- ```
  mysql> CREATE FUNCTION 21CSfactorial (n INT)
      -> RETURNS INT
      -> BEGIN
      ->    DECLARE f,i INT DEFAULT 1;
      ->    myloop:LOOP
      ->    IF i > n THEN
      ->         LEAVE myloop;
      ->    ELSE
      ->         SET f = f * i;
      ->         SET i = i + 1;
      ->    ITERATE myloop;
      ->    END IF;
      ->    END LOOP;
      ->    RETURN f;
      -> END]
  ```

Function Query:

```
mysql> SELECT 21CSfactorial (5)]
```

OUTPUT:



```
+-------------------+
| 21CSfactorial (5) |
+-------------------+
|               120 |
+-------------------+
```

Stored Procedure:

- ```
  mysql> CREATE PROCEDURE 21CSfactorial (IN n INT, OUT fact INT)
      -> BEGIN
      ->    DECLARE f,i INT DEFAULT 1;
      ->     myloop:LOOP
      ->     IF i > n THEN
      ->         LEAVE myloop;
      ->    ELSE
  ```

```
->          SET f = f * i;
->          SET i = i + 1;
->    ITERATE myloop;
->    END IF;
->    END LOOP;
->    SET fact = f;
-> END]
```

Stored Procedure Query:

```
mysql> CALL 21CSfactorial (6,@fact)]
mysql> SELECT @fact]
```

OUTPUT:

```
+-------+
| @fact |
+-------+
|   720 |
+-------+
```

3. Write a SQL function and stored procedure to print fibonacci series upto n terms and its sum.

Function:

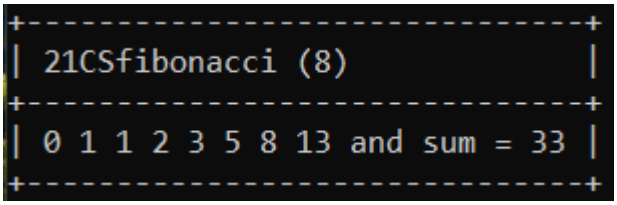
- ```
  mysql> CREATE FUNCTION 21CSfibonacci (n INT)
      -> RETURNS VARCHAR(1000)
      -> BEGIN
      ->    DECLARE i INT DEFAULT 3;
      ->    DECLARE a, temp INT DEFAULT 0;
      ->    DECLARE b, sum INT DEFAULT 1;
      ->    DECLARE str VARCHAR(1000);
      ->    SET str = CAST(a AS CHAR(2));
      ->    SET str = CONCAT(str, " ");
      ->    myloop:LOOP
      ->    IF i > n THEN
      ->         LEAVE myloop;
      ->    ELSE
      ->         SET temp = a + b;
      ->         SET a = b;
      ->         SET b = temp;
  ```
  ```

```
->          SET i = i + 1;
->          SET sum = sum + temp;
->          SET str = CONCAT(str, CAST(a AS CHAR(2)));
->          SET str = CONCAT(str, " ");
->     END IF;
->     END LOOP;
->     SET str = CONCAT(str, CAST(b AS CHAR(2)));
->     SET str = CONCAT(str, " and sum = ");
->     SET str = CONCAT(str, CAST(sum AS CHAR(3)));
->     RETURN str;
-> END]
```

Function Query:

```
mysql> SELECT 21CSfibonacci (8)]
```

OUTPUT:



```
+-------------------------------+
| 21CSfibonacci (8)             |
+-------------------------------+
| 0 1 1 2 3 5 8 13 and sum = 33 |
+-------------------------------+
```

Stored Procedure:

- ```
  mysql> CREATE PROCEDURE 21CSfibonacci (IN n INT, OUT retStr
  VARCHAR(1000))
  ```

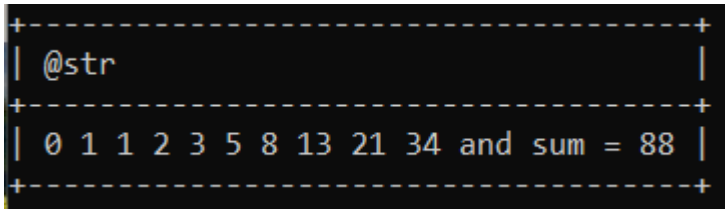
```
-> BEGIN
->     DECLARE i INT DEFAULT 3;
->     DECLARE a, temp INT DEFAULT 0;
->     DECLARE b, sum INT DEFAULT 1;
->     DECLARE str VARCHAR(1000);
->     SET str = CAST(a AS CHAR(2));
->     SET str = CONCAT(str, " ");
->     myloop:LOOP
->     IF i > n THEN
->          LEAVE myloop;
->     ELSE
->          SET temp = a + b;
->          SET a = b;
->          SET b = temp;
```
```

```
->          SET i = i + 1;
->          SET sum = sum + temp;
->          SET str = CONCAT(str, CAST(a AS CHAR(2)));
->          SET str = CONCAT(str, " ");
->     END IF;
->     END LOOP;
->     SET str = CONCAT(str, CAST(b AS CHAR(2)));
->     SET str = CONCAT(str, " and sum = ");
->     SET str = CONCAT(str, CAST(sum AS CHAR(3)));
->     SET retStr = str;
-> END]
```

Stored Procedure Query:

```
mysql> CALL 21CSfibonacci (10,@str)]
mysql> SELECT @str]
```

OUTPUT:



4. Write a SQL function and stored procedure to calculate age.

Function:

- ```
  mysql> CREATE FUNCTION 21CScalcAge(dat DATE)
      -> RETURNS VARCHAR(25)
      -> BEGIN
      ->     DECLARE curDate DATE DEFAULT CURRENT_DATE();
      ->     DECLARE tempDate DATE;
      ->     DECLARE year, month, date INT DEFAULT 0;
      ->     DECLARE str VARCHAR(25) DEFAULT "";
      ->     SET year = TIMESTAMPDIFF(YEAR, dat, curDate);
      ->     SET month = TIMESTAMPDIFF(MONTH, dat, curDate);
      ->     SET month = month - (year * 12);
      ->     SET tempDate = DATE_ADD(dat, INTERVAL year YEAR);
      ->     SET tempDate = DATE_ADD(tempDate, INTERVAL month
  MONTH);
  ```
  ```

```
    ->     SET date = DATEDIFF(curDate, tempDate) + 1;
    ->     SET str = CONCAT(str, CAST(year AS char(2)));
    ->     SET str = CONCAT(str, "Y ");
    ->     SET str = CONCAT(str, CAST(month AS char(2)));
    ->     SET str = CONCAT(str, "M ");
    ->     SET str = CONCAT(str, CAST(date AS char(2)));
    ->     SET str = CONCAT(str, "D");
    ->     RETURN str;
    -> END]
```

Function Query:

```
mysql> SELECT 21CScalcAge ('2002-03-30')]
```

OUTPUT:

```
+----------------------------+
| 21CScalcAge ('2002-03-30') |
+----------------------------+
| 20Y 0M 28D                 |
+----------------------------+
```

Stored Procedure:

- ```
  mysql> CREATE PROCEDURE 21CScalcAge(IN dat DATE, OUT retStr VARCHAR(25))
      -> BEGIN
      ->     DECLARE curDate DATE DEFAULT CURRENT_DATE();
      ->     DECLARE tempDate DATE;
      ->     DECLARE year, month, date INT DEFAULT 0;
      ->     DECLARE str VARCHAR(25) DEFAULT "";
      ->     SET year = TIMESTAMPDIFF(YEAR, dat, curDate);
      ->     SET month = TIMESTAMPDIFF(MONTH, dat, curDate);
      ->     SET month = month - (year * 12);
      ->     SET tempDate = DATE_ADD(dat, INTERVAL year YEAR);
      ->     SET tempDate = DATE_ADD(tempDate, INTERVAL month
  MONTH);
      ->     SET date = DATEDIFF(curDate, tempDate) + 1;
      ->     SET str = CONCAT(str, CAST(year AS char(2)));
      ->     SET str = CONCAT(str, "Y ");
      ->     SET str = CONCAT(str, CAST(month AS char(2)));
      ->     SET str = CONCAT(str, "M ");
  ```
  ```

```
    ->     SET str = CONCAT(str, CAST(date AS char(2)));
    ->     SET str = CONCAT(str, "D");
    ->     SET retStr = str;
    -> END]
```

Stored Procedure Query:

```
mysql> CALL 21CScalcAge ('2001-06-26', @age)]
mysql> SELECT @age]
```

OUTPUT:

```
+------------+
| @age       |
+------------+
| 20Y 10M 1D |
+------------+
```

5. Write a SQL function and stored procedure to count the total number of employees present in the employee table.

Employee table used:

```
mysql> SELECT * FROM Employee3]
```

```
+----------+------+----------+--------------+
| name     | sex  | salary   | deptName     |
+----------+------+----------+--------------+
| Amit     | M    | 30000.00 | Management   |
| Rita     | F    | 60000.00 | Headquarters |
| Jitendra | M    | 80000.00 | Headquarters |
| Riya     | F    | 40000.00 | Research     |
| Ravish   | M    | 20000.00 | Outreach     |
| Prachi   | F    | 25000.00 | Management   |
| Sita     | F    | 65000.00 | Research     |
| Utkarsh  | M    | 30000.00 | Research     |
+----------+------+----------+--------------+
```

Function:

-
```
mysql> CREATE FUNCTION 21CStotalNoEmployees()
    -> RETURNS INT
    -> BEGIN
    ->     DECLARE s INT;
    ->     SELECT COUNT(*) FROM Employee3 INTO s;
    ->     RETURN S;
    -> END]
```

Function Query:

```
mysql> SELECT 21CStotalNoEmployees ()]
```

OUTPUT:

```
+-----------------------+
| 21CStotalNoEmployees() |
+-----------------------+
|                     8 |
+-----------------------+
```

Stored Procedure:

- ```
  mysql> CREATE PROCEDURE 21CStotalNoEmployees (OUT count INT)
      -> BEGIN
      ->     DECLARE s INT;
      ->     SELECT COUNT(*) FROM Employee3 INTO s;
      ->     SET count = s;
      -> END]
  ```

Stored Procedure Query:

```
mysql> CALL 21CStotalNoEmployees (@res)]
mysql> SELECT @res]
```

OUTPUT:

```
+------+
| @res |
+------+
|    8 |
+------+
```

6. Write a SQL function and stored procedure to calculate the budget of the department.

Employee table used:

// from Assignment 5

```
mysql> SELECT * FROM Employee2]
```

```
+--------+----------+--------+-------+
| emp_id | emp_name | salary | d_no  |
+--------+----------+--------+-------+
|    101 | Amit     |  25000 | D1001 |
|    102 | Sunil    |  20000 | D1002 |
|    103 | Rakesh   |  18000 | D1003 |
|    104 | Ajay     |  16000 | D1001 |
|    105 | Suhail   |  20000 | D1002 |
|    106 | Arif     |  18000 | D1004 |
|    107 | Suresh   |  24000 | D1002 |
|    108 | Vijay    |  22000 | D1003 |
+--------+----------+--------+-------+
```
```

Department table used: // from Assignment 5

```
mysql> SELECT * FROM Department3]
```

d_no	dept_name
D1001	IT
D1002	Sales
D1003	Marketing
D1004	HR

Function:

-
```
mysql> CREATE FUNCTION 21CScalcBudget (dept VARCHAR(30))
    -> RETURNS INT
    -> BEGIN
    ->     DECLARE deptnumber VARCHAR(5);
    ->     DECLARE budget INT DEFAULT 0;
    ->     SELECT d_no FROM Department3 WHERE dept_name = dept
INTO deptnumber;
    ->     SELECT SUM(salary) FROM Employee2 WHERE d_no =
deptnumber INTO budget;
    ->     RETURN budget;
    -> END]
```

Function Query:

```
mysql> SELECT 21CScalcBudget ('Marketing')]
```

OUTPUT:

21CScalcBudget ('Marketing')
40000

Stored Procedure:

-
```
mysql> CREATE PROCEDURE 21CScalcBudget (IN dept VARCHAR(30),
OUT budget INT)

    -> BEGIN

    ->     DECLARE deptnumber VARCHAR(5);

    ->     DECLARE sumSal INT DEFAULT 0;

    ->     SELECT d_no FROM Department3 WHERE dept_name = dept
INTO deptnumber;

    ->     SELECT SUM(salary) FROM Employee2 WHERE d_no =
deptnumber INTO sumSal;

    ->     SET budget = sumSal;

    -> END]
```

Stored Procedure Query:

```
mysql> CALL 21CScalcBudget ('IT', @budget)]
mysql> SELECT @budget]
```

OUTPUT:

```
+---------+
| @budget |
+---------+
|   41000 |
+---------+
```

7. Write a SQL function and stored procedure to print the following message:

   "Hello <name> How are you".

Function:

- ```
  mysql> CREATE FUNCTION 21CSprintMsg (name VARCHAR(50))
      -> RETURNS VARCHAR(100)
      -> BEGIN
      ->    DECLARE msg VARCHAR(100) DEFAULT "Hello ";
      ->    SET msg = CONCAT(msg, name);
      ->    SET msg = CONCAT(msg, " How are you?");
      ->    RETURN msg;
      -> END]
  ```

Function Query:

```
mysql> SELECT 21CSprintMsg ('Faizan')]
```

OUTPUT:

```
+---------------------------+
| 21CSprintMsg ('Faizan')   |
+---------------------------+
| Hello Faizan How are you? |
+---------------------------+
```

Stored Procedure:

- ```
  mysql> CREATE PROCEDURE 21CSprintMsg (IN name VARCHAR(50), OUT message VARCHAR(100))
      -> BEGIN
      ->    DECLARE msg VARCHAR(100) DEFAULT "Hello ";
      ->    SET msg = CONCAT(msg, name);
      ->    SET msg = CONCAT(msg, " How are you?");
      ->    SET message = msg;
      -> END]
  ```

Stored Procedure Query:

```
mysql> CALL 21CSprintMsg ('Faizan', @msg)]
mysql> SELECT @msg]
```

OUTPUT:

```
+--------------------------+
| @msg                     |
+--------------------------+
| Hello Faizan How are you? |
+--------------------------+
```

8. Triggers

Creating Tables:

a) Employee

```
mysql> CREATE TABLE Employees2 AS SELECT * FROM Employees;
          // similar table used in Assignment 10
mysql> SELECT * FROM Employees2;
```

```
+------+-------+--------+
| eid  | ename | salary |
+------+-------+--------+
|    1 | Ajay  |  30000 |
|    2 | Ajith |  85000 |
|    3 | Arnab |  50000 |
|    4 | Harry |  45000 |
|    5 | Ron   |  90000 |
|    6 | Josh  |  75000 |
|    7 | Ram   | 100000 |
|    8 | John  |  75000 |
|    9 | Uttam |  25000 |
+------+-------+--------+
```

b) LogTable

```
mysql> CREATE TABLE LogTable
    -> (
    -> User   VARCHAR(50),
    -> Operation VARCHAR(20),
    -> Time   VARCHAR(20),
    -> Peid   VARCHAR(5),
    -> Pename VARCHAR(50),
    -> Pesal  VARCHAR(6),
    -> Neid   VARCHAR(5),
    -> Nename VARCHAR(50),
    -> Nesal  VARCHAR(6)
    -> );
```

1. INSERT TRIGGER

- mysql> CREATE TRIGGER insertTrig

  -> AFTER INSERT ON Employees2

  -> FOR EACH ROW

  -> BEGIN

  -> INSERT INTO LogTable VALUES

  -> (user(), 'Insert', now(), '-', '-', '-', new.eid, new.ename, new.salary);

  -> END]

- mysql> INSERT INTO Employees2 VALUES

  -> (10, 'Yousuf', 69000)]

- mysql> SELECT * FROM LogTable]

OUTPUT:

User	Operation	Time	Peid	Pename	Pesal	Neid	Nename	Nesal
root@localhost	Insert	2022-04-26 18:55:22	-	-	-	10	Yousuf	69000

2. UPDATE TRIGGER

- mysql> CREATE TRIGGER updateTrig

  -> AFTER UPDATE ON Employees2

  -> FOR EACH ROW

  -> BEGIN

  -> INSERT INTO LogTable VALUES

  -> (user(), 'Update', now(), old.eid, old.ename, old.salary, new.eid, new.ename, new.salary);

  -> END]

- mysql> UPDATE Employees2

  -> SET salary = 25000

  -> WHERE eid = 3]

- mysql> SELECT * FROM LogTable]

OUTPUT:

User	Operation	Time	Peid	Pename	Pesal	Neid	Nename	Nesal
root@localhost	Insert	2022-04-26 18:55:22	-	-	-	10	Yousuf	69000
root@localhost	Update	2022-04-26 23:56:08	3	Arnab	50000	3	Arnab	25000

3. DELETE TRIGGER

- mysql> CREATE TRIGGER deleteTrig

```
    -> AFTER DELETE ON Employees2
    -> FOR EACH ROW
    -> BEGIN
    ->    INSERT INTO LogTable VALUES
    ->    (user(), 'Delete', now(), old.eid, old.ename,
old.salary, '-', '-', '-');
    -> END]
```

- mysql> DELETE FROM Employees2

```
    -> WHERE eid = 3]
```

- mysql> SELECT * FROM LogTable]

OUTPUT:

User	Operation	Time	Peid	Pename	Pesal	Neid	Nename	Nesal
root@localhost	Insert	2022-04-26 18:55:22	-	-	-	10	Yousuf	69000
root@localhost	Update	2022-04-26 23:56:08	3	Arnab	50000	3	Arnab	25000
root@localhost	Delete	2022-04-27 00:00:48	3	Arnab	25000	-	-	-

4. CURSOR

Write a cursor to output salary of all employees in a string.

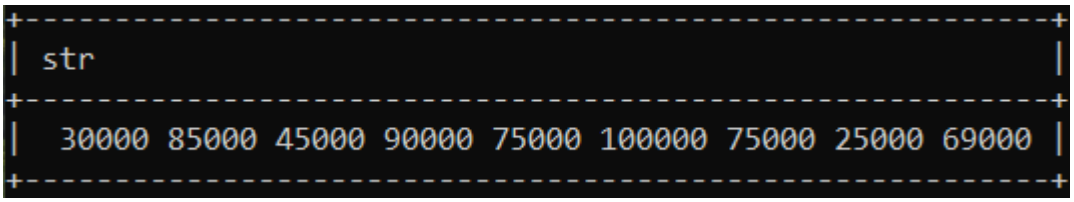
- mysql> CREATE PROCEDURE outSal (OUT s VARCHAR(15))

```
    -> BEGIN
    ->    DECLARE f INT DEFAULT 1;
    ->    DECLARE str LONGTEXT DEFAULT "";
    ->    DECLARE cur CURSOR FOR SELECT salary FROM
Employees2;
    ->    DECLARE continue HANDLER FOR NOT FOUND SET f=0;
    ->    OPEN cur;
    ->    myloop:LOOP
    ->    FETCH cur INTO s;
    ->    IF f=0 THEN
    ->         LEAVE myloop;
    ->    ELSE
    ->         SET str = CONCAT(str, " ", s);
    ->    END IF;
```

```
    ->     END LOOP;
    ->     CLOSE cur;
    ->     SELECT str;
    -> END]
```

- `mysql> CALL outSal (@s)]`

OUTPUT:



```
+-------------------------------------------------------------+
| str                                                         |
+-------------------------------------------------------------+
|  30000 85000 45000 90000 75000 100000 75000 25000 69000 |
+-------------------------------------------------------------+
```