SJSU ID: 015956473,016197896,016078998,016099304

Q1.a.

Following are the DML commands which I have performed in the 'employees' database

1. Create Table-creating a new table 'emp_leave'

```
mysql> create table emp_leave(leave_id int, leave_status varchar(10),leave_start_date date);
Query OK, 0 rows affected (0.05 sec)

mysql> describe emp_leave;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| leave_id         | int         | YES  |     | NULL    |       |
| leave_status     | varchar(10) | YES  |     | NULL    |       |
| leave_start_date | date        | YES  |     | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

Fig 1 : Create command

2. Alter table-Adding column 'salary_month' in salaries table.

```
mysql> describe salaries;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| emp_no    | int  | NO   | PRI | NULL    |       |
| salary    | int  | NO   |     | NULL    |       |
| from_date | date | NO   | PRI | NULL    |       |
| to_date   | date | NO   |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> alter table salaries add salary_month char(10);
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe salaries;
+--------------+---------+------+-----+---------+-------+
| Field        | Type    | Null | Key | Default | Extra |
+--------------+---------+------+-----+---------+-------+
| emp_no       | int     | NO   | PRI | NULL    |       |
| salary       | int     | NO   |     | NULL    |       |
| from_date    | date    | NO   | PRI | NULL    |       |
| to_date      | date    | NO   |     | NULL    |       |
| salary_month | char(10)| YES  |     | NULL    |       |
+--------------+---------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Fig 2 : Alter command

3. **Alter -Modifying column ‘gender varchar(20)’ in manager table to ‘gender varchar(10)’**

```
mysql> describe manager;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| manager_id | int         | NO   | PRI | NULL    |       |
| name       | varchar(20) | YES  |     | NULL    |       |
| department | varchar(20) | YES  |     | NULL    |       |
| phone_no   | int         | YES  |     | NULL    |       |
| gender     | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.19 sec)

mysql> alter table manager modify column gender varchar(10);
Query OK, 0 rows affected (0.29 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe manager;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| manager_id | int         | NO   | PRI | NULL    |       |
| name       | varchar(20) | YES  |     | NULL    |       |
| department | varchar(20) | YES  |     | NULL    |       |
| phone_no   | int         | YES  |     | NULL    |       |
| gender     | varchar(10) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Fig 3 : Alter command 2

4. **Alter table - renaming table ‘dept_emp’ to ‘department_employee’**

```
mysql> show tables;
+----------------------+
| Tables_in_employees  |
+----------------------+
| current_dept_emp     |
| departments          |
| dept_emp             |
| dept_emp_latest_date |
| dept_manager         |
| emp_leave            |
| employees            |
| manager              |
| salaries             |
| titles               |
+----------------------+
10 rows in set (0.04 sec)

mysql> alter table dept_emp rename to department_employee;
Query OK, 0 rows affected (0.23 sec)
```

Fig 4 : Alter command 3

5. Alter table -creating index on 'titles' table.

```
mysql> describe titles;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| emp_no    | int         | NO   | PRI | NULL    |       |
| title     | varchar(50) | NO   | PRI | NULL    |       |
| from_date | date        | NO   | PRI | NULL    |       |
| to_date   | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.06 sec)

mysql> create index emp_title on titles (emp_no,title);
Query OK, 0 rows affected (2.49 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Fig 5 : Creating index.

Q1.b.

1. Inserting values in Departments table-

```
mysql> insert into departments values('d010','Executive Data Analyst');
Query OK, 1 row affected (0.12 sec)

mysql> insert into departments values('d011','Manager Data Analyst');
Query OK, 1 row affected (0.22 sec)

mysql> insert into departments values('d012','Executive insights');
Query OK, 1 row affected (0.05 sec)
```

Fig 6 : Insert command 1

2. Inserting values in Employees table-

```
mysql> insert into employees values ('5000000','1971-09-08','John','Snow','M','2000-09-11');
Query OK, 1 row affected (0.03 sec)

mysql> insert into employees values ('5000001','1969-11-03','Martha','Dsouza','F','2002-03-04');
Query OK, 1 row affected (0.04 sec)
```

Fig 7 : Insert command 2

3. Inserting values in Manager table-

```
mysql> insert into manager values ('5000000','John','Engineering','8739287547','M');
ERROR 1264 (22003): Out of range value for column 'phone_no' at row 1
mysql> insert into manager values ('5000000','John','Engineering','87392875','M');
Query OK, 1 row affected (0.04 sec)

mysql> insert into manager values ('5000001','Martha','Engineering','67329355','F');
Query OK, 1 row affected (0.03 sec)
```

Fig 8 : Insert command 3

4. Inserting values in department_employee table-

```
mysql> insert into department_employee values('5000000','d010','2000-09-08','2010-01-01');
Query OK, 1 row affected (0.10 sec)

mysql> insert into department_employee values('5000001','d011','2002-09-08','2003-01-01');
Query OK, 1 row affected (0.05 sec)
```

Fig 9 : Insert command 4

5. Inserting values in Salaries table-

```
mysql> insert into salaries values('5000000','7000000','2000-09-08','March');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
mysql> insert into salaries values('5000000','7000000','2000-09-08','2010-01-01','March');
Query OK, 1 row affected (0.08 sec)

mysql> insert into salaries values('5000001','5000000','2002-09-08','2003-01-01','March');
Query OK, 1 row affected (0.01 sec)
```

Fig 10 : Insert command 5

Q1.c.

1. Add an attribute, Phone, to the employees table

```
mysql> alter table employees add phone varchar(15);
Query OK, 0 rows affected (0.50 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe employees;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_no     | int           | NO   | PRI | NULL    |       |
| birth_date | date          | NO   |     | NULL    |       |
| first_name | varchar(14)   | NO   |     | NULL    |       |
| last_name  | varchar(16)   | NO   |     | NULL    |       |
| gender     | enum('M','F') | NO   |     | NULL    |       |
| hire_date  | date          | NO   |     | NULL    |       |
| phone      | varchar(15)   | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
7 rows in set (0.02 sec)
```

Fig 11 : Adding attribute

2. Remove the titles table

```
mysql> drop table titles;
Query OK, 0 rows affected (0.19 sec)
```

Fig 12 : Removing title table

3. Change the dept_no field from 4 characters to 5 characters

```
mysql> describe dept_manager;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| emp_no    | int     | NO   | PRI | NULL    |       |
| dept_no   | char(4) | NO   | PRI | NULL    |       |
| from_date | date    | NO   |     | NULL    |       |
| to_date   | date    | NO   |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> alter table dept_manager modify column dept_no char(5);
ERROR 1832 (HY000): Cannot change column 'dept_no': used in a foreign key constraint 'dept_manager_ibfk_2'
```

Fig 13 : Changing Field

'dept_no' field cannot be altered because 'dept_no' is itself a Primary Key and it cannot be altered.

4. Add a new department called "Medical" with number d0010 and later delete it

```
mysql> insert into departments values('d0010','Medical');
ERROR 1406 (22001): Data too long for column 'dept_no' at row 1
mysql> describe departments;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dept_no   | char(4)     | NO   | PRI | NULL    |       |
| dept_name | varchar(40) | NO   | UNI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.02 sec)
```

Fig 14 : Adding values

Values cannot be added because the dept_no has length limit (4) and 'd0010' has 5 values

5. Modify the department name from “Development” to “Engineering”

```
mysql> select * from departments;
+---------+----------------------+
| dept_no | dept_name            |
+---------+----------------------+
| d009    | Customer Service     |
| d005    | Development          |
| d010    | Executive Data Analyst |
| d012    | Executive insights   |
| d002    | Finance              |
| d003    | Human Resources      |
| d011    | Manager Data Analyst |
| d001    | Marketing            |
| d004    | Production           |
| d006    | Quality Management   |
| d008    | Research             |
| d007    | Sales                |
+---------+----------------------+
12 rows in set (0.00 sec)

mysql> update departments set dept_name = 'Engineering' where dept_name = 'Development';
Query OK, 1 row affected (0.07 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from departments;
+---------+----------------------+
| dept_no | dept_name            |
+---------+----------------------+
| d009    | Customer Service     |
| d005    | Engineering          |
| d010    | Executive Data Analyst |
| d012    | Executive insights   |
| d002    | Finance              |
| d003    | Human Resources      |
| d011    | Manager Data Analyst |
| d001    | Marketing            |
| d004    | Production           |
| d006    | Quality Management   |
| d008    | Research             |
| d007    | Sales                |
+---------+----------------------+
12 rows in set (0.00 sec)
```

Fig 14 : Modifying values

Q2.a.

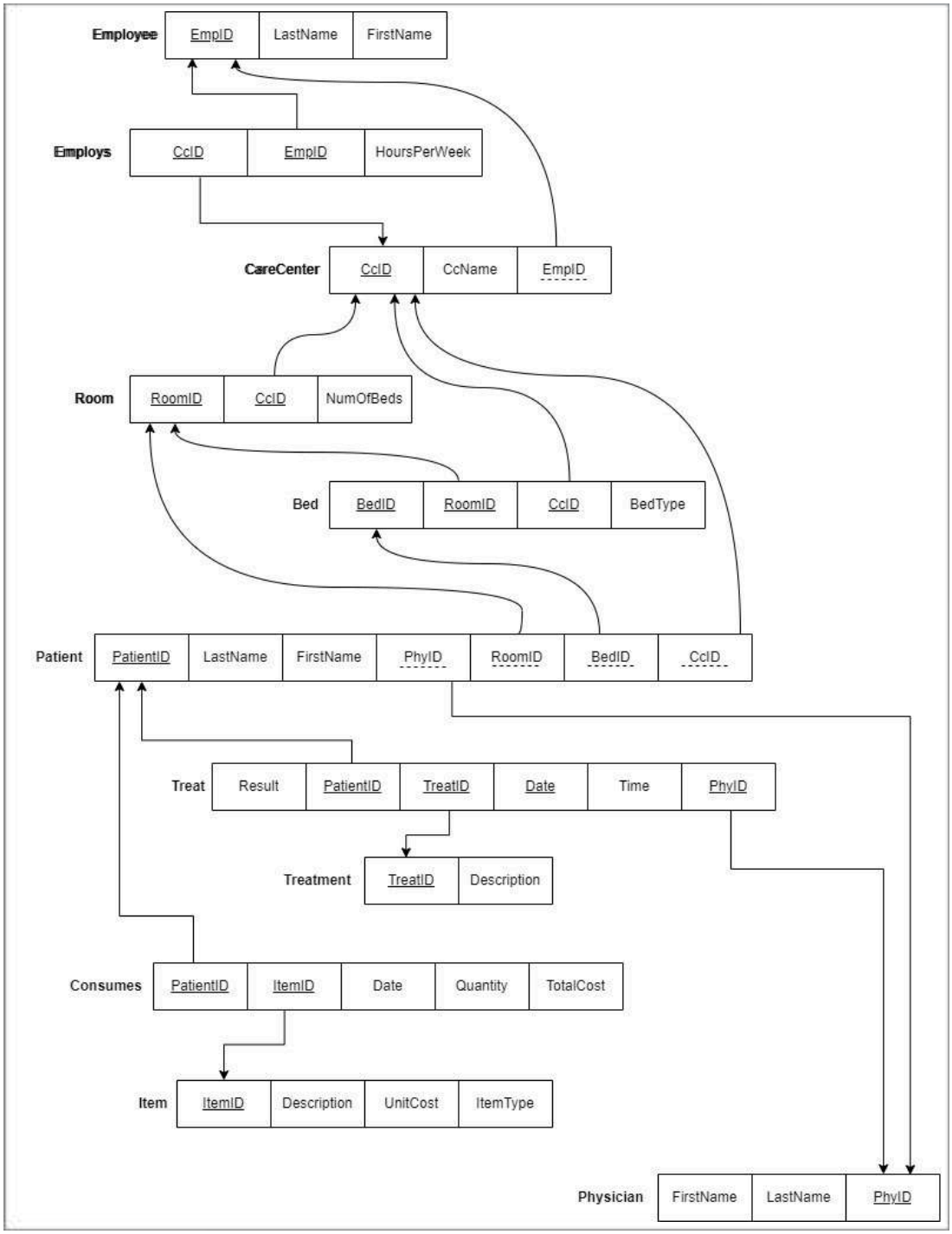


Fig 14 : Relational Schema

Q2.b.

Created a database for the Mountain View Community Hospital ER Diagram named ***MVC_Hospital***.

The screenshots below have SQL DDL statements for creating each table in the database ensuring domain, entity and referential integrity constraints.

```
MySQL 8.0 Command Line Client
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| employees          |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
7 rows in set (0.00 sec)

mysql> create database MVC_Hospital;
Query OK, 1 row affected (0.00 sec)

mysql> use MVC_Hospital;
Database changed
```

The table ***Employee*** is a strong entity with *EmpID* as primary key.

```
mysql> create table Employee(EmpID int not null, LastName varchar(20), FirstName varchar(20), constraint employee_pk primary key(EmpID));
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> describe Employee;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| EmpID     | int         | NO   | PRI | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

CareCenter has a primary key *CcID* and foreign key from *Employee* table; *EmpID*.

```
mysql> create table CareCenter(CcID int not null, CcName varchar(20), EmpID int, constraint carecenter_pk primary key(CcID), constraint employee_fk foreign key(EmpID) refer
ences Employee(EmpID) on update cascade);
Query OK, 0 rows affected (0.03 sec)

mysql> describe CareCenter;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| CcID   | int         | NO   | PRI | NULL    |       |
| CcName | varchar(20) | YES  |     | NULL    |       |
| EmpID  | int         | YES  | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Employs is an associative entity that intersects *Employee* and *CareCenter* table. Its primary key is a composite of *EmpID* and *CcID*, foreign keys from both the tables.

```
mysql> create table Employs(CcID int not null, EmpID int not null, HoursPerWeek int, constraint employs_pk primary key(CcID, EmpID), constraint carecenter_fk foreign key(Cc
ID) references CareCenter(CcID) on update restrict, constraint employee_fk1 foreign key(EmpID) references Employee(EmpID) on update restrict);
Query OK, 0 rows affected (0.01 sec)

mysql> describe Employs;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| CcID        | int  | NO   | PRI | NULL    |       |
| EmpID       | int  | NO   | PRI | NULL    |       |
| HoursPerWeek | int | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

The table ***Room*** has a primary key composite of its own unique key *RoomID* and foreign key *CcID* from *CareCenter* table.

```
mysql> create table Room(RoomID int not null, CcID int not null, NumOfBeds int, constraint room_pk primary key(RoomID, CcID), constraint carecenter_fk1 foreign key(CcID) re
ferences CareCenter(CcID) on update restrict);
Query OK, 0 rows affected (0.02 sec)

mysql> describe Room;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| RoomID    | int  | NO   | PRI | NULL    |       |
| CcID      | int  | NO   | PRI | NULL    |       |
| NumOfBeds | int  | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Bed table is a weak entity which cannot exist on its own. It has a composite primary key which consists of *BedID*; its own unique key and foreign keys *RoomID* and *CcID* from tables *Room* and *CareCenter*.

```
mysql> create table Bed(BedID int not null, RoomID int not null, CcID int not null, BedType varchar(50), constraint bed_pk primary key(BedID, RoomID, CcID), constraint room
_fk foreign key(RoomID) references Room(RoomID) on update restrict, constraint carecenter_fk2 foreign key(CcID) references CareCenter(CcID) on update restrict);
Query OK, 0 rows affected (0.02 sec)

mysql> describe Bed;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| BedID   | int         | NO   | PRI | NULL    |       |
| RoomID  | int         | NO   | PRI | NULL    |       |
| CcID    | int         | NO   | PRI | NULL    |       |
| BedType | varchar(50) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Physician is a strong entity with *PhyID* as its primary key.

```
mysql> create table Physician(PhyID int not null, LastName varchar(20), FirstName varchar(20), constraint physician_pk primary key(PhyID));
Query OK, 0 rows affected (0.01 sec)

mysql> describe Physician;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| PhyID     | int         | NO   | PRI | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

The table ***Treatment*** has a primary key *TreatID*.

```
mysql> create table Treatment(TreatID int not null, Description varchar(200), constraint treatment_pk primary key(TreatID));
Query OK, 0 rows affected (0.01 sec)

mysql> describe Treatment;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| TreatID     | int          | NO   | PRI | NULL    |       |
| Description | varchar(200) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Item is a strong entity with primary key *ItemID*.

```
mysql> create table Item(ItemID int not null, Description varchar(200), UnitCost decimal, ItemType varchar(50), constraint item_pk primary key(ItemID));
Query OK, 0 rows affected (0.01 sec)

mysql> describe Item;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| ItemID      | int          | NO   | PRI | NULL    |       |
| Description | varchar(200) | YES  |     | NULL    |       |
| UnitCost    | decimal(10,0)| YES  |     | NULL    |       |
| ItemType    | varchar(50)  | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Patient has a primary key PatientID with foreign keys PhyID, BedID, RoomId and CcID from tables Physician, Bed, Room and CareCenter.

```
mysql> create table Patient(PatientID int not null, LastName varchar(20), FirstName varchar(20), PhyID int, BedID int, RoomID int, CcID int, constraint patient_pk primary k
ey(PatientID), constraint physician_fk foreign key(PhyID) references Physician(PhyID) on update cascade, constraint bed_fk foreign key(BedID) references Bed(BedID) on updat
e cascade, constraint room_fk2 foreign key(RoomID) references Room(RoomID) on update cascade, constraint carecenter_fk3 foreign key(CcID) references CareCenter(CcID) on upd
ate cascade);
Query OK, 0 rows affected (0.02 sec)

mysql> describe Patient;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| PatientID | int         | NO   | PRI | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
| PhyID     | int         | YES  | MUL | NULL    |       |
| BedID     | int         | YES  | MUL | NULL    |       |
| RoomID    | int         | YES  | MUL | NULL    |       |
| CcID      | int         | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Treat has a composite primary key with attributes *TreatID*, *PhyID*, *PatientID* and *Date*. Date is used here as one of the attributes to uniquely identify each row of the table because one patient can visit the same physician for the same treatment multiple times, but each appointment will have a unique date.

```
mysql> create table Treat(PhyID int not null, PatientID int not null, TreatID int not null, Date date not null, Time time, Result varchar(200), constraint treat_pk primary
key(PhyID, PatientID, TreatID, Date), constraint physician_fk2 foreign key(PhyID) references Physician(PhyID) on update restrict, constraint patient_fk foreign key(PatientI
D) references Patient(PatientID) on update restrict, constraint treatment_fk foreign key(TreatID) references Treatment(TreatID) on update restrict);
Query OK, 0 rows affected (0.02 sec)

mysql> describe Treat;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| PhyID     | int          | NO   | PRI | NULL    |       |
| PatientID | int          | NO   | PRI | NULL    |       |
| TreatID   | int          | NO   | PRI | NULL    |       |
| Date      | date         | NO   | PRI | NULL    |       |
| Time      | time         | YES  |     | NULL    |       |
| Result    | varchar(200) | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Consumes is an associative entity bridging the tables *Patient* and *Item*. It has a composite primary key with attributes *PatientID* and *ItemID*.

```
mysql> create table Consumes(PatientID int not null, ItemID int not null, Date date, Quantity int, TotalCost float, constraint consumes_pk primary key(PatientID, ItemID), c
onstraint patient_fk2 foreign key(PatientID) references Patient(PatientID) on update restrict, constraint item_fk foreign key(ItemID) references Item(ItemID) on update rest
rict);
Query OK, 0 rows affected (0.02 sec)

mysql> describe Consumes;
+-----------+-------+------+-----+---------+-------+
| Field     | Type  | Null | Key | Default | Extra |
+-----------+-------+------+-----+---------+-------+
| PatientID | int   | NO   | PRI | NULL    |       |
| ItemID    | int   | NO   | PRI | NULL    |       |
| Date      | date  | YES  |     | NULL    |       |
| Quantity  | int   | YES  |     | NULL    |       |
| TotalCost | float | YES  |     | NULL    |       |
+-----------+-------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

All the tables in the database -

Q3.a.

Below, we have attached screenshots to 10 SQL constructs covered during lectures. Each query captures a concept unique to our recent learning.

i. **Auto increment or Generated Always As Identity** – The following create table query has the primary key *course_id* increment by 1 every time a user inputs new data because of the *auto_increment* keyword used during table creation

ii. **On Update Set Null** – In the create table query below, we have exercised referential integrity *on update set null* on the foreign key *emp_no* referencing the primary key of *employees* table. This will set the *emp_no* data to *Null* in *certification_level* table every time there's an update corresponding to the referenced *emp_no* row in *employees* table

iii. **Check and Composite Primary Key** – We have exercised two SQL constructs in the following query. Introduced a *Check* constraint while defining the attribute *certificate* to check that it is one of the 5 values mentioned in the parenthesis. If not, it can be *Null*

The primary key of *course_completion* table is a composite of two foreign keys; *course_id* from table *courses* and *c_level* from table *certification_level*

```
mysql> create table course_completion(course_id int not null, c_level int not null, certificate int check(certificate in('1', '2', '3', '4', '5')), constraint ccpk primary
key(course_id, c_level), constraint ccfk1 foreign key(course_id) references courses(course_id), constraint ccfk2 foreign key(c_level) references certification_level(c_level
));
Query OK, 0 rows affected (0.03 sec)

mysql> describe course_completion;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| course_id   | int  | NO   | PRI | NULL    |       |
| c_level     | int  | NO   | PRI | NULL    |       |
| certificate | int  | YES  |     | NULL    |       |
+-------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

iv. **Alter column.. Set default** – The table action *set default* allows database designer to set a default value to an attribute when no input is provided for incoming data

In the query below, the column *certificate* of table *course_completion* will automatically insert "0" for every row where user does not enter data

```
mysql> alter table course_completion alter column certificate set default 0;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe course_completion;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| course_id   | int  | NO   | PRI | NULL    |       |
| c_level     | int  | NO   | PRI | NULL    |       |
| certificate | int  | YES  |     | 0       |       |
+-------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

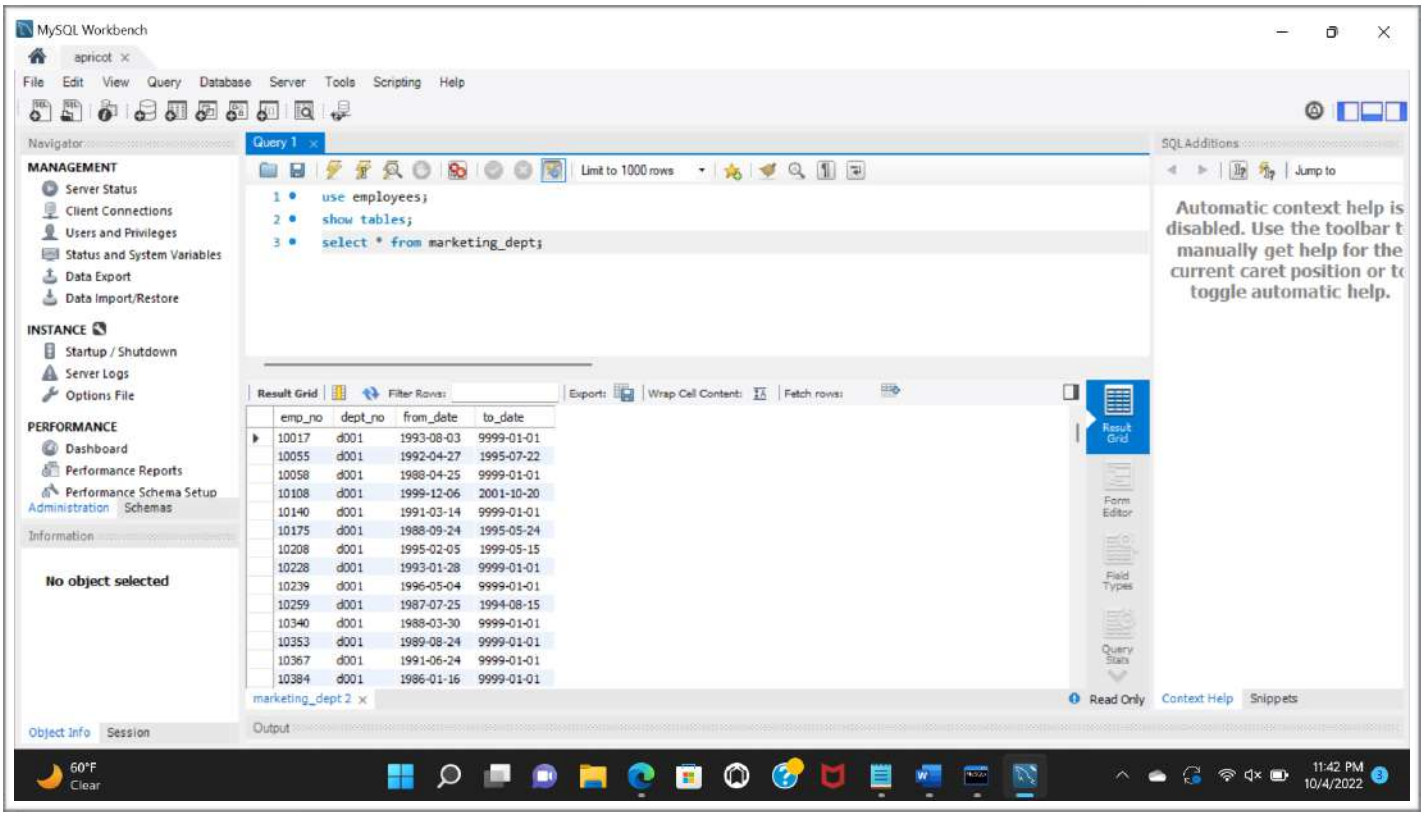
v. The following screenshots cover two non-traditional insert statements –

Insert specifying attribute names – This is used to insert data in specific attributes while leaving other attributes Null or default

```
mysql> insert into courses(course_name, duration_in_months) values ('Data Analytics', '3');
Query OK, 1 row affected (0.00 sec)

mysql> insert into courses(course_name, duration_in_months) values ('Machine Learning', '12');
Query OK, 1 row affected (0.00 sec)

mysql> insert into courses(course_name, duration_in_months) values ('Automata Theory', '6');
Query OK, 1 row affected (0.00 sec)

mysql> select * from courses;
+-----------+-----------------+--------------------+
| course_id | course_name     | duration_in_months |
+-----------+-----------------+--------------------+
|         1 | Data Analytics  |                  3 |
|         2 | Machine Learning|                 12 |
|         3 | Automata Theory |                  6 |
+-----------+-----------------+--------------------+
3 rows in set (0.00 sec)
```

Inserting from another table – Conditions specified in the *where* clause are used to import data from another table

```
mysql> create table marketing_dept (emp_no int not null, dept_no char(4) not null, from_date date not null, to_date date not null);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into marketing_dept select * from dept_emp where dept_no = "d001";
Query OK, 20211 rows affected (0.35 sec)
Records: 20211  Duplicates: 0  Warnings: 0

mysql> describe marketing_dept;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| emp_no    | int     | NO   |     | NULL    |       |
| dept_no   | char(4) | NO   |     | NULL    |       |
| from_date | date    | NO   |     | NULL    |       |
| to_date   | date    | NO   |     | NULL    |       |
+-----------+---------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```



vi. **Select.. From.. Where.. Group by.. Having.. Order by** – The following query implements all the mentioned clauses including an aggregate function *AVG* to return the average salary of department managers in each department.

```
mysql> select dept_no, avg(salary) from dept_manager, salaries where dept_no in ('d001', 'd002', 'd003', 'd004', 'd005') group by dept_no having avg(salary)>50000 order by
dept_no;
+---------+-------------+
| dept_no | avg(salary) |
+---------+-------------+
| d001    |  63810.7092 |
| d002    |  63810.7092 |
| d003    |  63810.7092 |
| d004    |  63810.7092 |
| d005    |  63810.7092 |
+---------+-------------+
5 rows in set (40.84 sec)
```

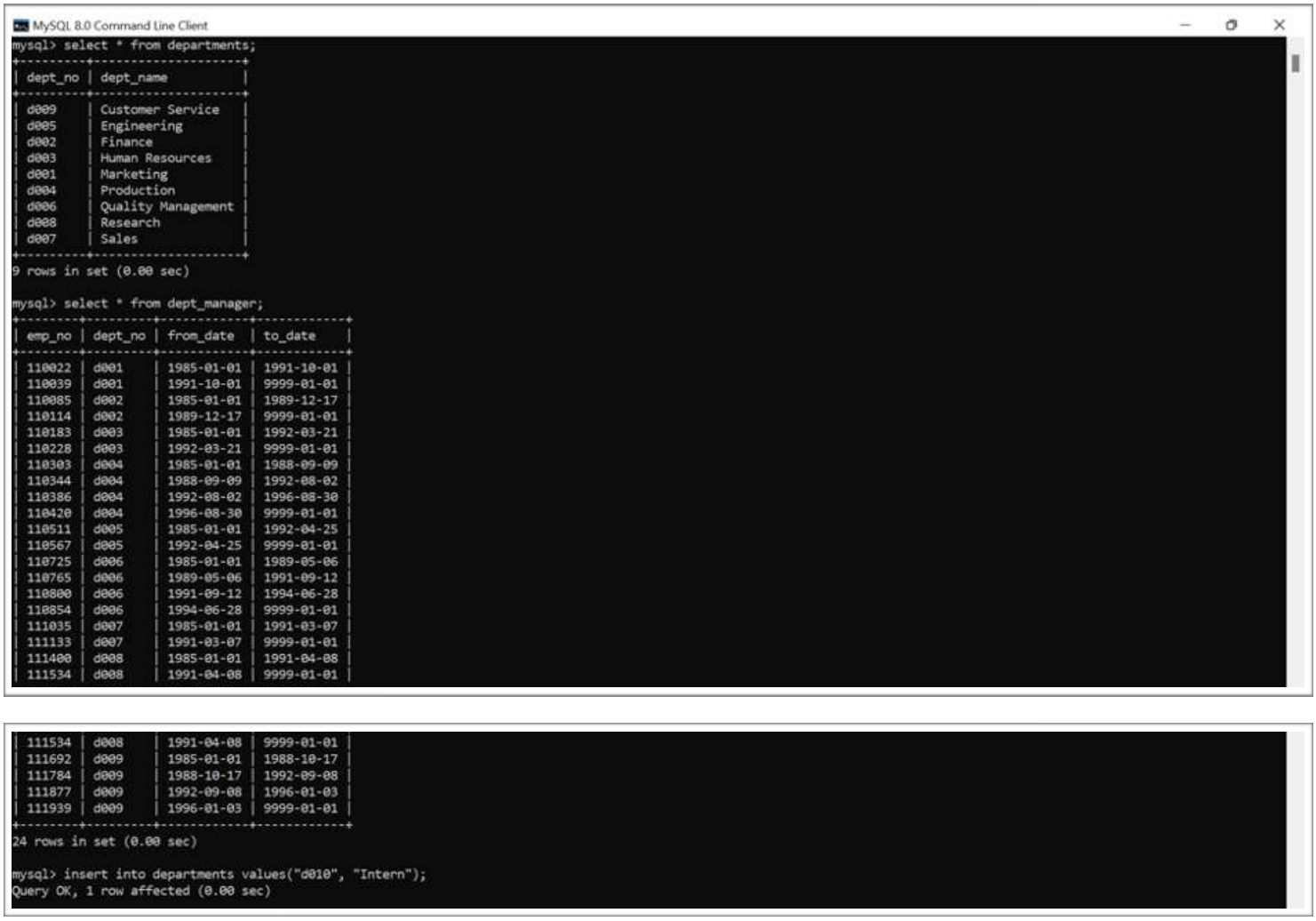
vii. **Like and Boolean operators** – Using the mentioned constructs to identify courses named starting Data or ending Theory with duration equal to or greater than 6 months.

```
mysql> select * from courses;
+-----------+-----------------+--------------------+
| course_id | course_name     | duration_in_months |
+-----------+-----------------+--------------------+
|         1 | Data Analytics  |                  3 |
|         2 | Machine Learning|                 12 |
|         3 | Automata Theory |                  6 |
|         4 | Data Science    |                  6 |
+-----------+-----------------+--------------------+
4 rows in set (0.00 sec)

mysql> select course_name, duration_in_months from courses where (course_name like 'Data%' or course_name like '%Theory') and duration_in_months>=6;
+-----------------+--------------------+
| course_name     | duration_in_months |
+-----------------+--------------------+
| Automata Theory |                  6 |
| Data Science    |                  6 |
+-----------------+--------------------+
2 rows in set (0.00 sec)
```

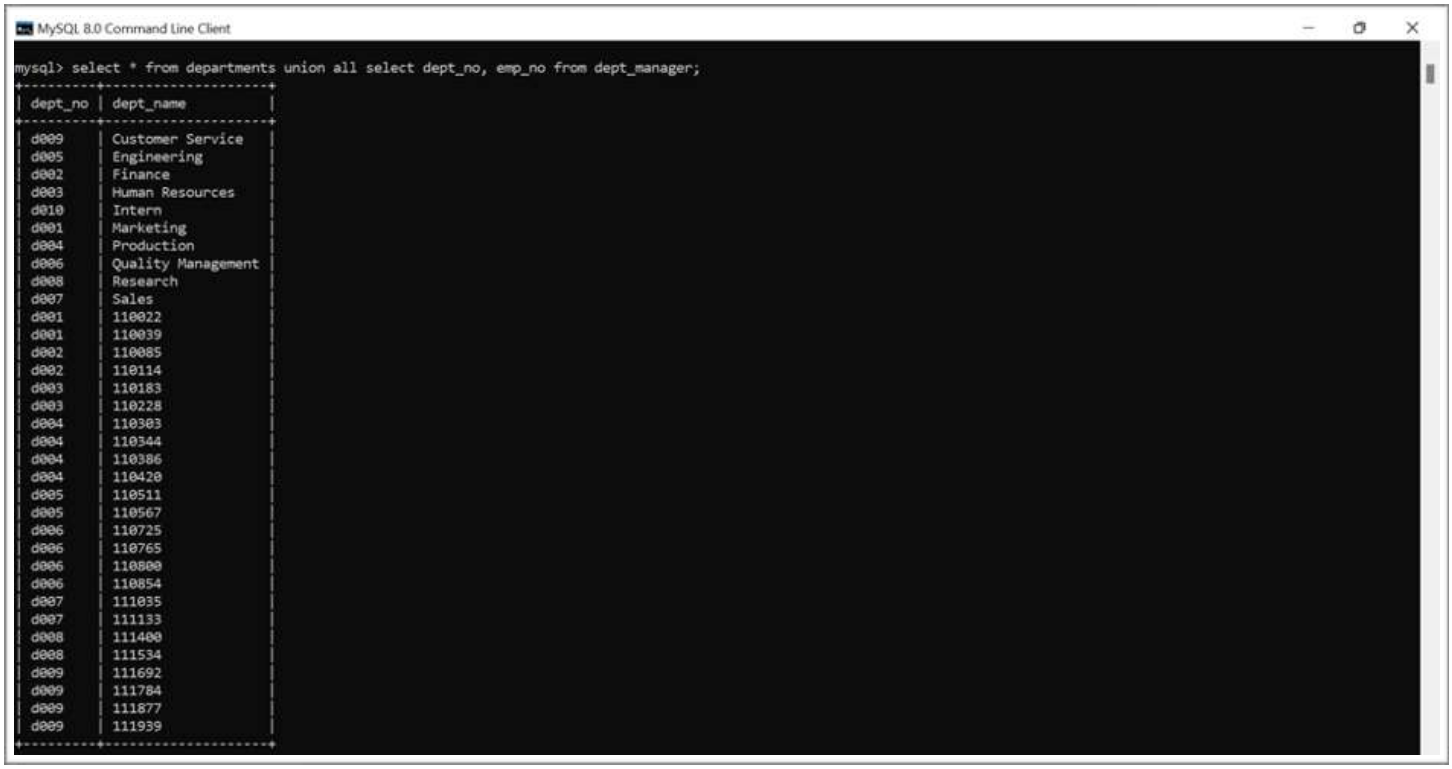
JOIN operations

The following screenshots cover three types of joins on two tables; *departments* and *dept_manager* of the employees database.

To differentiate more clearly, we added a new department “Intern” with dept_no “d010” to the table departments.

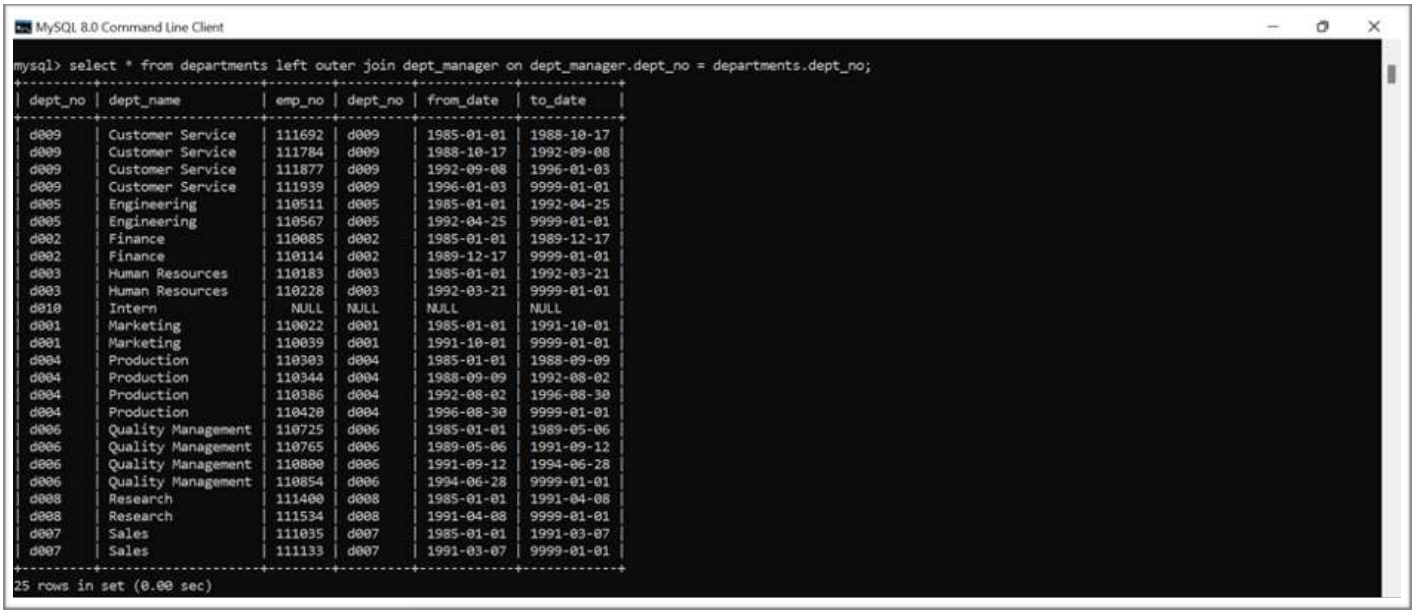


viii. **UNION ALL** – The union all operation works only when same number of columns are specified for all participating tables. In this case, union works on attributes *dept_no, dept_name* from table *departments* and *dept_no, emp_no* from table *dept_manager* with *dept_no* being the common attribute. As the table *departments* is mentioned first in the query, we notice that attribute name for second column is *dept_name* even though *emp_no* values exist in the column. Union all operation in this case does not provide valuable insights.

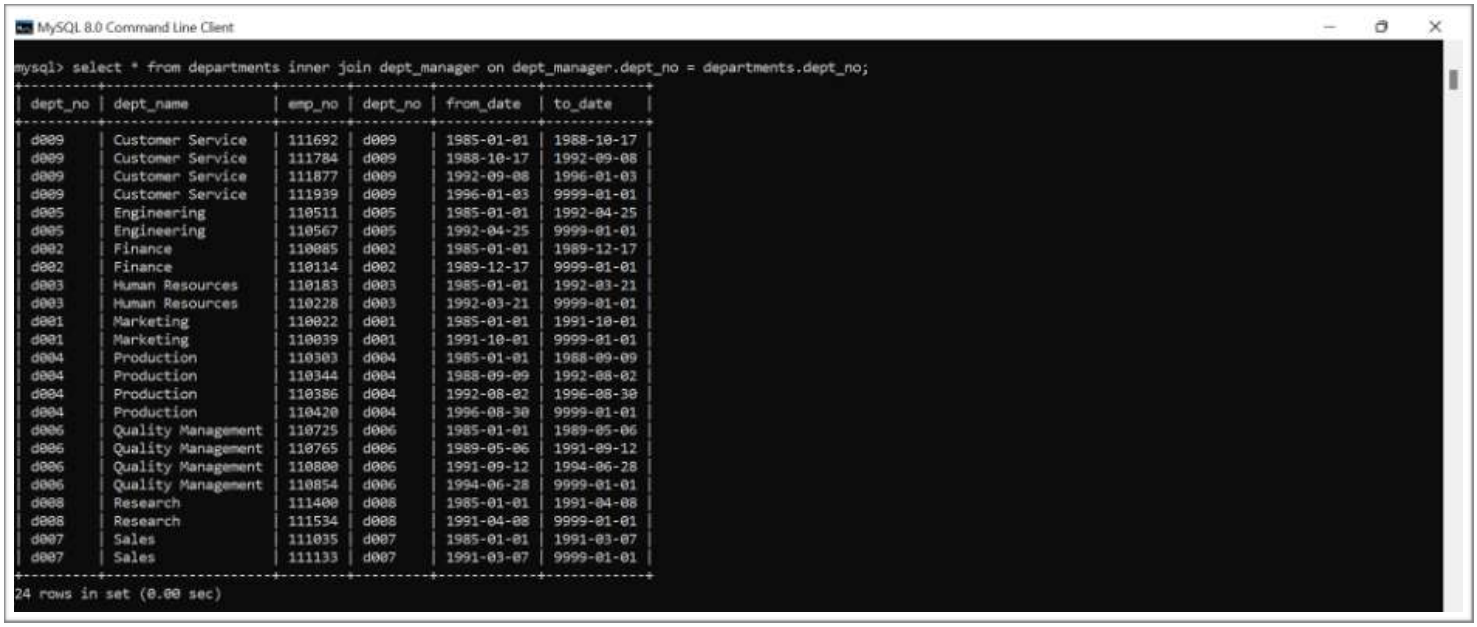


ix. **LEFT OUTER JOIN** – Left join operation displays all the data from table mentioned first (left side of the clause) and intersecting data from the second table.

In the screenshot below, we notice Null values corresponding to the row *dept_no* = “d010”. This is because even though the values exist in the left table *departments*, there are no corresponding values in the table *dept_manager*.



x. **INNER JOIN** – Inner join displays intersecting data from participating tables. We notice the row corresponding to dept_no = “d010” missing from the below screenshot when compared to values from left outer join.



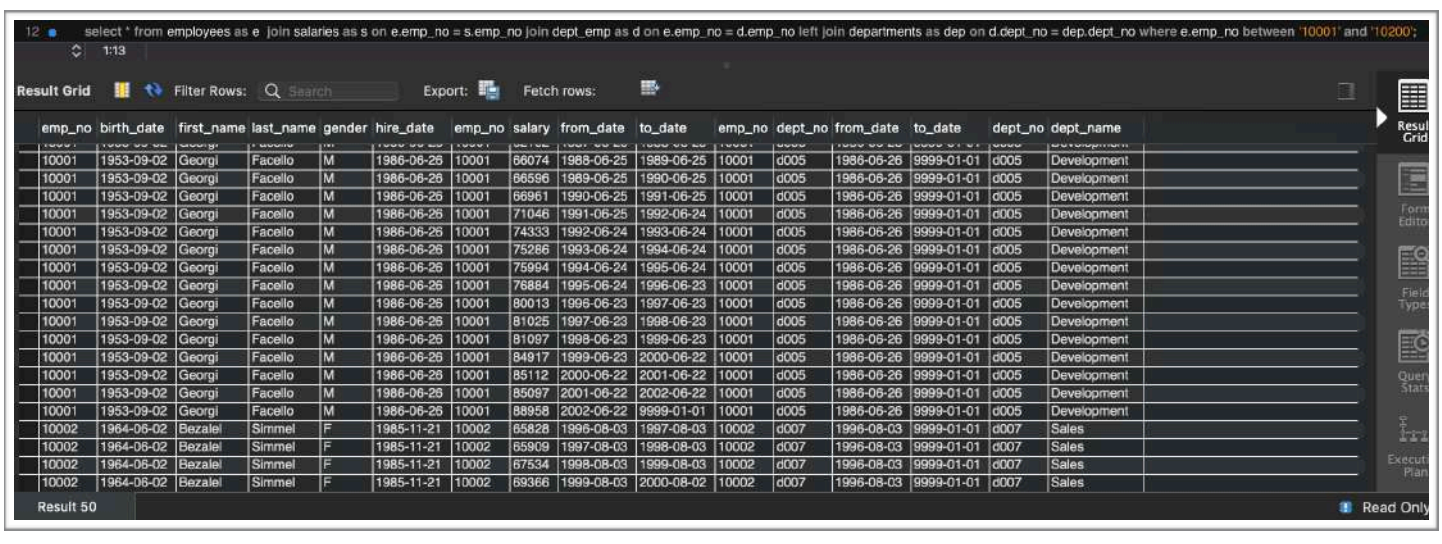
```
MySQL 8.0 Command Line Client

mysql> select * from departments inner join dept_manager on dept_manager.dept_no = departments.dept_no;
+---------+--------------------+--------+---------+------------+------------+
| dept_no | dept_name          | emp_no | dept_no | from_date  | to_date    |
+---------+--------------------+--------+---------+------------+------------+
| d009    | Customer Service   | 111692 | d009    | 1985-01-01 | 1988-10-17 |
| d009    | Customer Service   | 111784 | d009    | 1988-10-17 | 1992-09-08 |
| d009    | Customer Service   | 111877 | d009    | 1992-09-08 | 1996-01-03 |
| d009    | Customer Service   | 111939 | d009    | 1996-01-03 | 9999-01-01 |
| d005    | Engineering        | 110511 | d005    | 1985-01-01 | 1992-04-25 |
| d005    | Engineering        | 110567 | d005    | 1992-04-25 | 9999-01-01 |
| d002    | Finance            | 110085 | d002    | 1985-01-01 | 1989-12-17 |
| d002    | Finance            | 110114 | d002    | 1989-12-17 | 9999-01-01 |
| d003    | Human Resources    | 110183 | d003    | 1985-01-01 | 1992-03-21 |
| d003    | Human Resources    | 110228 | d003    | 1992-03-21 | 9999-01-01 |
| d001    | Marketing          | 110022 | d001    | 1985-01-01 | 1991-10-01 |
| d001    | Marketing          | 110039 | d001    | 1991-10-01 | 9999-01-01 |
| d004    | Production         | 110303 | d004    | 1985-01-01 | 1988-09-09 |
| d004    | Production         | 110344 | d004    | 1988-09-09 | 1992-08-02 |
| d004    | Production         | 110386 | d004    | 1992-08-02 | 1996-08-30 |
| d004    | Production         | 110420 | d004    | 1996-08-30 | 9999-01-01 |
| d006    | Quality Management | 110725 | d006    | 1985-01-01 | 1989-05-06 |
| d006    | Quality Management | 110765 | d006    | 1989-05-06 | 1991-09-12 |
| d006    | Quality Management | 110800 | d006    | 1991-09-12 | 1994-06-28 |
| d006    | Quality Management | 110854 | d006    | 1994-06-28 | 9999-01-01 |
| d008    | Research           | 111400 | d008    | 1985-01-01 | 1991-04-08 |
| d008    | Research           | 111534 | d008    | 1991-04-08 | 9999-01-01 |
| d007    | Sales              | 111035 | d007    | 1985-01-01 | 1991-03-07 |
| d007    | Sales              | 111133 | d007    | 1991-03-07 | 9999-01-01 |
+---------+--------------------+--------+---------+------------+------------+
24 rows in set (0.00 sec)
```

Q3.b.

- Hospital and Employees database are both in 3rd Normal form.
- We can see that both hospital and Employees table are relations, There is no multivalues attributes and also every values is atomic in both the database, so it qualifies for the 1 Normal form.
- In both the database, every non-key attributes is fully functionally dependent on the primary key also there are no partial functional dependency so we can say that both the database are in 2 Normal form.
- We can say that both the databases are in 3 Normal form because in addition to the two points written above, there are no transitive dependencies between the non-key attributes and the key attributes.

Q4.a.

- We have taken taken employee database for this question.
- We have Join three tables i.e. employees table, salaries table and department table.
- Since the data is very large, we took top 200 employees starting from 10001 - 10200.
- Below is the Screenshot of the query that we have have run on my sql workbench.-



```
12  select * from employees as e  join salaries as s on e.emp_no = s.emp_no join dept_emp as d on e.emp_no = d.emp_no left join departments as dep on d.dept_no = dep.dept_no where e.emp_no between '10001' and '10200';
    1:13

Result Grid   Filter Rows: Search   Export:   Fetch rows:
```

emp_no	birth_date	first_name	last_name	gender	hire_date	emp_no	salary	from_date	to_date	emp_no	dept_no	from_date	to_date	dept_no	dept_name
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	66074	1988-06-25	1989-06-25	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	66596	1989-06-25	1990-06-25	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	66961	1990-06-25	1991-06-25	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	71046	1991-06-25	1992-06-24	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	74333	1992-06-24	1993-06-24	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	75286	1993-06-24	1994-06-24	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	75994	1994-06-24	1995-06-24	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	76884	1995-06-24	1996-06-23	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	80013	1996-06-23	1997-06-23	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	81025	1997-06-23	1998-06-23	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	81097	1998-06-23	1999-06-23	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	84917	1999-06-23	2000-06-22	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	85112	2000-06-22	2001-06-22	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	85097	2001-06-22	2002-06-22	10001	d005	1986-06-26	9999-01-01	d005	Development
10001	1953-09-02	Georgi	Facello	M	1986-06-26	10001	88958	2002-06-22	9999-01-01	10001	d005	1986-06-26	9999-01-01	d005	Development
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	10002	65828	1996-08-03	1997-08-03	10002	d007	1996-08-03	9999-01-01	d007	Sales
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	10002	65909	1997-08-03	1998-08-03	10002	d007	1996-08-03	9999-01-01	d007	Sales
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	10002	67534	1998-08-03	1999-08-03	10002	d007	1996-08-03	9999-01-01	d007	Sales
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	10002	69366	1999-08-03	2000-08-02	10002	d007	1996-08-03	9999-01-01	d007	Sales

```
Result 50                                                    Read Only
```

We have attach Q4(a).csv file where we have analyse some data using pivot tables.
There are multiple picot tables which gives meaningful data.
In Pivot 1, we have employees name and their salaries department wise.
In Pivot 2, we have same employees and their salaries department wise but we have added Gender attribute as a filter.
In Pivot 3, we have shown the salaries of each department in which we can see which department has the highest salaries.
In Pivot 4, we have made a derived attribute 'Age' from the 'date of birth' of the employees using the formula 'today() - (date_of_birth)/365'. In Pivot 4, we have shown age wise salary of the employees.

Q4.b.

1. **Joining department and dept_emp to get dept_name.**

SQL Query - select d.dept_no,e.emp_no,d.dept_name from departments as d join dept_emp as e on e.dept_no=d.dept_no where (e.emp_no between '10001' and '10200') or (e.emp_no between '110000' and '120000');

VLookup formula- =VLOOKUP(A2,departments!$A$1:$B$10,2,0)

2. **Joining department and dept_manager to get dept_name of managers.**

SQL Query- select d.dept_no,e.emp_no,d.dept_name from departments as d join dept_manager as e on e.dept_no=d.dept_no where (e.emp_no between '10001' and '10200') or (e.emp_no between '110000' and '120000');

VLookup formula - =VLOOKUP(A2,departments!$A$1:$B$10,2,0)

3. **Joining dept_emp and employees to get genders of employees.**

SQL Query-select d.dept_no,d.emp_no,e.gender from dept_emp as d join employees as e on d.emp_no=e.emp_no where (e.emp_no between '10001' and '10200') or (e.emp_no between '110000' and '120000');

VLookup formula- =VLOOKUP(A2,employees!$A$1:$F$226,5,0)

4. **Joining employees and salaries to get first name and last names of employees.**

SQL Query- select s.emp_no,s.salary,e.first_name,e.last_name from employees as e join salaries as s on e.emp_no=s.emp_no where (e.emp_no between '10001' and '10200') or (e.emp_no between '110000' and '120000');

VLookup formula - =VLOOKUP(A2,employees!$A$1:$F$226,3,0)
=VLOOKUP(A2,employees!$A$1:$F$226,4,0)

5. **Joining employees and titles to get title of the employees.**

SQL Query- select e.emp_no,e.first_name,e.last_name,t.title from employees as e left join titles as t on e.emp_no=t.emp_no where (e.emp_no between '10001' and '10200') or (e.emp_no between '110000' and '120000');

VLookup formula - =VLOOKUP(A2,Titles!$A$1:$D$353,2,0)

Q5.a.

We have interviewed Mr. Sohail Khan. He is a Senior Data Engineer at Persistent Systems Limited in Pune,India. Please find below the interview we conducted.

1. **Please introduce yourself.**

   "I am Sohail Khan. I am a Senior Data Engineer at Persistent Systems Limited, Pune. I have over 4 years of experience in designing and implementing ETL pipelines. I have worked across various domains of business such as payments, finance and healthcare and helped the business design their data warehouse and migrate them to Google Cloud Platform"

2. **What is/was your roles and responsibilities in your company?**

   "Data Warehouses are developed for different Analytical purposes, hence we always develop the data model around that analytical use case. Also, data in the data warehouse come from various source and various formats and also is usually of a large size. Hence there is significant compute performance required to gather the data and perform 'transformation' on it before loading in to the data warehouse. I am also responsible in making key decisions on designing the right architecture and selecting the right technical stack for the development of the Data Pipelines."

3. **What tools you use for managing retrieving, cleaning and analysis of the data?**

   "There are mainly two ways you can retreive data, in batch and in stream. The batches are generally in the form of files and are transmitted over SFTP to your server or bucket while in case of stream, it can be an API, a Message Queue like Kafka or even sensor data transmitted over the internet. Now before you can put this data into the warehouse, you clean it based on the business's analytical use case. In a batch load you have a finite set of records that you receive in a file whereas in a stream load you do not. Based on this, you either develop a batch pipeline or a stream pipeline. Apache Spark, Apache Spark Streaming, Scala, PySpark and Apache Beam are some of the most common Data Pipelining SDKs available for development of the ETL pipelines. If we talk about tools, Abinitio, Talend, Informatica and DMExpress are some tools that help you develop ETL pipelines. To make these processes autonomous, Unix scripting or Apache Airfloow is used to orchestrate various process and also follow a run schedule. Finally once you have into the data warehouse which is typically an OLAP like AWS Redshift, Google Bigquery or Netezza etc, analytical tools like Looker, Tableau and PowerBI are used for reporting."

4. **What are the most common Data Warehousing use cases you see?**

   "Generally, data warehouses are used for analytical purposes, such as trend analysis to help businesses make informed decisions and forensic analysis, such as detecting fraud.It is also used to explore new

business opportunities to study the customer's behavior and profile the right person to advertise or send promotions or discounts. In more recent times, Data Warehouses are now becoming the source for Machine Learning use cases and AI development as well."

5. **At what level of normal form you normalise data in your company?**

   "My company generally denormalizes warehouse databases with some tables normalized to 1NF."

6. **What are some of the key challenges that you see come up during the datawarehouse development process?**

   "ETL pipelines and warehouses can also be built using simple python scripts and Mysql databases but it would take forever to traverse over a million records with it. Hence Big Data Technologies like HDFS, Spark are used. Cloud has become the go-to- place for development of ETL pipelines for their ability to autoscale and handle spikes in the data loads."

7. **Have you ever used surrogate keys?**

   "Yes, we have to use surrogate keys in data warehouse. Also, data in the warehouse can also be duplicate at times and this could break the pipeline at times. As we do not want to lose on the data load tables are de-duplicated as and when required. For this, when data is loaded in to the data warehouse, Unique keys are generated by the system that are attached to the record while they are populated into the database."

8. **What challenge do you face while adding new data and how do you manage a growing database?**

   "This schema evolution problem is pretty common as with time your source systems may change and consequently their schema may change. It becomes a major problems when a new record is added. Your target DB has to be redefined to be able to accept that column. This use case is becoming a feature in various OLAP engines now. For management of large datawarehouses we generally use Google Bigquery that stores data on to it's own storage known as Colossus. However, generally a cluster is used that can autoscale as the data grows"

9. **What challenges you face when you have to merge 2 or more big database to an existing one?**

   "The major challenge is merging of the different fact tables in the schema. The joins become complex and also some tables need changes in schema or column definitions. Sometimes data types are also not same and need to be changed accordingly. If you are merging from two heterogenous platforms, like Oracle to Bigquery, then datetime formats are also to be taken care of."

10. **Any recommendation or advise for someone who want to work in the data space?**

    "Yes, I would happily recommend to work in data domain, if you have knowledge and skills, and if you like analysis for particular domain "

Below are the take aways from the interview.

1. There are mainly 2 ways in which a person can retrieve data into a data warehouse i.e. in Batches or Streams. In batches data is generally transmitted in the form of files to your server, whereas in

stream, the data might be transmitted in the form of an API, a message queue (like Kafka) or even a sensor data transmitted online.

2. A variety of different tools and software are used to retrieve data into a data warehouse.
   To create ETL pipelines software like Apache Spark, Apache Spark Streaming, Scala, PySpark and Apache Beam or tools like Abinitio, Talend, Informatica and DMExpress, are some popular SDKs used. To make these process autonomouse Unix scripting or Apache Airfloow is used.
   Finally after we import data into the data warehouse which is usually an OLAP like AWS Redshift, Google Bigquery or Netezza etc, analytical tools like Looker, Tableau and PowerBI are used for reporting.

3. Some of the most common use cases of Data Warehouses are trend analysis which help the business in making informed decisions or forensic analysis which help detect fraudulent transactions. Also data is used to study customer's behaviour and pattern to advertise and send promotions or discounts. Data warehouses are also being used to develop ML models and AI.

4. One of the main challenges that people face during development of a data warehouse is performance. Platform, technical stack and amount of data must be properly chosen. ETL pipelines and warehouses can be developed using MySQL databases but they will take ages to process millions of records. Hence, proper Big Data Tools like HDFS are used. Cloud has also become a very popular choice due to its ability to auto scale and handle huge spikes in data.

5. One of the most common problem with adding new data to the data warehouse is that of schema evolution. When adding new data, your target database doesn't always allow addition of a new column. This use case is becoming a function of many OLAP engines now.
   Another way of getting around this is using clusters which can auto scale as the data grows.

6. Surrogate keys are very commons in data warehouses as many OLAP engines do not the function of primary keys. Also many times duplicate data is present in the warehouse which can break pipelines. To overcome this, when data is loaded into the warehouse, unique keys are automatically generated, attached to the record and populated into the database.

Q5.b.

1. According to the first wikipedia link on Database Normalization, we get to know the emergence of data normalization which was done by Edgar F. Codd in 1970 when he was working on relational model during his tenure at IBM. The main objective of normalization of the database was to query and manipulate the data easily using SQL.

2. Further he states that normalization will also help to reduce the chances of redesigning the structure of database when there is a need to extend it. When normalization is achieved to its fullest it will help in the extension of the database structure without affecting the existing structure.

3. Apart from the well known normal forms i.e 1NF,2NF,3NF,BCNF,5NF and 6NF, we get to learn about one more form in this article which comes between 3NF and BCNF called “EKNF - Elementary Key Normal Form”. EKNF has the qualities and features of 3NF and BCNF both and is also free from the problems which usually arises in them.

4. In addition to this there is one more form which comes between 4NF and 5NF called “ETNF - Essential Tuple Normal Form”. ETNF is believed to be not as strong as 5NF but it is also effective in the elimination of redundant tuples as the 5NF.

5. In the Technicspub link Steve Hoberman has stated many data modeling challenges that he came across, one such interesting challenge mentioned in the link is whether an attribute which is an indicator can be NULL i.e can it have an empty value. He mentions about the two possible values of an indicator which is usually Active/Inactive, On/Off, True/False or Yes/No, the question here is whether there can be a third value which will be an empty value (NULL) or can the attribute be considered as optional. If the attribute field is required then the NULL value can be replaced by a value that is default for example “Active”.

6. Another such challenge he mentioned is whether a foreign key should be included on a logical data model. A foreign key in the logical data model helps a user to identify the relationship between two entities which can be informative and will be helpful in knowing the design better and in terms of readability. But some argue that the foreign key should only be present in the physical model as they are constructs of relational database.

7. In one of the books on data modeling called ”Data Modeling Made Simple ” by Steve Hoberman, he mentions the importance of data modeling through the technique of way finding and how it is important and easier to fully understand the database and its structure by this technique. He explains data modeling as a tool which is a combination of texts and symbols which gives us valuable information from the real world’s complex information.

8. The ansi.org website mentions about the updated SQL 2016 standard which has added support for analytical functionalities which includes logarithmic and trigonometric functions. It has also made enhancements on the stored modules and object bindings.

9. The wiscorp site of SQL standards talks about the SQL 2003 standard which is the replacement of the SQL 1999 standard. The new features includes the introduction of new data types such as BIGINT and MULTISET, and also the enhancement of the Unbounded Array data type.It also introduces new scalar, aggregate and hypothetical functions such as LN, EXP, POWER (scalar), STDDEV_POP, STDDEV_POP (aggregate) and RANK within group and DENSE_RANK within group (hypothetical).

10. Among the various mentioned SQL tutorial websites the well renowned khan academy website provides a brief explanation of basic sql commands, operators and relational queries. Apart from that it also helps us to learn and write complex and advanced sql queries using the operators in the form of challenges, such challenges include the song selector in karaoke and playlist maker etc.