*Practical 1: Defining schema for applications

Built in Schemas in SQL: (dbo, guest, sys, INFORMATION_SCHEMA)

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.05 sec)
```

Create a schema in SQL:

>>> CREATE SCHEMA College;

```
mysql> CREATE SCHEMA College;
Query OK, 1 row affected (0.04 sec)
```

Show schemas in SQL:

>>> SHOW SCHEMAS;

```
mysql> show schemas;
+--------------------+
| Database           |
+--------------------+
| college            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

Drop a schema in SQL: (Used to delete schema and its related objects)

>>> DROP SCHEMA College;

```
mysql> DROP SCHEMA college;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> Show Schemas;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

***Practical 2:** Creating tables, Renaming tables, Data constraints (Primary key, Foreign key, Not Null), Data insertion into a table.

Creation of a database in SQL:

>>> CREATE DATABASE College;

```
mysql> CREATE DATABASE College;
Query OK, 1 row affected (0.06 sec)

mysql>
```

Show list of databses in SQL:

>>> SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| college            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.04 sec)

mysql>
```

How to use a database you have created in SQL:

>>> use College;

```
mysql> use college;
Database changed
```

How to create a table in SQL :

>>> CREATE TABLE Student(ID int(10), name varchar(50), Branch varchar(30), city varchar(50));

```
mysql> CREATE TABLE Student( ID  int(10), name varchar(50), Branch varchar(30), city varchar(50));
Query OK, 0 rows affected, 1 warning (0.10 sec)
```

desc command in SQL: (to describe or show structure of table)

>>> desc student; *or_* describe student;

```
mysql> desc student;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| ID     | int         | YES  |     | NULL    |       |
| name   | varchar(50) | YES  |     | NULL    |       |
| Branch | varchar(30) | YES  |     | NULL    |       |
| city   | varchar(50) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

Renaming tables in SQL: (Using RENAME TABLE Statement)

>>> RENAME TABLE student TO student_details;

```
mysql> RENAME TABLE student TO student_details;
Query OK, 0 rows affected (0.06 sec)
```

- The above query will rename/change the name of the student table into student_details

Renaming tables in SQL: (Using ALTER TABLE Statement)

>>> ALTER TABLE student_details RENAME student;

```
mysql> ALTER TABLE student_details RENAME student;
Query OK, 0 rows affected (0.05 sec)
```

- The above query will rename/change the name of the student_details table into student

How to drop a table you have created in SQL:

>>> DROP TABLE student;

```
mysql> drop table student;
Query OK, 0 rows affected (0.06 sec)
```

- The above query will remove/drop the table which you have created

Data Constraints in SQL: (Primary key, Foreign key, Not Null)

- **NOT NULL:**

>>> CREATE TABLE info(id int(10) NOT NULL, name varchar(30) NOT NULL, city varchar(30));

```
mysql> CREATE TABLE info( id  int(10) NOT NULL, name  varchar(30) NOT NULL, city  varchar(30) );
Query OK, 0 rows affected, 1 warning (0.07 sec)

mysql> desc info;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | NO   |     | NULL    |       |
| name  | varchar(30) | NO   |     | NULL    |       |
| city  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- The query will specify NOT NULL to field id and name as shown above

- **PRIMARY KEY:**

>>> CREATE TABLE Personal (id int(10) NOT NULL UNIQUE, name varchar(30) , city varchar(30), PRIMARY KEY(id));

```
mysql> CREATE TABLE Personal ( id  int(10) NOT NULL UNIQUE, name  varchar(30) , city  varchar(30), PRIMARY KEY(id) );
Query OK, 0 rows affected, 1 warning (0.07 sec)

mysql> desc personal;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
| city  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

- **FOREIGN KEY:**

Consider we have two tables, table 1 and table 2,
Field **id** in table 1 is a **primary key** and the **foreign key** field **city** from table 1 is linked with primary key of table 2 i.e. with field **cid** as shown below

Table 1 : **Student**

id	name	City_id
1	Rahul	1
2	Aman	2
3	Shradhha	1

PRIMARY KEY

FOREIGN KEY

Table 2 : **City**

cid	city
1	Nanded
2	Latur

PRIMARY KEY

Syntax:

>>> CREATE TABLE student (id int(10) NOT NULL, name varchar(30) , city_id int(10) not null, PRIMARY KEY(id),FOREIGN KEY(city_id) REFERENCES city(cid));

```
mysql> CREATE TABLE student ( id  int(10) NOT NULL, name  varchar(30) , city_id int(10) not null, PRIMARY KEY(id),FOREIGN KEY(city_id) REFERENCES city(cid) );
Query OK, 0 rows affected, 2 warnings (0.09 sec)

mysql> desc student;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| id      | int         | NO   | PRI | NULL    |       |
| name    | varchar(30) | YES  |     | NULL    |       |
| city_id | int         | NO   | MUL | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Data insertion in SQL: (INSERT INTO Statement)

- **First create a table using create table staement:**

>>> CREATE TABLE Customer(cust_id int(10), cust_name varchar(50), address varchar(50), age int(10));

```
mysql> CREATE TABLE Customer(cust_id int(10), cust_name varchar(50), address varchar(50), age int(10));
Query OK, 0 rows affected, 2 warnings (0.11 sec)
```

1) **Inserting data values using INSERT INTO statement:**

>>> INSERT INTO Customer values(101,'John wick', 'new york', 30);

```
mysql> INSERT INTO Customer values(101,'John wick', 'new york', 30);
Query OK, 1 row affected (0.04 sec)
```

- similarly we can insert multiple values one by one

>>> INSERT INTO Customer values(102,'Bruce Banner', 'Downtown', 34);
>>> INSERT INTO Customer values(103,'Teth Adam', 'Georgia', 50);
>>> INSERT INTO Customer values(104,'Tony Stark', 'new york', 40);

```
mysql> INSERT INTO Customer values(102,'Bruce Banner', 'Downtown', 34);
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO Customer values(103,'Teth Adam', 'Georgia', 50);
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Customer values(104,'Tony Stark', 'new york', 40);
Query OK, 1 row affected (0.04 sec)
```

- **To see the inserted records in a table or to access the records in a database we use SELECT statement**

>>> SELECT * FROM Customer;

```
mysql> SELECT * FROM Customer;
+---------+--------------+----------+------+
| cust_id | cust_name    | address  | age  |
+---------+--------------+----------+------+
|     101 | John wick    | new york |   30 |
|     102 | Bruce Banner | Downtown |   34 |
|     103 | Teth Adam    | Georgia  |   50 |
|     104 | Tony Stark   | new york |   40 |
+---------+--------------+----------+------+
4 rows in set (0.00 sec)
```

2) **Inserting only in specified columns :**

>>> INSERT INTO Customer(cust_id, cust_name, address, age) values(105,'Diana Queen', 'Los Angeles', 25);

```
mysql>  INSERT INTO Customer( cust_id, cust_name, address, age) values(105,'Diana Queen', 'Los Angeles', 25);
Query OK, 1 row affected (0.05 sec)

mysql> SELECT * FROM Customer;
+---------+--------------+-------------+------+
| cust_id | cust_name    | address     | age  |
+---------+--------------+-------------+------+
|     101 | John wick    | new york    |   30 |
|     102 | Bruce Banner | Downtown    |   34 |
|     103 | Teth Adam    | Georgia     |   50 |
|     104 | Tony Stark   | new york    |   40 |
|     105 | Diana Queen  | Los Angeles |   25 |
+---------+--------------+-------------+------+
5 rows in set (0.00 sec)
```

- similarly we can insert multiple values in a specific column one by one

>>> INSERT INTO Customer(cust_id, cust_name, address) values(105,'Elona holms', 'Downtown');
>>> INSERT INTO Customer(cust_id, cust_name) values(107,'Peter Parker');
>>> INSERT INTO Customer(cust_id, cust_name, age) values(108,'Jack rose', 23);

```
mysql>  INSERT INTO Customer( cust_id, cust_name, address) values(105,'Elona holms', 'Downtown');
Query OK, 1 row affected (0.01 sec)

mysql>  INSERT INTO Customer( cust_id, cust_name) values(107,'Peter Parker');
Query OK, 1 row affected (0.03 sec)

mysql> INSERT INTO Customer( cust_id, cust_name, age) values(108,'Jack rose', 23);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Customer;
+---------+--------------+-------------+------+
| cust_id | cust_name    | address     | age  |
+---------+--------------+-------------+------+
|     101 | John wick    | new york    |   30 |
|     102 | Bruce Banner | Downtown    |   34 |
|     103 | Teth Adam    | Georgia     |   50 |
|     104 | Tony Stark   | new york    |   40 |
|     105 | Diana Queen  | Los Angeles |   25 |
|     105 | Elona holms  | Downtown    | NULL |
|     107 | Peter Parker | NULL        | NULL |
|     108 | Jack rose    | NULL        |   23 |
+---------+--------------+-------------+------+
8 rows in set (0.00 sec)
```

- **How to select a specific field or a column whose values you want to fetch :**

Syntax: >>> SELECT column1, coumn2….. FROM table_name;

e.g.

>>> select cust_id,cust_name from customer;

```
mysql> select cust_id,cust_name from customer;
+---------+--------------+
| cust_id | cust_name    |
+---------+--------------+
|     101 | John wick    |
|     102 | Bruce Banner |
|     103 | Teth Adam    |
|     104 | Tony Stark   |
|     105 | Diana Queen  |
|     105 | Elona holms  |
|     107 | Peter Parker |
|     108 | Jack rose    |
+---------+--------------+
8 rows in set (0.00 sec)
```

***Practical 3:** Grouping data, Aggregate functions, Oracle functions (Mathematical, Character functions).

Grouping Data:

i) ORDER BY: It is used to sort the records in ascending or descending order.

Consider following student table,

```
mysql> SELECT * FROM student;
+--------+-------+--------+-----------+
| Rollno | Name  | Branch | city      |
+--------+-------+--------+-----------+
|      1 | Max   | CSE    | Berlin    |
|      2 | Ana   | ME     | London    |
|      3 | John  | AI     | Ankara    |
|      4 | Amber | CE     | Boston    |
|      5 | Brad  | CSE    | Manhatton |
+--------+-------+--------+-----------+
5 rows in set (0.00 sec)
```

>>> SELECT *FROM Student
ORDER BY name ASC; - - - ASC= Ascending

```
mysql> SELECT * FROM Student ORDER BY Name ASC;
+--------+-------+--------+-----------+
| Rollno | Name  | Branch | city      |
+--------+-------+--------+-----------+
|      4 | Amber | CE     | Boston    |
|      2 | Ana   | ME     | London    |
|      5 | Brad  | CSE    | Manhatton |
|      3 | John  | AI     | Ankara    |
|      1 | Max   | CSE    | Berlin    |
+--------+-------+--------+-----------+
5 rows in set (0.03 sec)
```

- The above statement will sort the records of 'Name' in Ascending Order as shown in above

>>> SELECT *FROM Student
ORDER BY Rollno DESC; - - - DESC= Descending

```
mysql> SELECT * FROM Student ORDER BY Rollno DESC;
+--------+-------+--------+-----------+
| Rollno | Name  | Branch | city      |
+--------+-------+--------+-----------+
|      5 | Brad  | CSE    | Manhatton |
|      4 | Amber | CE     | Boston    |
|      3 | John  | AI     | Ankara    |
|      2 | Ana   | ME     | London    |
|      1 | Max   | CSE    | Berlin    |
+--------+-------+--------+-----------+
5 rows in set (0.00 sec)
```

- The above statement will sort the records of 'Rollno' in Descending Order as shown in above

```
>>> SELECT city  FROM Student
    ORDER BY city DESC;
```

```
mysql> SELECT city  FROM Student ORDER BY city DESC;
+-----------+
| city      |
+-----------+
| Manhatton |
| London    |
| Boston    |
| Berlin    |
| Ankara    |
+-----------+
5 rows in set (0.00 sec)
```

ii) GROUP BY: It is used to group rows that have same values.
We often use Group by and Having Clause with
Aggregate functions (COUNT(), SUM(), AVG(), MAX(), MIN())

Consider following student_marks table,

```
mysql> SELECT * FROM Student_marks;
+------+--------+-------+---------+
| sid  | sname  | marks | city    |
+------+--------+-------+---------+
|    1 | Amol   |    60 | Pune    |
|    2 | Rajesh |    70 | Nanded  |
|    3 | Mahi   |    90 | Pune    |
|    4 | Rani   |    55 | Nanded  |
|    5 | Rohan  |    84 | Chennai |
+------+--------+-------+---------+
5 rows in set (0.00 sec)
```

***Query:** Find total marks of each student from each city.

```
>>> SELECT SUM(Marks),city
    FROM student_marks GROUP BY City;
```

```
mysql> SELECT SUM(Marks),city from student_marks GROUP BY City;
+------------+---------+
| SUM(Marks) | city    |
+------------+---------+
|        150 | Pune    |
|        125 | Nanded  |
|         84 | Chennai |
+------------+---------+
3 rows in set (0.00 sec)
```

- The above statement will find total marks of each student from each city.

Iii) Having: It is used to group rows that have same values
Consider above student_marks table,

***Query:** Find total marks of each student from city Pune and Chennai.

```
>>> SELECT SUM(marks), city
    FROM Student_marks GROUP BY City
    HAVING City IN ('Pune','Chennai');
```

```
mysql> SELECT SUM(marks), city FROM Student_marks GROUP BY City HAVING City IN ('Pune','Chennai');
+------------+---------+
| SUM(marks) | city    |
+------------+---------+
|        150 | Pune    |
|         84 | Chennai |
+------------+---------+
2 rows in set (0.00 sec)
```

***Query:** Find the Count of students from city Pune.

```
>>> SELECT Count(sname),city
    FROM Student_marks GROUP BY City
    HAVING City IN ('Pune');
```

```
mysql> SELECT Count(sname), city FROM Student_marks GROUP BY City HAVING City IN ('Pune');
+--------------+------+
| Count(sname) | city |
+--------------+------+
|            2 | Pune |
+--------------+------+
1 row in set (0.00 sec)
```

Aggregate Functions:

- An Aggregate function allows you to perform calculations on a set of values to return a single value.
- Various Aggregate functions are:
  i) COUNT()
  ii) SUM()
  iii) AVG()
  iv) MAX()
  v) MIN().

Consider following test table

```
mysql> select* from test;
+------+-------+-------+----------+
| id   | Name  | value | city     |
+------+-------+-------+----------+
|   11 | John  | 100   | New YOrk |
|   22 | Robin | 200   | Brooklyn |
|   33 | Brad  | 300   | Georgia  |
| NULL | Animo | 400   | Bellwood |
+------+-------+-------+----------+
4 rows in set (0.00 sec)
```

- **COUNT() :** This function is used to count the number of rows in a given table

>>> SELECT COUNT(*) FROM Test;

```
mysql> SELECT COUNT(*) FROM Test;
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
1 row in set (0.06 sec)
```

>>> SELECT COUNT(ALL ,id) FROM Test;

```
+---------------+
| COUNT(ALL id) |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)
```

- **SUM() :** This function is used to get the sum of a numeric column or a specified column

>>> SELECT SUM(Value) FROM Test;

```
mysql> SELECT COUNT(ALL id) FROM Test;
+---------------+
| COUNT(ALL id) |
+---------------+
|             3 |
+---------------+
1 row in set (0.00 sec)
```

>>> SELECT SUM(Value) FROM Test WHERE Value> 200;

```
mysql> SELECT SUM(Value) FROM Test WHERE Value> 200;
+------------+
| SUM(Value) |
+------------+
|        700 |
+------------+
1 row in set (0.04 sec)
```

- **AVG() :** This function is used to get the average of records of a numeric column or a specified column.

>>> SELECT AVG(Value) FROM Test;

```
mysql> SELECT AVG(Value) FROM Test;
+------------+
| AVG(Value) |
+------------+
|        250 |
+------------+
1 row in set (0.00 sec)
```

>>> SELECT AVG(id) FROM Test;

```
mysql> SELECT AVG(id) FROM Test;
+---------+
| AVG(id) |
+---------+
| 22.0000 |
+---------+
1 row in set (0.00 sec)
```

- **MAX() :** This function is used to get the maximum record from a column.

>>> SELECT MAX(value) FROM Test;

```
mysql> SELECT MAX(value) FROM Test;
+------------+
| MAX(value) |
+------------+
| 400        |
+------------+
1 row in set (0.02 sec)
```

>>> SELECT MAX(id) FROM Test;

```
mysql> SELECT MAX(id) FROM Test;
+---------+
| MAX(id) |
+---------+
|      33 |
+---------+
1 row in set (0.00 sec)
```

- **MIN() :** This function is used to get the minimum record from a column.

>>> SELECT MIN(id) FROM Test;

```
mysql>  SELECT MIN(id) FROM Test;
+---------+
| MIN(id) |
+---------+
|      11 |
+---------+
1 row in set (0.00 sec)
```

>>> SELECT MIN(value) FROM Test;

```
mysql> SELECT MIN(value) FROM Test;
+------------+
| MIN(value) |
+------------+
| 100        |
+------------+
1 row in set (0.00 sec)
```

Oracle Functions: (Mathematical, Character Functions)

- An Oracle function is a subprogram that used to return a single value.
- i) Mathematical Functions.
  Ii) Character Functions.

i) Mathematical Functions: Numeric Functions are used to perform operations on numbers and return numbers. Numeric functions are sometimes called mathematical functions.

• **ABS():** It returns the absolute value of a number.

>>> SELECT ABS(2);

```
mysql> SELECT ABS(2);
+--------+
| ABS(2) |
+--------+
|      2 |
+--------+
1 row in set (0.06 sec)
```

>>> SELECT ABS(-6);

```
mysql> SELECT ABS(-6);
+---------+
| ABS(-6) |
+---------+
|       6 |
+---------+
1 row in set (0.00 sec)
```

• **ACOS():** It returns the arc cosine of a number.

>>> SELECT ACOS(0.11);

```
mysql>  SELECT ACOS(0.11);
+------------------+
| ACOS(0.11)       |
+------------------+
| 1.46057327680715 |
+------------------+
1 row in set (0.00 sec)
```

>>>SELECT ACOS(0.21);

```
mysql>  SELECT ACOS(.21);
+-------------------+
| ACOS(.21)         |
+-------------------+
| 1.359221367036801 |
+-------------------+
1 row in set (0.00 sec)
```

• **ASIN():** It returns the arc sine of a number.

>>> SELECT ASIN(1);

```
mysql> SELECT ASIN(1);
+--------------------+
| ASIN(1)            |
+--------------------+
| 1.5707963267948966 |
+--------------------+
1 row in set (0.00 sec)
```

• **POW():** It returns m raised to the nth power.

>>> Select POWER(25,2); - - - It means 25 raise to 2 I.e. 625

```
mysql> Select POWER(25,2) ;
+-------------+
| POWER(25,2) |
+-------------+
|         625 |
+-------------+
1 row in set (0.04 sec)
```

• **SQRT():** It returns the square root of a number.

```
mysql> Select sqrt(25);
+----------+
| sqrt(25) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> Select sqrt(145);
+--------------------+
| sqrt(145)          |
+--------------------+
| 12.041594578792296 |
+--------------------+
1 row in set (0.00 sec)
```

- **ROUND():** It returns a number rounded to a certain number of decimal places

>>> Select round(172.41);

```
mysql> Select round(172.41) ;
+---------------+
| round(172.41) |
+---------------+
|           172 |
+---------------+
1 row in set (0.00 sec)
```

- **MOD():** It returns the remainder of n divided by m.

>>> Select mod(25,2);

```
mysql> Select mod(25,2);
+-----------+
| mod(25,2) |
+-----------+
|         1 |
+-----------+
1 row in set (0.04 sec)
```

- **CEIL():** It returns the smallest integer value that is greater than or equal to a number.

>>> select ceil(11.5);

```
mysql> select ceil(11.5);
+------------+
| ceil(11.5) |
+------------+
|         12 |
+------------+
1 row in set (0.00 sec)
```

- **COS():** It returns the cosine of a number.
- **SIN():** It returns the sine of a number.
- **TAN():** It returns the tangent of a number.
- **COT():** It returns the cotangent of a number.

```
mysql> select cos(12);
+--------------------+
| cos(12)            |
+--------------------+
| 0.8438539587324921 |
+--------------------+
1 row in set (0.00 sec)

mysql> Select sin(22);
+-----------------------+
| sin(22)               |
+-----------------------+
| -0.008851309290403876 |
+-----------------------+
1 row in set (0.05 sec)

mysql> Select tan(14);
+--------------------+
| tan(14)            |
+--------------------+
| 7.2446066160948055 |
+--------------------+
1 row in set (0.00 sec)

mysql> select cot(12);
+---------------------+
| cot(12)             |
+---------------------+
| -1.5726734063976893 |
+---------------------+
1 row in set (0.03 sec)
```

- **LN():** It returns the natural logarithm of a number.
- **LOG10():** It returns the base-10 logarithm of a number.

```
mysql> select LN(25);
+--------------------+
| LN(25)             |
+--------------------+
| 3.2188758248682006 |
+--------------------+
1 row in set (0.00 sec)

mysql> select Log10(25);
+--------------------+
| Log10(25)          |
+--------------------+
| 1.3979400086720377 |
+--------------------+
1 row in set (0.00 sec)
```

- **GREATEST():** It returns the greatest value in a list of expressions.

>>> Select greatest(25, 26, 02, 58, 65, 41, 74);

```
mysql> Select greatest(25, 26, 02, 58, 65, 41, 74);
+---------------------------------------+
| greatest(25, 26, 02, 58, 65, 41, 74) |
+---------------------------------------+
|                                    74 |
+---------------------------------------+
1 row in set (0.03 sec)
```

- **LEAST():** It returns the smallest value in a list of expressions.

>>> Select least(25, 26, 02, 58, 65, 41, 74);

```
mysql> Select least(25, 26, 02, 58, 65, 41, 74);
+-----------------------------------+
| least(25, 26, 02, 58, 65, 41, 74) |
+-----------------------------------+
|                                 2 |
+-----------------------------------+
1 row in set (0.00 sec)
```

ii) Character Functions: Character functions are used to perform an operation on an input string or a character and return an output string or a character**.** String functions are sometimes called Character functions.

- **ASCII():** This function is used to find the ASCII value of a character.

>>>SELECT ASCII('A');

```
mysql> SELECT ascii('A');
+------------+
| ascii('A') |
+------------+
|         65 |
+------------+
1 row in set (0.05 sec)
```

- **SUBSTR():** This function is used to find a sub string from the a string from the given position.

>>> SELECT SUBSTR('MASTAN', 2, 4); - - - means start from position 2 and end upto 4

```
mysql> SELECT SUBSTR('Mastan', 2, 4);
+------------------------+
| SUBSTR('Mastan', 2, 4) |
+------------------------+
| asta                   |
+------------------------+
1 row in set (0.00 sec)
```

s

- **UPPER:** This function is used to make the string in upper case.

>>> SELECT UPPER('matoshri college');

```
mysql> SELECT UPPER('matoshri college');
+---------------------------+
| UPPER('matoshri college') |
+---------------------------+
| MATOSHRI COLLEGE          |
+---------------------------+
1 row in set (0.00 sec)
```

- **LOWER():** This function is used to convert the given string into lower case.

>>> SELECT LOWER('BATMAN');

```
mysql>  SELECT LOWER('BATMAN');
+-----------------+
| LOWER('BATMAN') |
+-----------------+
| batman          |
+-----------------+
1 row in set (0.00 sec)
```

- **CONCAT():** This function is used to add two words or strings.

```
mysql> Select concat('Avengers', 'Assemble');
+------------------------------+
| concat('Avengers', 'Assemble') |
+------------------------------+
| AvengersAssemble             |
+------------------------------+
1 row in set (0.00 sec)
```

- **LENGTH():** This function is used to find the length of a word.

```
mysql> Select length('Engineering');
+-----------------------+
| length('Engineering') |
+-----------------------+
|                    11 |
+-----------------------+
1 row in set (0.00 sec)
```

- **FORMAT():** This function is used to display a number in the given format.

```
mysql>  SELECT FORMAT(123456789, '##-##-#####');
+----------------------------------+
| FORMAT(123456789, '##-##-#####') |
+----------------------------------+
| 123,456,789                      |
+----------------------------------+
1 row in set, 1 warning (0.00 sec)
```

- **LEFT():** This function is used to SELECT a sub string from the left of given size or characters.

```
mysql>  SELECT LEFT('Matoshri College', 4);
+-----------------------------+
| LEFT('Matoshri College', 4) |
+-----------------------------+
| Mato                        |
+-----------------------------+
1 row in set (0.00 sec)
```

- **REPEAT():** This function is used to write the given string again and again till the number of times mentioned.

```
mysql> SELECT REPEAT('Tamam', 2);
+--------------------+
| REPEAT('Tamam', 2) |
+--------------------+
| TamamTamam         |
+--------------------+
1 row in set (0.00 sec)
```

- **REPLACE():** This function is used to cut the given string by removing the given sub string.

```
mysql> SELECT REPLACE('HELLO Pari', 'P', 'H');
+--------------------------------+
| REPLACE('HELLO Pari', 'P', 'H') |
+--------------------------------+
| HELLO Hari                     |
+--------------------------------+
1 row in set (0.00 sec)
```

- **REVERSE():** This function is used to reverse a string.

```
mysql> SELECT REVERSE('Hmm');
+----------------+
| REVERSE('Hmm') |
+----------------+
| mmH            |
+----------------+
1 row in set (0.00 sec)
```

- **RIGHT():** This function is used to SELECT a sub string from the right end of the given size.

```
mysql>  SELECT RIGHT('College', 3);
+---------------------+
| RIGHT('College', 3) |
+---------------------+
| ege                 |
+---------------------+
1 row in set (0.00 sec)
```

- **TRIM():** This function is used to cut the given symbol from the string.

```
mysql> SELECT TRIM(LEADING '0' FROM '000123');
+--------------------------------+
| TRIM(LEADING '0' FROM '000123') |
+--------------------------------+
| 123                            |
+--------------------------------+
1 row in set (0.00 sec)
```

*Practical 4: Sub-Queries, SET Operations, Joins.

Sub-Queries:

A Sub-query is a query within another query. The outer query is called as **main query** and inner query is called as **subquery**.

Consider following employee table:

```
mysql> select*from employee;
+------+----------+------+----------+--------+
| id   | Name     | Age  | Address  | Salary |
+------+----------+------+----------+--------+
|    1 | John     |   20 | US       |   2000 |
|    2 | Stephan  |   26 | Dubai    |   1500 |
|    3 | David    |   27 | Bangkok  |   2000 |
|    4 | Alina    |   29 | UK       |   6500 |
|    5 | Katherin |   34 | Banglore |   8500 |
|    6 | Harry    |   42 | China    |   4500 |
|    7 | Damon    |   25 | Mizoram  |  10000 |
+------+----------+------+----------+--------+
7 rows in set (0.00 sec)
```

i) Subqueries with the Select Statement :

```
>>> SELECT * FROM employee
    WHERE id IN (
            SELECT id FROM employee
            WHERE Salary > 4500);
```

```
mysql> SELECT * FROM employee WHERE id IN ( SELECT id FROM employee WHERE Salary > 4500);
+------+----------+------+----------+--------+
| id   | Name     | Age  | Address  | Salary |
+------+----------+------+----------+--------+
|    4 | Alina    |   29 | UK       |   6500 |
|    5 | Katherin |   34 | Banglore |   8500 |
|    7 | Damon    |   25 | Mizoram  |  10000 |
+------+----------+------+----------+--------+
3 rows in set (0.01 sec)
```

ii) Subqueries with the INSERT Statement :

```
>>> INSERT INTO employee_bkp
    SELECT * FROM employee
    WHERE id IN (
            SELECT id FROM employee);
```

```
mysql> INSERT INTO employee_bkp SELECT * FROM employee WHERE id IN ( SELECT id FROM employee);
Query OK, 7 rows affected (0.03 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

- The above statement will copy the complete employee table into employee_bkp table

iii) Subqueries with the UPDATE Statement :

```
>>> UPDATE employee SET Salary=Salary*0.25
    WHERE Age IN (
                SELECT Age FROM employee_bkp
                WHERE Age>=29);
```

```
mysql>  UPDATE employee SET Salary=Salary*0.25 WHERE Age IN ( SELECT Age FROM employee_bkp WHERE Age>=29);
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> Select * from employee;
+------+----------+------+----------+--------+
| id   | Name     | Age  | Address  | Salary |
+------+----------+------+----------+--------+
|    1 | John     |   20 | US       |   2000 |
|    2 | Stephan  |   26 | Dubai    |   1500 |
|    3 | David    |   27 | Bangkok  |   2000 |
|    4 | Alina    |   29 | UK       |   1625 |
|    5 | Katherin |   34 | Banglore |   2125 |
|    6 | Harry    |   42 | China    |   1125 |
|    7 | Damon    |   25 | Mizoram  |  10000 |
+------+----------+------+----------+--------+
7 rows in set (0.00 sec)
```

iv) Subqueries with the DELETE Statement :

```
>>> DELETE FROM employee
    WHERE Age IN (
                SELECT Age FROM employee_bkp
                WHERE Age>=29);
```

```
mysql> DELETE FROM employee WHERE Age IN ( SELECT Age FROM employee_bkp WHERE Age>=29);
Query OK, 3 rows affected (0.06 sec)

mysql> SELECT * FROM employee;
+------+---------+------+---------+--------+
| id   | Name    | Age  | Address | Salary |
+------+---------+------+---------+--------+
|    1 | John    |   20 | US      |   2000 |
|    2 | Stephan |   26 | Dubai   |   1500 |
|    3 | David   |   27 | Bangkok |   2000 |
|    7 | Damon   |   25 | Mizoram |  10000 |
+------+---------+------+---------+--------+
4 rows in set (0.00 sec)
```

SET Operations :

A SET is a collection of elements on which UNION, INTERSECTION and difference operations can be formed.

i) UNION
ii) INTERSECT
iii) EXCEPT.

i) UNION :

- A UNION clause is used to combine two tables into single table of all matching rows.
- Consider following two tables:

Table: Sale1

```
mysql> select * from sale1;
+------+--------+
| name | amount |
+------+--------+
| Joe  |   1000 |
| Alex |   2000 |
| Bob  |   5000 |
+------+--------+
3 rows in set (0.05 sec)
```

Table: Sale2

```
mysql> select * from sale2;
+------+--------+
| name | amount |
+------+--------+
| Joe  |   2000 |
| Alex |   2000 |
| Zach |   3000 |
+------+--------+
3 rows in set (0.04 sec)
```

```
>>> SELECT * FROM sale1
    UNION
    SELECT * FROM sale2;
```

```
mysql>  SELECT * FROM sale1 UNION SELECT * FROM sale2;
+------+--------+
| name | amount |
+------+--------+
| Joe  |   1000 |
| Alex |   2000 |
| Bob  |   5000 |
| Joe  |   2000 |
| Zach |   3000 |
+------+--------+
5 rows in set (0.04 sec)
```

- **UNION ALL :** This operation is similar to Union. But it shows the duplicate rows.

```
>>> SELECT * FROM sale1
    UNION ALL
    SELECT * FROM sale2;
```

```
mysql>  SELECT * FROM sale1 UNION ALL SELECT * FROM sale2;
+------+--------+
| name | amount |
+------+--------+
| Joe  |   1000 |
| Alex |   2000 |
| Bob  |   5000 |
| Joe  |   2000 |
| Alex |   2000 |
| Zach |   3000 |
+------+--------+
6 rows in set (0.00 sec)
```

ii) INTERSECT :

- Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements.
- In case of Intersect the number of columns and datatype must be same..

>>> MySQL does not support INTERSECT Operator.

The First table,

ID	NAME
1	abhi
2	adam

The Second table,

ID	NAME
2	adam
3	Chester

>>> SELECT * FROM First INTERSECT SELECT * FROM Second;

Output :

ID	NAME
2	adam

iii) EXCEPT :

- The SQL EXCEPT clause/operator is used to combine two SELECT statements and returns returns only rows, which are not available in the second SELECT statement.

>>> MySQL does not support EXCEPT Operator.

>>> SELECT * FROM First EXCEPT SELECT * FROM Second;

Output :

ID	NAME
1	abhi

JOINS :

The SQL JOIN clause is used to combine records from two pr more tables from a database.

- JOIN Operations:
  - i) INNER JOIN
  - ii) LEFT JOIN
  - iii) RIGHT JOIN
  - iv) FULL JOIN

  OUTER JOIN

- Consider following two tables: (one column common in both tables)

```
mysql> select *from students;
+------------+---------+--------------+
| student_id | city_id | student_name |
+------------+---------+--------------+
|          1 |     101 | Abhi         |
|          2 |     101 | Shyam        |
|          3 |     102 | Hari         |
|          4 |    NULL | Govind       |
+------------+---------+--------------+
4 rows in set (0.00 sec)
```

```
mysql> select *from city;
+---------+-----------+
| city_id | city_name |
+---------+-----------+
|     101 | Nanded    |
|     102 | Pune      |
|     103 | Chennai   |
+---------+-----------+
3 rows in set (0.00 sec)
```

i) INNER JOIN :

- The SQL INNER JOIN joins two tables based on a common column, and selects records that have matching values in these columns.

>>> SELECT * FROM students
INNER JOIN city ON students.city_id = city.city_id;

```
mysql> SELECT * FROM students
    -> INNER JOIN city ON students.city_id = city.city_id;
+------------+---------+--------------+---------+-----------+
| student_id | city_id | student_name | city_id | city_name |
+------------+---------+--------------+---------+-----------+
|          1 |     101 | Abhi         |     101 | Nanded    |
|          2 |     101 | Shyam        |     101 | Nanded    |
|          3 |     102 | Hari         |     102 | Pune      |
+------------+---------+--------------+---------+-----------+
3 rows in set (0.04 sec)
```

ii) LEFT JOIN :

- The SQL LEFT JOIN joins all rows from the left table even if there is no matches in right table

>>> SELECT * FROM students
LEFT JOIN city ON students.city_id = city.city_id;

```
mysql> SELECT * FROM students
    -> LEFT JOIN city ON students.city_id = city.city_id;
+------------+---------+--------------+---------+-----------+
| student_id | city_id | student_name | city_id | city_name |
+------------+---------+--------------+---------+-----------+
|          1 |     101 | Abhi         |     101 | Nanded    |
|          2 |     101 | Shyam        |     101 | Nanded    |
|          3 |     102 | Hari         |     102 | Pune      |
|          4 |    NULL | Govind       |    NULL | NULL      |
+------------+---------+--------------+---------+-----------+
4 rows in set (0.04 sec)
```

iii) RIGHT JOIN :

- The SQL RIGHT JOIN joins all rows from the right table even if there is no matches in left table

```
>>> SELECT * FROM students
    RIGHT JOIN city ON students.city_id = city.city_id;
```

```
mysql> SELECT * FROM students
    -> RIGHT JOIN city ON students.city_id = city.city_id;
+------------+---------+--------------+---------+-----------+
| student_id | city_id | student_name | city_id | city_name |
+------------+---------+--------------+---------+-----------+
|          2 |     101 | Shyam        |     101 | Nanded    |
|          1 |     101 | Abhi         |     101 | Nanded    |
|          3 |     102 | Hari         |     102 | Pune      |
|       NULL |    NULL | NULL         |     103 | Chennai   |
+------------+---------+--------------+---------+-----------+
4 rows in set (0.00 sec)
```

iv) FULL JOIN :

- The SQL FULL JOIN combines the results of both left and right outer joins.
- it means it joins all rows from both the tables even if there is no matches.

>>> *MySQL does not support FULL OUTER JOIN*

- Unlike SQL Server, MySQL does not support FULL OUTER JOIN as a separate JOIN type. However, to get the results same to FULL OUTER JOIN, you can combine LEFT OUTER JOIN and RIGHT OUTER JOIN as follows:

```
>>> SELECT * FROM students
    LEFT JOIN city ON students.city_id = city.city_id
    UNION
    SELECT * FROM students
    RIGHT JOIN city ON students.city_id = city.city_id;
```

```
mysql> SELECT * FROM students
    -> LEFT JOIN city ON students.city_id = city.city_id
    -> UNION
    -> SELECT * FROM students
    -> RIGHT JOIN city ON students.city_id = city.city_id;
+------------+---------+--------------+---------+-----------+
| student_id | city_id | student_name | city_id | city_name |
+------------+---------+--------------+---------+-----------+
|          1 |     101 | Abhi         |     101 | Nanded    |
|          2 |     101 | Shyam        |     101 | Nanded    |
|          3 |     102 | Hari         |     102 | Pune      |
|          4 |    NULL | Govind       |    NULL | NULL      |
|       NULL |    NULL | NULL         |     103 | Chennai   |
+------------+---------+--------------+---------+-----------+
5 rows in set (0.00 sec)
```

***Practical 5:** Creation of databases, writing SQL and PL/SQL queries to retrieve information from thev databases.

Creation of databases:

>>> CREATE DATABASE Pract5;

```
mysql> CREATE DATABASE Pract5;
Query OK, 1 row affected (0.08 sec)
```

- The above query will create a database named as pract5 as shown.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| college            |
| dbms               |
| information_schema |
| mpgi               |
| mysql              |
| performance_schema |
| pract5             |
| sys                |
| view               |
+--------------------+
9 rows in set (0.00 sec)
```

SQL queries to retrieve information from database:

- Consider following table:

```
mysql> Select * from student;
+----+-----------------+---------+------+
| id | name            | address | age  |
+----+-----------------+---------+------+
|  1 | Aman Rathod     | Nanded  |   26 |
|  2 | Rahul Wagh      | Pune    |   23 |
|  3 | Neha Sharma     | Latur   |   23 |
|  4 | Birudev Bhaskar | Pune    |   26 |
|  5 | Abhi Shinde     | Mumbai  |   22 |
|  6 | Asta Adhik      | Nanded  |   21 |
+----+-----------------+---------+------+
6 rows in set (0.05 sec)
```

***Query 1: Write a query to display all rows or records from student table.**

>>> SELECT * FROM student;

```
mysql> Select * from student;
+----+-----------------+---------+------+
| id | name            | address | age  |
+----+-----------------+---------+------+
|  1 | Aman Rathod     | Nanded  |   26 |
|  2 | Rahul Wagh      | Pune    |   23 |
|  3 | Neha Sharma     | Latur   |   23 |
|  4 | Birudev Bhaskar | Pune    |   26 |
|  5 | Abhi Shinde     | Mumbai  |   22 |
|  6 | Asta Adhik      | Nanded  |   21 |
+----+-----------------+---------+------+
6 rows in set (0.05 sec)
```

***Query 2: Write a query to display all rows or records from student table in descending order of name.**

```
>>> SELECT * FROM student
    ORDER BY name DESC;
```

```
mysql> Select * from student ORDER BY name DESC;
+----+-----------------+---------+------+
| id | name            | address | age  |
+----+-----------------+---------+------+
|  2 | Rahul Wagh      | Pune    |   23 |
|  3 | Neha Sharma     | Latur   |   23 |
|  4 | Birudev Bhaskar | Pune    |   26 |
|  6 | Asta Adhik      | Nanded  |   21 |
|  1 | Aman Rathod     | Nanded  |   26 |
|  5 | Abhi Shinde     | Mumbai  |   22 |
+----+-----------------+---------+------+
6 rows in set (0.03 sec)
```

- The above query will retrieve the data in descending order of names of the student.

***Query 3: Write a query to display names of the student who lives in 'Nanded'**

```
>>> SELECT * FROM student
    WHERE address='Nanded';
```

```
mysql> Select * from student WHERE address = 'Nanded' ;
+----+-------------+---------+------+
| id | name        | address | age  |
+----+-------------+---------+------+
|  1 | Aman Rathod | Nanded  |   26 |
|  6 | Asta Adhik  | Nanded  |   21 |
+----+-------------+---------+------+
2 rows in set (0.04 sec)
```

- The above query will retrieve the data of the students who lives in Nanded.

***Query 4: Write a query to display students whose age > 24.**

```
>>> SELECT * FROM student
    WHERE age > 24;
```

```
mysql> Select * from student WHERE age > 24 ;
+----+-----------------+---------+------+
| id | name            | address | age  |
+----+-----------------+---------+------+
|  1 | Aman Rathod     | Nanded  |   26 |
|  4 | Birudev Bhaskar | Pune    |   26 |
+----+-----------------+---------+------+
2 rows in set (0.04 sec)
```

- The above query will retrieve the data of the students whose age is greater than 24.

***Query 5: Write a query to Count total number of students.**

```
>>> SELECT COUNT(*) FROM student;
```

```
mysql> Select COUNT(*) from student;
+----------+
| COUNT(*) |
+----------+
|        6 |
+----------+
1 row in set (0.04 sec)
```

PL/SQL queries to retrieve information from database:

List of all PL/SQL Queries :

1. Create definition query:

```
BEGIN
      create table table name (
      colm name 1 data type(size),
      colm name 2 data type(size),
      ….
      colm name N data type(size)
      );
END;
```

In the above syntax, we use create table statement to create a new table; here, the specified table name means the actual table name that we need to create.

```
>>> create table studentA(
                  stud_id number(10) not null,
                  stud_name varchar2(30) not null,
                   stud_city varchar2(30)
                 );
```

```
Table created.
```

2. Data Insertion query:

```
BEGIN
        insert into  table name(colm 1,colm 2, ……colm N)
        values(col value1, colm value 2,….colm value 3);
END;
```

In the above syntax, we use to insert into a statement to insert the records into the specified table, here insert into keyword is mandatory

```
>>>  insert into studentA(stud_id, stud_name, stud_city) values(101,'Jenny','Mumbai');
      insert into studentA(stud_id, stud_name, stud_city) values(102,'Johan','Mumbai');
      insert into studentA(stud_id, stud_name, stud_city) values(103,'Pooja','London');
      insert into studentA(stud_id, stud_name, stud_city) values(104,'Sameer','London');
      insert into studentA(stud_id, stud_name, stud_city) values(105,'Rohit', 'London');

>>>  select * from studentA;
```

STUD_ID	STUD_NAME	STUD_CITY
101	Jenny	Mumbai
102	Johan	Mumbai
102	Johan	Mumbai
104	Sameer	London
105	Rohit	London

3. PL/SQL Update query:

```
BEGIN
      update table name
      set colm name 1 = colm value 1,
          colm name 2 = colm value 2, ……
          colm name N = colm value N
      where [condition];
END;
```

In the above syntax, we use an update query to update the records from the specified table

```
>>> update studentA set stud_city='Pune' where stud_id=101;
```

STUD_ID	STUD_NAME	STUD_CITY
101	Jenny	Pune
102	Johan	Mumbai
102	Johan	Mumbai
104	Sameer	London
105	Rohit	London

4. PL/SQL Deletion query:

```
BEGIN
     delete from table name where [specified condition] ;
END;
```

Syntax of delete query is very simple as shown in above syntax

5. PL/SQL Select query:

```
BEGIN
     select * from specified table name; or
END;
```

Basically, there are two ways to perform the select query in PL/SQL, as shown in the above syntax.

6. PL/SQL Arithmetic query:

```
select * from  table name where [specified condition];
```

In the above syntax, we use a select statement with specified table names with the specified arithmetic operators.
PL/SQL also provides some intermediate Query to the user as follows.

1. Currval and Nextval: It is used to generate the sequential number in increment order.
2. Rowid: It is used to return the rowid of any specified table.
3. Rownum: It displays the row from a specified table in which row may get selected.

Similarly, we have Comparison operators, Set Operator, %ISOPEN, Taking input from the user, Index-By table, Calling a Function, and %ROWCOUNT.

***Practical 6:** Assignment on Triggers and Cursors.

Triggers:

Triggers are the sql statements that are automatically executed when there is any change in the databases. It responces

Example 1:

- Create a table in MySQL:

```
>>> CREATE TABLE student(
        Id integer PRIMARY KEY,
        first_name varchar(50),
        last_name varchar(50),
        full_name varchar(50)
        );
```

```
mysql> CREATE TABLE student(Id integer PRIMARY KEY, first_name varchar(50), last_name varchar(50), full_name varchar(50));
Query OK, 0 rows affected (0.07 sec)

mysql> desc student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Id         | int         | NO   | PRI | NULL    |       |
| first_name | varchar(50) | YES  |     | NULL    |       |
| last_name  | varchar(50) | YES  |     | NULL    |       |
| full_name  | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.06 sec)
```

- Creation of triggers:

```
>>> create trigger student_name
    after INSERT
    on student
    for each row
     UPDATE student set full_name = first_name || ' ' || last_name;
```

```
mysql> create trigger student_name
    -> after INSERT
    -> on
    -> student
    -> for each row
    ->    UPDATE student set full_name = first_name || ' ' || last_name
    -> ;
Query OK, 0 rows affected, 2 warnings (0.06 sec)
```

- Insert records in a table:

```
>>> INSERT INTO student(id, first_name, last_name) VALUES(1,'Alvaro', 'Morte');
>>> INSERT INTO student(id, first_name, last_name) VALUES(2,'Ursula', 'Corbero');
>>> INSERT INTO student(id, first_name, last_name) VALUES(3,'Itziar', 'Ituno');
>>> INSERT INTO student(id, first_name, last_name) VALUES(4,'Pedro', 'Alonso');
>>> INSERT INTO student(id, first_name, last_name) VALUES(5,'Alba', 'Flores');
```

Output:

emp_id	first_name	last_name	full_name
1	Alvaro	Morte	Alvaro Morte
2	Ursula	Corbero	Ursula Corbero
3	Itziar	Ituno	Itziar Ituno
4	Pedro	Alonso	Pedro Alonso
5	Alba	Flores	Alba Flores

- Display Triggers in SQL:

```
>>> SHOW TRIGGERS \G;
```

--- \G rotates the table visually to vertical mode.

```
mysql> SHOW TRIGGERS \G;
*************************** 1. row ***************************
             Trigger: student_name
               Event: INSERT
               Table: student
           Statement: UPDATE student set full_name = first_name || ' ' || last_name
              Timing: AFTER
             Created: 2023-01-17 00:47:34.99
            sql_mode: STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
             Definer: root@localhost
character_set_client: cp850
collation_connection: cp850_general_ci
  Database Collation: utf8mb4_0900_ai_ci
1 row in set (0.05 sec)
```

- Drop Triggers in SQL:

```
>>> DROP TRIGGER student_name;
```

This will erase the trigger from the database.

Example 2:

Create table:

```
>>> create table subject(
                tid int(4),
                name varchar(30),
                subj1 int(2),
                subj2 int(2),
                subj3 int(2),
                total int(3),
                per int(3)
                );
```

```
mysql> create table subject(tid int(4), name varchar(30), subj1 int(2), subj2 int(2), subj3 int(2), total int(3), per int(3));
Query OK, 0 rows affected, 6 warnings (0.06 sec)

mysql> desc subject;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| tid   | int         | YES  |     | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
| subj1 | int         | YES  |     | NULL    |       |
| subj2 | int         | YES  |     | NULL    |       |
| subj3 | int         | YES  |     | NULL    |       |
| total | int         | YES  |     | NULL    |       |
| per   | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

Create Trigger:

```
>>>  create trigger stud_marks
       before INSERT
       on
       Student
       for each row
       set Student.total = Student.subj1 + Student.subj2 + Student.subj3,
       Student.per = Student.total * 60 / 100;
```

```
mysql> insert into Student values(0, "ABCDE", 20, 20, 20, 0, 0);
Query OK, 1 row affected (0.09 sec)

mysql> select * from Student;
+-----+-------+-------+-------+-------+-------+------+
| tid | name  | subj1 | subj2 | subj3 | total | per  |
+-----+-------+-------+-------+-------+-------+------+
| 100 | ABCDE |    20 |    20 |    20 |    60 |   36 |
+-----+-------+-------+-------+-------+-------+------+
1 row in set (0.00 sec)
```

Cursors:

- Whenever DML statements are executed, a temporary work area is created in the system memory and it is called a cursor.
- A cursor in database is a construct which allows you to iterate/traversal the records of a table.

Create a table:

```
>>> CREATE TABLE Book (
     ID INT PRIMARY KEY,
    TITLE VARCHAR(100),
    AUTHOR VARCHAR(40),
    DATE VARCHAR(40)
   );
```

```
mysql> CREATE TABLE Book (ID INT PRIMARY KEY, TITLE VARCHAR(100), AUTHOR VARCHAR(40), DATE VARCHAR(40));
Query OK, 0 rows affected (0.07 sec)
```

Insert values in table:

```
>>> insert into tutorials values(1, 'Java', 'Krishna', '2019-09-01');
>>> insert into tutorials values(2, 'JFreeCharts', 'Satish', '2019-05-01');
>>> insert into tutorials values(3, 'JavaSprings', 'Amit', '2019-05-01');
>>> insert into tutorials values(4, 'Android', 'Ram', '2019-03-01');
>>> insert into tutorials values(5, 'Cassandra', 'Pruthvi', '2019-04-06');
```

```
mysql> insert into book values(1, 'Java', 'Krishna', '2019-09-01');
Query OK, 1 row affected (0.04 sec)

mysql> insert into book values(2, 'JFreeCharts', 'Satish', '2019-05-01');
Query OK, 1 row affected (0.04 sec)

mysql> insert into book values(3, 'JavaSprings', 'Amit', '2019-05-01');
Query OK, 1 row affected (0.03 sec)

mysql> insert into book values(4, 'Android', 'Ram', '2019-03-01');
Query OK, 1 row affected (0.01 sec)

mysql> insert into book values(5, 'Cassandra', 'Pruthvi', '2019-04-06');
Query OK, 1 row affected (0.05 sec)
```

create another table to back up the data:

```
>>> CREATE TABLE backup (
     ID INT,
    TITLE VARCHAR(100),
    AUTHOR VARCHAR(40),
    DATE VARCHAR(40)
   );
```

```
mysql> CREATE TABLE Backup (ID INT, TITLE VARCHAR(100), AUTHOR VARCHAR(40), DATE VARCHAR(40));
Query OK, 0 rows affected (0.07 sec)
```

Following procedure backups the contents of the Book table to the backup table using cursors:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE ExampleProc()
    ->    BEGIN
    ->       DECLARE done INT DEFAULT 0;
    ->       DECLARE BookID INTEGER;
    ->       DECLARE BookTitle, BookAuthor, BookDate VARCHAR(20);
    ->       DECLARE cur CURSOR FOR SELECT * FROM Book;
    ->       DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    ->       OPEN cur;
    ->       label: LOOP
    ->       FETCH cur INTO BookID, BookTitle, BookAuthor, BookDate;
    ->       INSERT INTO backup VALUES(BookID, BookTitle, BookAuthor, BookDate);
    ->       IF done = 1 THEN LEAVE label;
    ->       END IF;
    ->       END LOOP;
    ->       CLOSE cur;
    ->    END//
Query OK, 0 rows affected (0.09 sec)
```

To call the above procedure:

>>> CALL ExampleProc;

```
mysql> CALL ExampleProc;
Query OK, 1 row affected (0.06 sec)
```

If we **retrieve the data from backup table** then **we get same inserted records from books** into backup table as follows:

```
mysql> select * from book;
+----+------------+---------+------------+
| ID | TITLE      | AUTHOR  | DATE       |
+----+------------+---------+------------+
|  1 | Java       | Krishna | 2019-09-01 |
|  2 | JFreeCharts | Satish | 2019-05-01 |
|  3 | JavaSprings | Amit   | 2019-05-01 |
|  4 | Android    | Ram     | 2019-03-01 |
|  5 | Cassandra  | Pruthvi | 2019-04-06 |
+----+------------+---------+------------+
5 rows in set (0.00 sec)

mysql> select * from backup;
+------+------------+---------+------------+
| ID   | TITLE      | AUTHOR  | DATE       |
+------+------------+---------+------------+
|    1 | Java       | Krishna | 2019-09-01 |
|    2 | JFreeCharts | Satish | 2019-05-01 |
|    3 | JavaSprings | Amit   | 2019-05-01 |
|    4 | Android    | Ram     | 2019-03-01 |
|    5 | Cassandra  | Pruthvi | 2019-04-06 |
+----+------------+---------+------------+
5 rows in set (0.00 sec)
```

***Practical 8:** Design and implement ER model diagrams (Entity Relationship Model) of database systems such as office automation, Hotel management and Hospital Management.

HOTEL MANAGEMENT :

The hotel management system is a set of hotel management apps that keep things running smoothly. Accounting software, customer relationship management software, and a huge number of industry-specific programs are all available.

HOTEL MANAGEMENT SYSTEM DATABASE:

1. Customer table
2. Room Table
3. Employees Table
4. Room class Table
5. Reservation Table
6. Payment Table

Create A Database named as Hotel Management System:

>>> CREATE DATABASE Hotel_Management_System;

```
mysql> CREATE DATABASE Hotel_Management_System;
Query OK, 1 row affected (0.04 sec)
```

>>> use Hotel_Management_System;

```
mysql> use Hotel_Management_System;
Database changed
mysql>
```

Create Tables in database:

1. Customer table:

```
>>> CREATE TABLE Customer(
      Cust_id int(11) PRIMARY KEY,
      Firstname varchar(255),
      Lastname varchar(255),
      Address varchar(255),
      Status varchar(255),
      MobNo. Int(11)
    );
```

```
mysql> CREATE TABLE Customer(
    -> Cust_id int(11) PRIMARY KEY,
    -> Firstname varchar(255),
    -> Lastname varchar(255),
    -> Address varchar(255),
    -> Status varchar(255),
    -> MobNo int(11)
    -> );
Query OK, 0 rows affected, 2 warnings (0.04 sec)

mysql> desc Customer;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| Cust_id   | int          | NO   | PRI | NULL    |       |
| Firstname | varchar(255) | YES  |     | NULL    |       |
| Lastname  | varchar(255) | YES  |     | NULL    |       |
| Address   | varchar(255) | YES  |     | NULL    |       |
| Status    | varchar(255) | YES  |     | NULL    |       |
| MobNo     | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.04 sec)
```

2. Room table:

```
>>> CREATE TABLE Room(
      room_id int(11) PRIMARY KEY,
      room_type varchar(30),
      description text
    );
```

```
mysql> CREATE TABLE room(
    -> room_id int(11) PRIMARY KEY,
    -> room_type varchar(30),
    -> description text
    -> );
Query OK, 0 rows affected, 1 warning (0.08 sec)

mysql> desc room;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| room_id     | int         | NO   | PRI | NULL    |       |
| room_type   | varchar(30) | YES  |     | NULL    |       |
| description | text        | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

3. Employees table:

```
>>> CREATE TABLE Employees(
    employee_id int(11) PRIMARY KEY,
    Firstname varchar(30),
    Lastname varchar(30),
    Address text,
    post varchar(30),
    Username varchar(30),
    Password varchar(30));
```

```
mysql> CREATE TABLE Employees(
    -> employee_id int(11) PRIMARY KEY,
    -> Firstname varchar(30),
    -> Lastname varchar(30),
    -> Address text,
    -> post varchar(30),
    -> Username varchar(30),
    -> Password varchar(30)
    -> );
Query OK, 0 rows affected, 1 warning (0.07 sec)

mysql> desc employees;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| employee_id | int         | NO   | PRI | NULL    |       |
| Firstname   | varchar(30) | YES  |     | NULL    |       |
| Lastname    | varchar(30) | YES  |     | NULL    |       |
| Address     | text        | YES  |     | NULL    |       |
| post        | varchar(30) | YES  |     | NULL    |       |
| Username    | varchar(30) | YES  |     | NULL    |       |
| Password    | varchar(30) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.05 sec)
```

4. Room Class table:

```
>>> CREATE TABLE RoomClass(
    Class_id int(11) PRIMARY KEY,
    name varchar(30),
    Price int(11)
   );
```

```
mysql> CREATE TABLE RoomClass(
    -> Class_id int(11) PRIMARY KEY,
    -> name varchar(30),
    -> Price int(11)
    -> );
Query OK, 0 rows affected, 2 warnings (0.09 sec)

mysql> desc RoomClass;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Class_id | int         | NO   | PRI | NULL    |       |
| name     | varchar(30) | YES  |     | NULL    |       |
| Price    | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

5. Reservation table:

```
>>> CREATE TABLE Reservation(
      Reservation_id int(11) PRIMARY KEY,
      Cust_id int(11),
       room_id int(11),
      R_date date,
      Date_in date,
      Date_out date,
      Foreign key (cust_id) REFERENCES Customer(cust_id),
      Foreign key (room_id) REFERENCES room(room_id)
     );
```

```
mysql> CREATE TABLE Reservation(
    -> Reservation_id int(11) PRIMARY KEY,
    -> Cust_id int(11),
    -> room_id int(11),
    -> R_date date,
    -> Date_in date,
    -> Date_out date,
    -> Foreign key (cust_id) REFERENCES Customer(cust_id),
    -> Foreign key (room_id) REFERENCES room(room_id)
    -> );
Query OK, 0 rows affected, 3 warnings (0.09 sec)

mysql> desc reservation;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| Reservation_id | int  | NO   | PRI | NULL    |       |
| Cust_id        | int  | YES  | MUL | NULL    |       |
| room_id        | int  | YES  | MUL | NULL    |       |
| R_date         | date | YES  |     | NULL    |       |
| Date_in        | date | YES  |     | NULL    |       |
| Date_out       | date | YES  |     | NULL    |       |
+----------------+------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

6. Payment table:

```
>>>  CREATE TABLE Payment(
       payment_id int(11) PRIMARY KEY,
       Cust_id int(11),
       Payment_date date,
       Foreign key (cust_id) REFERENCES Customer(cust_id)
     );
```

```
mysql> CREATE TABLE Payment(
    -> payment_id int(11) PRIMARY KEY,
    -> Cust_id int(11),
    -> Payment_date date,
    -> Foreign key (cust_id) REFERENCES Customer(cust_id)
    -> );
Query OK, 0 rows affected, 2 warnings (0.06 sec)

mysql> desc payment;
+--------------+------+------+-----+---------+-------+
| Field        | Type | Null | Key | Default | Extra |
+--------------+------+------+-----+---------+-------+
| payment_id   | int  | NO   | PRI | NULL    |       |
| Cust_id      | int  | YES  | MUL | NULL    |       |
| Payment_date | date | YES  |     | NULL    |       |
+--------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

Tables in Hotel Management System database:

```
>>>  SHOW TABLES;
```

```
mysql> show tables;
+-----------------------------------+
| Tables_in_hotel_management_system |
+-----------------------------------+
| customer                          |
| employees                         |
| payment                           |
| reservation                       |
| room                              |
| roomclass                         |
+-----------------------------------+
6 rows in set (0.04 sec)
```

***Practical 9:** Java Database Connectivity:

1. Create a database:

>>> CREATE DATBASE mydb;

```
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.01 sec)
```

2. **Create a table in that database:**

>>> CREATE TABLEe designation(
code int PRIMARY KEY auto_increment,
title char(35) NOT NULL UNIQUE
);

```
mysql> create table designation(
    -> code int primary key auto_increment,
    -> title char(35) not null unique
    -> );
Query OK, 0 rows affected (0.03 sec)
```

3. Now, we want to access the data of this table using Java database connectivity.

I) create a directory in your main drive (named gfg).

II) now, inside gfg created two more directories one named as 'src' and the other 'lib'.

```
C:\WINDOWS\system32\cmd.exe

C:\>md gfg

C:\>cd gfg

C:\gfg>md lib

C:\gfg>md src

C:\gfg>dir
 Volume in drive C has no label.
 Volume Serial Number is DC5F-2A04

 Directory of C:\gfg

14-12-2021  08:03    <DIR>          .
14-12-2021  08:03    <DIR>          ..
14-12-2021  08:03    <DIR>          lib
14-12-2021  08:03    <DIR>          src
               0 File(s)              0 bytes
               4 Dir(s)  51,382,362,112 bytes free

C:\gfg>_
```

III) put the MySQL connector java jar file in the lib folder.

```
C:\gfg>cd lib

C:\gfg\lib>dir
 Volume in drive C has no label.
 Volume Serial Number is DC5F-2A04

 Directory of C:\gfg\lib

14-12-2021  08:08    <DIR>          .
14-12-2021  08:08    <DIR>          ..
09-03-2020  11:19         2,385,601 mysql-connector-java-8.0.20.jar
               1 File(s)      2,385,601 bytes
               2 Dir(s)  51,388,645,376 bytes free
```

4. we will write connectivity code in the src folder, To write connectivity code user must know the following information:

Driver class:- The driver class for connectivity of MySQL database "com.mysql.cj.jdbc.Driver"
URL for Connection:- The connection URL for the mysql database is jdbc:mysql://localhost:3306/mydb ('mydb' is the name of database).

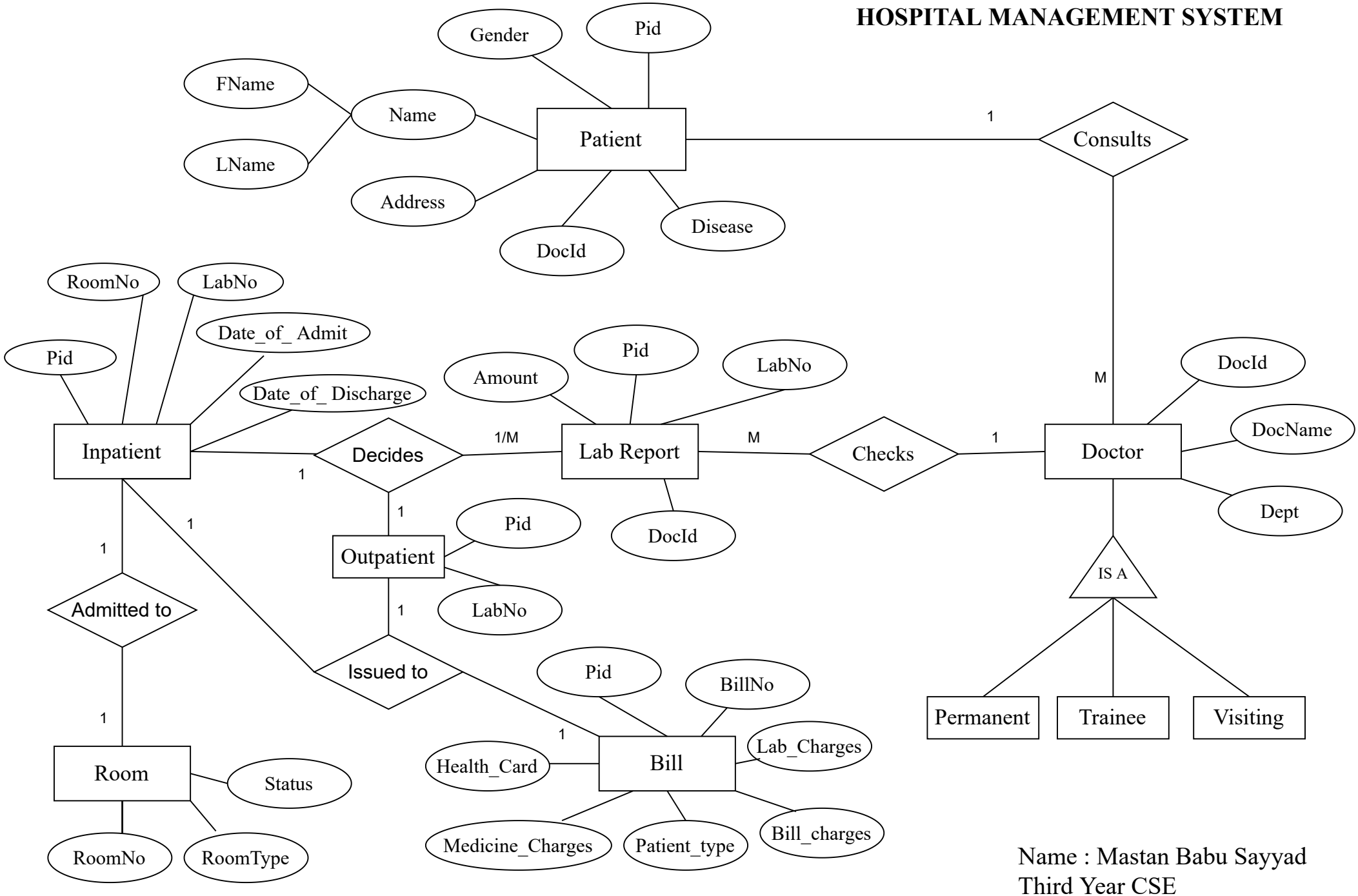
To get more clarification follow the connectivity code below.

5. In this src code, we will set up the connection and get all the data from the table. we have created the 'check.java' file in the src folder

```java
import java.sql.*;
public class GFG {
    public static void main(String arg[])
    {
        Connection connection = null;
        try {
            // below two lines are used for connectivity.
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/mydb",
                "mydbuser", "mydbuser");

            // mydb is database
            // mydbuser is name of database
            // mydbuser is password of database

            Statement statement;
            statement = connection.createStatement();
            ResultSet resultSet;
            resultSet = statement.executeQuery(
                "select * from designation");
            int code;
            String title;
            while (resultSet.next()) {
                code = resultSet.getInt("code");
                title = resultSet.getString("title").trim();
                System.out.println("Code : " + code
                                   + " Title : " + title);
            }
            resultSet.close();
            statement.close();
            connection.close();
        }
        catch (Exception exception) {
            System.out.println(exception);
        }
    } // function ends
} // class ends
```

```
C:\gfg\src>javac -classpath ..\lib\mysql-connector-java-8.0.20.jar;. Check.java

C:\gfg\src>java -classpath ..\lib\mysql-connector-java-8.0.20.jar;. Check
Code : 2 Title : CEO
Code : 3 Title : cook
Code : 1 Title : dancer
Code : 5 Title : manager
Code : 31 Title : null
Code : 8 Title : security
Code : 6 Title : waiter

C:\gfg\src>_
```

HOSPITAL MANAGEMENT SYSTEM



Name : Mastan Babu Sayyad
Third Year CSE