राष्ट्रीय प्रौद्योगिकी संस्थान जमशेदपुर
NATIONAL INSTITUTE OF TECHNOLOGY JAMSHEDPUR
(An Institute of National importance under MoE, Govt. of India)

Database Management System (CS3306)

Practical Lab File

MCA II Year – 3rd Semester (2023-24)

Department of Computer Science and Engineering

National Institute of Technology, Jamshedpur

Jamshedpur 831014

Student Name.: Somdeep Misra

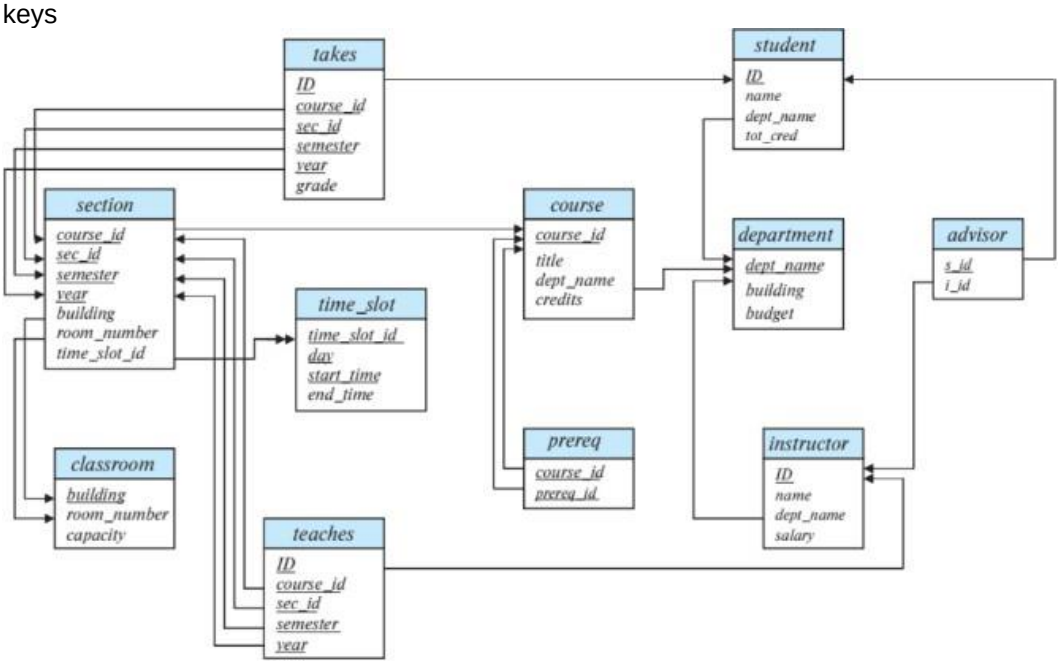
Student Roll No.: 2022PGCSCA061

Index

Assignment 1:

Q1) Create All tables using CREATE TABLE command in University Database) for the University Schema as given in

Figure. Make any reasonable assumptions about data types, and be sure to declare primary and foreign keys



Classroom:

```
mysql> create table classroom (
    ->      building varchar(15),
    ->      room_number varchar(7),
    ->      capacity numeric(4,0),
    ->      primary key(building,room_number)
    ->      );
Query OK, 0 rows affected (0.03 sec)
```

Department:

```
Query OK, 0 rows affected (0.03 sec)

mysql> create table department (
    ->       dept_name varchar(20),
    ->       building varchar(15),
    ->       budget numeric(12,2) check (budget > 0),
    ->       primary key (dept_name)
    ->       );
Query OK, 0 rows affected (0.02 sec)
```

Course:

```
mysql> create table course(
    ->       course_id varchar(7),
    ->       title varchar(50),
    ->       dept_name varchar(20),
    ->       credits numeric(2,0) check (credits > 0),
    ->       primary key (course_id),
    ->       foreign key (dept_name) references department(dept_name) on delete set null);
Query OK, 0 rows affected (0.03 sec)
```

Instructor:

```
mysql>       create table instructor (
    ->       ID varchar(5),
    ->       name varchar(20) not null,
    ->       dept_name varchar(20),
    ->       salary numeric(8,2) check(salary > 29000),
    ->       primary key(ID),
    ->       foreign key (dept_name) references department(dept_name) on delete set null);
Query OK, 0 rows affected (0.03 sec)
```

Student:

```
mysql> create table student
    -> (ID          varchar (5),
    ->  name        varchar (20) not null,
    ->  dept_name   varchar (20),
    ->  tot_cred    numeric (3,0) check (tot_cred>=0),
    ->  primary key(ID),
    ->  foreign key(dept_name) references department(dept_name) on delete set NULL);
Query OK, 0 rows affected (0.03 sec)
```

Advisor:

```
mysql> create table advisor
    -> (s_ID        varchar (5),
    ->  i_ID        varchar (5),
    ->  primary key (s_ID),
    ->  foreign key (i_ID) references instructor (ID) on delete set null,
    ->  foreign key (s_ID) references student (ID) on delete cascade);
Query OK, 0 rows affected (0.03 sec)
```

Prereq:

```
mysql> create table prereq
    -> (course_id   varchar(8),
    ->  prereq_id   varchar(8),
    ->  primary key (course_id,prereq_id),
    ->  foreign key (course_id) references course(course_id)
    ->                on delete cascade,
    ->  foreign key (prereq_id) references course(course_id));
Query OK, 0 rows affected (0.03 sec)
```

Timeslot:

```
mysql> create table timeslot
    -> (time_slot_id  varchar(4),
    ->  day           varchar(1) check (day in ('M', 'T', 'W', 'R', 'F', 'S', 'U') ),
    ->  start_time    time,
    ->  end_time      time,
    ->  primary key  (time_slot_id,day,start_time));
Query OK, 0 rows affected (0.02 sec)
```

Section:

```
mysql> create table section (
    ->       course_id varchar(8),
    ->       sec_id varchar(8),
    ->       semester varchar(6) check (semester in ('Fall','Winter',
    ->       'Spring','Summer')),
    ->       year numeric(4,0) check(year > 1701 and year < 2100),
    ->       building varchar(15),
    ->       room_number varchar(7),
    ->       time_slot_id varchar(4),
    ->       primary key(course_id,sec_id,semester,year),
    ->       foreign key(course_id) references course(course_id) on delete cascade,
    ->       foreign key(building, room_number) references classroom(building, room_number) on delete set null,
    ->       foreign key(time_slot_id) references timeslot(time_slot_id) on delete set null);
Query OK, 0 rows affected (0.07 sec)
```

Teaches:

```
mysql> create table teaches
    -> (ID          varchar (5),
    ->  course_id   varchar (8),
    ->  sec_id      varchar (8),
    ->  semester    varchar (6),
    ->  year        numeric (4,0),
    ->  primary key(ID,course_id,sec_id,semester,year),
    ->  foreign key(course_id,sec_id,semester,year) references section(course_id,sec_id,semester,year)
    ->              on delete cascade,
    ->  foreign key(ID) references instructor(ID)
    ->              on delete cascade);
Query OK, 0 rows affected (0.03 sec)
```

Takes:

```
mysql> create table takes
    -> (ID          varchar (5),
    ->  course_id   varchar (8),
    ->  sec_id      varchar (8),
    ->  semester    varchar (6),
    ->  year        numeric (4,0),
    ->  grade       varchar (2),
    ->  primary key (ID,course_id,sec_id,semester,year),
    ->  foreign key (course_id,sec_id,semester,year) references section (course_id,sec_id,semester,year)
    ->                  on delete cascade,
    ->  foreign key (ID) references student(ID)
    ->                  on delete cascade);
Query OK, 0 rows affected (0.04 sec)
```

Viewing all the tables after creation:

```
mysql> show tables;
+----------------------+
| Tables_in_university |
+----------------------+
| advisor              |
| classroom            |
| course               |
| department           |
| instructor           |
| prereq               |
| section              |
| student              |
| takes                |
| teaches              |
| timeslot             |
+----------------------+
11 rows in set (0.01 sec)
```

Q2) Populate (Fill) all the tables of the University Schema with the Sample Data as given in Appendix A of the Text Book

Classroom:

```
mysql> INSERT INTO classroom (building, room_number, capacity)
    -> VALUES ('Packard', '101', 50),
    ->        ('Painter', '514', 10),
    ->        ('Taylor', '3128', 70),
    ->        ('Watson', '100', 30),
    ->        ('Watson', '120', 50);
Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> Select * from classroom;
+----------+-------------+----------+
| building | room_number | capacity |
+----------+-------------+----------+
| Packard  | 101         |       50 |
| Painter  | 514         |       10 |
| Taylor   | 3128        |       70 |
| Watson   | 100         |       30 |
| Watson   | 120         |       50 |
+----------+-------------+----------+
5 rows in set (0.00 sec)
```

Department:

```
mysql> INSERT INTO department (dept_name, building, budget)
    -> VALUES ('Biology', 'Watson', 90000),
    ->        ('Comp. Sci.', 'Taylor', 100000),
    ->        ('Elec. Eng.', 'Taylor', 85000),
    ->        ('Finance', 'Painter', 120000),
    ->        ('History', 'Painter', 50000),
    ->        ('Music', 'Packard', 80000),
    ->        ('Physics', 'Watson', 7000);
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from department;
+------------+----------+-----------+
| dept_name  | building | budget    |
+------------+----------+-----------+
| Biology    | Watson   |  90000.00 |
| Comp. Sci. | Taylor   | 100000.00 |
| Elec. Eng. | Taylor   |  85000.00 |
| Finance    | Painter  | 120000.00 |
| History    | Painter  |  50000.00 |
| Music      | Packard  |  80000.00 |
| Physics    | Watson   |   7000.00 |
+------------+----------+-----------+
7 rows in set (0.00 sec)
```

Course:

```
mysql> INSERT INTO course (course_id, title, dept_name, credits)
    -> VALUES
    ->     ('BIO-101', 'Intro. to Biology', 'Biology', 4),
    ->     ('BIO-301', 'Genetics', 'Biology', 4),
    ->     ('BIO-399', 'Computational Biology', 'Biology', 3),
    ->     ('CS-101', 'Intro. to Computer Science', 'Comp. Sci.', 4),
    ->     ('CS-190', 'Game Design', 'Comp. Sci.', 4),
    ->     ('CS-315', 'Robotics', 'Comp. Sci.', 3),
    ->     ('CS-319', 'Image Processing', 'Comp. Sci.', 3),
    ->     ('CS-347', 'Database System Concepts', 'Comp. Sci.', 3),
    ->     ('EE-181', 'Intro. to Digital Systems', 'Elec. Eng.', 3),
    ->     ('FIN-201', 'Investment Banking', 'Finance', 3),
    ->     ('HIS-351', 'World History', 'History', 3),
    ->     ('MU-199', 'Music Video Production', 'Music', 3),
    ->     ('PHY-101', 'Physical Principles', 'Physics', 4);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from course;
+-----------+---------------------------+------------+---------+
| course_id | title                     | dept_name  | credits |
+-----------+---------------------------+------------+---------+
| BIO-101   | Intro. to Biology         | Biology    |       4 |
| BIO-301   | Genetics                  | Biology    |       4 |
| BIO-399   | Computational Biology     | Biology    |       3 |
| CS-101    | Intro. to Computer Science | Comp. Sci. |       4 |
| CS-190    | Game Design               | Comp. Sci. |       4 |
| CS-315    | Robotics                  | Comp. Sci. |       3 |
| CS-319    | Image Processing          | Comp. Sci. |       3 |
| CS-347    | Database System Concepts  | Comp. Sci. |       3 |
| EE-181    | Intro. to Digital Systems | Elec. Eng. |       3 |
| FIN-201   | Investment Banking        | Finance    |       3 |
| HIS-351   | World History             | History    |       3 |
| MU-199    | Music Video Production    | Music      |       3 |
| PHY-101   | Physical Principles       | Physics    |       4 |
+-----------+---------------------------+------------+---------+
13 rows in set (0.00 sec)
```

Instructor:

```
mysql> INSERT INTO instructor (id, name, dept_name, salary)
    -> VALUES
    ->     ('10101', 'Srinivasan', 'Comp. Sci.', 65000),
    ->     ('12121', 'Wu', 'Finance', 90000),
    ->     ('15151', 'Mozart', 'Music', 40000),
    ->     ('22222', 'Einstein', 'Physics', 95000),
    ->     ('32343', 'El Said', 'History', 60000),
    ->     ('33456', 'Gold', 'Physics', 87000),
    ->     ('45565', 'Katz', 'Comp. Sci.', 75000),
    ->     ('58583', 'Califieri', 'History', 62000),
    ->     ('76543', 'Singh', 'Finance', 80000),
    ->     ('76766', 'Crick', 'Biology', 72000),
    ->     ('83821', 'Brandt', 'Comp. Sci.', 92000),
    ->     ('98345', 'Kim', 'Elec. Eng.', 80000);
Query OK, 12 rows affected (0.01 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from instructor;
+-------+------------+------------+----------+
| ID    | name       | dept_name  | salary   |
+-------+------------+------------+----------+
| 10101 | Srinivasan | Comp. Sci. | 65000.00 |
| 12121 | Wu         | Finance    | 90000.00 |
| 15151 | Mozart     | Music      | 40000.00 |
| 22222 | Einstein   | Physics    | 95000.00 |
| 32343 | El Said    | History    | 60000.00 |
| 33456 | Gold       | Physics    | 87000.00 |
| 45565 | Katz       | Comp. Sci. | 75000.00 |
| 58583 | Califieri  | History    | 62000.00 |
| 76543 | Singh      | Finance    | 80000.00 |
| 76766 | Crick      | Biology    | 72000.00 |
| 83821 | Brandt     | Comp. Sci. | 92000.00 |
| 98345 | Kim        | Elec. Eng. | 80000.00 |
+-------+------------+------------+----------+
12 rows in set (0.00 sec)
```

Timeslot:

```
mysql> INSERT INTO timeslot (time_slot_id, day, start_time, end_time)
    -> VALUES
    ->     ('A', 'M', '08:00', '08:50'),
    ->     ('A', 'W', '08:00', '08:50'),
    ->     ('A', 'F', '08:00', '08:50'),
    ->     ('B', 'M', '09:00', '09:50'),
    ->     ('B', 'W', '09:00', '09:50'),
    ->     ('B', 'F', '09:00', '09:50'),
    ->     ('C', 'M', '11:00', '11:50'),
    ->     ('C', 'W', '11:00', '11:50'),
    ->     ('C', 'F', '11:00', '11:50'),
    ->     ('D', 'M', '13:00', '13:50'),
    ->     ('D', 'W', '13:00', '13:50'),
    ->     ('D', 'F', '13:00', '13:50'),
    ->     ('E', 'T', '10:30', '11:45'),
    ->     ('E', 'R', '10:30', '11:45'),
    ->     ('F', 'T', '14:30', '15:45'),
    ->     ('F', 'R', '14:30', '15:45'),
    ->     ('G', 'M', '16:00', '16:50'),
    ->     ('G', 'W', '16:00', '16:50'),
    ->     ('G', 'F', '16:00', '16:50'),
    ->     ('H', 'W', '10:00', '12:30');
Query OK, 20 rows affected (0.01 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from timeslot;
+--------------+-----+------------+----------+
| time_slot_id | day | start_time | end_time |
+--------------+-----+------------+----------+
| A            | F   | 08:00:00   | 08:50:00 |
| A            | M   | 08:00:00   | 08:50:00 |
| A            | W   | 08:00:00   | 08:50:00 |
| B            | F   | 09:00:00   | 09:50:00 |
| B            | M   | 09:00:00   | 09:50:00 |
| B            | W   | 09:00:00   | 09:50:00 |
| C            | F   | 11:00:00   | 11:50:00 |
| C            | M   | 11:00:00   | 11:50:00 |
| C            | W   | 11:00:00   | 11:50:00 |
| D            | F   | 13:00:00   | 13:50:00 |
| D            | M   | 13:00:00   | 13:50:00 |
| D            | W   | 13:00:00   | 13:50:00 |
| E            | R   | 10:30:00   | 11:45:00 |
| E            | T   | 10:30:00   | 11:45:00 |
| F            | R   | 14:30:00   | 15:45:00 |
| F            | T   | 14:30:00   | 15:45:00 |
| G            | F   | 16:00:00   | 16:50:00 |
| G            | M   | 16:00:00   | 16:50:00 |
| G            | W   | 16:00:00   | 16:50:00 |
| H            | W   | 10:00:00   | 12:30:00 |
+--------------+-----+------------+----------+
20 rows in set (0.00 sec)
```

Section:

```
mysql> INSERT INTO section (course_id, sec_id, semester, year, building, room_number, time_slot_id)
    -> VALUES
    ->     ('BIO-101', '1', 'Summer', 2017, 'Painter', '514', 'B'),
    ->     ('BIO-301', '1', 'Summer', 2018, 'Painter', '514', 'A'),
    ->     ('CS-101', '1', 'Fall', 2017, 'Packard', '101', 'H'),
    ->     ('CS-101', '1', 'Spring', 2018, 'Packard', '101', 'F'),
    ->     ('CS-190', '1', 'Spring', 2017, 'Taylor', '3128', 'E'),
    ->     ('CS-190', '2', 'Spring', 2017, 'Taylor', '3128', 'A'),
    ->     ('CS-315', '1', 'Spring', 2018, 'Watson', '120', 'D'),
    ->     ('CS-319', '1', 'Spring', 2018, 'Watson', '100', 'B'),
    ->     ('CS-319', '2', 'Spring', 2018, 'Taylor', '3128', 'C'),
    ->     ('CS-347', '1', 'Fall', 2017, 'Taylor', '3128', 'A'),
    ->     ('EE-181', '1', 'Spring', 2017, 'Taylor', '3128', 'C'),
    ->     ('FIN-201', '1', 'Spring', 2018, 'Packard', '101', 'B'),
    ->     ('HIS-351', '1', 'Spring', 2018, 'Painter', '514', 'C'),
    ->     ('MU-199', '1', 'Spring', 2018, 'Packard', '101', 'D'),
    ->     ('PHY-101', '1', 'Fall', 2017, 'Watson', '100', 'A');
Query OK, 15 rows affected (0.01 sec)
Records: 15  Duplicates: 0  Warnings: 0

mysql> Select * from section;
+-----------+--------+----------+------+----------+-------------+--------------+
| course_id | sec_id | semester | year | building | room_number | time_slot_id |
+-----------+--------+----------+------+----------+-------------+--------------+
| BIO-101   | 1      | Summer   | 2017 | Painter  | 514         | B            |
| BIO-301   | 1      | Summer   | 2018 | Painter  | 514         | A            |
| CS-101    | 1      | Fall     | 2017 | Packard  | 101         | H            |
| CS-101    | 1      | Spring   | 2018 | Packard  | 101         | F            |
| CS-190    | 1      | Spring   | 2017 | Taylor   | 3128        | E            |
| CS-190    | 2      | Spring   | 2017 | Taylor   | 3128        | A            |
| CS-315    | 1      | Spring   | 2018 | Watson   | 120         | D            |
| CS-319    | 1      | Spring   | 2018 | Watson   | 100         | B            |
| CS-319    | 2      | Spring   | 2018 | Taylor   | 3128        | C            |
| CS-347    | 1      | Fall     | 2017 | Taylor   | 3128        | A            |
| EE-181    | 1      | Spring   | 2017 | Taylor   | 3128        | C            |
| FIN-201   | 1      | Spring   | 2018 | Packard  | 101         | B            |
| HIS-351   | 1      | Spring   | 2018 | Painter  | 514         | C            |
| MU-199    | 1      | Spring   | 2018 | Packard  | 101         | D            |
| PHY-101   | 1      | Fall     | 2017 | Watson   | 100         | A            |
+-----------+--------+----------+------+----------+-------------+--------------+
15 rows in set (0.00 sec)
```

Teaches

```
mysql> INSERT INTO teaches (ID, course_id, sec_id, semester, year)
    -> VALUES
    ->     ('10101', 'CS-101', 1, 'Fall', 2017),
    ->     ('10101', 'CS-315', 1, 'Spring', 2018),
    ->     ('10101', 'CS-347', 1, 'Fall', 2017),
    ->     ('12121', 'FIN-201', 1, 'Spring', 2018),
    ->     ('15151', 'MU-199', 1, 'Spring', 2018),
    ->     ('22222', 'PHY-101', 1, 'Fall', 2017),
    ->     ('32343', 'HIS-351', 1, 'Spring', 2018),
    ->     ('45565', 'CS-101', 1, 'Spring', 2018),
    ->     ('45565', 'CS-319', 1, 'Spring', 2018),
    ->     ('76766', 'BIO-101', 1, 'Summer', 2017),
    ->     ('76766', 'BIO-301', 1, 'Summer', 2018),
    ->     ('83821', 'CS-190', 1, 'Spring', 2017),
    ->     ('83821', 'CS-190', 2, 'Spring', 2017),
    ->     ('83821', 'CS-319', 2, 'Spring', 2018),
    ->     ('98345', 'EE-181', 1, 'Spring', 2017);
Query OK, 15 rows affected (0.00 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

```
mysql> select * from teaches;
+-------+-----------+--------+----------+------+
| ID    | course_id | sec_id | semester | year |
+-------+-----------+--------+----------+------+
| 76766 | BIO-101   | 1      | Summer   | 2017 |
| 76766 | BIO-301   | 1      | Summer   | 2018 |
| 10101 | CS-101    | 1      | Fall     | 2017 |
| 45565 | CS-101    | 1      | Spring   | 2018 |
| 83821 | CS-190    | 1      | Spring   | 2017 |
| 83821 | CS-190    | 2      | Spring   | 2017 |
| 10101 | CS-315    | 1      | Spring   | 2018 |
| 45565 | CS-319    | 1      | Spring   | 2018 |
| 83821 | CS-319    | 2      | Spring   | 2018 |
| 10101 | CS-347    | 1      | Fall     | 2017 |
| 98345 | EE-181    | 1      | Spring   | 2017 |
| 12121 | FIN-201   | 1      | Spring   | 2018 |
| 32343 | HIS-351   | 1      | Spring   | 2018 |
| 15151 | MU-199    | 1      | Spring   | 2018 |
| 22222 | PHY-101   | 1      | Fall     | 2017 |
+-------+-----------+--------+----------+------+
15 rows in set (0.00 sec)
```

Student:

```
mysql> INSERT INTO student (ID, name, dept_name, tot_cred)
    -> VALUES
    ->     ('00128', 'Zhang', 'Comp. Sci.', 102),
    ->     ('12345', 'Shankar', 'Comp. Sci.', 32),
    ->     ('19991', 'Brandt', 'History', 80),
    ->     ('23121', 'Chavez', 'Finance', 110),
    ->     ('44553', 'Peltier', 'Physics', 56),
    ->     ('45678', 'Levy', 'Physics', 46),
    ->     ('54321', 'Williams', 'Comp. Sci.', 54),
    ->     ('55739', 'Sanchez', 'Music', 38),
    ->     ('70557', 'Snow', 'Physics', 0),
    ->     ('76543', 'Brown', 'Comp. Sci.', 58),
    ->     ('76653', 'Aoi', 'Elec. Eng.', 60),
    ->     ('98765', 'Bourikas', 'Elec. Eng.', 98),
    ->     ('98988', 'Tanaka', 'Biology', 120);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from student;
+-------+----------+------------+----------+
| ID    | name     | dept_name  | tot_cred |
+-------+----------+------------+----------+
| 00128 | Zhang    | Comp. Sci. |      102 |
| 12345 | Shankar  | Comp. Sci. |       32 |
| 19991 | Brandt   | History    |       80 |
| 23121 | Chavez   | Finance    |      110 |
| 44553 | Peltier  | Physics    |       56 |
| 45678 | Levy     | Physics    |       46 |
| 54321 | Williams | Comp. Sci. |       54 |
| 55739 | Sanchez  | Music      |       38 |
| 70557 | Snow     | Physics    |        0 |
| 76543 | Brown    | Comp. Sci. |       58 |
| 76653 | Aoi      | Elec. Eng. |       60 |
| 98765 | Bourikas | Elec. Eng. |       98 |
| 98988 | Tanaka   | Biology    |      120 |
+-------+----------+------------+----------+
13 rows in set (0.00 sec)
```

Takes:

```
mysql> INSERT INTO takes (ID, course_id, sec_id, semester, year, grade)
    -> VALUES
    ->     ('00128', 'CS-101', '1', 'Fall', 2017, 'A'),
    ->     ('00128', 'CS-347', '1', 'Fall', 2017, 'A'),
    ->     ('12345', 'CS-101', '1', 'Fall', 2017, 'C'),
    ->     ('12345', 'CS-190', '1', 'Spring', 2017, 'A'),
    ->     ('12345', 'CS-319', '2', 'Spring', 2018, 'A'),
    ->     ('12345', 'CS-347', '1', 'Fall', 2017, 'A'),
    ->     ('19991', 'HIS-351', '1', 'Spring', 2018, 'B'),
    ->     ('23121', 'FIN-201', '1', 'Spring', 2018, 'C+'),
    ->     ('44553', 'PHY-101', '1', 'Fall', 2017, 'B-'),
    ->     ('45678', 'CS-101', '1', 'Fall', 2017, 'F'),
    ->     ('45678', 'CS-101', '1', 'Spring', 2018, 'B+'),
    ->     ('45678', 'CS-319', '1', 'Spring', 2018, 'B'),
    ->     ('54321', 'CS-101', '1', 'Fall', 2017, 'A-'),
    ->     ('54321', 'CS-190', '1', 'Spring', 2017, 'B+'),
    ->     ('55739', 'MU-199', '1', 'Spring', 2018, 'A-'),
    ->     ('76543', 'CS-101', '1', 'Fall', 2017, 'A'),
    ->     ('76543', 'CS-319', '2', 'Spring', 2018, 'A'),
    ->     ('76653', 'EE-181', '1', 'Spring', 2017, 'C'),
    ->     ('98765', 'CS-101', '1', 'Fall', 2017, 'C-'),
    ->     ('98765', 'CS-315', '1', 'Spring', 2018, 'B'),
    ->     ('98988', 'BIO-101', '1', 'Summer', 2017, 'A'),
    ->     ('98988', 'BIO-301', '1', 'Summer', 2018, NULL);
Query OK, 22 rows affected (0.01 sec)
Records: 22  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from takes;
+-------+----------+--------+----------+------+-------+
| ID    | course_id | sec_id | semester | year | grade |
+-------+----------+--------+----------+------+-------+
| 00128 | CS-101    | 1      | Fall     | 2017 | A     |
| 00128 | CS-347    | 1      | Fall     | 2017 | A     |
| 12345 | CS-101    | 1      | Fall     | 2017 | C     |
| 12345 | CS-190    | 1      | Spring   | 2017 | A     |
| 12345 | CS-319    | 2      | Spring   | 2018 | A     |
| 12345 | CS-347    | 1      | Fall     | 2017 | A     |
| 19991 | HIS-351   | 1      | Spring   | 2018 | B     |
| 23121 | FIN-201   | 1      | Spring   | 2018 | C+    |
| 44553 | PHY-101   | 1      | Fall     | 2017 | B-    |
| 45678 | CS-101    | 1      | Fall     | 2017 | F     |
| 45678 | CS-101    | 1      | Spring   | 2018 | B+    |
| 45678 | CS-319    | 1      | Spring   | 2018 | B     |
| 54321 | CS-101    | 1      | Fall     | 2017 | A-    |
| 54321 | CS-190    | 1      | Spring   | 2017 | B+    |
| 55739 | MU-199    | 1      | Spring   | 2018 | A-    |
| 76543 | CS-101    | 1      | Fall     | 2017 | A     |
| 76543 | CS-319    | 2      | Spring   | 2018 | A     |
| 76653 | EE-181    | 1      | Spring   | 2017 | C     |
| 98765 | CS-101    | 1      | Fall     | 2017 | C-    |
| 98765 | CS-315    | 1      | Spring   | 2018 | B     |
| 98988 | BIO-101   | 1      | Summer   | 2017 | A     |
| 98988 | BIO-301   | 1      | Summer   | 2018 | NULL  |
+-------+----------+--------+----------+------+-------+
```

Advisor:

```
mysql> INSERT INTO advisor (s_id, i_id)
    -> VALUES
    ->     ('00128', '45565'),
    ->     ('12345', '10101'),
    ->     ('23121', '76543'),
    ->     ('44553', '22222'),
    ->     ('45678', '22222'),
    ->     ('76543', '45565'),
    ->     ('76653', '98345'),
    ->     ('98765', '98345'),
    ->     ('98988', '76766');
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from advisor;
+-------+-------+
| s_ID  | i_ID  |
+-------+-------+
| 12345 | 10101 |
| 44553 | 22222 |
| 45678 | 22222 |
| 00128 | 45565 |
| 76543 | 45565 |
| 23121 | 76543 |
| 98988 | 76766 |
| 76653 | 98345 |
| 98765 | 98345 |
+-------+-------+
9 rows in set (0.00 sec)
```

Prereq:

```
mysql> INSERT INTO prereq (course_id, prereq_id)
    -> VALUES
    ->     ('BIO-301', 'BIO-101'),
    ->     ('BIO-399', 'BIO-101'),
    ->     ('CS-190', 'CS-101'),
    ->     ('CS-315', 'CS-101'),
    ->     ('CS-319', 'CS-101'),
    ->     ('CS-347', 'CS-101'),
    ->     ('EE-181', 'PHY-101');
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from prereq;
+-----------+-----------+
| course_id | prereq_id |
+-----------+-----------+
| BIO-301   | BIO-101   |
| BIO-399   | BIO-101   |
| CS-190    | CS-101    |
| CS-315    | CS-101    |
| CS-319    | CS-101    |
| CS-347    | CS-101    |
| EE-181    | PHY-101   |
+-----------+-----------+
7 rows in set (0.00 sec)
```

Q3) Write SQL DDL (Use commands CREATE, DROP, DELETE, ALTER) corresponding to the Insurance Database schema as shown in Figure. Make any reasonable assumptions about data types, and be sure to declare primary and foreign keys.

person (*driver_id*, *name*, *address*)
car (*license_plate*, *model*, *year*)
accident (*report_number*, *year*, *location*)
owns (*driver_id*, *license_plate*)
participated (*report_number*, *license_plate*, *driver_id*, *damage_amount*)

Person:

```
mysql> create table person (
    ->      driver_id varchar(15),
    ->      name varchar(15),
    ->      address varchar(15),
    ->      primary key(driver_id)
    ->    );
Query OK, 0 rows affected (0.03 sec)
```

Car:

```
mysql> create table car (
    ->      license_plate varchar(15),
    ->      model varchar(15),
    ->      year year,
    ->      primary key(license_plate)
    ->    );
Query OK, 0 rows affected (0.03 sec)
```

Accident:

```
mysql> create table accident (
    ->      report_number varchar(15),
    ->      year year,
    ->      location varchar(15),
    ->      primary key(report_number)
    ->    );
Query OK, 0 rows affected (0.03 sec)
```

Owns:

```
mysql> create table owns (
    ->      driver_id varchar(15),
    ->      license_plate varchar(15),
    ->      primary key(driver_id,license_plate),
    ->      foreign key (license_plate) references car (license_plate)
    ->                                  on delete cascade,
    ->      foreign key (driver_id) references person (driver_id)
    ->                                  on delete cascade
    ->    );
Query OK, 0 rows affected (0.04 sec)
```

Participate:

```
mysql> create table participated (
    ->     report_number varchar(15),
    ->     license_plate varchar(15),
    ->     driver_id varchar(15),
    ->     damage_amount numeric(10,2),
    ->     primary key(report_number,license_plate),
    ->     foreign key (report_number) references accident (report_number)
    ->                                 on delete cascade,
    ->     foreign key (license_plate,driver_id) references owns (license_plate,driver_id)
    ->                                 on delete cascade
    ->     );
Query OK, 0 rows affected (0.05 sec)
```

All Tables After Creation:

```
mysql> show tables;
+----------------------+
| Tables_in_insurance |
+----------------------+
| accident             |
| car                  |
| owns                 |
| participated         |
| person               |
+----------------------+
5 rows in set (0.00 sec)
```

ALTER:

```
mysql> ALTER TABLE accident CHANGE location place_of_accident varchar(15);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

DELETE:

```
mysql> INSERT into person(driver_id,name,address) VALUES('DRIVER101','DUMMY_DRIVER','JSR');
Query OK, 1 row affected (0.07 sec)

mysql> Select * from person;
+-----------+-------------+---------+
| driver_id | name        | address |
+-----------+-------------+---------+
| DRIVER101 | DUMMY_DRIVER | JSR     |
+-----------+-------------+---------+
1 row in set (0.00 sec)

mysql> Delete from person;
Query OK, 1 row affected (0.10 sec)

mysql> Select * from person;
Empty set (0.00 sec)
```

DROP:

```
mysql> DROP table participated;
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+----------------------+
| Tables_in_insurance |
+----------------------+
| accident             |
| car                  |
| owns                 |
| person               |
+----------------------+
4 rows in set (0.00 sec)
```

Assignment 2:

Find the names of all the instructors from Biology department

```
mysql> SELECT name
    -> FROM instructor
    -> WHERE dept_name="Biology";
+-------+
| name  |
+-------+
| Crick |
+-------+
```

Find the names of courses in Computer science department which have 3 credits

```
mysql> SELECT title
    -> FROM course
    -> WHERE credits="3" AND dept_name="Comp. Sci.";
+--------------------------+
| title                    |
+--------------------------+
| Robotics                 |
| Image Processing         |
| Database System Concepts |
+--------------------------+
```

For the student with ID 12345 (or any other value), show all course_id and title of all courses registered for by the student.

```
mysql> SELECT takes.course_id,title
    -> FROM takes,course
    -> WHERE course.course_id=takes.course_id AND ID="12345";
+-----------+----------------------------+
| course_id | title                      |
+-----------+----------------------------+
| CS-101    | Intro. to Computer Science |
| CS-190    | Game Design                |
| CS-319    | Image Processing           |
| CS-347    | Database System Concepts   |
+-----------+----------------------------+
```

As above, but show the total number of credits for such courses (taken by that student). Don't display the tot_creds value from the student table, you should use SQL aggregation on courses taken by the student.

```
mysql> SELECT sum(credits)
    -> FROM takes,course
    -> WHERE course.course_id=takes.course_id AND ID="12345";
+--------------+
| sum(credits) |
+--------------+
|           14 |
+--------------+
```

As above, but display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted)

```
mysql> SELECT id,sum(credits)
    -> FROM takes,course
    -> WHERE course.course_id=takes.course_id
    -> GROUP by id;
```

```
+-------+-------------+
| id    | sum(credits) |
+-------+-------------+
| 98988 |           8 |
| 00128 |           7 |
| 12345 |          14 |
| 45678 |          11 |
| 54321 |           8 |
| 76543 |           7 |
| 98765 |           7 |
| 76653 |           3 |
| 23121 |           3 |
| 19991 |           3 |
| 55739 |           3 |
| 44553 |           4 |
+-------+-------------+
```

Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names)

```
mysql> Select distinct name
    -> FROM takes,course,student
    -> WHERE course.course_id=takes.course_id
    -> AND takes.id=student.id
    -> AND course.dept_name="Comp. Sci.";
```

```
+-----------+
| name      |
+-----------+
| Zhang     |
| Shankar   |
| Levy      |
| Williams  |
| Brown     |
| Bourikas  |
+-----------+
```

Display the IDs of all instructors who have never taught a course (Notesad1) Oracle uses the keyword minus in place of except; (2) interpret "taught" as "taught or is scheduled to teach")

```
mysql> Select id
    -> FROM instructor
    -> EXCEPT
    -> Select id
    -> FROM teaches;
```

```
+-------+
| id    |
+-------+
| 76543 |
| 58583 |
| 33456 |
+-------+
```

As above, but display the names of the instructors also, not just the IDs.

```
mysql> Select instructor.id,name
    -> FROM instructor
    -> EXCEPT
    -> Select instructor.id,name
    -> FROM teaches,instructor
    -> WHERE teaches.id=instructor.id;
```

```
+-------+-----------+
| id    | name      |
+-------+-----------+
| 33456 | Gold      |
| 58583 | Califieri |
| 76543 | Singh     |
+-------+-----------+
```

9) You need to create a movie database. Create three tables, one for actors(AID, name), one for movies(MID, title) and one for actor_role(MID, AID, rolename). Use appropriate data types for each of the attributes, and add appropriate primary/foreign key constraints.

```
mysql> create database movie;
mysql> use movie;
mysql> CREATE table ACTOR(
    ->        AID      varchar(15),
    ->        name     varchar(15),
    ->        primary key (AID)
    ->        );
mysql> CREATE table MOVIES(
    ->        MID      varchar(15) primary key,
    ->        title     varchar(15)
    ->        );
mysql> CREATE table ACTOR_ROLE(
    ->       MID      varchar(15),
    ->       AID      varchar(15),
    ->       rolename varchar(15),
    ->       primary key (MID,AID,rolename),
    ->       foreign key (MID) references MOVIES (MID),
    ->       foreign key (AID) references ACTORS (AID)
    ->       );
```

Insert data to the above tables (approx 3 to 6 rows in each table), including data for actor "Charlie Chaplin", and for yourself (using your roll number as ID).

Movies:

```
mysql> INSERT INTO movies
    -> VALUES ("MOV001", "DUMMY MOVIE 1"),
    ->        ("MOV002", "DUMMY MOVIE 2"),
    ->        ("MOV003", "DUMMY MOVIE 3");
Query OK, 3 rows affected (0.05 sec)

mysql> SELECT * FROM MOVIES;
+--------+---------------+
| MID    | title         |
+--------+---------------+
| MOV001 | DUMMY MOVIE 1 |
| MOV002 | DUMMY MOVIE 2 |
| MOV003 | DUMMY MOVIE 3 |
+--------+---------------+
```

Actors

```
mysql> INSERT INTO actors
    -> VALUES ("ACT001", "DUMMY ACTOR 1"),
    ->        ("ACT002", "CHARLIE CHAPLIN"),
    ->        ("2022PGCSCA061", "SOMDEEP");
```

Actor_role:

```
mysql> SELECT * FROM ACTORS;
+--------------+----------------+
| AID          | name           |
+--------------+----------------+
| 2022PGCSCA061 | SOMDEEP        |
| ACT001       | DUMMY ACTOR 1  |
| ACT002       | CHARLIE CHAPLIN |
+--------------+----------------+
3 rows in set (0.00 sec)

mysql> INSERT INTO ACTOR_ROLE
    -> VALUES ("MOV001", "ACT002", "FUNNY GUY"),
    ->        ("MOV001", "ACT001", "DUMMY GUY"),
    ->        ("MOV002", "ACT002", "TWIN 1"),
    ->        ("MOV002", "ACT002", "TWIN 2"),
    ->        ("MOV003", "ACT001", "NARRATOR");
mysql> SELECT * FROM ACTOR_ROLE;
+--------+--------+-----------+
| MID    | AID    | rolename  |
+--------+--------+-----------+
| MOV001 | ACT001 | DUMMY GUY |
| MOV003 | ACT001 | NARRATOR  |
| MOV001 | ACT002 | FUNNY GUY |
| MOV002 | ACT002 | TWIN 1    |
| MOV002 | ACT002 | TWIN 2    |
+--------+--------+-----------+
```

Write a query to list all movies in which actor "Charlie Chaplin" has acted, along with the number of roles he had in that movie.

```
mysql> SELECT title,count(rolename)
    -> FROM movies,actor_role,actors
    -> WHERE movies.MID=actor_role.MID
    -> AND actor_role.AID=actors.AID
    -> AND name="CHARLIE CHAPLIN"
    -> GROUP BY title;
+---------------+-----------------+
| title         | count(rolename) |
+---------------+-----------------+
| DUMMY MOVIE 1 |               1 |
| DUMMY MOVIE 2 |               2 |
+---------------+-----------------+
```

Write a query to list all actors who have not acted in any movie

```
mysql> SELECT name
    -> FROM actors
    -> EXCEPT
    -> SELECT name
    -> FROM actors,actor_role
    -> WHERE actors.AID=actor_role.aid;
+---------+
| name    |
+---------+
| SOMDEEP |
+---------+
```

13) List names of actors, along with titles of movies they have acted in. If they have not acted in any movie, show the movie title as null. (Do not use SQL outerjoin syntax here, write it from scratch.

```
mysql> SELECT name,title
    -> FROM actor_role,movies,actors
    -> WHERE actors.AID=actor_role.AID AND movies.MID = actor_role.MID
    -> GROUP BY title,name;
+----------------+--------------+
| name           | title        |
+----------------+--------------+
| SOMDEEP        | NULL         |
| DUMMY ACTOR 1  | DUMMY MOVIE 1 |
| DUMMY ACTOR 1  | DUMMY MOVIE 3 |
| CHARLIE CHAPLIN | DUMMY MOVIE 1 |
| CHARLIE CHAPLIN | DUMMY MOVIE 2 |
+----------------+--------------+
```

Assignment 3

Write an SQL query using the university schema to find the ID of each student who has never taken a course at the university. Do this using no subqueries and no set operations (use an outer join)

```
mysql> Select Student.ID
    -> From Student
    -> LEFT JOIN takes
    -> On Student.id = takes.id
    -> Where takes.id IS NULL;
+-------+
| ID    |
+-------+
| 70557 |
+-------+
```

Express the following query in SQL using no subqueries and no set operations. select ID
from student except select s id from advisor
where i ID is not null

```
mysql> Select ID
    -> From Student
    -> LEFT JOIN advisor
    -> On id = s_id
    -> Where s_id is NULL;
+-------+
| ID    |
+-------+
| 54321 |
| 19991 |
| 55739 |
| 70557 |
+-------+
```

Write the SQL statements using the university schema to perform the following operations:
Create a new course "CS-001", titled "Weekly Seminar", with 1 credit.

```
mysql> INSERT into course
    -> VALUES("CS-001","Weekly Seminar","Comp. Sci.",1);
mysql> Select * from course;
+-----------+----------------------------+------------+---------+
| course_id | title                      | dept_name  | credits |
+-----------+----------------------------+------------+---------+
| BIO-101   | Intro. to Biology          | Biology    |       4 |
| BIO-301   | Genetics                   | Biology    |       4 |
| BIO-399   | Computational Biology      | Biology    |       3 |
| CS-001    | Weekly Seminar             | Comp. Sci. |       1 |
| CS-101    | Intro. to Computer Science | Comp. Sci. |       4 |
| CS-190    | Game Design                | Comp. Sci. |       4 |
| CS-315    | Robotics                   | Comp. Sci. |       3 |
| CS-319    | Image Processing           | Comp. Sci. |       3 |
| CS-347    | Database System Concepts   | Comp. Sci. |       3 |
| EE-181    | Intro. to Digital Systems  | Elec. Eng. |       3 |
| FIN-201   | Investment Banking         | Finance    |       3 |
| HIS-351   | World History              | History    |       3 |
| MU-199    | Music Video Production     | Music      |       3 |
| PHY-101   | Physical Principles        | Physics    |       4 |
+-----------+----------------------------+------------+---------+
```

Create a section of this course in Fall 2017, with secID of 1, and with the location of this section not yet specified.

```
mysql> INSERT into section(course_id,sec_id,semester,year)
    -> VALUES("CS-001","1","Fall",2017);
mysql> Select * from section;
+-----------+--------+----------+------+----------+-------------+--------------+
| course_id | sec_id | semester | year | building | room_number | time_slot_id |
+-----------+--------+----------+------+----------+-------------+--------------+
| BIO-101   | 1      | Summer   | 2017 | Painter  | 514         | B            |
| BIO-301   | 1      | Summer   | 2018 | Painter  | 514         | A            |
| CS-001    | 1      | Fall     | 2017 | NULL     | NULL        | NULL         |
| CS-101    | 1      | Fall     | 2017 | Packard  | 101         | H            |
| CS-101    | 1      | Spring   | 2018 | Packard  | 101         | F            |
| CS-190    | 1      | Spring   | 2017 | Taylor   | 3128        | E            |
| CS-190    | 2      | Spring   | 2017 | Taylor   | 3128        | A            |
| CS-315    | 1      | Spring   | 2018 | Watson   | 120         | D            |
| CS-319    | 1      | Spring   | 2018 | Watson   | 100         | B            |
| CS-319    | 2      | Spring   | 2018 | Taylor   | 3128        | C            |
| CS-347    | 1      | Fall     | 2017 | Taylor   | 3128        | A            |
| EE-181    | 1      | Spring   | 2017 | Taylor   | 3128        | C            |
| FIN-201   | 1      | Spring   | 2018 | Packard  | 101         | B            |
| HIS-351   | 1      | Spring   | 2018 | Painter  | 514         | C            |
| MU-199    | 1      | Spring   | 2018 | Packard  | 101         | D            |
| PHY-101   | 1      | Fall     | 2017 | Watson   | 100         | A            |
+-----------+--------+----------+------+----------+-------------+--------------+
```

Enroll every student in the Comp. Sci. department in the above section.

```
mysql> INSERT into takes(id,course_id,sec_id,semester,year)
    -> SELECT student.id,"CS-001","1","Fall",2017
    -> FROM student
    -> WHERE student.dept_name="Comp. Sci.";
```

```
mysql> Select * from takes
    -> Where course_id="CS-001";
+-------+-----------+--------+----------+------+-------+
| ID    | course_id | sec_id | semester | year | grade |
+-------+-----------+--------+----------+------+-------+
| 00128 | CS-001    | 1      | Fall     | 2017 | NULL  |
| 12345 | CS-001    | 1      | Fall     | 2017 | NULL  |
| 54321 | CS-001    | 1      | Fall     | 2017 | NULL  |
| 76543 | CS-001    | 1      | Fall     | 2017 | NULL  |
+-------+-----------+--------+----------+------+-------+
```

Delete enrollments in the above section where the student's ID is 12345.

```
mysql> DELETE from takes
    -> where ID = 12345;
```

Delete the course CS-001. What will happen if you run this delete state-ment without first deleting offerings (sections) of this course?

```
mysql> Delete from course
    -> where course_id="CS-001";
Query OK, 1 row affected (0.01 sec)
```

Delete all takes tuples corresponding to any section of any course with the word "advanced" as a part of the title; ignore case when matching the word with the title.

```
mysql> DELETE takes from takes
    -> JOIN course using (course_id)
    -> WHERE title like "%advanced%";
Query OK, 0 rows affected (0.00 sec)
```

Using the university schema, write an SQL query to find the names of those departments whose budget is higher than that of Philosophy. List them in alphabetic Order.

```
mysql> Select dept_name
    -> From department
    -> Where Budget >
    -> (Select budget
    ->  From department
    ->  Where dept_name = "Philosophy"
    ->  );
Empty set (0.53 sec)
```

Using the university schema, use SQL to do the following: For each student who has retaken a course at least twice (i.e., the student has taken the course at least three times), show the course ID and the student's ID.
Please display your results in order of course ID and do not display duplicate rows

```
mysql> Select distinct ID
    ->  From takes
    ->  Group by ID,course_ID
    ->  Having count(ID) >= 3;
Empty set (0.00 sec)
```

Using the university schema, write an SQL query to find the name and ID of each History student whose name begins with the letter 'D' and who has not taken at least five Music courses.

```
mysql> Select Name
    -> From Student
    -> Join takes on takes.ID=Student.ID
    -> Join course on course.course_id=takes.course_id
    -> WHERE student.dept_name="History"
    -> AND Name Like "B%"
    -> AND course.dept_name="HISTORY"
    -> GROUP BY Student.id,course.course_id
    -> HAVING count(Student.id)>0;
+--------+
| Name   |
+--------+
| Brandt |
+--------+
```

Using the university schema, write an SQL query to find section(s) with maximum enrollment. The result columns should appear in the order "course_id, sec_id, year, semester, num". (It may be convenient to use the with construct.)

```
mysql> Select course_id,sec_id,year,semester,count(course_id)
    -> from takes
    -> GROUP BY sec_id,year,semester,course_id
    -> ORDER BY count(course_id) DESC
    -> Limit 1;
+-----------+--------+------+----------+------------------+
| course_id | sec_id | year | semester | count(course_id) |
+-----------+--------+------+----------+------------------+
| CS-101    | 1      | 2017 | Fall     |                6 |
+-----------+--------+------+----------+------------------+
```

Using the university schema, write an SQL query to find the ID and name of each instructor who has never given an A grade in any course she or he has taught. (Instructors who have never taught a course trivially satisfy this condition.)

```
mysql> Select ID,name
    -> From Instructor
    -> Except
    -> Select instructor.ID,name
    -> From teaches
    -> Natural join instructor
    -> Join takes using (course_id,sec_id,semester,year)
    -> Where grade = 'A';
+-------+-----------+
| ID    | name      |
+-------+-----------+
| 12121 | Wu        |
| 15151 | Mozart    |
| 22222 | Einstein  |
| 32343 | El Said   |
| 33456 | Gold      |
| 45565 | Katz      |
| 58583 | Califieri |
| 76543 | Singh     |
| 98345 | Kim       |
+-------+-----------+
```

Rewrite the preceding query, but also ensure that you include only instructors who have given at least one other non-null grade in some course.

```
mysql> Select instructor.ID,name
    -> From teaches
    -> Natural join instructor
    -> Join takes using (course_id,sec_id,semester,year)
    -> Except
    -> Select instructor.ID,name
    -> From teaches
    -> Natural join instructor
    -> Join takes using (course_id,sec_id,semester,year)
    -> Where grade like 'A%';
+-------+-----------+
| ID    | name      |
+-------+-----------+
| 45565 | Katz      |
| 98345 | Kim       |
| 12121 | Wu        |
| 32343 | El Said   |
| 22222 | Einstein  |
+-------+-----------+
```

Assignment 4

A database is being constructed for storing sales information system which store the information of
Client. The client have it own unique client number, client name, client addresses, city, address, pin code, state and total balance to be required to paid. Consider the following schema:
Client (Client_ID, Client_Name, Address, City, Pin, State, Bal_Due) Execute the following queries:

Create a View called Client_View1 having all data of Client table.

```
mysql> CREATE VIEW Client_View1 as
    -> Select * from Client;
Query OK, 0 rows affected (0.16 sec)
```

```
mysql> Select * from Client_View1;
+-----------+---------------+-------------+----------+--------+---------+----------+
| client_id | client_name   | address     | city     | pin    | state   | bal_due  |
+-----------+---------------+-------------+----------+--------+---------+----------+
| cn01001   | alak roy      |  b.d.para   | amarpur  | 799101 | tripura |   390.90 |
| cn01002   | suman roy     |  puk para   | agartala | 799102 | tripura |  2390.90 |
| cn01003   | moytree nayak |  pachmile   | tezpur   | 799152 | assam   |  9990.90 |
| cn01004   | priya das     |  chora para | kamalpur | 799301 | tripura |  1390.90 |
| cn01005   | mita mia      |  kamal para | singur   | 799721 | tripura |   190.90 |
| cn01006   | pulak roy     |  bircity    | sonitpur | 799141 | assam   |   110.90 |
| cn01007   | munni das     |  nappam     | sonitpur | 799152 | assam   |   190.90 |
| cn01008   | kusum roy     |  city_dos   | tezpur   | 799141 | assam   |   110.90 |
| cn01009   | mina das      |  pachmile   | tezpur   | 799152 | assam   |   190.90 |
| cn01010   | pauri mia     |  pachmile   | tezpur   | 799152 | assam   |  2990.90 |
| cn01011   | manali das    |  satmile    | tezpur   | 799154 | assam   |  3990.90 |
+-----------+---------------+-------------+----------+--------+---------+----------+
11 rows in set (0.04 sec)
```

Create a view called Client_vw2 having Client_ID , city and Bal_Due attributes of client table.

```
mysql> Create view Client_vw2 as
    -> Select client_id,city,bal_due
    -> from client;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> Select * from Client_vw2;
+-----------+----------+---------+
| client_id | city     | bal_due |
+-----------+----------+---------+
| cn01001   | amarpur  |  390.90 |
| cn01002   | agartala | 2390.90 |
| cn01003   | tezpur   | 9990.90 |
| cn01004   | kamalpur | 1390.90 |
| cn01005   | singur   |  190.90 |
| cn01006   | sonitpur |  110.90 |
| cn01007   | sonitpur |  190.90 |
| cn01008   | tezpur   |  110.90 |
| cn01009   | tezpur   |  190.90 |
| cn01010   | tezpur   | 2990.90 |
| cn01011   | tezpur   | 3990.90 |
+-----------+----------+---------+
11 rows in set (0.03 sec)
```

Create a view called Client_vw3 with renaming Client_ID as CID , Client_Name as cname and Address as Addr of client table.

```
mysql> Create view Client_vw3 as
    -> Select client_id as CID, client_name as cname, address as Addr
    -> From Client;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> Select * from Client_vw3;
+---------+---------------+-------------+
| CID     | cname         | Addr        |
+---------+---------------+-------------+
| cn01001 | alak roy      |  b.d.para   |
| cn01002 | suman roy     |  puk para   |
| cn01003 | moytree nayak |  pachmile   |
| cn01004 | priya das     |  chora para |
| cn01005 | mita mia      |  kamal para |
| cn01006 | pulak roy     |  bircity    |
| cn01007 | munni das     |  nappam     |
| cn01008 | kusum roy     |  city_dos   |
| cn01009 | mina das      |  pachmile   |
| cn01010 | pauri mia     |  pachmile   |
| cn01011 | manali das    |  satmile    |
+---------+---------------+-------------+
11 rows in set (0.01 sec)
```

Using Client_view1, print client_name and Balance of Client whose ID is ‘cn01001’.

```
mysql> Select client_name,bal_due
    -> From client_view1
    -> Where client_id="cn01001";
+-------------+---------+
| client_name | bal_due |
+-------------+---------+
| alak roy    |  390.90 |
+-------------+---------+
1 row in set (0.00 sec)
```

Insert a row into Client_vw2 (‘cn02003’, ‘alld’, 5000).

```
mysql> Insert into client_vw3
    -> VALUES ("cn02003","alld",5000);
Query OK, 1 row affected (0.07 sec)
```

Modify view Client_vw2 such that bal_due of Client_ID CN01004 now become 1000.

```
mysql> UPDATE client_vw2
    -> Set bal_due=1000
    -> Where client_id="cn01004";
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> Select * from client_vw2
    -> Where client_id="cn01004";
+-----------+----------+---------+
| client_id | city     | bal_due |
+-----------+----------+---------+
| cn01004   | kamalpur | 1000.00 |
+-----------+----------+---------+
1 row in set (0.00 sec)
```

Delete row from view client_vw2 where Client_ID=’CN02003’.

```
mysql> Delete from client_vw2
    -> Where client_id="cn02003";
Query OK, 1 row affected (0.12 sec)
```

Delete view client_vw3 from memory.

```
mysql> Drop view client_vw3;
Query OK, 0 rows affected (0.08 sec)
```

Consider another table Client2 (ClientID, Phone). Create a view client_vw4 which has clientID, Client_name, bal_due and phone. Use both The tables Client and Client2.

```
mysql> CREATE TABLE Client2(
    ->        ClientID     varchar(10),
    ->        Phone        varchar(15),
    ->        primary key (ClientID)
    ->        );
Query OK, 0 rows affected (0.20 sec)
```

```
mysql> Insert into client2
    -> values
    -> ('cn01002',"8349603502"),
    -> ('cn01003',"8023357127"),
    -> ('cn01004',"9101172812"),
    -> ('cn01005',"9875409872"),
    -> ('cn01006',"7698102938");
Query OK, 5 rows affected (0.10 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> Select * from client2;
+----------+------------+
| ClientID | Phone      |
+----------+------------+
| cn01001  | 8910987654 |
| cn01002  | 8349603502 |
| cn01003  | 8023357127 |
| cn01004  | 9101172812 |
| cn01005  | 9875409872 |
| cn01006  | 7698102938 |
+----------+------------+
6 rows in set (0.00 sec)
```

```
mysql> Create view client_vw4 as
    -> Select ClientID,client_name,bal_due,Phone
    -> From Client,Client2
    -> Where CLientID=client_id;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> Select * from client_vw4;
+----------+---------------+---------+------------+
| ClientID | client_name   | bal_due | Phone      |
+----------+---------------+---------+------------+
| cn01001  | alak roy      |  390.90 | 8910987654 |
| cn01002  | suman roy     | 2390.90 | 8349603502 |
| cn01003  | moytree nayak | 9990.90 | 8023357127 |
| cn01004  | priya das     | 1000.00 | 9101172812 |
| cn01005  | mita mia      |  190.90 | 9875409872 |
| cn01006  | pulak roy     |  110.90 | 7698102938 |
+----------+---------------+---------+------------+
```

A database is being constructed for storin
g sales information system. The customer who buy the
item or order the item have it own unique customer id, customer name, customer city, grade and salesman id of those salesman from which they buy the item. Each customer order is to buy item from the salesman. In the order, it has unique sales order number, sales order date, customer id, salesman id and purchase amount to be paid. The elitsalesman have salesman id, name, city and commission which shows the personal details of all salesman. Consider the following schema:
CUSTOMER (c_id, c_name, city, grade, s_id) SALESMAN (s_id, name, city, commission)
ORDERS (o_no, purchase_amt, o_date, c_id, s_id) ELITSALESMAN (s_id, name, city, commission)

Create a view for those salesmen who belong to the city 'New York'.

```
mysql> Create View View1 as
    -> Select * from salesman
    -> Where city = "New York";
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> Select * from View1;
+------+-----------+----------+------------+
| s_id | name      | city     | commission |
+------+-----------+----------+------------+
|  101 | Salesman1 | New York |       0.15 |
|  103 | Salesman3 | New York |       0.14 |
+------+-----------+----------+------------+
```

Create a view for all salesmen with columns salesman_id, name and city.

```
mysql> Create View View2 as
    -> Select s_id as salesman_id,name,city
    -> From salesman;
Query OK, 0 rows affected (0.13 sec)
```

```
mysql> Select * from View2;
+-------------+-----------+---------------+
| salesman_id | name      | city          |
+-------------+-----------+---------------+
|         101 | Salesman1 | New York      |
|         102 | Salesman2 | Los Angeles   |
|         103 | Salesman3 | New York      |
|         104 | Salesman4 | Chicago       |
|         105 | Salesman5 | San Francisco |
+-------------+-----------+---------------+
```

Find the salesmen of the city New York who achieved the commission more than 13%.

```
mysql> Select *
    -> From salesman
    -> Where commission>0.13;
+------+-----------+----------+------------+
| s_id | name      | city     | commission |
+------+-----------+----------+------------+
|  101 | Salesman1 | New York |       0.15 |
|  103 | Salesman3 | New York |       0.14 |
+------+-----------+----------+------------+
2 rows in set (0.00 sec)
```

Create a view to getting a count of how many customers we have at each level of a grade.

```
mysql> Create View View3 as
    -> Select grade,count(c_id)
    -> From Customer
    -> GROUP BY(grade);
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> Select * from View3;
+-------+-------------+
| grade | count(c_id) |
+-------+-------------+
| A     |           2 |
| B     |           2 |
| C     |           1 |
+-------+-------------+
3 rows in set (0.00 sec)
```

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

```
mysql> Create View View4 as
    -> SELECT o_date,
    ->        COUNT(DISTINCT c_id) AS "customer count",
    ->        COUNT(DISTINCT s_id) AS "salesman count"
    -> FROM orders
    -> GROUP BY o_date;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> Select * from view4;
+------------+----------------+----------------+
| o_date     | customer count | salesman count |
+------------+----------------+----------------+
| 2023-09-01 |              1 |              1 |
| 2023-09-02 |              1 |              1 |
| 2023-09-03 |              2 |              2 |
| 2023-09-04 |              1 |              1 |
| 2023-09-05 |              1 |              1 |
+------------+----------------+----------------+
5 rows in set (0.00 sec)
```

Create a view that shows for each order the salesman and customer by name.

```
mysql> Create View View5 as
    -> SELECT o_no, c_name AS "customer name", name AS "salesman name"
    -> FROM orders
    -> JOIN customer USING (c_id)
    -> JOIN salesman ON salesman.s_id = orders.s_id;
Query OK, 0 rows affected (0.16 sec)
```

```
mysql> Select * from View5;
+------+---------------+---------------+
| o_no | customer name | salesman name |
+------+---------------+---------------+
|    1 | Customer1     | Salesman1     |
|    3 | Customer1     | Salesman1     |
|    2 | Customer2     | Salesman2     |
|    4 | Customer3     | Salesman3     |
|    5 | Customer4     | Salesman4     |
|    6 | Customer5     | Salesman5     |
+------+---------------+---------------+
6 rows in set (0.02 sec)
```

Create a view that finds the salesman who has the customer with the highest order of a day.

```
mysql> Create View View6 as
    -> WITH T1 AS (
    ->   SELECT c_id, COUNT(c_id) AS order_count,s_id
    ->   FROM orders
    ->   GROUP BY c_id,s_id
    -> )
    -> SELECT DISTINCT s_id
    -> FROM T1
    -> WHERE c_id = (
    ->   SELECT c_id
    ->   FROM T1
    ->   WHERE order_count = (
    ->     SELECT MAX(order_count) AS maxx
    ->     FROM T1
    ->   )
    -> );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> Select  * from View6;
+------+
| s_id |
+------+
|  101 |
+------+
1 row in set (0.00 sec)
```

Create a view that finds the salesman who has the customer with the highest order at least 3 times on a day.

```
mysql> CREATE VIEW highestorder3 AS
    -> SELECT DISTINCT
    ->     o1.s_id AS salesman_id,
    ->     o1.o_date AS order_date,
    ->     COUNT(o1.c_id) AS order_count
    -> FROM
    ->     ORDERS o1
    -> WHERE
    ->     o1.purchase_amt = (
    ->         SELECT
    ->             MAX(o2.purchase_amt)
    ->         FROM
    ->             ORDERS o2
    ->         WHERE
    ->             DATE(o1.o_date) = DATE(o2.o_date)
    ->     )
    -> GROUP BY
    ->     o1.s_id, o1.o_date
    -> HAVING
    ->     COUNT(o1.c_id) >= 3;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM highestorder3;
Empty set (0.01 sec)
```

Create a view that shows all of the customers who have the highest grade.

```
mysql> Create View View7 as
    -> Select * from customer
    -> Where grade=(
    -> Select max(grade)
    -> From customer
    -> );
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> Select * from View7;
+------+-----------+---------+-------+------+
| c_id | c_name    | city    | grade | s_id |
+------+-----------+---------+-------+------+
|    4 | Customer4 | Chicago | C     |  104 |
+------+-----------+---------+-------+------+
1 row in set (0.00 sec)
```

Create a view that shows the number of the salesman in each city

```
mysql> Create View View8 as
    -> Select city,count(s_id)
    -> From salesman
    -> Group By city;
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> Select * from view8;
+---------------+-------------+
| city          | count(s_id) |
+---------------+-------------+
| New York      |           2 |
| Los Angeles   |           1 |
| Chicago       |           1 |
| San Francisco |           1 |
+---------------+-------------+
4 rows in set (0.00 sec)
```

Create a view that shows the average and total orders for each salesman after his or her name.

```
mysql> CREATE VIEW orderSummary AS
    -> SELECT
    ->     s.s_id AS salesman_id,
    ->     s.name AS salesman_name,
    ->     AVG(o.purchase_amt) AS avg_order_amount,
    ->     SUM(o.purchase_amt) AS total_order_amount
    -> FROM
    ->     SALESMAN s
    -> LEFT JOIN
    ->     ORDERS o ON s.s_id = o.s_id
    -> GROUP BY
    ->     s.s_id, s.name;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM orderSummary;
+-------------+--------------+-----------------+-------------------+
| salesman_id | salesman_name | avg_order_amount | total_order_amount |
+-------------+--------------+-----------------+-------------------+
|         101 | Salesman1    |      1100.000000 |           2200.00 |
|         102 | Salesman2    |       800.000000 |            800.00 |
|         103 | Salesman3    |       500.000000 |            500.00 |
|         104 | Salesman4    |      1500.000000 |           1500.00 |
|         105 | Salesman5    |       900.000000 |            900.00 |
+-------------+--------------+-----------------+-------------------+
```

(Assume all names are unique)

Create a view that shows each salesman with more than one customers.

```
mysql> CREATE VIEW salesClient AS
    -> SELECT
    ->     s.s_id AS salesman_id,
    ->     s.name AS salesman_name,
    ->     s.city AS salesman_city,
    ->     s.commission AS salesman_commission,
    ->     COUNT(DISTINCT c.c_id) AS customer_count
    -> FROM
    ->     SALESMAN s
    -> INNER JOIN
    ->     CUSTOMER c ON s.s_id = c.s_id
    -> GROUP BY
    ->     s.s_id, s.name, s.city, s.commission
    -> HAVING
    ->     COUNT(DISTINCT c.c_id) > 1;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM salesClient;
Empty set (0.01 sec)
```

Create a view that shows all matches of customers with salesman such that at least one customer in the city of customer served by a salesman in the city of the salesman.

```
mysql> CREATE VIEW customerMatch AS
    -> SELECT
    ->     c.c_id AS customer_id,
    ->     c.c_name AS customer_name,
    ->     c.city AS customer_city,
    ->     s.s_id AS salesman_id,
    ->     s.name AS salesman_name,
    ->     s.city AS salesman_city
    -> FROM
    ->     CUSTOMER c
    -> JOIN
    ->     SALESMAN s ON c.city = s.city;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM customeMatch;
ERROR 1146 (42S02): Table 'sales.customematch' doesn't exist
mysql> SELECT * FROM customerMatch;
+-------------+--------------+---------------+-------------+---------------+---------------+
| customer_id | customer_name | customer_city | salesman_id | salesman_name | salesman_city |
+-------------+--------------+---------------+-------------+---------------+---------------+
|           3 | Customer3    | New York      |         101 | Salesman1     | New York      |
|           1 | Customer1    | New York      |         101 | Salesman1     | New York      |
|           2 | Customer2    | Los Angeles   |         102 | Salesman2     | Los Angeles   |
|           3 | Customer3    | New York      |         103 | Salesman3     | New York      |
|           1 | Customer1    | New York      |         103 | Salesman3     | New York      |
|           4 | Customer4    | Chicago       |         104 | Salesman4     | Chicago       |
|           5 | Customer5    | San Francisco |         105 | Salesman5     | San Francisco |
+-------------+--------------+---------------+-------------+---------------+---------------+
```

Create a view that shows the number of orders in each day

```
mysql> CREATE VIEW orderByDay AS
    -> SELECT
    ->     DATE(o_date) AS order_date,
    ->     COUNT(o_no) AS order_count
    -> FROM
    ->     ORDERS
    -> GROUP BY
    ->     order_date;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM orderByDay;
+------------+-------------+
| order_date | order_count |
+------------+-------------+
| 2011-09-01 |           1 |
| 2011-09-02 |           1 |
| 2011-09-03 |           2 |
| 2012-09-04 |           1 |
| 2013-09-05 |           1 |
+------------+-------------+
```

Create a view that finds the salesmen who issued orders on October 10th, 2012

```
mysql> CREATE VIEW oct10 AS
    -> SELECT DISTINCT
    ->     s.s_id AS salesman_id,
    ->     s.name AS salesman_name,
    ->     s.city AS salesman_city,
    ->     s.commission AS salesman_commission
    -> FROM
    ->     SALESMAN s
    -> JOIN
    ->     ORDERS o ON s.s_id = o.s_id
    -> WHERE
    ->     DATE(o.o_date) = '2012-10-10';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM oct10;
Empty set (0.00 sec)
```

Create a view that finds the salesmen who issued orders on either August 17th, 2012 or October 10th, 2012.

```
mysql> CREATE VIEW onDate AS
    -> SELECT DISTINCT
    ->     s.s_id AS salesman_id,
    ->     s.name AS salesman_name,
    ->     s.city AS salesman_city,
    ->     s.commission AS salesman_commission
    -> FROM
    ->     SALESMAN s
    -> JOIN
    ->     ORDERS o ON s.s_id = o.s_id
    -> WHERE
    ->     DATE(o.o_date) IN ('2012-08-17', '2012-10-10');
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM onDate;
Empty set (0.00 sec)
```

Assignment 5:

Q1: Create the tables described below:

Table Name: CLIENT_MASTER
Description: Used to store client information.

Column Name	Data Type	Size	Default	Attributes
CLIENTNO	Varchar	6		Primary Key / first letter must start with 'C'
NAME	Varchar	20		Not Null
ADDRESS1	Varchar	30		
ADDRESS2	Varchar	30		
CITY	Varchar	15		
PINCODE	Int	8		
STATE	Varchar	15		
BALDUE	Float	10,2		

Table Name: PRODUCT_MASTER
Description: Used to store product information.

Column Name	Data Type	Size	Default	Attributes
PRODUCTNO	Varchar	6		Primary Key / first letter must start with 'P'
DESCRIPTION	Varchar	15		Not Null
PROFITPERCENT	Decimal	4.2		Not Null
UNITMEASURE	Varchar	10		Not Null
QTYONHAND	Int	8		Not Null
REORDERLVL	Int	8		Not Null
SELLPRICE	Decimal	8.2		Not Null, Cannot be 0
COSTPRICE	Decimal	8.2		Not Null, Cannot be 0

Table Name: SALESMAN_MASTER
Description: Used to store salesman information working for the company.

Column Name	Data Type	Size	Default	Attributes
SALESMANNO	Varchar	6		Primary Key / first letter must start with 'S'
SALESMANNAME	Varchar	20		Not Null
ADDRESS1	Varchar	30		Not Null
ADDRESS2	Varchar	30		
CITY	Varchar	20		
PINCODE	Int	8		
STATE	Varchar	20		
SALAMT	Float	8,2		Not Null, Cannot be 0
TOTTOGET	Float	6,2		Not Null, Cannot be 0
YTDSALES	Float	6,2		Not Null
REMARKS	Varchar	60		

Table Name: SALES_ORDER
Description: Used to store client's orders.

Column Name	Data Type	Size	Default	Attributes
ORDERNO	Varchar	6		Primary Key / first letter must start with 'O'
CLIENTNO	Varchar	6		Foreign Key references ClientNo of Client_Master table
ORDERDATE	Date			Not Null
DELYADDR	Varchar	25		
SALESMANNO	Varchar	6		Foreign Key references SalesmanNo of Salesman_Master table
DELYTYPE	Char	1	F	Delivery: part (P) / full (F)
BILLYN	Char	1		
DELYDATE	Date			Cannot be less than Order_Date
ORDERSTATUS	Varchar	10		Values ('In Process', 'Fulfilled', 'BackOrder', 'Cancelled')

Table Name: SALES_ORDER_DETAILS
Description: Used to store client's orders with details of each product ordered.

Column Name	Date Type	Size	Default	Attributes
ORDERNO	Varchar	6		Foreign Key reference OrderNo of Sales_Order table
PRODUCTNO	Varchar	6		Foreign Key reference ProductNo of Product_Master table
QTYORDERED	Int	8		
QTYDISP	Int	8		
PRODUCTRATE	Float	10,2		

```
mysql> create database company1;
Query OK, 1 row affected (0.06 sec)
```

```
mysql> use company1;
Database changed
```

```
mysql> CREATE TABLE CLIENT_MASTER(
    ->     CLIENTNO VARCHAR(6) PRIMARY KEY CHECK (LEFT(CLIENTNO, 1) = 'C'),
    ->     NAME VARCHAR(20) NOT NULL,
    ->     ADDRESS1 VARCHAR(30),
    ->     ADDRESS2 VARCHAR(30),
    ->     CITY VARCHAR(15),
    ->     PINCODE INT(8),
    ->     STATE VARCHAR(15),
    ->     BALDUE FLOAT(10, 2)
    -> );
Query OK, 0 rows affected, 2 warnings (1.49 sec)

mysql> CREATE TABLE PRODUCT_MASTER (
    ->     PRODUCTNO VARCHAR(6) PRIMARY KEY CHECK (LEFT(PRODUCTNO, 1) = 'P'),
    ->     DESCRIPTION VARCHAR(15) NOT NULL,
    ->     PROFITPERCENT DECIMAL(4, 2) NOT NULL,
    ->     UNITMEASURE VARCHAR(10) NOT NULL,
    ->     QTYONHAND INT(8) NOT NULL,
    ->     REORDERLVL INT(8) NOT NULL,
    ->     SELLPRICE DECIMAL(8, 2) NOT NULL CHECK (SELLPRICE <> 0),
    ->     COSTPRICE DECIMAL(8, 2) NOT NULL CHECK (COSTPRICE <> 0)
    -> );
Query OK, 0 rows affected, 2 warnings (0.20 sec)
```

```
mysql> CREATE TABLE SALESMAN_MASTER (
    ->     SALESMANNO VARCHAR(6) PRIMARY KEY CHECK (LEFT(SALESMANNO, 1) = 'S'),
    ->     SALESMANNAME VARCHAR(20) NOT NULL,
    ->     ADDRESS1 VARCHAR(30) NOT NULL,
    ->     ADDRESS2 VARCHAR(30),
    ->     CITY VARCHAR(20),
    ->     PINCODE INT(8),
    ->     STATE VARCHAR(20),
    ->     SALAMT FLOAT(8, 2) NOT NULL CHECK (SALAMT <> 0),
    ->     TOTTOGET FLOAT(6, 2) NOT NULL CHECK (TOTTOGET <> 0),
    ->     YTDSALES FLOAT(6, 2) NOT NULL,
    ->     REMARKS VARCHAR(60)
    -> );
Query OK, 0 rows affected, 4 warnings (0.18 sec)

mysql> CREATE TABLE SALES_ORDER(
    ->     ORDERNO VARCHAR(6) PRIMARY KEY CHECK (LEFT(ORDERNO, 1) = 'O'),
    ->     CLIENTNO VARCHAR(6) REFERENCES CLIENT_MASTER(ClientNo),
    ->     ORDERDATE DATE NOT NULL,
    ->     DELYADDR VARCHAR(25),
    ->     SALESMANNO VARCHAR(6) REFERENCES Salesman_Master(SalesmanNo),
    ->     DELYTYPE CHAR(1) DEFAULT 'F' CHECK (DELYTYPE IN ('P', 'F')),
    ->     BILLYN CHAR(1),
    ->     DELYDATE DATE,
    ->     ORDERSTATUS VARCHAR(10) CHECK (ORDERSTATUS IN ('In Process', 'Fulfilled', 'BackOrder', 'Cancelled'))
    -> );
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE SALES_ORDER_DETAILS(
    ->     ORDERNO VARCHAR(6) REFERENCES SALES_ORDER(OrderNo),
    ->     PRODUCTNO VARCHAR(6) REFERENCES Product_Master(ProductNo),
    ->     QTYORDERED INT(8),
    ->     QTYDISP INT(8),
    ->     PRODUCTRATE FLOAT(10, 2),
    ->     PRIMARY KEY (ORDERNO,PRODUCTNO)
    -> );
Query OK, 0 rows affected, 3 warnings (0.19 sec)

mysql> show tables;
+--------------------+
| Tables_in_company1 |
+--------------------+
| client_master      |
| product_master     |
| sales_order        |
| sales_order_details |
| salesman_master    |
+--------------------+
5 rows in set (0.95 sec)
```

2)

a. Re-insert the data generated for tables CLIENT_MASTER, PRODUCT_MASTER and SALESMAN_MASTER.

b. Data for Sales_Order table:

OrderNo	ClientNo	OrderDate	SalesmanNo	DelyType	BillYN	DelyDate	OrderStatus
O19001	C00001	12-June-04	S00001	F	N	20-July-02	In Process
O19002	C00002	25-June-04	S00002	P	N	27-June-02	Cancelled
O46865	C00003	18-Feb-04	S00003	F	Y	20-Feb-02	Fulfilled
019003	C00001	03-Apr-04	S00001	F	Y	07-Apr-02	Fulfilled
O46866	C00004	20-May-04	S00002	P	N	22-May-02	Cancelled
O19008	C00005	24-May-04	S00004	F	N	26-July-02	In Process

c. Data for Sales_Order_Details table:

OrderNo	ProductNo	QtyOrdered	QtyDisp	ProductRate
O19001	P00001	4	4	525
O19001	P07965	2	1	8400
O19001	P07885	2	1	5250
O19002	P00001	10	0	525
O46865	P07868	3	3	3150
O46865	P07885	3	1	5250
O46865	P00001	10	10	525
O46865	P0345	4	4	1050
O19003	P03453	2	2	1050
O19003	P06734	1	1	12000
O46866	P07965	1	0	8400
O46866	P07975	1	0	1050
O19008	P00001	10	5	525
O19008	P07975	5	3	1050

```
mysql> INSERT INTO CLIENT_MASTER
    -> VALUES
    ->     ('C00001', 'Ivan Bayross', 'F', '111', 'Bombay', '400054', 'Maharashtra', 15000),
    ->     ('C00002', 'Vandana', 'F', '112', 'Madras', '780001', 'Tamil Nadu', 0),
    ->     ('C00003', 'Praveen', 'F', '113', 'Bombay', '400057', 'Maharashtra', 5000),
    ->     ('C00004', 'Basu', 'F', '114', 'Bangalore', '560001', 'Karnataka', NULL),
    ->     ('C00005', 'Ravi', 'F', '115', 'Delhi', '110005', 'Delhi', 2000),
    ->     ('C00006', 'Rukhmani', 'F', '116', 'Bombay', '400060', 'Maharashtra', NULL);
Query OK, 6 rows affected (0.43 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Product_Master
    -> VALUES
    ->     ('P00001', '1.44 Floppies', '5', 'Price', '100', '20', '525', '500'),
    ->     ('P00002', 'Monitors', '6', 'Price', '10', '3', '12000', '11280'),
    ->     ('P00003', 'Mouse', '5', 'Price', '20', '5', '1050', '1000'),
    ->     ('P00004', '1.22 Floppies', '5', 'Price', '100', '20', '525', '500'),
    ->     ('P00005', 'Keyboard', '2', 'Price', '10', '3', '3150', '3050'),
    ->     ('P00006', '540 HDD', '2.5', 'Price', '10', '3', '5250', '5100');
Query OK, 6 rows affected (0.09 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
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');
Query OK, 4 rows affected (0.09 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Sales_Order
    -> VALUES
    ->     ('O19001', 'C00001', '2004-06-12', 'F-111' , 'S00001', 'F', 'N', '2002-07-20', 'In Process'),
    ->     ('O19002', 'C00002', '2004-06-25', 'F-112' , 'S00002', 'P', 'N', '2002-06-27', 'Cancelled'),
    ->     ('O46865', 'C00003', '2004-02-18', 'F-113' , 'S00003', 'F', 'Y', '2002-02-20', 'Fulfilled'),
    ->     ('O19003', 'C00001', '2004-04-03', 'F-114' , 'S00001', 'F', 'Y', '2002-04-07', 'Fulfilled'),
    ->     ('O46866', 'C00004', '2004-05-20', 'F-115' , 'S00002', 'P', 'N', '2002-05-22', 'Cancelled'),
    ->     ('O19008', 'C00005', '2004-05-24', 'F-116' , 'S00004', 'F', 'N', '2002-07-26', 'In Process');
Query OK, 6 rows affected (0.05 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Sales_Order_Details
    -> VALUES
    ->     ('O19001', 'P00001', 4, 4, 525),
    ->     ('O19001', 'P07965', 2, 1, 8400),
    ->     ('O19001', 'P07885', 2, 1, 5250),
    ->     ('O19002', 'P00001', 10, 0, 525),
    ->     ('O46865', 'P07868', 3, 3, 3150),
    ->     ('O46865', 'P07885', 3, 1, 5250),
    ->     ('O46865', 'P00001', 10, 10, 525),
    ->     ('O46865', 'P0345', 4, 4, 1050),
    ->     ('O19003', 'P03453', 2, 2, 1050),
    ->     ('O19003', 'P06734', 1, 1, 12000),
    ->     ('O46866', 'P07965', 1, 0, 8400),
    ->     ('O46866', 'P07975', 1, 0, 1050),
    ->     ('O19008', 'P00001', 10, 5, 525),
    ->     ('O19008', 'P07975', 5, 3, 1050);
Query OK, 14 rows affected (0.12 sec)
Records: 14  Duplicates: 0  Warnings: 0
```

Q3)

List the names of all clients having ‘a’ as the second letter in their names.

```
mysql> Select NAME
    -> From Client_Master
    -> Where (LEFT(Name,2) like '%A') or (LEFT(Name,2) like '%a');
+---------+
| NAME    |
+---------+
| Vandana |
| Basu    |
| Ravi    |
+---------+
3 rows in set (0.07 sec)
```

List the clients who stay in a city whose First letter is ‘M’.

```
mysql> Select Name
    -> From Client_Master
    -> Where City Like "M%";
+---------+
| Name    |
+---------+
| Vandana |
+---------+
1 row in set (7.34 sec)
```

List all clients who stay in 'Bangalore' or 'Mangalore'

```
mysql> Select Name
    -> From Client_Master
    -> Where City="Bangalore" or City="Mangalore";
+------+
| Name |
+------+
| Basu |
+------+
1 row in set (1.57 sec)
```

List all clients whose BalDue is greater than value 10000.

```
mysql> Select Name
    -> From Client_Master
    -> Where BalDue > 10000;
+--------------+
| Name         |
+--------------+
| Ivan Bayross |
+--------------+
1 row in set (0.00 sec)
```

List all information from the Sales_Order table for orders placed in the month of June.

```
mysql> Select *
    -> From Sales_Order
    -> Where MONTH(OrderDate) = 6;
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
| ORDERNO | CLIENTNO | ORDERDATE  | DELYADDR | SALESMANNO | DELYTYPE | BILLYN | DELYDATE   | ORDERSTATUS |
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
| O19001  | C00001   | 2004-06-12 | F-111    | S00001     | F        | N      | 2002-07-20 | In Process  |
| O19002  | C00002   | 2004-06-25 | F-112    | S00002     | P        | N      | 2002-06-27 | Cancelled   |
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
2 rows in set (2.57 sec)
```

List the order information for ClientNo 'C00001' and 'C00002'.

```
mysql> Select *
    -> From Sales_Order
    -> Where ClientNO = "C00001" or CLientNO = "C00002";
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
| ORDERNO | CLIENTNO | ORDERDATE  | DELYADDR | SALESMANNO | DELYTYPE | BILLYN | DELYDATE   | ORDERSTATUS |
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
| O19001  | C00001   | 2004-06-12 | F-111    | S00001     | F        | N      | 2002-07-20 | In Process  |
| O19002  | C00002   | 2004-06-25 | F-112    | S00002     | P        | N      | 2002-06-27 | Cancelled   |
| O19003  | C00001   | 2004-04-03 | F-114    | S00001     | F        | Y      | 2002-04-07 | Fulfilled   |
+---------+----------+------------+----------+------------+----------+--------+------------+-------------+
3 rows in set (0.04 sec)
```

List products whose selling price is greater than 500 and less than or equal to 750.

```
mysql> Select Description
    -> From Product_Master
    -> Where SELLPRICE > 500 and SELLPRICE <= 750;
+---------------+
| Description   |
+---------------+
| 1.44 Floppies |
| 1.22 Floppies |
+---------------+
2 rows in set (0.63 sec)
```

List products whose selling price is more than 500. Calculate a new selling price as, original selling price*.15. Rename the new column in the output of the above query as new_price.

```
mysql>  SELECT
    ->     ProductNo,
    ->     Description,
    ->     SellPrice AS OriginalSellingPrice,
    ->     (SellPrice * 0.15) AS new_price
    -> FROM Product_Master
    -> WHERE SellPrice > 500;
+-----------+--------------+----------------------+-----------+
| ProductNo | Description  | OriginalSellingPrice | new_price |
+-----------+--------------+----------------------+-----------+
| P00001    | 1.44 Floppies |               525.00 |   78.7500 |
| P00002    | Monitors     |             12000.00 | 1800.0000 |
| P00003    | Mouse        |              1050.00 |  157.5000 |
| P00004    | 1.22 Floppies |               525.00 |   78.7500 |
| P00005    | Keyboard     |              3150.00 |  472.5000 |
| P00006    | 540 HDD      |              5250.00 |  787.5000 |
+-----------+--------------+----------------------+-----------+
```

List the names, city and state of clients who are not in the state of 'Maharashtra'.

```
mysql> Select name,city,state
    -> From Client_Master
    -> Where State <> "Maharashtra";
+---------+-----------+------------+
| name    | city      | state      |
+---------+-----------+------------+
| Vandana | Madras    | Tamil Nadu |
| Basu    | Bangalore | Karnataka  |
| Ravi    | Delhi     | Delhi      |
+---------+-----------+------------+
3 rows in set (0.00 sec)
```

Count the total Int of orders.

```
mysql> Select count(OrderNo)
    -> From sales_order;
+----------------+
| count(OrderNo) |
+----------------+
|              6 |
+----------------+
1 row in set (5.32 sec)
```

Calculate the average price of all the products.

```
mysql> Select avg(sellprice)
    -> From Product_Master;
+---------------+
| avg(sellprice) |
+---------------+
|    3750.000000 |
+---------------+
1 row in set (0.19 sec)
```

Determine the maximum and minimum product prices. Rename the output as max_price and min_price respectively.

```
mysql> Select max(sellprice) as max_price, min(sellprice) as min_price
    -> From Product_Master;
+-----------+-----------+
| max_price | min_price |
+-----------+-----------+
|  12000.00 |    525.00 |
+-----------+-----------+
```

Count the Int of products having price less than or equal to 500.

```
mysql> Select count(ProductNo)
    -> From Product_Master
    -> Where sellprice <=500;
+------------------+
| count(ProductNo) |
+------------------+
|                0 |
+------------------+
1 row in set (0.02 sec)
```

List all the products whose QtyOnHand is less than recorder level.

```
mysql> Select description as Products
    -> From Product_Master
    -> Where QTYONHAND < REORDERLVL ;
Empty set (0.03 sec)
```

iList the month (in alphabets) and date when the orders must be delivered.

```
mysql> Select monthname(delydate) as month , date(delydate) as date
    -> From Sales_order;
+----------+------------+
| month    | date       |
+----------+------------+
| July     | 2002-07-20 |
| June     | 2002-06-27 |
| April    | 2002-04-07 |
| July     | 2002-07-26 |
| February | 2002-02-20 |
| May      | 2002-05-22 |
+----------+------------+
6 rows in set (0.01 sec)
```

List the OrderDate in the format 'DD-Month-YY'. E.g. 12-February-02.

```
mysql> SELECT DATE_FORMAT(ORDERDATE, '%d-%M-%y') AS FormattedOrderDate
    -> FROM Sales_Order;
+--------------------+
| FormattedOrderDate |
+--------------------+
| 12-June-04         |
| 25-June-04         |
| 03-April-04        |
| 24-May-04          |
| 18-February-04     |
| 20-May-04          |
+--------------------+
6 rows in set (1.18 sec)
```

List the date, 15 days after today's date.

```
mysql> Select ORDERNO,DATE_ADD(ORDERDATE, INTERVAL 15 DAY) as new_date
    -> From Sales_Order;
+---------+------------+
| ORDERNO | new_date   |
+---------+------------+
| O19001  | 2004-06-27 |
| O19002  | 2004-07-10 |
| O19003  | 2004-04-18 |
| O19008  | 2004-06-08 |
| O46865  | 2004-03-04 |
| O46866  | 2004-06-04 |
+---------+------------+
6 rows in set (0.21 sec)
```

4)

Print the description and total qty sold for each product.

```
mysql> Select description, count(productno) as total_sold
    -> from sales_order_details
    -> join product_master using(productno)
    -> group by productno;
+--------------+------------+
| description  | total_sold |
+--------------+------------+
| 1.44 Floppies |          4 |
+--------------+------------+
```

Find the value of each product sold.

```
mysql> Select productno, productrate
    -> From Sales_order_details;
+-----------+-------------+
| productno | productrate |
+-----------+-------------+
| P00001    |      525.00 |
| P07885    |     5250.00 |
| P07965    |     8400.00 |
| P00001    |      525.00 |
| P03453    |     1050.00 |
| P06734    |    12000.00 |
| P00001    |      525.00 |
| P07975    |     1050.00 |
| P00001    |      525.00 |
| P0345     |     1050.00 |
| P07868    |     3150.00 |
| P07885    |     5250.00 |
| P07965    |     8400.00 |
| P07975    |     1050.00 |
+-----------+-------------+
14 rows in set (0.00 sec)
```

Calculate the average qty sold for each client that has a maximum order value of 15000.00.

```
mysql> Select avg(qtyordered)
    -> From Sales_order_details
    -> Join Sales_Order using (OrderNo)
    -> Where ClientNo in
    -> (Select distinct ClientNo from sales_order_details join sales_order
    -> where productrate>15000)
    -> Group by ClientNo;
Empty set (0.01 sec)
```

Find out the total of all the billed orders for the month ofJune.

```
mysql> Select sum(qtyordered*productrate)
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Where Month(OrderDate) = 6;
+------------------------------+
| sum(qtyordered*productrate) |
+------------------------------+
|                     34650.00 |
+------------------------------+
1 row in set (0.00 sec)
```

i)

Find out the products, which have been sold to 'Ivan Bayross'.

```
mysql> Select ProductNo
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Client_Master using (ClientNo)
    -> Where Name="Ivan Bayross";
+-----------+
| ProductNo |
+-----------+
| P00001    |
| P07885    |
| P07965    |
| P03453    |
| P06734    |
+-----------+
5 rows in set (0.00 sec)
```

Find the names of clients who have purchased 'Trousers'.

```
mysql> Select distinct name
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Client_Master using (ClientNo)
    -> Join Product_Master using (ProductNo)
    -> Where Description="Trousers";
Empty set (0.00 sec)
```

List the products and orders from customers who have ordered less than 5 units of 'Pull Overs'.

```
mysql> Select name,description
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Client_Master using (ClientNo)
    -> Join Product_Master using (ProductNo)
    -> Where name not in(
    -> Select distinct name
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Client_Master using (ClientNo)
    -> Join Product_Master using (ProductNo)
    -> Where Description="Trousers"
    -> );
+-------------+---------------+
| name        | description   |
+-------------+---------------+
| Ivan Bayross | 1.44 Floppies |
| Vandana     | 1.44 Floppies |
| Ravi        | 1.44 Floppies |
| Praveen     | 1.44 Floppies |
+-------------+---------------+
4 rows in set (0.00 sec)
```

Find the products and their quantities for the orders placed by 'Ivan Bayross' and 'Mamta Muzumdar'

```
mysql> Select name,description
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Client_Master using (ClientNo)
    -> Join Product_Master using (ProductNo)
    -> Group By Name,Description
    -> Having Name in ("Ivan Bayross", "Mamta Muzumdar");
+--------------+---------------+
| name         | description   |
+--------------+---------------+
| Ivan Bayross | 1.44 Floppies |
+--------------+---------------+
1 row in set (0.00 sec)
```

Find the products and their quantities for the orders placed by ClientNo 'C00001' and C00002'.

```
mysql> Select ClientNO,description,qtyordered
    -> From Sales_order_details
    -> Join Sales_order using (OrderNo)
    -> Join Product_Master using (ProductNo)
    -> Where ClientNo in ("C00001", "C00002")
    -> Group By ClientNo,description,qtyordered;
+----------+---------------+------------+
| ClientNO | description   | qtyordered |
+----------+---------------+------------+
| C00001   | 1.44 Floppies |          4 |
| C00002   | 1.44 Floppies |         10 |
+----------+---------------+------------+
2 rows in set (0.00 sec)
```

ii)

Find the ProductNo and description of non-moving products i.e. products not being sold.

```
mysql> select productno, description from
    -> product_master
    -> Where productno not in
    -> (Select productno from sales_order_details);
+-----------+---------------+
| productno | description   |
+-----------+---------------+
| P00002    | Monitors      |
| P00003    | Mouse         |
| P00004    | 1.22 Floppies |
| P00005    | Keyboard      |
| P00006    | 540 HDD       |
+-----------+---------------+
```

List the customer Name, Address1, Address2, City and PinCode for the client who has placed order no 'O19001'.

```
mysql> select name, address1,address2,city,pincode
    -> from client_master
    -> join sales_order using(clientno)
    -> where orderno = 'O19001';
+-------------+----------+----------+--------+---------+
| name        | address1 | address2 | city   | pincode |
+-------------+----------+----------+--------+---------+
| Ivan Bayross | F       | 111      | Bombay |  400054 |
+-------------+----------+----------+--------+---------+
1 row in set (0.00 sec)
```

List if the product 'Lycra Top' has been ordered by any client and print the Client_no, Name to whom it was sold

```
mysql> select clientno,name from client_master
    -> join sales_order using (clientno)
    -> join sales_order_details using (orderno)
    -> join product_master using(productno)
    -> where product_master.description = 'Lycra Top';
Empty set (0.00 sec)
```

List the names of clients who have placed orders worth Rs. 10000 or more.

```
mysql> select distinct name from client_master
    -> join sales_order using (clientno)
    -> join sales_order_details using (orderno)
    -> where qtyordered * productrate >= 10000;
+--------------+
| name         |
+--------------+
| Ivan Bayross |
| Praveen      |
+--------------+
```

Assignment- 6

Write a PL/SQL program to print “HELLO WORLD”.

```
SQL> set serveroutput on
SQL> BEGIN
  2  dbms_output.put_line ('Hello World..');
  3  END;
  4  /
Hello World..
```

Write a PL/SQL code for inverting a number. (Example: 1234 – 432 1)

```
SQL> set serveroutput on
SQL> DECLARE
  2     n number;
  3     a number :=0;
  4  BEGIN
  5        n := &n;
  6        WHILE n > 0 LOOP
  7            a := a*10;
  8            a := a + MOD(n,10);
  9            n := FLOOR(n/10);
 10        END LOOP;
 11        dbms_output.put_line('value of a: ' || a);
 12  END;
 13  /
Enter value for n: 1567
old   5:       n := &n;
new   5:       n := 1567;
value of a: 7651

PL/SQL procedure successfully completed.
```

Write a PL/SQL code to find the greatest number among three with Anonymous blocks.

```
SQL> DECLARE
  2     x number;
  3     a number;
  4     b number;
  5     c number;
  6  BEGIN
  7        a := &a;
  8        b := &b;
  9        c := &c;
 10        IF a>=b and a>=c THEN
 11           dbms_output.put_line('Largest number is: ' || a);
 12        ELSIF b>=c THEN
 13           dbms_output.put_line('Largest number is: ' || b);
 14        ELSE
 15           dbms_output.put_line('Largest number is: ' || c);
 16        END IF;
 17  END;
 18  /
Enter value for a: 7
old   7:       a := &a;
new   7:       a := 7;
Enter value for b: 5
old   8:       b := &b;
new   8:       b := 5;
Enter value for c: 9
old   9:       c := &c;
new   9:       c := 9;
Largest number is: 9

PL/SQL procedure successfully completed.
```

Write a PL/SQL code to calculate the area of a circle where radius takes values from 3 to 7. Store the calculated area in Table AREA. The schema of table is given below: AREA (Radius, Area).

```
SQL> Set serveroutput on;
SQL> Declare
  2     i number;
  3  Begin
  4     FOR i in 3 .. 7 LOOP
  5         Insert into Area
  6         Values(i,i*i*3.14);
  7     END LOOP;
  8  END;
  9  /

PL/SQL procedure successfully completed.

SQL> Select * from Area;

    RADIUS       AREA
---------- ----------
         3      28.26
         4      50.24
         5       78.5
         6     113.04
         7     153.86

SQL>
```

Write a PL/SQL program to accept a number and find the factorial of the number.

```
SQL> DECLARE
  2     f number := 1;
  3     i number;
  4     n number;
  5  BEGIN
  6        n := &n;
  7        FOR i in 1 .. n LOOP
  8           f := f*i;
  9        END LOOP;
 10        dbms_output.put_line('value of f: ' || f);
 11  END;
 12  /
Enter value for n: 5
old   6:       n := &n;
new   6:       n := 5;
value of f: 120
```

Write a PL/SQL program to display the months between two dates of a year.

```
SQL> Set serveroutput on;
SQL> Declare
  2     start_date DATE;
  3     end_date DATE;
  4
  5  Begin
  6     start_date := TO_DATE ('12-3-2000','dd-mm-yyyy');
  7     end_date := TO_DATE ('12-11-2000','dd-mm-yyyy');
  8     IF to_char(start_date, 'MONTH') <> to_char(end_date, 'MONTH') THEN
  9        start_date := ADD_MONTHS(start_date,1);
 10     END IF;
 11     While to_char(start_date, 'MONTH') <> to_char(end_date, 'MONTH') LOOP
 12     dbms_output.put_line(to_char(start_date, 'MONTH'));
 13         start_date := ADD_MONTHS(start_date,1);
 14     END LOOP;
 15  END;
 16  /
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER

PL/SQL procedure successfully completed.
```

7. Create an Account_Master table & insert the tuples as given the question. Write a PL/SQL code that will accept an account number from user. If the balance of account is less than minimum balance, (i.e 1000) deducts Rs 100 from balance.

The schema of table is given below:

Acc_Master (acct_no, acct_holder_name , Balance);

Create table Account_Master(acct_no number(5) primarykey,acct_holder_name varchar2(10),balance number(10));

Tuples to be inserted are:

insert into Account_Master

values(1,'John',1000); insert into

Account_Master values(2,'Denis',100);

insert into Account_Master

values(3,'Albert',1100); insert into

Account_Master values(4,'Charles',700);

insert into Account_Master

values(5,'Darwin',1700);

```
SQL> Create table Account_Master(
  2  acct_no number,
  3  acct_holder_name varchar2(10),
  4  balance number,
  5  Primary key(acct_no)
  6  );

Table created.
```

```
SQL> Insert into Account_Master values(1,'John',1000);

1 row created.

SQL> Insert into Account_Master values(2,'Denis',100);

1 row created.

SQL> Insert into Account_Master values(3,'Albert',1100);

1 row created.

SQL> Insert into Account_Master values(4,'Charles',700);

1 row created.

SQL> Insert into Account_Master values(5,'Darwin',1700);

1 row created.
```

```
SQL> Select * from Account_Master;

   ACCT_NO ACCT_HOLDE    BALANCE
---------- ---------- ----------
         1 John             1000
         2 Denis             100
         3 Albert           1100
         4 Charles           700
         5 Darwin           1700
```

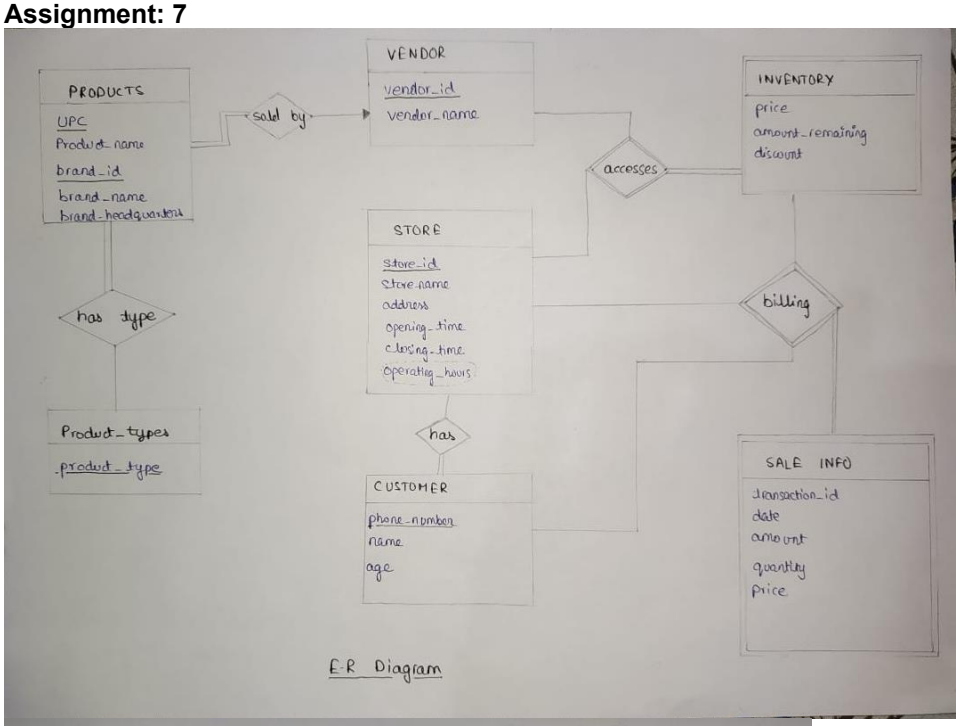
```
SQL> Set serveroutput on;
SQL> Declare
  2     minbalance number := 1000;
  3     temp_act_number number;
  4     bal number;
  5  Begin
  6     temp_act_number := &x;
  7     Select balance into bal
  8     From Account_Master
  9     Where acct_no = temp_act_number;
 10     IF bal < minbalance THEN
 11         Dbms_output.put_line('Account balance lower than minimum balance');
 12         UPDATE Account_Master
 13         Set balance=balance-100
 14         Where acct_no= temp_act_number;
 15     END IF;
 16  End;
 17  /
Enter value for x: 2
old   6:     temp_act_number := &x;
new   6:     temp_act_number := 2;
Account balance lower than minimum balance

PL/SQL procedure successfully completed.
```

```
SQL> Select * from Account_Master;

   ACCT_NO ACCT_HOLDE    BALANCE
---------- ---------- ----------
         1 John             1000
         2 Denis               0
         3 Albert           1100
         4 Charles           700
         5 Darwin           1700
```

Assignment: 7



E-R Diagram

Now we'll form the relations and check whether our design is good. If not, we'll normalize and make our D.B design better.

i) Products :

R: { UPC, Product_name, brand_id, brand_name, headquarters, vendor_id }

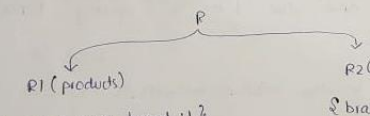
FD's: UPC → Product_name

brand_id → brand_name, headquarters, vendor_id

1NF ✓

It's not in 2NF as the P.K is UPC, brand_id but they both derive some non-prime attribute individually,

So, we'll decompose the relation as:



{ UPC, product_name, brand_id }

{ brand_id, brand_name, headquarters, vendor_id }

Our new relations are now,

Products: { UPC, Product_name, brand_id }

Brands: { brand_id, brand_name, headquarters, vendorzid }

Both of them can be proved to be normalized upto 3NF.

(ii) Vendors:

R: { vendor_id, vendor_name }

F.D's:

vendor_id → vendor_name

As there is only one fd and the L.H.S is super key, hence it's in BCNF.

So, no change is required in this relation.

(iii) Stores:

R: { store_id, store_name, address, opening_time, closing_time, operating_hours }

F.D's:

store_id → store_name, address, opening_time, closing_time, operating_hours

As there is only one F.D and the L.H.S is superkey hence it's in BCNF.

So, no change is required in this relation as well.

(iv) Product types:

R: { UPC, product_type }

No non-trivial F.D exists here

P.K = UPC, product_type

(v) Inventory:

R: { store_id, product_id, price, stock_remaining, discount }

FD's:

store_id, product_id → price, stock_remaining, discount

1NF ✓

2NF ✓ As no partial dependency exists

3NF ✓ As no transitive dependency exists

(vi) Customers:

R: { phone_no, name, age, store_id }

F.D's

phone_no → name, age, store_id

It is in BCNF.

Hence, no need to decompose

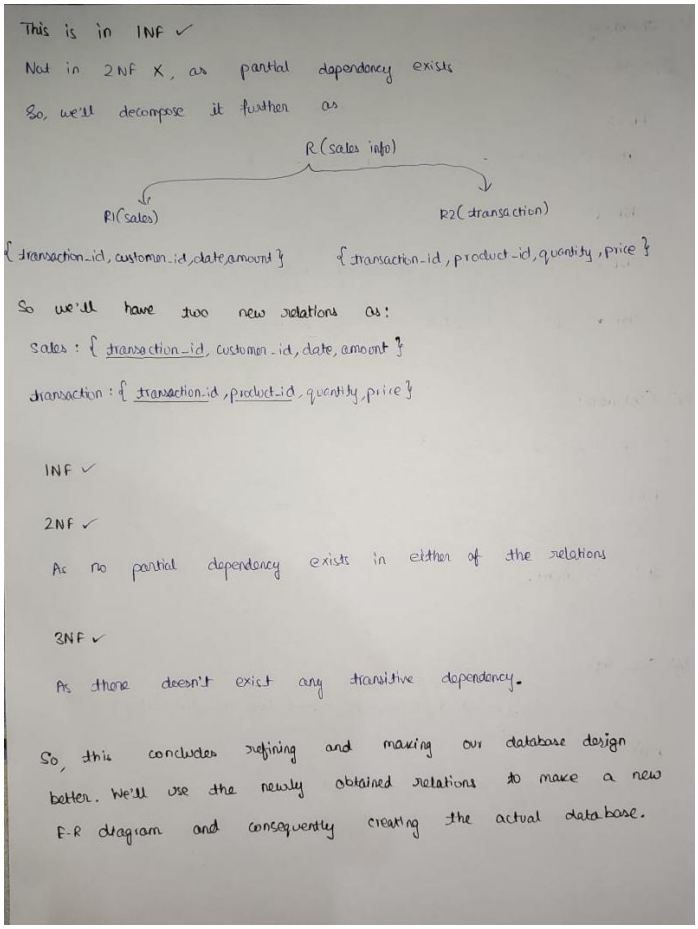
(vii) Sales info:

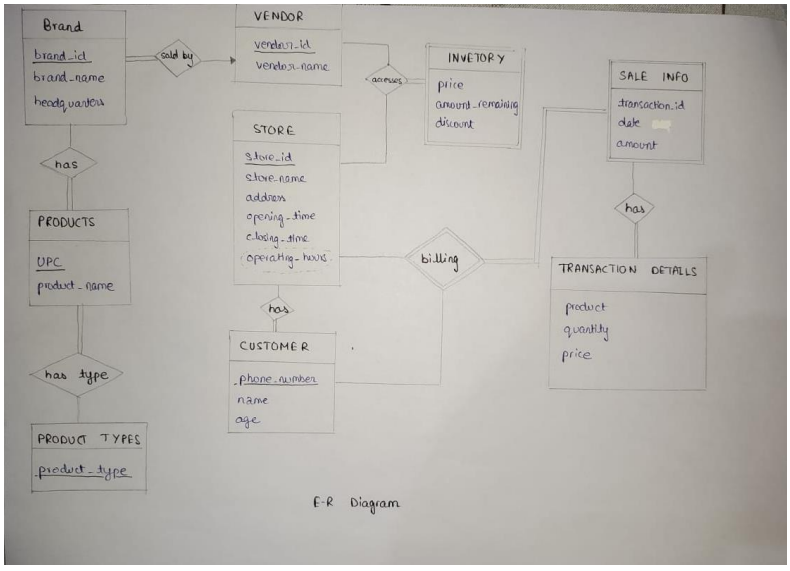
R: { transaction_id, product_id, customer_id, store_id, date, amount, quantity, price }

F.D's

transaction_id → customer_id, store_id, date, amount

product_id, transaction_id → amount, quantity, price





Creating Tables:
create database enterprise;

```
use enterprise;

create table vendor(
	vendor_id varchar(10),
	vendor_name varchar(20),
	primary key (vendor_id)
);

create table brand(
	brand_id varchar(10),
	brand_name varchar(20),
	headquarters  varchar(20),
	vendor_id varchar(10),
	primary key (brand_id),
	foreign key (vendor_id) references vendor(vendor_id)
);

create table products(
	UPC varchar(10),
	product_name varchar(20),
	brand_id  varchar(10),
	primary key(UPC),
	foreign key (brand_id) references brand(brand_id)
);

create table product_types(
	UPC varchar(10),
	product_type varchar(20),
	primary key (UPC,product_type),
	foreign key (UPC) references products(UPC)
);

create table stores(
	store_id varchar(10),
	store_name varchar(20),
	store_address varchar(20),
	opening_time time,
	closing_time time,
	primary key (store_id)
);

create table inventory(
	store_id varchar(10),
	product_id varchar(10),
	price numeric(10,2),
	stock_remaining integer,
	discount numeric(4,2) default '0.0',
	primary key (store_id,product_id),
	foreign key (store_id) references stores(store_id),
	foreign key (product_id) references products(UPC)
);

create table customers(
	phone_num numeric(10,0),
	name varchar(20),
	age integer,
	store_id varchar(10),
	primary key (phone_num),
	foreign key(store_id) references stores(store_id)
```

```
);

create table sales(
	transaction_id varchar(10),
	customer_id numeric(10,0),
	date date,
	amount numeric(12,2),
	primary key (transaction_id),
	foreign key (customer_id) references customers(phone_num)
);

create table transaction(
	transaction_id varchar(10),
	store_id varchar(10),
	product_id varchar(10),
	quantity integer,
	price numeric(8,2),
	primary key (transaction_id,product_id),
	foreign key (product_id) references products(UPC),
	foreign key (store_id) references stores(store_id)
);
```

Populating the tables:

```
INSERT INTO vendor (vendor_id, vendor_name) VALUES
('VEN0000001', 'Vendor 1'),
('VEN0000002', 'Vendor 2'),
('VEN0000003', 'Vendor 3'),
('VEN0000004', 'Vendor 4'),
('VEN0000005', 'Vendor 5');

INSERT INTO brand (brand_id, brand_name, headquarters, vendor_id) VALUES
('BRAN000001', 'Brand 1', 'Mumbai', 'VEN0000001'),
('BRAN000002', 'Brand 2', 'Delhi', 'VEN0000002'),
('BRAN000003', 'Brand 3', 'Bangalore', 'VEN0000003'),
('BRAN000004', 'Brand 4', 'Chennai', 'VEN0000004'),
('BRAN000005', 'Brand 5', 'Hyderabad', 'VEN0000005'),
('BRAN000006', 'Brand 6', 'Kolkata', 'VEN0000001'),
('BRAN000007', 'Brand 7', 'Pune', 'VEN0000002'),
('BRAN000008', 'Brand 8', 'Jaipur', 'VEN0000003'),
('BRAN000009', 'Brand 9', 'Ahmedabad', 'VEN0000004'),
('BRAN000010', 'Brand 10', 'Lucknow', 'VEN0000005');

INSERT INTO products (UPC, product_name, brand_id) VALUES
('UPC0000001', 'Product A', 'BRAN000001'),
('UPC0000002', 'Product B', 'BRAN000002'),
('UPC0000003', 'Product C', 'BRAN000003'),
('UPC0000004', 'Product D', 'BRAN000004'),
('UPC0000005', 'Product E', 'BRAN000005'),
('UPC0000006', 'Product F', 'BRAN000006'),
('UPC0000007', 'Product G', 'BRAN000007'),
('UPC0000008', 'Product H', 'BRAN000008'),
('UPC0000009', 'Product I', 'BRAN000009'),
('UPC0000010', 'Product J', 'BRAN000010'),
('UPC0000011', 'Product K', 'BRAN000001'),
('UPC0000012', 'Product L', 'BRAN000002'),
('UPC0000013', 'Product M', 'BRAN000003'),
('UPC0000014', 'Product N', 'BRAN000004'),
('UPC0000015', 'Product O', 'BRAN000005'),
```

('UPC0000016', 'Product P', 'BRAN000006'),
('UPC0000017', 'Product Q', 'BRAN000007'),
('UPC0000018', 'Product R', 'BRAN000008'),
('UPC0000019', 'Product S', 'BRAN000009'),
('UPC0000020', 'Product T', 'BRAN000010'),
('UPC0000021', 'Product U', 'BRAN000001'),
('UPC0000022', 'Product V', 'BRAN000002'),
('UPC0000023', 'Product W', 'BRAN000003'),
('UPC0000024', 'Product X', 'BRAN000004'),
('UPC0000025', 'Product Y', 'BRAN000005'),
('UPC0000026', 'Product Z', 'BRAN000006'),
('UPC0000027', 'Product AA', 'BRAN000007'),
('UPC0000028', 'Product BB', 'BRAN000008'),
('UPC0000029', 'Product CC', 'BRAN000009'),
('UPC0000030', 'Product DD', 'BRAN000010'),
('UPC0000031', 'Product EE', 'BRAN000001'),
('UPC0000032', 'Product FF', 'BRAN000002'),
('UPC0000033', 'Product GG', 'BRAN000003'),
('UPC0000034', 'Product HH', 'BRAN000004'),
('UPC0000035', 'Product II', 'BRAN000005'),
('UPC0000036', 'Product JJ', 'BRAN000006'),
('UPC0000037', 'Product KK', 'BRAN000007'),
('UPC0000038', 'Product LL', 'BRAN000008'),
('UPC0000039', 'Product MM', 'BRAN000009'),
('UPC0000040', 'Product NN', 'BRAN000010');

INSERT INTO product_types (UPC, product_type) VALUES
('UPC0000001', 'Soda'),
('UPC0000002', 'Kitchen Item'),
('UPC0000003', 'Beverage'),
('UPC0000004', 'Cleaner'),
('UPC0000005', 'Drug'),
('UPC0000006', 'Soda'),
('UPC0000007', 'Kitchen Item'),
('UPC0000008', 'Beverage'),
('UPC0000009', 'Cleaner'),
('UPC0000010', 'Drug'),
('UPC0000011', 'Soda'),
('UPC0000012', 'Kitchen Item'),
('UPC0000013', 'Beverage'),
('UPC0000014', 'Cleaner'),
('UPC0000015', 'Drug'),
('UPC0000016', 'Soda'),
('UPC0000017', 'Kitchen Item'),
('UPC0000018', 'Beverage'),
('UPC0000019', 'Cleaner'),
('UPC0000020', 'Drug'),
('UPC0000021', 'Soda'),
('UPC0000022', 'Kitchen Item'),
('UPC0000023', 'Beverage'),
('UPC0000024', 'Cleaner'),
('UPC0000025', 'Drug'),
('UPC0000026', 'Soda'),
('UPC0000027', 'Kitchen Item'),
('UPC0000028', 'Beverage'),
('UPC0000029', 'Cleaner'),
('UPC0000030', 'Drug'),
('UPC0000031', 'Soda'),
('UPC0000032', 'Kitchen Item'),
('UPC0000033', 'Beverage'),

('UPC0000034', 'Cleaner'),
('UPC0000035', 'Drug'),
('UPC0000036', 'Soda'),
('UPC0000037', 'Kitchen Item'),
('UPC0000038', 'Beverage'),
('UPC0000039', 'Cleaner'),
('UPC0000040', 'Drug');

INSERT INTO stores (store_id, store_name, store_address, opening_time, closing_time) VALUES
('STO0000001', 'Store A', 'Mumbai', '08:00:00 ', '20:00:00 '),
('STO0000002', 'Store B', 'Delhi', '09:00:00 ', '21:30:00 '),
('STO0000003', 'Store C', 'Bangalore', '08:00:00 ', '20:00:00 '),
('STO0000004', 'Store D', 'Chennai', '09:00:00 ', '21:30:00 '),
('STO0000005', 'Store E', 'Hyderabad', '08:00:00 ', '20:00:00 '),
('STO0000006', 'Store F', 'Kolkata', '10:00:00 ', '22:30:00 '),
('STO0000007', 'Store G', 'Pune', '08:00:00 ', '20:00:00 '),
('STO0000008', 'Store H', 'Jaipur', '09:00:00 ', '21:30:00 '),
('STO0000009', 'Store I', 'Ahmedabad', '08:00:00 ', '20:00:00 '),
('STO0000010', 'Store J', 'Lucknow', '09:00:00 ', '21:30:00 ');

INSERT INTO customers (phone_num, name, age, store_id) VALUES
('8910123456', 'Aarav Sharma', 28, 'STO0000001'),
('9123123456', 'Aditi Patel', 32, 'STO0000002'),
('6289123456', 'Arjun Gupta', 24, 'STO0000003'),
('9433123456', 'Diya Singh', 40, 'STO0000004'),
('8910345678', 'Kavya Kapoor', 35, 'STO0000005'),
('9123345678', 'Rohan Verma', 28, 'STO0000006'),
('6289345678', 'Isha Singh', 36, 'STO0000007'),
('9433345678', 'Vihaan Reddy', 30, 'STO0000008'),
('8910567890', 'Advait Choudhary', 27, 'STO0000009'),
('9123567890', 'Myra Yadav', 32, 'STO0000010'),
('6289567890', 'Anika Bhat', 31, 'STO0000001'),
('9433567890', 'Advik Kumar', 26, 'STO0000002'),
('8910789012', 'Nia Singh', 29, 'STO0000003'),
('9123789012', 'Reyansh Verma', 33, 'STO0000004'),
('6289789012', 'Kia Chopra', 35, 'STO0000005'),
('9433789012', 'Vihaan Raj', 30, 'STO0000006'),
('8910901234', 'Anaya Kapoor', 27, 'STO0000007'),
('9123901234', 'Vivaan Reddy', 28, 'STO0000008'),
('6289901234', 'Saanvi Yadav', 32, 'STO0000009'),
('9433901234', 'Daksh Bhat', 24, 'STO0000010');

INSERT INTO inventory (store_id, product_id, price, stock_remaining, discount) VALUES
('STO0000001', 'UPC0000001', 19.99, 100, 0),
('STO0000001', 'UPC0000002', 29.99, 75, 5),
('STO0000001', 'UPC0000003', 14.99, 50, 10),
('STO0000001', 'UPC0000004', 39.99, 120, 0),
('STO0000001', 'UPC0000005', 9.99, 60, 0),
('STO0000002', 'UPC0000001', 24.99, 90, 5),
('STO0000002', 'UPC0000002', 19.99, 80, 0),
('STO0000002', 'UPC0000003', 34.99, 70, 5),
('STO0000002', 'UPC0000004', 12.99, 110, 0),
('STO0000002', 'UPC0000005', 27.99, 65, 10),
('STO0000003', 'UPC0000001', 19.99, 100, 0),
('STO0000003', 'UPC0000002', 29.99, 75, 5),
('STO0000003', 'UPC0000003', 14.99, 50, 10),
('STO0000003', 'UPC0000004', 39.99, 120, 0),
('STO0000003', 'UPC0000005', 9.99, 60, 0),
('STO0000004', 'UPC0000001', 24.99, 90, 5),
('STO0000004', 'UPC0000002', 19.99, 80, 0),

('STO0000004', 'UPC0000003', 34.99, 70, 5),
('STO0000004', 'UPC0000004', 12.99, 110, 0),
('STO0000004', 'UPC0000005', 27.99, 65, 10),
('STO0000005', 'UPC0000001', 19.99, 100, 0),
('STO0000005', 'UPC0000002', 29.99, 75, 5),
('STO0000005', 'UPC0000003', 14.99, 50, 10),
('STO0000005', 'UPC0000004', 39.99, 120, 0),
('STO0000005', 'UPC0000005', 9.99, 60, 0),
('STO0000006', 'UPC0000001', 24.99, 90, 5),
('STO0000006', 'UPC0000002', 19.99, 80, 0),
('STO0000006', 'UPC0000003', 34.99, 70, 5),
('STO0000006', 'UPC0000004', 12.99, 110, 0),
('STO0000006', 'UPC0000005', 27.99, 65, 10),
('STO0000007', 'UPC0000001', 19.99, 100, 0),
('STO0000007', 'UPC0000002', 29.99, 75, 5),
('STO0000007', 'UPC0000003', 14.99, 50, 10),
('STO0000007', 'UPC0000004', 39.99, 120, 0),
('STO0000007', 'UPC0000005', 9.99, 60, 0),
('STO0000008', 'UPC0000001', 24.99, 90, 5),
('STO0000008', 'UPC0000002', 19.99, 80, 0),
('STO0000008', 'UPC0000003', 34.99, 70, 5),
('STO0000008', 'UPC0000004', 12.99, 110, 0),
('STO0000008', 'UPC0000005', 27.99, 65, 10),
('STO0000009', 'UPC0000001', 19.99, 100, 0),
('STO0000009', 'UPC0000002', 29.99, 75, 5),
('STO0000009', 'UPC0000003', 14.99, 50, 10),
('STO0000009', 'UPC0000004', 39.99, 120, 0),
('STO0000009', 'UPC0000005', 9.99, 60, 0),
('STO0000010', 'UPC0000001', 24.99, 90, 5),
('STO0000010', 'UPC0000002', 19.99, 80, 0),
('STO0000010', 'UPC0000003', 34.99, 70, 5),
('STO0000010', 'UPC0000004', 12.99, 110, 0),
('STO0000010', 'UPC0000005', 27.99, 65, 10);

INSERT INTO transaction (transaction_id, store_id, product_id , quantity, price) VALUES
('TRA0000001', 'STO0000001', 'UPC0000001', 2, NULL),
('TRA0000001', 'STO0000001', 'UPC0000002', 3, NULL),
('TRA0000001', 'STO0000001', 'UPC0000003', 4, NULL),
('TRA0000001', 'STO0000001', 'UPC0000004', 1, NULL),
('TRA0000002', 'STO0000002', 'UPC0000002', 3, NULL),
('TRA0000002', 'STO0000002', 'UPC0000003', 4, NULL),
('TRA0000002', 'STO0000002', 'UPC0000004', 1, NULL),
('TRA0000002', 'STO0000002', 'UPC0000005', 5, NULL),
('TRA0000003', 'STO0000003', 'UPC0000003', 2, NULL),
('TRA0000003', 'STO0000003', 'UPC0000004', 3, NULL),
('TRA0000003', 'STO0000003', 'UPC0000005', 4, NULL),
('TRA0000003', 'STO0000003', 'UPC0000001', 1, NULL),
('TRA0000004', 'STO0000004', 'UPC0000004', 2, NULL),
('TRA0000004', 'STO0000004', 'UPC0000005', 3, NULL),
('TRA0000004', 'STO0000004', 'UPC0000003', 4, NULL),
('TRA0000004', 'STO0000004', 'UPC0000001', 1, NULL),
('TRA0000005', 'STO0000005', 'UPC0000005', 2, NULL),
('TRA0000005', 'STO0000005', 'UPC0000003', 3, NULL),
('TRA0000005', 'STO0000005', 'UPC0000004', 4, NULL),
('TRA0000005', 'STO0000005', 'UPC0000002', 1, NULL),
('TRA0000006', 'STO0000006', 'UPC0000001', 2, NULL),
('TRA0000006', 'STO0000006', 'UPC0000004', 3, NULL),
('TRA0000006', 'STO0000006', 'UPC0000002', 4, NULL),
('TRA0000006', 'STO0000006', 'UPC0000003', 1, NULL),

```
('TRA0000007', 'STO0000007', 'UPC0000001', 2, NULL),
('TRA0000007', 'STO0000007', 'UPC0000002', 3, NULL),
('TRA0000007', 'STO0000007', 'UPC0000003', 4, NULL),
('TRA0000007', 'STO0000007', 'UPC0000004', 1, NULL),
('TRA0000008', 'STO0000008', 'UPC0000001', 2, NULL),
('TRA0000008', 'STO0000008', 'UPC0000002', 3, NULL),
('TRA0000008', 'STO0000008', 'UPC0000003', 4, NULL),
('TRA0000008', 'STO0000008', 'UPC0000005', 1, NULL),
('TRA0000009', 'STO0000009', 'UPC0000001', 2, NULL),
('TRA0000009', 'STO0000009', 'UPC0000002', 3, NULL),
('TRA0000009', 'STO0000009', 'UPC0000003', 4, NULL),
('TRA0000009', 'STO0000009', 'UPC0000004', 1, NULL),
('TRA0000010', 'STO0000010', 'UPC0000001', 2, NULL),
('TRA0000010', 'STO0000010', 'UPC0000002', 3, NULL),
('TRA0000010', 'STO0000010', 'UPC0000003', 4, NULL),
('TRA0000010', 'STO0000010', 'UPC0000005', 1, NULL);

Update transaction
Set price = (select price-(discount*price/100)
        from inventory
        where inventory.store_id=transaction.store_id
        and transaction.product_id=inventory.product_id);


INSERT INTO sales (transaction_id, customer_id, date) VALUES
('TRA0000001', '8910123456', '2023-11-01'),
('TRA0000002', '9123123456', '2023-11-02'),
('TRA0000003', '6289123456', '2023-11-03'),
('TRA0000004', '9433123456', '2023-11-04'),
('TRA0000005', '8910123456', '2023-11-05'),
('TRA0000006', '9123345678', '2023-11-06'),
('TRA0000007', '6289345678', '2023-11-07'),
('TRA0000008', '9433345678', '2023-11-08'),
('TRA0000009', '8910567890', '2023-11-09'),
('TRA0000010', '9123567890', '2023-11-10');

Update sales
SET amount = (
   select sum(transaction.quantity*transaction.price)
   from transaction
   where transaction.transaction_id=sales.transaction_id);
```

Queries:

Customers whose name starts with 'A':

```
select * from customers
where name like 'A%';
```

Customers whose name start with either 'K' or 'N':

```
select * from customers
where name like 'K%'
or name like 'N%';
```

Customers whose age is between 25 and 30:

```
select * from customers
where age between 25 and 30;
select * from customers
```

```
where name like 'K%'
or name like 'N%';
```

Month where most orders were placed:

```
select monthname(date) as Month from sales
group by monthname(date)
order by count(date)
limit 1;
```

Count of stores where coke outsell pepsi:

```
SELECT COUNT(*) as stores_count
FROM (
   SELECT t.store_id,
        SUM(CASE WHEN p.product_name = 'Coke' THEN quantity ELSE 0 END) as coke_sales,
        SUM(CASE WHEN p.product_name = 'Pepsi' THEN quantity ELSE 0 END) as pepsi_sales
   FROM Sales as sa
   JOIN customers ON customer_id=phone_num
         JOIN  transaction as t ON t.transaction_id=sa.transaction_id
   JOIN Products  as p ON t.product_id = p.UPC
   GROUP BY t.store_id
) sales_comparison
WHERE coke_sales >= pepsi_sales;
```

Name of top 5 customers with highest order value:

```
select name,amount from sales
join customers on customer_id=phone_num
order by amount desc
limit 5;
```

Customer with most orders:

```
select name from customers
join sales on phone_num=customer_id
group by phone_num
order by count(name) desc
limit 1;
```

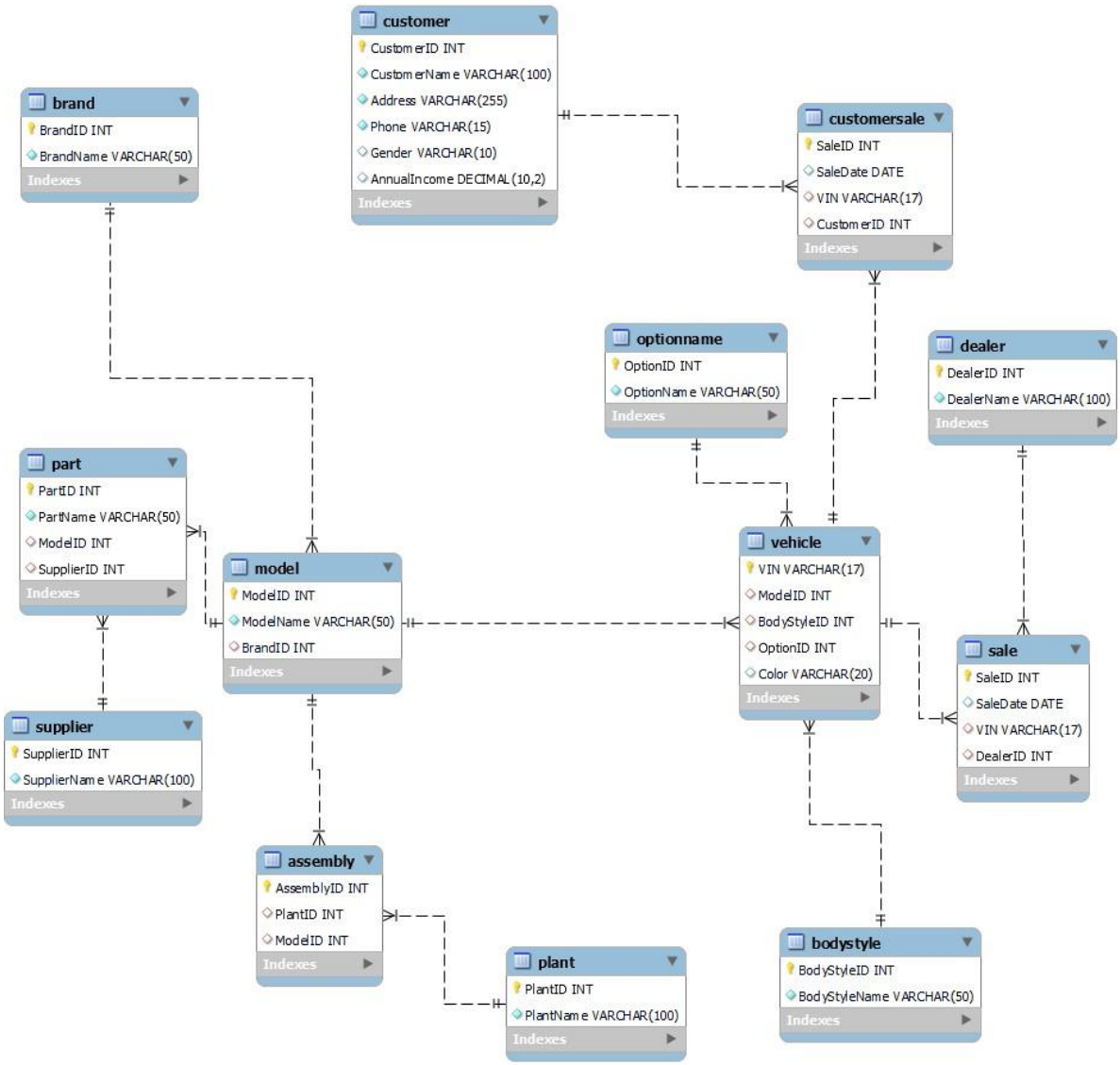
Top selling product at each store:

```
with t as(
select stores.store_name,product_name,quantity from transaction
inner join stores
join products on transaction.product_id=upc
group by stores.store_id,upc,quantity)
select * from t
group by store_name,product_name,quantity
having quantity= (Select max(quantity) from t  as t1
                                                    where t1.store_name=t.store_name

order by store_name,quantity desc;
```

Assignment 8:

ER Diagram:



Creating Tables:

```
CREATE DATABASE AUTOMOBILE_ENTERPRISE;

USE AUTOMOBILE_ENTERPRISE;

CREATE TABLE Brand (
    BrandID INT PRIMARY KEY,
    BrandName VARCHAR(50) NOT NULL
);

CREATE TABLE Model (
    ModelID INT PRIMARY KEY,
    ModelName VARCHAR(50) NOT NULL,
    BrandID INT,
    FOREIGN KEY (BrandID) REFERENCES Brand(BrandID)
);

CREATE TABLE BodyStyle (
    BodyStyleID INT PRIMARY KEY,
```

```
    BodyStyleName VARCHAR(50) NOT NULL
);

CREATE TABLE Option (
    OptionID INT PRIMARY KEY,
    OptionName VARCHAR(50) NOT NULL
);

CREATE TABLE Vehicle (
    VIN VARCHAR(17) PRIMARY KEY,
    ModelID INT,
    BodyStyleID INT,
    OptionID INT,
    Color VARCHAR(20),
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID),
    FOREIGN KEY (BodyStyleID) REFERENCES BodyStyle(BodyStyleID),
    FOREIGN KEY (OptionID) REFERENCES OptionName(OptionID)
);

CREATE TABLE Dealer (
    DealerID INT PRIMARY KEY,
    DealerName VARCHAR(100) NOT NULL
);

CREATE TABLE Sale (
    SaleID INT PRIMARY KEY,
    SaleDate DATE,
    VIN VARCHAR(17),
    DealerID INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN),
    FOREIGN KEY (DealerID) REFERENCES Dealer(DealerID)
);

CREATE TABLE Supplier (
    SupplierID INT PRIMARY KEY,
    SupplierName VARCHAR(100) NOT NULL
);

CREATE TABLE Part (
    PartID INT PRIMARY KEY,
    PartName VARCHAR(50) NOT NULL,
    ModelID INT,
    SupplierID INT,
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID),
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);

CREATE TABLE Plant (
    PlantID INT PRIMARY KEY,
    PlantName VARCHAR(100) NOT NULL
);

CREATE TABLE Assembly (
    AssemblyID INT PRIMARY KEY,
    PlantID INT,
    ModelID INT,
    FOREIGN KEY (PlantID) REFERENCES Plant(PlantID),
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID)
);
```

```
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Phone VARCHAR(15) NOT NULL,
    Gender VARCHAR(10),
    AnnualIncome DECIMAL(10, 2)
);

CREATE TABLE CustomerSale (
    SaleID INT PRIMARY KEY,
    SaleDate DATE,
    VIN VARCHAR(17),
    CustomerID INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);


CREATE TABLE Brand (
    BrandID INT PRIMARY KEY,
    BrandName VARCHAR(50) NOT NULL
);

CREATE TABLE Model (
    ModelID INT PRIMARY KEY,
    ModelName VARCHAR(50) NOT NULL,
    BrandID INT,
    FOREIGN KEY (BrandID) REFERENCES Brand(BrandID)
);

CREATE TABLE BodyStyle (
    BodyStyleID INT PRIMARY KEY,
    BodyStyleName VARCHAR(50) NOT NULL
);

CREATE TABLE OptionName (
    OptionID INT PRIMARY KEY,
    OptionName VARCHAR(50) NOT NULL
);

CREATE TABLE Vehicle (
    VIN VARCHAR(17) PRIMARY KEY,
    ModelID INT,
    BodyStyleID INT,
    OptionID INT,
    Color VARCHAR(20),
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID),
    FOREIGN KEY (BodyStyleID) REFERENCES BodyStyle(BodyStyleID),
    FOREIGN KEY (OptionID) REFERENCES Option(OptionID)
);

CREATE TABLE Dealer (
    DealerID INT PRIMARY KEY,
    DealerName VARCHAR(100) NOT NULL
);

CREATE TABLE Sale (
    SaleID INT PRIMARY KEY,
    SaleDate DATE,
```

```
    VIN VARCHAR(17),
    DealerID INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN),
    FOREIGN KEY (DealerID) REFERENCES Dealer(DealerID)
);

CREATE TABLE Supplier (
    SupplierID INT PRIMARY KEY,
    SupplierName VARCHAR(100) NOT NULL
);

CREATE TABLE Part (
    PartID INT PRIMARY KEY,
    PartName VARCHAR(50) NOT NULL,
    ModelID INT,
    SupplierID INT,
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID),
    FOREIGN KEY (SupplierID) REFERENCES Supplier(SupplierID)
);

CREATE TABLE Plant (
    PlantID INT PRIMARY KEY,
    PlantName VARCHAR(100) NOT NULL
);

CREATE TABLE Assembly (
    AssemblyID INT PRIMARY KEY,
    PlantID INT,
    ModelID INT,
    FOREIGN KEY (PlantID) REFERENCES Plant(PlantID),
    FOREIGN KEY (ModelID) REFERENCES Model(ModelID)
);

CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Phone VARCHAR(15) NOT NULL,
    Gender VARCHAR(10),
    AnnualIncome DECIMAL(10, 2)
);

CREATE TABLE CustomerSale (
    SaleID INT PRIMARY KEY,
    SaleDate DATE,
    VIN VARCHAR(17),
    CustomerID INT,
    FOREIGN KEY (VIN) REFERENCES Vehicle(VIN),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)
);
```

Populating the database:

```
INSERT INTO Brand (BrandID, BrandName)
VALUES
    (1, 'GM'),
    (2, 'Volkswagen');
```

```
INSERT INTO Model (ModelID, ModelName, BrandID)
VALUES
    (1, 'Enclave', 1),
    (2, 'LaCrosse', 1),
    (3, 'Mariner', 2),
    (4, 'Milan', 2);


INSERT INTO BodyStyle (BodyStyleID, BodyStyleName)
VALUES
    (1, '4-door'),
    (2, 'Wagon');


INSERT INTO OptionName (OptionID, OptionName)
VALUES
    (1, 'Red'),
    (2, 'Automatic'),
    (3, 'V6 Engine');


INSERT INTO Vehicle (VIN, ModelID, BodyStyleID, OptionID, Color)
VALUES
    ('ABC12345678901234', 1, 1, 1, 'Red'),
    ('DEF12345678901234', 2, 2, 2, 'Blue');


INSERT INTO Dealer (DealerID, DealerName)
VALUES
    (1, 'ABC Dealership'),
    (2, 'XYZ Dealership');


INSERT INTO Sale (SaleID, SaleDate, VIN, DealerID)
VALUES
    (1, '2023-01-15', 'ABC12345678901234', 1),
    (2, '2023-02-20', 'DEF12345678901234', 2);


INSERT INTO Supplier (SupplierID, SupplierName)
VALUES
    (1, 'Parts Supplier A'),
    (2, 'Parts Supplier B');


INSERT INTO Part (PartID, PartName, ModelID, SupplierID)
VALUES
    (1, 'Engine Part', 1, 1),
    (2, 'Transmission Part', 2, 2);


INSERT INTO Plant (PlantID, PlantName)
VALUES
    (1, 'Assembly Plant A'),
    (2, 'Assembly Plant B');


INSERT INTO Assembly (AssemblyID, PlantID, ModelID)
VALUES
```

```
    (1, 1, 1),
    (2, 2, 2);


INSERT INTO Customer (CustomerID, CustomerName, Address, Phone, Gender, AnnualIncome)
VALUES
    (1, 'John Doe', '123 Main St', '555-1234', 'Male', 50000.00),
    (2, 'Jane Smith', '456 Oak St', '555-5678', 'Female', 60000.00),
    (3, 'Bob Johnson', '789 Pine St', '555-9876', 'Male', 75000.00);


INSERT INTO CustomerSale (SaleID, SaleDate, VIN, CustomerID)
VALUES
    (101, '2023-01-15', 'ABC12345678901234', 1),
    (102, '2023-02-20', 'DEF12345678901234', 2);
```

Queries:

Sample Queries:

Top 2 brands by unit sales in the past year.

```
SELECT
    Brand.BrandID,
    Brand.BrandName,
    COUNT(Sale.SaleID) AS UnitSale
FROM
    Brand
        JOIN
    Model ON Brand.BrandID = Model.BrandID
        JOIN
    Vehicle ON Model.ModelID = Vehicle.ModelID
        JOIN
    Sale ON Vehicle.VIN = Sale.VIN
WHERE
    Sale.SaleDate BETWEEN DATE_SUB(CURDATE(), INTERVAL 1 YEAR) AND CURDATE()
GROUP BY Brand.BrandID , Brand.BrandName
ORDER BY UnitSale DESC
LIMIT 2;
```

Retrieve Vehicle Information for a Specific Customer

```
SELECT V.VIN, M.model_name, O.color, O.engine_type, O.transmisson, S.date, S.price
FROM Vehicle V
JOIN Options O ON V.model_id = O.model_id AND V.option_id = O.option_id
JOIN Model M ON V.model_id = M.model_id
JOIN Sale S ON V.VIN = S.VIN
WHERE V.customer_id = 'C1';
```

Retrieve Parts Ordered by Manufacturer

```
SELECT PO.order_id, P.part_type, S.name AS supplier_name, PO.date
FROM Part_Order PO
JOIN Part P ON PO.part_id = P.part_id
JOIN Supplier S ON PO.supplier_id = S.supplier_id
WHERE PO.manufacturer_id = 'TM1';
```

Calculate Total Sales Revenue for a Dealer

```
SELECT D.name AS dealer_name, SUM(S.price) AS total_revenue
FROM Dealer D
JOIN Sale S ON D.dealer_id = S.dealer_id
WHERE D.name = 'Sunrise Auto Dealers'
GROUP BY D.name;
```

Find those dealers who keep a vehicle in inventory for the longest average time.

```
SELECT
   Dealer.DealerID,
   Dealer.DealerName,
   AVG(DATEDIFF(CURDATE(), Sale.SaleDate)) AS AverageDaysInInventory
FROM
   Dealer
JOIN Sale ON Dealer.DealerID = Sale.DealerID
GROUP BY
   Dealer.DealerID, Dealer.DealerName
ORDER BY
   AverageDaysInInventory DESC;
```

Suppose that it is found that transmissions made by supplier Getrag between two given dates are defective. Find the VIN of each car containing such a transmission and the customer to which it was sold. If your design allows, suppose the defective transmissions all come from only one of Getrag’ s plants.

```
SELECT
   Sale.VIN,
   Customer.CustomerName
FROM
   Sale
JOIN Vehicle ON Sale.VIN = Vehicle.VIN
JOIN Model ON Vehicle.ModelID = Model.ModelID
JOIN Part ON Model.ModelID = Part.ModelID
JOIN Supplier ON Part.SupplierID = Supplier.SupplierID
JOIN Plant ON Supplier.SupplierID = Plant.PlantID
JOIN Assembly ON Model.ModelID = Assembly.ModelID
JOIN CustomerSale ON Sale.SaleID = CustomerSale.SaleID
JOIN Customer ON CustomerSale.CustomerID = Customer.CustomerID
WHERE
   Sale.SaleDate BETWEEN 'start_date' AND 'end_date'
   AND Plant.PlantName = 'Assembly Plant A';
```