

## **BASIC CONCEPTS OF SQL**

### **Introduction to SQL**

SQL stands for “Structured Query Language” and can be pronounced as “SQL” or “sequel – (Structured English Query Language)”. It is a query language used for accessing and modifying information in the database. IBM first developed SQL in 1970s. Also it is an ANSI/ISO standard. It has become a Standard Universal Language used by most of the relational database management systems (RDBMS). Some of the RDBMS systems are: Oracle, Microsoft SQL server, Sybase etc. Most of these have provided their own implementation thus enhancing its feature and making it a powerful tool. Few of the SQL commands used in SQL programming are SELECT Statement, UPDATE Statement, INSERT INTO Statement, DELETE Statement, WHERE Clause, ORDER BY Clause, GROUP BY Clause, ORDER Clause, Joins, Views, GROUP Functions, Indexes etc.

### **SQL Commands**

SQL commands are instructions used to communicate with the database to perform specific task that work with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can create tables, add data to tables, or modify data, drop the table, set permissions for users.

### **CREATE TABLE Statement**

The CREATE TABLE Statement is used to create tables to store data. Integrity Constraints like primary key, unique key and foreign key can be defined for the columns while creating the table. The integrity constraints can be defined at column level or table level. The implementation and the syntax of the CREATE Statements differs for different RDBMS.

**The Syntax for the CREATE TABLE Statement is:**

```
CREATE TABLE  
table_name  
  
(column_name1 datatype constraint,  
column_name2 datatype, ...  
column_nameN datatype);
```

**SQL Data Types:**

char(size)	Fixed-length character string. Size is specified in parenthesis. Max 255 bytes.
Varchar2(size)	Variable-length character string. Max size is specified in parenthesis.
number(size) or int	Number value with a max number of column digits specified in parenthesis.
Date	Date value in „dd-mon-yy“. Eg., „07-jul-2004“
number(size,d) or real	Number value with a maximum number of digits of "size" total, with a maximum number of "d" digits to the right of the decimal.

**SQL Integrity Constraints:**

Integrity Constraints are used to apply business rules for the database tables. The constraints available in SQL are **Foreign Key, Primary key, Not Null, Unique, Check**.

Constraints can be defined in two ways:

1. The constraints can be specified immediately after the column definition. This is called column-level definition.
2. The constraints can be specified after all the columns are defined. This is called table-level definition.

**1) Primary key:**

This constraint defines a column or combination of columns which uniquely identify each row in the table.

**Syntax to define a Primary key at column level:**

```
Column_namedatatype[CONSTRAINTconstraint_name] PRIMARYKEY
```

### Syntax to define a Primary key at table level:

```
[CONSTRAINT constraint_name] PRIMARY KEY(column_name1,  
column_name2,..)
```

### Foreign key or Referential Integrity:

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. For a column to be defined as a Foreign Key, it should be defined as a Primary Key in the table which it is referring. One or more columns can be defined as foreign key.

### Syntax to define a Foreign key at column level:

```
[CONSTRAINTconstraint_name] REFERENCES
```

```
referenced_table_name(column_name)
```

### Not Null Constraint:

This constraint ensures all rows in the table contain a definite value for the column which is specified as not null. Which means a null value is not allowed.

```
[CONSTRAINTconstraintname] NOTNULL
```

### Syntax to define a Not Null constraint:

### Unique Key:

This constraint ensures that a column or a group of columns in each row have a distinct value.

A column(s) can have a null value but the values cannot be duplicated.

```
[CONSTRAINT constraint_name] UNIQUE
```

**Syntax to define a Unique key at column level:**

**Syntax to define a Unique key at table level:**

```
[CONSTRAINT constraint_name] UNIQUE (column_name)
```

```
[CONSTRAINT constraint_name] CHECK (condition)
```

## ALTER TABLE Statement

The SQL ALTER TABLE command is used to modify the definition (structure) of a table by modifying the definition of its columns. The ALTER command is used to perform the following functions.

- 1) Add, drop, modify table columns
- 2) Add and drop constraints
- 3) Enable and Disable constraints

## He HAVING clause

The HAVING clause can be used to restrict the display of grouped rows. The result of the grouped query is passed on to the HAVING clause for output filtration.

- **The INSERT INTO Statement**

The INSERT INTO statement is used to insert a row in a table.

- **The UPDATE Statement**

The UPDATE statement is used to update existing records in a table.

- **The DELETE Statement**

The DELETE statement is used to delete rows in a table. SQL DELETE

- **Commit command**

Commit command is used to permanently save any transaction into database.

- **Rollback command**

This command restores the database to last committed state. It is also used with savepoint command to jump to a save point in a transaction.

- **Save point command**

**Save point** command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

1) **Create a table called Employee & execute the following.**

**Employee(EMPNO,ENAME,JOB, MANAGER\_NO, SAL, COMMISSION)**

1. **Create a user and grant all permissions to the user.**
2. **Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER\_NO, SAL, COMMISSION and use rollback. Check the result.**
3. **Add primary key constraint and not null constraint to the employee table.**
4. **Insert null values to the employee table and verify the result.**

```
CREATE TABLE Employee (  
  
    EMPNO INT PRIMARY KEY,  
  
    ENAME VARCHAR(255),  
  
    JOB VARCHAR(255),  
  
    MANAGER_NO INT,  
  
    SAL DECIMAL(10, 2),  
  
    COMMISSION DECIMAL(10, 2)  
  
);
```

1. -- Create a new user

```
CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';
```

-- Grant all privileges to the user on a specific database

```
GRANT ALL PRIVILEGES ON your_database_name.* TO 'new_user'@'localhost';
```

-- Flush privileges to apply changes

```
FLUSH PRIVILEGES;
```

Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER\_NO, SAL, COMMISSION and use rollback. Check the result.

2. -- Begin transaction

START TRANSACTION;

-- Insert three records

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)  
VALUES

(1, 'John Doe', 'Manager', NULL, 5000, 1000),

(2, 'Jane Smith', 'Developer', 1, 4000, NULL),

(3, 'Alice Johnson', 'Salesperson', 1, 3000, 500);

-- Rollback the transaction

ROLLBACK;

Add primary key constraint and not null constraint to the employee table.

3. -- Add primary key constraint to EMPNO column

ALTER TABLE Employee

ADD CONSTRAINT PK\_Employee\_EMPNO PRIMARY KEY (EMPNO);

-- Add not null constraints to EMPNO, ENAME, JOB, and SAL columns

ALTER TABLE Employee

MODIFY EMPNO INT NOT NULL,

MODIFY ENAME VARCHAR(255) NOT NULL,

MODIFY JOB VARCHAR(255) NOT NULL,

MODIFY SAL DECIMAL(10, 2) NOT NULL;

4. -- Inserting NULL values into MANAGER\_NO and COMMISSION columns

INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER\_NO, SAL, COMMISSION)

VALUES

(4, 'Bob Brown', 'Intern', NULL, 2500, NULL),

(5, 'Emma White', 'Assistant', NULL, 3000, NULL);

-- Selecting all records from the Employee table

SELECT \* FROM Employee;

```
student@student1: ~
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> create database Employee;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| Company |
| Employee |
| IA2 |
| Library |
| Salesman |
| employee |
| information_schema |
| internal |
| library |
| library1 |
| movie |
| mysql |
| performance_schema |
| prg1 |
| publisher |
| salesman |
| sys |
+-----+
17 rows in set (0.00 sec)

mysql> use Employee;
ERROR 1049 (42000): Unknown database 'Employee'
mysql> use Employee;
Database changed
mysql> CREATE TABLE Employee (
-> EMPNO INT PRIMARY KEY,
-> ENAME VARCHAR(255),
-> JOB VARCHAR(255),
-> MANAGER_NO INT,
-> SAL DECIMAL(10, 2),
-> COMMISSION DECIMAL(10, 2)
-> );
Query OK, 0 rows affected (0.04 sec)

mysql>
mysql>
mysql> Create a user and grant all permissions to the user.
->
-> -- Create a new user
-> CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right
```



```
student@student1: ~  
mysql> Create a user and grant all permissions to the user.  
->  
-> -- Create a new user  
-> CREATE USER 'new_user'@'localhost' IDENTIFIED BY 'password';  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'a user and grant all permissions to the user.'  
  
CREATE USER 'new_user'@'localhos' at line 1  
mysql>  
mysql> -- Grant all privileges to the user on a specific database  
mysql> GRANT ALL PRIVILEGES ON your_database_name.* TO 'new_user'@'localhost';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql>  
mysql> -- Flush privileges to apply changes  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>  
mysql> Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.  
->  
-> -- Begin transaction  
-> START TRANSACTION;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'any three records in the employee table contains attributes EMPNO,ENAME JOB, MAN' at line 1  
mysql>  
mysql> -- Insert three records  
mysql> INSERT INTO Employee (EMPNO, ENAME, JOB, MANAGER_NO, SAL, COMMISSION)  
-> VALUES  
-> (1, 'John Doe', 'Manager', NULL, 5000, 1000),  
-> (2, 'Jane Smith', 'Developer', 1, 4000, NULL),  
-> (3, 'Alice Johnson', 'Salesperson', 1, 3000, 500);  
Query OK, 3 rows affected (0.02 sec)  
Records: 3 Duplicates: 0 Warnings: 0  
  
mysql>  
mysql> -- Rollback the transaction  
mysql> ROLLBACK;  
Query OK, 0 rows affected (0.00 sec)  
  
mysql>  
mysql> Add primary key constraint and not null constraint to the employee table.  
->  
-> -- Add primary key constraint to EMPNO column  
-> ALTER TABLE Employee  
-> ADD CONSTRAINT PK_Employee_EMPNO PRIMARY KEY (EMPNO);  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Add primary key constraint and not null constraint to the employee table.'  
  
ALTER at line 1
```

**2) Create a table called Employee that contain attributes EMPNO,ENAME,JOB, MGR,SAL &execute the following.**

- 1. Add a column commission with domain to the Employee table.**
- 2. Insert any five records into the table.**
- 3. Update the column details of job**
- 4. Rename the column of Employ table using alter command.**
- 5. Delete the employee whose Empno is 105.**

**-- Step 1:** Create the Employee table with attributes EMPNO, ENAME, JOB, MGR, SAL

```
CREATE TABLE Employee (
```

```
    EMPNO INT,
```

```
    ENAME VARCHAR(50),
```

```
    JOB VARCHAR(50),
```

```
    MGR INT,
```

```
    SAL DECIMAL(10, 2)
```

```
);
```

**-- Step 2:** Add a column commission to the Employee table

```
ALTER TABLE Employee
```

```
ADD COLUMN COMMISSION DECIMAL(10, 2);
```

**-- Step 3:** Insert five records into the table

```
INSERT INTO Employee (EMPNO, ENAME, JOB, MGR, SAL, COMMISSION) VALUES
```

```
(101, 'John Doe', 'Manager', NULL, 5000.00, NULL),
```

```
(102, 'Jane Smith', 'Developer', 101, 4000.00, NULL),
```

```
(103, 'Alice Johnson', 'Designer', 101, 4500.00, NULL),
```

```
(104, 'Bob Brown', 'Analyst', 101, 4200.00, NULL),
```

```
(105, 'Emma Wilson', 'Tester', 102, 3800.00, NULL);
```

**-- Step 4:** Update the column details of the JOB column

UPDATE Employee

SET JOB = CONCAT('Senior ', JOB);

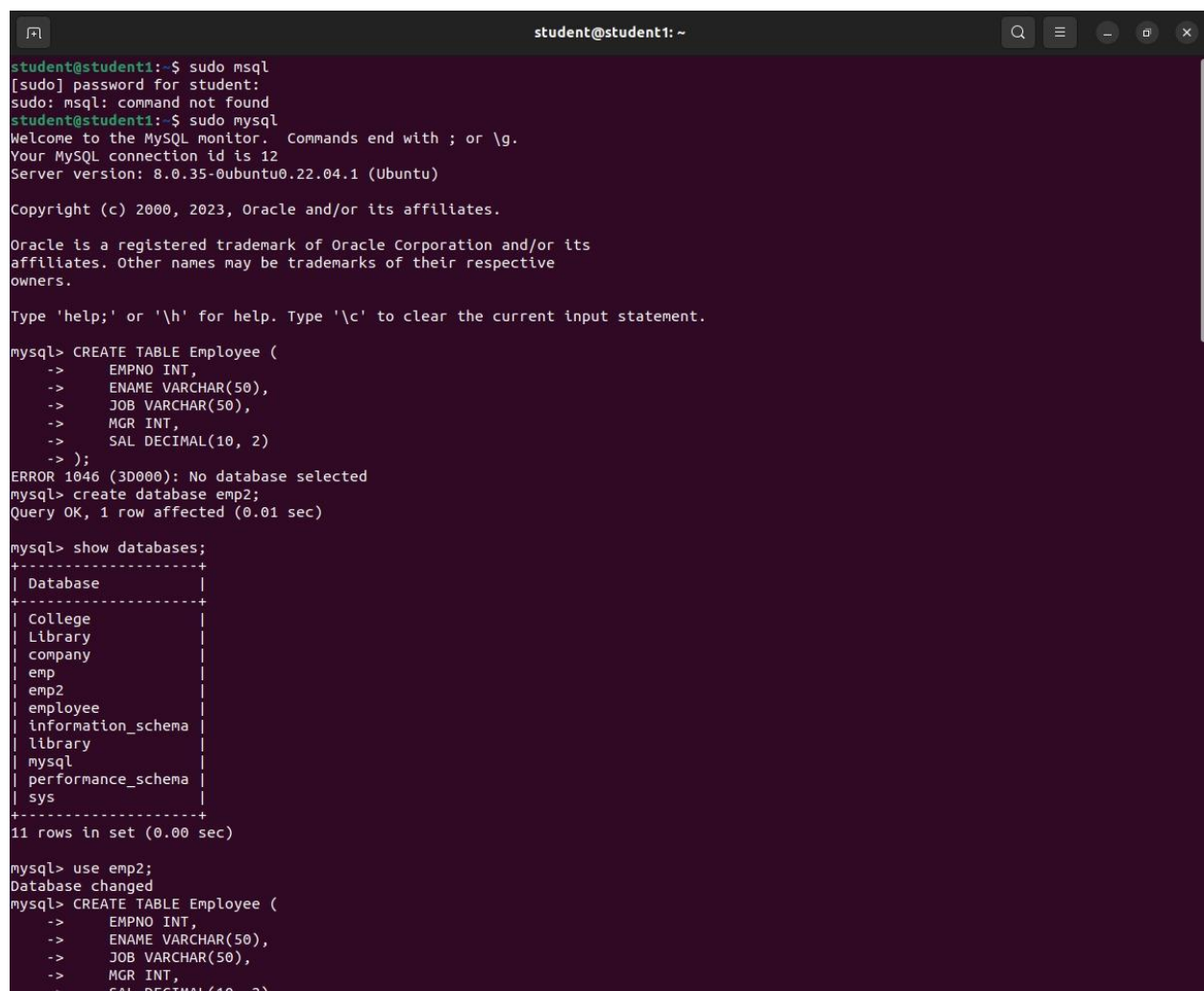
-- **Step 5:** Rename the column MGR to MANAGER\_NO

ALTER TABLE Employee

CHANGE COLUMN MGR MANAGER\_NO INT;

-- **Step 6:** Delete the employee whose EMPNO is 105

DELETE FROM Employee WHERE EMPNO = 105;



```
student@student1: ~
student@student1:~$ sudo mysql
[sudo] password for student:
sudo: mysql: command not found
student@student1:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.35-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2023, 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> CREATE TABLE Employee (
  ->     EMPNO INT,
  ->     ENAME VARCHAR(50),
  ->     JOB VARCHAR(50),
  ->     MGR INT,
  ->     SAL DECIMAL(10, 2)
  -> );
ERROR 1046 (3D000): No database selected
mysql> create database emp2;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| College |
| Library |
| company |
| emp      |
| emp2     |
| employee |
| information_schema |
| library  |
| mysql    |
| performance_schema |
| sys      |
+-----+
11 rows in set (0.00 sec)

mysql> use emp2;
Database changed
mysql> CREATE TABLE Employee (
  ->     EMPNO INT,
  ->     ENAME VARCHAR(50),
  ->     JOB VARCHAR(50),
  ->     MGR INT,
  ->     SAL DECIMAL(10, 2)
```

```

mysql> UPDATE Employee
  -> SET JOB = 'Senior ' || JOB
  -> WHERE EMPNO IN (101, 102, 103, 104, 105);
ERROR 1292 (22007): Truncated incorrect DOUBLE value: 'Senior '
mysql> update Employee;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> UPDATE Employee
  -> SET JOB = CONCAT('Senior ', JOB);
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int | YES | | NULL | |
| ENAME | varchar(50) | YES | | NULL | |
| JOB | varchar(50) | YES | | NULL | |
| MGR | int | YES | | NULL | |
| SAL | decimal(10,2) | YES | | NULL | |
| COMMISSION | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> ALTER TABLE Employee
  -> CHANGE COLUMN MGR MANAGER_NO INT;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int | YES | | NULL | |
| ENAME | varchar(50) | YES | | NULL | |
| JOB | varchar(50) | YES | | NULL | |
| MANAGER_NO | int | YES | | NULL | |
| SAL | decimal(10,2) | YES | | NULL | |
| COMMISSION | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> DELETE FROM Employee WHERE EMPNO = 105;
Query OK, 1 row affected (0.01 sec)

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int | YES | | NULL | |
| ENAME | varchar(50) | YES | | NULL | |

```

```

| SAL | decimal(10,2) | YES | | NULL | |
| COMMISSION | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> create database emp3;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| College |
| Library |
| company |
| emp |
| emp2 |
| emp3 |
| employee |
| information_schema |
| library |
| mysql |
| performance_schema |
| sys |
+-----+
12 rows in set (0.00 sec)

mysql> use emp3;
Database changed
mysql> CREATE TABLE Employee (
  -> E_id INT PRIMARY KEY,
  -> E_name VARCHAR(50),
  -> Age INT,
  -> Salary DECIMAL(10, 2)
  -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Employee (
  -> E_id INT PRIMARY KEY,
  -> E_name VARCHAR(50),
  -> Age INT,
  -> Salary DECIMAL(10, 2)
  -> );
ERROR 1050 (42501): Table 'Employee' already exists
mysql> CREATE TABLE Employee3 (
  -> E_id INT PRIMARY KEY,
  -> E_name VARCHAR(50),
  -> Age INT,
  -> Salary DECIMAL(10, 2)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql> desc Employee3;

```

```
student@student1: ~
1 row in set (0.00 sec)

mysql> desc Employee3;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| E_id  | int           | NO   | PRI | NULL    |       |
| E_name | varchar(50)   | YES  |     | NULL    |       |
| Age   | int           | YES  |     | NULL    |       |
| Salary | decimal(10,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT COUNT(*) AS num_employees_high_salary
-> FROM Employee
-> WHERE Salary > 8000;
+-----+-----+
| num_employees_high_salary |
+-----+-----+
| 0 |
+-----+-----+
1 row in set (0.01 sec)

mysql> INSERT INTO Employee3 (E_id, E_name, Age, Salary)
-> VALUES
-> (101, 'Anu', 22, 9000),
-> (102, 'Shane', 29, 8000),
-> (103, 'Rohan', 34, 6000),
-> (104, 'Scott', 44, 10000),
-> (105, 'Tiger', 35, 8000),
-> (106, 'Alex', 27, 7000),
-> (107, 'Abhi', 29, 8000);
Query OK, 7 rows affected (0.00 sec)
Records: 7 Duplicates: 0 Warnings: 0

mysql> desc Employee3;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| E_id  | int           | NO   | PRI | NULL    |       |
| E_name | varchar(50)   | YES  |     | NULL    |       |
| Age   | int           | YES  |     | NULL    |       |
| Salary | decimal(10,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from Employee3;
+-----+-----+-----+-----+
| E_id | E_name | Age | Salary |
+-----+-----+-----+-----+
| 101  | Anu    | 22  | 9000.00 |
| 102  | Shane  | 29  | 8000.00 |
| 103  | Rohan  | 34  | 6000.00 |
```

**3) Queries using aggregate functions (COUNT,AVG,MIN,MAX,SUM),Group by, Order by. Employee(E\_id, E\_name, Age, Salary)**

- 1. Create Employee table containing all Records E\_id, E\_name, Age, Salary.**
- 2. Count number of employee names from employee table**
- 3. Find the Maximum age from employee table.**
- 4. Find the Minimum age from employee table.**
- 5. Find salaries of employee in Ascending Order.**
- 6. Find grouped salaries of employees.**

1. Create the Employee table:

```
CREATE TABLE Employee (  
    E_id INT PRIMARY KEY,  
    E_name VARCHAR(50),  
    Age INT,  
    Salary DECIMAL(10, 2)  
);
```

2. Count the number of employee names:

```
SELECT COUNT(E_name) AS num_employees  
FROM Employee;
```

3. Find the maximum age:

```
SELECT MAX(Age) AS max_age  
FROM Employee;
```

4. Find the minimum age:

```
SELECT MIN(Age) AS min_age  
FROM Employee;
```

5. Find salaries of employees in ascending order:

```
SELECT E_name, Salary
```

```
FROM Employee
```

```
ORDER BY Salary ASC;
```

6. Find grouped salaries of employees:

```
SELECT Salary, COUNT(*) AS num_employees
```

```
FROM Employee
```

```
GROUP BY Salary;
```

These SQL queries should help you achieve the specified tasks using aggregate functions, GROUP BY, and ORDER BY clauses in SQL. You can execute them in your SQL environment to obtain the desired results.

```
student@student1: ~
mysql> UPDATE Employee
  -> SET JOB = 'Senior ' || JOB
  -> WHERE EMPNO IN (101, 102, 103, 104, 105);
ERROR 1292 (22007): Truncated incorrect DOUBLE value: 'Senior '
mysql> update Employee;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1
mysql> UPDATE Employee
  -> SET JOB = CONCAT('Senior ', JOB);
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int  | YES  |     | NULL    |       |
| ENAME | varchar(50) | YES  |     | NULL    |       |
| JOB   | varchar(50) | YES  |     | NULL    |       |
| MGR   | int  | YES  |     | NULL    |       |
| SAL   | decimal(10,2) | YES  |     | NULL    |       |
| COMMISSION | decimal(10,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> ALTER TABLE Employee
  -> CHANGE COLUMN MGR MANAGER_NO INT;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int  | YES  |     | NULL    |       |
| ENAME | varchar(50) | YES  |     | NULL    |       |
| JOB   | varchar(50) | YES  |     | NULL    |       |
| MANAGER_NO | int | YES  |     | NULL    |       |
| SAL   | decimal(10,2) | YES  |     | NULL    |       |
| COMMISSION | decimal(10,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> DELETE FROM Employee WHERE EMPNO = 105;
Query OK, 1 row affected (0.01 sec)

mysql> desc Employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int  | YES  |     | NULL    |       |
| ENAME | varchar(50) | YES  |     | NULL    |       |
| JOB   | varchar(50) | YES  |     | NULL    |       |
| MGR   | int  | YES  |     | NULL    |       |
| SAL   | decimal(10,2) | YES  |     | NULL    |       |
| COMMISSION | decimal(10,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```



```

student@student1: ~
| SAL | decimal(10,2) | YES | | NULL | |
| COMMISSION | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> create database emp3;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| College |
| Library |
| company |
| emp |
| emp2 |
| emp3 |
| employee |
| information_schema |
| library |
| mysql |
| performance_schema |
| sys |
+-----+
12 rows in set (0.00 sec)

mysql> use emp3;
Database changed

mysql> CREATE TABLE Employee (
-> E_id INT PRIMARY KEY,
-> E_name VARCHAR(50),
-> Age INT,
-> Salary DECIMAL(10, 2)
-> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE Employee (
-> E_id INT PRIMARY KEY,
-> E_name VARCHAR(50),
-> Age INT,
-> Salary DECIMAL(10, 2)
-> );
ERROR 1050 (42S01): Table 'Employee' already exists

mysql> CREATE TABLE Employee3 (
-> E_id INT PRIMARY KEY,
-> E_name VARCHAR(50),
-> Age INT,
-> Salary DECIMAL(10, 2)
-> );
Query OK, 0 rows affected (0.04 sec)

mysql> desc Employee3;

```

```

student@student1: ~
1 row in set (0.00 sec)

mysql> desc Employee3;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| E_id | int | NO | PRI | NULL | |
| E_name | varchar(50) | YES | | NULL | |
| Age | int | YES | | NULL | |
| Salary | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT COUNT(*) AS num_employees_high_salary
-> FROM Employee
-> WHERE Salary > 8000;
+-----+
| num_employees_high_salary |
+-----+
| 0 |
+-----+
1 row in set (0.01 sec)

mysql> INSERT INTO Employee3 (E_id, E_name, Age, Salary)
-> VALUES
-> (101, 'Anu', 22, 9000),
-> (102, 'Shane', 29, 8000),
-> (103, 'Rohan', 34, 6000),
-> (104, 'Scott', 44, 10000),
-> (105, 'Tiger', 35, 8000),
-> (106, 'Alex', 27, 7000),
-> (107, 'Abhi', 29, 8000);
Query OK, 7 rows affected (0.00 sec)
Records: 7 Duplicates: 0 Warnings: 0

mysql> desc Employee3;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| E_id | int | NO | PRI | NULL | |
| E_name | varchar(50) | YES | | NULL | |
| Age | int | YES | | NULL | |
| Salary | decimal(10,2) | YES | | NULL | |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from Employee3;
+-----+-----+-----+-----+
| E_id | E_name | Age | Salary |
+-----+-----+-----+-----+
| 101 | Anu | 22 | 9000.00 |
| 102 | Shane | 29 | 8000.00 |
| 103 | Rohan | 34 | 6000.00 |

```



```
student@student1: ~
+-----+
1 row in set (0.01 sec)

mysql> SELECT MAX(Age) AS max_age
-> FROM Employee3;
+-----+
| max_age |
+-----+
|      44 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT MIN(Age) AS min_age
-> FROM Employee3;
+-----+
| min_age |
+-----+
|      22 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT E_name, Salary
-> FROM Employee3
-> ORDER BY Salary ASC;
+-----+-----+
| E_name | Salary |
+-----+-----+
| Rohan  | 6000.00 |
| Alex   | 7000.00 |
| Shane  | 8000.00 |
| Tiger  | 8000.00 |
| Abhi   | 8000.00 |
| Anu    | 9000.00 |
| Scott  | 10000.00 |
+-----+-----+
7 rows in set (0.00 sec)

mysql> SELECT Salary, COUNT(*) AS num_employees
-> FROM Employee3
-> GROUP BY Salary;
+-----+-----+
| Salary | num_employees |
+-----+-----+
| 9000.00 | 1 |
| 8000.00 | 3 |
| 6000.00 | 1 |
| 10000.00 | 1 |
| 7000.00 | 1 |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

**4) Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary.**

**CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)**

```
show databases;
```

```
create database dbms;
```

```
use dbms;
```

```
mysql> CREATE TABLE customers (  
->     ID INT AUTO_INCREMENT PRIMARY KEY,  
->     NAME VARCHAR(100),  
->     AGE INT,  
->     ADDRESS VARCHAR(255),  
->     SALARY DECIMAL(10, 2)  
-> );
```

```
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> INSERT INTO customers (NAME, AGE, ADDRESS, SALARY) VALUES  
->     ('John Doe', 30, '123 Main St, Anytown, USA',  
50000.00),  
->     ('Jane Smith', 25, '456 Elm St, Othertown, USA',  
60000.00),  
->     ('Alice Johnson', 35, '789 Oak St, Anycity, USA',  
70000.00),  
->     ('Bob Brown', 40, '321 Pine St, Anothercity, USA',  
55000.00),  
->     ('Emily Davis', 28, '654 Maple St, Somewhere, USA',  
65000.00),  
->     ('Michael Wilson', 45, '987 Cedar St, Nowhere, USA',  
75000.00),  
->     ('Sarah Miller', 32, '159 Birch St, Anyplace, USA',  
62000.00),  
->     ('David Jones', 38, '852 Walnut St, Elsewhere, USA',  
68000.00),  
->     ('Jennifer Taylor', 27, '753 Cherry St, Elsewhere,  
USA', 59000.00),  
->     ('William Anderson', 33, '369 Oak St, Somewhere,  
USA', 71000.00);
```

```
Query OK, 10 rows affected (0.01 sec)
```

```
Records: 10  Duplicates: 0  Warnings: 0
```

```
mysql> select *from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	John Doe	30	123 Main St, Anytown, USA	50000.00
2	Jane Smith	25	456 Elm St, Othertown, USA	60000.00
3	Alice Johnson	35	789 Oak St, Anycity, USA	70000.00
4	Bob Brown	40	321 Pine St, Anothercity, USA	55000.00
5	Emily Davis	28	654 Maple St, Somewhere, USA	65000.00
6	Michael Wilson	45	987 Cedar St, Nowhere, USA	75000.00
7	Sarah Miller	32	159 Birch St, Anyplace, USA	62000.00
8	David Jones	38	852 Walnut St, Elsewhere, USA	68000.00
9	Jennifer Taylor	27	753 Cherry St, Elsewhere, USA	59000.00
10	William Anderson	33	369 Oak St, Somewhere, USA	71000.00

10 rows in set (0.00 sec)

```
mysql> DELIMITER //
```

```
mysql>
```

```
mysql> CREATE TRIGGER display_salary_changes_insert
-> BEFORE INSERT ON customers
-> FOR EACH ROW
-> BEGIN
->     INSERT INTO salary_changes_log (customer_id,
old_salary, new_salary, change_time)
->     VALUES (NEW.ID, NULL, NEW.salary, NOW());
-> END//
```

Query OK, 0 rows affected (0.01 sec)

```
mysql>
```

```
mysql> CREATE TRIGGER display_salary_changes_update
-> BEFORE UPDATE ON customers
-> FOR EACH ROW
-> BEGIN
->     INSERT INTO salary_changes_log (customer_id,
old_salary, new_salary, change_time)
->     VALUES (NEW.ID, OLD.salary, NEW.salary, NOW());
-> END//
```

Query OK, 0 rows affected (0.01 sec)

```
mysql>
```

```
mysql> CREATE TRIGGER display_salary_changes_delete
-> BEFORE DELETE ON customers
-> FOR EACH ROW
-> BEGIN
```

```

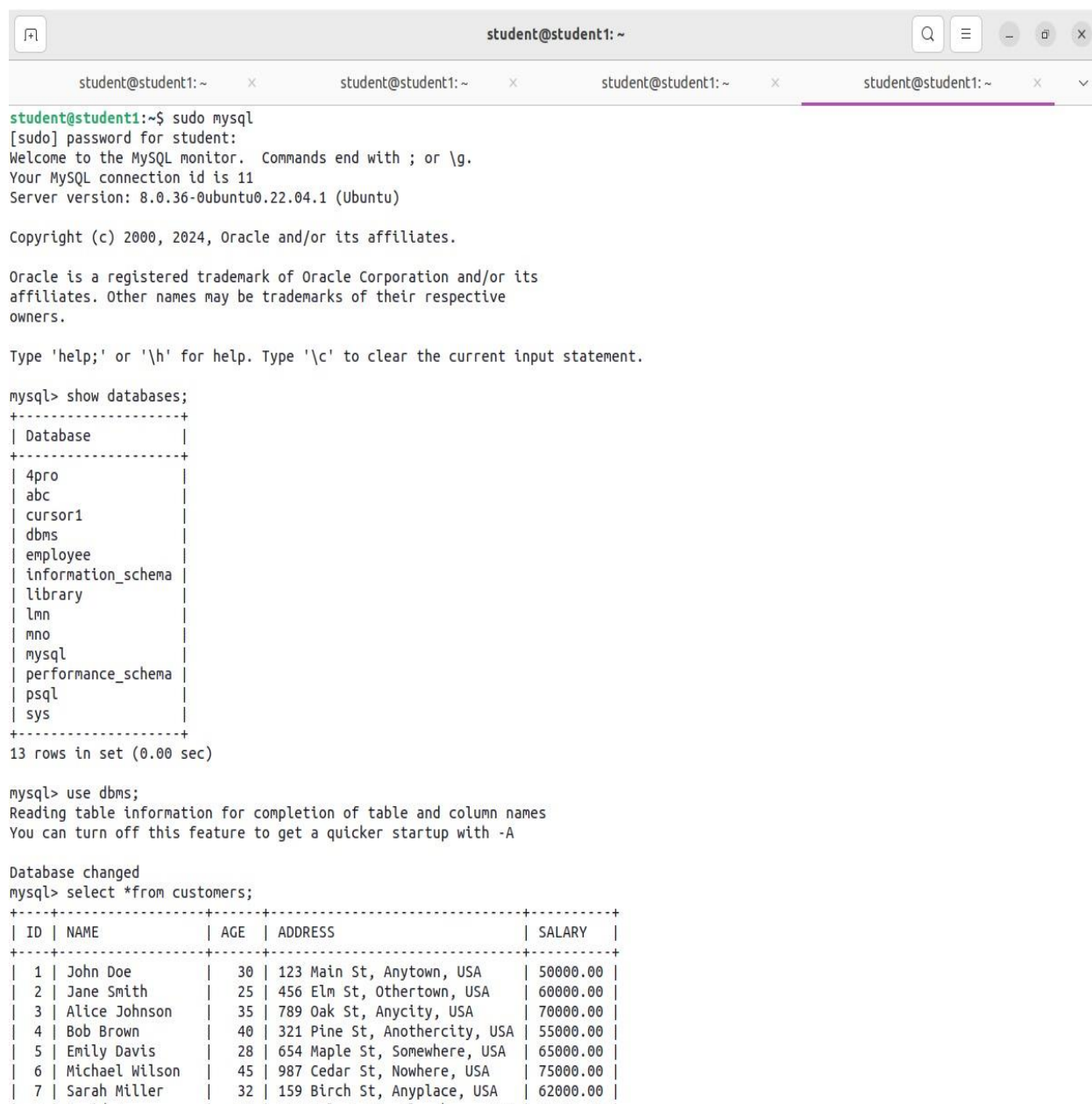
->      INSERT INTO salary_changes_log (customer_id,
old_salary, new_salary, change_time)
->      VALUES (OLD.ID, OLD.salary, NULL, NOW());
-> END//

```

Query OK, 0 rows affected (0.01 sec)

```
mysql> DELIMITER ;
```

```
mysql> SHOW TRIGGERS;
```



```

student@student1:~$ sudo mysql
[sudo] password for student:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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 |
+-----+
| 4pro     |
| abc      |
| cursor1  |
| dbms     |
| employee |
| information_schema |
| library  |
| lmn      |
| mno      |
| mysql    |
| performance_schema |
| psql     |
| sys      |
+-----+
13 rows in set (0.00 sec)

mysql> use dbms;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from customers;
+----+-----+-----+-----+-----+
| ID | NAME          | AGE | ADDRESS                      | SALARY |
+----+-----+-----+-----+-----+
| 1  | John Doe      | 30  | 123 Main St, Anytown, USA   | 50000.00 |
| 2  | Jane Smith    | 25  | 456 Elm St, Othertown, USA  | 60000.00 |
| 3  | Alice Johnson | 35  | 789 Oak St, Anycity, USA    | 70000.00 |
| 4  | Bob Brown     | 40  | 321 Pine St, Anothercity, USA | 55000.00 |
| 5  | Emily Davis   | 28  | 654 Maple St, Somewhere, USA | 65000.00 |
| 6  | Michael Wilson | 45  | 987 Cedar St, Nowhere, USA  | 75000.00 |
| 7  | Sarah Miller  | 32  | 159 Birch St, Anyplace, USA | 62000.00 |
+----+-----+-----+-----+-----+

```

1



```

mysql>
mysql> DELIMITER ;
mysql> SHOW TRIGGERS;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Trigger | Event | Table | Statement | Timing | Created | Definer | sql_mode | collation_connection | Database Collation |
+-----+-----+-----+-----+-----+-----+-----+-----+
| display_salary_changes_insert | INSERT | customers | BEGIN
INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
VALUES (NEW.ID, NULL, NEW.salary, NOW());
END | BEFORE | 2024-05-01 10:18:45.92 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION
_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_0900_ai_ci | utf8mb4_0900_ai_ci |
| customers_insert_trigger | INSERT | customers | BEGIN
INSERT INTO customers_log (action) VALUES (CONCAT('New record inserted for customer: ', NEW.NAME));
END | AFTER | 2024-05-01 09:49:51.18 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_
ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_0900_ai_ci | utf8mb4
_0900_ai_ci |
| display_salary_changes_update | UPDATE | customers | BEGIN
INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
VALUES (NEW.ID, OLD.salary, NEW.salary, NOW());
END | BEFORE | 2024-05-01 10:18:45.93 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZE
RO,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_0900_ai_ci | utf8mb4_0900_ai_ci |
| display_salary_changes_delete | DELETE | customers | BEGIN
INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
VALUES (OLD.ID, OLD.salary, NULL, NOW());
END | BEFORE | 2024-05-01 10:18:45.94 | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION
_BY_ZERO,NO_ENGINE_SUBSTITUTION | root@localhost | utf8mb4 | utf8mb4_0900_ai_ci | utf8mb4_0900_ai_ci |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
mysql> CREATE TRIGGER display_salary_changes_insert
-> -> BEFORE INSERT ON customers
-> -> FOR EACH ROW
-> -> BEGIN
-> -> INSERT INTO salary_changes_log (customer_id, old_salary, new_salary, change_time)
-> -> VALUES (11, NULL, 1000, NOW());
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the rig

```

**5) Create cursor for Employee table & extract the values from the table. Declare the variables, Open the cursor & extract the values from the cursor. Close the cursor.**

**Employee(E\_id, E\_name, Age, Salary).**

CURSOR

```
mysql> create database employee;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> use employee;
Database changed
```

```
mysql> CREATE TABLE employee (
->     E_id INT AUTO_INCREMENT PRIMARY KEY,
->     E_name VARCHAR(100),
->     Age INT,
->     Salary DECIMAL(10, 2)
-> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> INSERT INTO employee (E_name, Age, Salary) VALUES
-> ('John Doe', 30, 50000.00),
-> ('Alice Smith', 25, 60000.00),
-> ('Bob Johnson', 35, 70000.00);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM employee;
+-----+-----+-----+-----+
| E_id | E_name      | Age | Salary    |
+-----+-----+-----+-----+
| 1    | John Doe    | 30  | 50000.00  |
| 2    | Alice Smith | 25  | 60000.00  |
| 3    | Bob Johnson | 35  | 70000.00  |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> -- Create a stored procedure to define the cursor
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE employee_cursor_proc()
-> BEGIN
->     -- Declare variables to store values fetched by the
cursor
->     DECLARE done BOOLEAN DEFAULT FALSE;
->     DECLARE emp_id INT;
->     DECLARE emp_name VARCHAR(100);
```

```
-> DECLARE emp_age INT;
-> DECLARE emp_salary DECIMAL(10, 2);
->
-> -- Declare the cursor
-> DECLARE emp_cursor CURSOR FOR
->     SELECT E_id, E_name, Age, Salary FROM employee;
->
-> -- Declare handler for when no more rows are found
-> DECLARE CONTINUE HANDLER FOR NOT FOUND SET done =
TRUE;
->
-> -- Open the cursor
-> OPEN emp_cursor;
->
-> -- Loop through the result set
-> emp_loop: LOOP
->     -- Fetch data from the cursor into variables
->     FETCH emp_cursor INTO emp_id, emp_name, emp_age,
emp_salary;
->
->     -- Check if no more rows are found
->     IF done THEN
->         LEAVE emp_loop;
->     END IF;
->
->     -- Do something with the fetched data, for
example, print it
->     SELECT CONCAT('Employee ID: ', emp_id, ', Name:
', emp_name, ', Age: ', emp_age, ', Salary: ', emp_salary) AS
EmployeeInfo;
->
->     END LOOP;
->
-> -- Close the cursor
-> CLOSE emp_cursor;
->
-> END//
Query OK, 0 rows affected (0.02 sec)
```

```
mysql>
mysql> DELIMITER ;
mysql> CALL employee_cursor_proc();
```



```
+-----+
| EmployeeInfo |
+-----+
| Employee ID: 1, Name: John Doe, Age: 30, Salary: 50000.00 |
+-----+
1 row in set (0.00 sec)
```

```
+-----+
| EmployeeInfo |
+-----+
| Employee ID: 2, Name: Alice Smith, Age: 25, Salary: 60000.00 |
+-----+
1 row in set (0.00 sec)
```

```
+-----+
| EmployeeInfo |
+-----+
| Employee ID: 3, Name: Bob Johnson, Age: 35, Salary: 70000.00 |
+-----+
1 row in set (0.00 sec)
```

Query OK, 0 rows affected (0.00 sec)

```
student@student1: ~
student@student1:~$ sudo mysql
[sudo] password for student:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.36-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2024, 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> create database cursor;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'cursor' at line 1
mysql> create database employee;
Query OK, 1 row affected (0.01 sec)

mysql> use employee;
Database changed
mysql> CREATE TABLE employee (
  ->   E_id INT AUTO_INCREMENT PRIMARY KEY,
  ->   E_name VARCHAR(100),
  ->   Age INT,
  ->   Salary DECIMAL(10, 2)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO employee (E_name, Age, Salary) VALUES
  -> ('John Doe', 30, 50000.00),
  -> ('Alice Smith', 25, 60000.00),
  -> ('Bob Johnson', 35, 70000.00);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM employee;
+-----+-----+-----+-----+
| E_id | E_name | Age | Salary |
+-----+-----+-----+-----+
| 1    | John Doe | 30 | 50000.00 |
| 2    | Alice Smith | 25 | 60000.00 |
| 3    | Bob Johnson | 35 | 70000.00 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> -- Create a stored procedure to define the cursor
mysql> DELIMITER //
mysql> CREATE PROCEDURE employee_cursor_proc()
  -> BEGIN
  ->   -- Declare variables to store values fetched by the cursor
  ->
  ->   -- Open the cursor
  ->   OPEN emp_cursor;
  ->
  ->   -- Loop through the result set
  ->   emp_loop: LOOP
  ->     -- Fetch data from the cursor into variables
  ->     FETCH emp_cursor INTO emp_id, emp_name, emp_age, emp_salary;
  ->
  ->     -- Check if no more rows are found
  ->     IF done THEN
  ->       LEAVE emp_loop;
  ->     END IF;
  ->
  ->     -- Do something with the fetched data, for example, print it
  ->     SELECT CONCAT('Employee ID: ', emp_id, ', Name: ', emp_name, ', Age: ', emp_age, ', Salary: ', emp_salary) AS EmployeeInfo;
  ->
  ->   END LOOP;
  ->
  ->   -- Close the cursor
  ->   CLOSE emp_cursor;
  ->
  -> END//
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL employee_cursor_proc();
+-----+
| EmployeeInfo |
+-----+
| Employee ID: 1, Name: John Doe, Age: 30, Salary: 50000.00 |
+-----+
1 row in set (0.00 sec)

+-----+
| EmployeeInfo |
+-----+
| Employee ID: 2, Name: Alice Smith, Age: 25, Salary: 60000.00 |
+-----+
1 row in set (0.00 sec)

+-----+
| EmployeeInfo |
+-----+
| Employee ID: 3, Name: Bob Johnson, Age: 35, Salary: 70000.00 |
+-----+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql>
```

**6) Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N\_RollCall with the data available in the table O\_RollCall. If the data in the first table already exist in the second table then that data should be skipped.**

```
mysql> CREATE TABLE Employee (  
->     E_id INT AUTO_INCREMENT PRIMARY KEY,  
->     E_name VARCHAR(100),  
->     Age INT,  
->     Salary DECIMAL(10, 2)  
-> );  
Query OK, 0 rows affected (0.04 sec)  
  
mysql> INSERT INTO Employee (E_name, Age, Salary) VALUES  
-> ('John Doe', 30, 50000.00),  
-> ('Jane Smith', 28, 60000.00),  
-> ('Michael Johnson', 35, 75000.00);  
Query OK, 3 rows affected (0.01 sec)  
Records: 3  Duplicates: 0  Warnings: 0  
  
mysql> SELECT * FROM Employee;  
+-----+-----+-----+-----+  
| E_id | E_name          | Age | Salary  |  
+-----+-----+-----+-----+  
|    1 | John Doe        |   30 | 50000.00 |  
|    2 | Jane Smith      |   28 | 60000.00 |  
|    3 | Michael Johnson |   35 | 75000.00 |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> CREATE PROCEDURE merge_data()  
-> BEGIN  
->     DECLARE v_n_rollcall_id INT;  
->     DECLARE v_n_rollcall_data VARCHAR(255);  
->     DECLARE v_exists INT;  
->  
->     DECLARE done BOOLEAN DEFAULT FALSE;  
->  
->     -- Declare cursor for N_RollCall table  
->     DECLARE n_rollcall_cursor CURSOR FOR  
->         SELECT ID, Data  
->         FROM N_RollCall;  
->  
->     -- Declare continue handler for cursor  
->     DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  
->  
->     -- Open the cursor  
->     OPEN n_rollcall_cursor;  
->  
->     -- Loop through the cursor and merge data into O_RollCall  
->     n_rollcall_loop: LOOP
```

```
->          FETCH n_rollcall_cursor INTO v_n_rollcall_id,
v_n_rollcall_data;
->
->          IF done THEN
->              LEAVE n_rollcall_loop;
->          END IF;
->
->          -- Check if data exists in O_RollCall
->          SELECT COUNT(*)
->          INTO v_exists
->          FROM O_RollCall
->          WHERE Data = v_n_rollcall_data;
->
->          IF v_exists = 0 THEN
->              -- Insert data into O_RollCall
->              INSERT INTO O_RollCall (ID, Data)
->              VALUES (v_n_rollcall_id, v_n_rollcall_data);
->          ELSE
->              -- Data already exists, do nothing
->              SELECT 'Data with ID ', v_n_rollcall_id, ' already exists
in O_RollCall. Skipping...' AS Message;
->          END IF;
->      END LOOP;
->
->      -- Close the cursor
->      CLOSE n_rollcall_cursor;
-> END //
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> DELIMITER //
mysql>
mysql> CREATE PROCEDURE employee_cursor_proc()
-> BEGIN
->   -- Declare variables to store values fetched by the cursor
->   DECLARE done BOOLEAN DEFAULT FALSE;
->   DECLARE emp_id INT;
->   DECLARE emp_name VARCHAR(100);
->   DECLARE emp_age INT;
->   DECLARE emp_salary DECIMAL(10, 2);
->
->   -- Declare the cursor
->   DECLARE emp_cursor CURSOR FOR
->     SELECT E_id, E_name, Age, Salary FROM employee;
->
->   -- Declare handler for when no more rows are found
->   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
->
->   -- Open the cursor
->   OPEN emp_cursor;
->
->   -- Loop through the result set
->   emp_loop: LOOP
->     -- Fetch data from the cursor into variables
->     FETCH emp_cursor INTO emp_id, emp_name, emp_age, emp_salary;
->
->     -- Check if no more rows are found
->     IF done THEN
->       LEAVE emp_loop;
->     END IF;
->
->     -- Do something with the fetched data, for example, print it
->     SELECT CONCAT('Employee ID: ', emp_id, ', Name: ', emp_name, ', Age: ', emp_age, ', Salary: ', emp_salary) AS Employee
eInfo;
->
->     END LOOP;
->
->   -- Close the cursor
->   CLOSE emp_cursor;
->
-> END//
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL employee_cursor_proc();
+-----+
| EmployeeInfo |
+-----+
| Employee ID: 1, Name: John Doe, Age: 30, Salary: 50000.00 |
+-----+
1 row in set (0.00 sec)
```

**7) Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations**

MONGODB

How to run mongoDB in mysql ubuntu

To run MongoDB on Ubuntu, you don't need to use MySQL. MongoDB is a separate NoSQL database and doesn't require MySQL to be installed or running. However, if you want to run both MongoDB and MySQL on the same Ubuntu system, you can do so without conflicts.

Here's how you can install and run MongoDB on Ubuntu alongside MySQL:

1. Install MongoDB:
  - i) Update Package Index:  
`sudo apt update`
  - ii) Install MongoDB:  
`sudo apt install -y mongodb`
  - iii) Start MongoDB Service:  
`sudo systemctl start mongodb`
  - iv) Enable MongoDB Service to Start on Boot:  
`sudo systemctl enable mongodb`
  - v) Verify MongoDB Installation:  
`sudo systemctl status mongodb`
  - vi) Access MongoDB Shell:  
`mongo`

**I) Steps to Execute Program**

1. Create a collection named 'comments'
2. Insert One Row

```
db.comments.insert({  
  'name': 'Harry',  
  'lang': 'JavaScript',  
  'member_since': 5  
})
```

```
}}
```

### 3. Insert many Rows

```
db.comments.insertMany([  
  {'name': 'Harry',  
   'lang': 'JavaScript',  
   'member_since': 5  
  },  
  {'name': 'Rohan',  
   'lang': 'Python',  
   'member_since': 3  
  },  
  {'name': 'Lovish',  
   'lang': 'Java',  
   'member_since': 4  
}])
```

### 4. Search in a MongoDB Database

```
db.comments.find({'lang': 'Python'})
```

### 5. Update a row

```
db.comments.updateOne({'name': 'Shubham'},  
{'$set': {'name': 'Harry',  
          'lang': 'JavaScript',  
          'member_since': 51  
}}, {'upsert': true})
```

### 6. Mongodb Rename Operator

```
db.comments.update({'name': 'Rohan'},  
{'$rename': {  
  member_since: 'member'  
}})
```

### 7 .Delete Row

```
db.comments.remove({'name': 'Harry'})
```

## MONGODB

How to run mongoDB in mysql ubuntu

To run MongoDB on Ubuntu, you don't need to use MySQL. MongoDB is a separate NoSQL database and doesn't require MySQL to be installed or running. However, if you want to run both MongoDB and MySQL on the same Ubuntu system, you can do so without conflicts.

Here's how you can install and run MongoDB on Ubuntu alongside MySQL:

1. Install MongoDB:
  - i) Update Package Index:  
`sudo apt update`
  - ii) Install MongoDB:  
`sudo apt install -y mongodb`
  - iii) Start MongoDB Service:  
`sudo systemctl start mongodb`
  - iv) Enable MongoDB Service to Start on Boot:  
`sudo systemctl enable mongodb`
  - v) Verify MongoDB Installation:  
`sudo systemctl status mongodb`
  - vi) Access MongoDB Shell:  
`mongo`

#### I) Steps to Execute Program

1. Create a collection named 'comments'
2. Insert One Row

```
db.comments.insert({  
  'name': 'Harry',  
  'lang': 'JavaScript',  
  'member_since': 5  
})
```

#### 3. Insert many Rows

```
db.comments.insertMany([  
  {  
    'name': 'Harry',  
    'lang': 'JavaScript',  
    'member_since': 5  
  },  
  {  
    'name': 'Harry',  
    'lang': 'JavaScript',  
    'member_since': 5  
  },  
  {  
    'name': 'Harry',  
    'lang': 'JavaScript',  
    'member_since': 5  
  }  
])
```



### 3. Insert many Rows

```
db.comments.insertMany([  
  {'name': 'Harry',  
    'lang': 'JavaScript',  
    'member_since': 5  
  },  
  {'name': 'Rohan',  
    'lang': 'Python',  
    'member_since': 3  
  },  
  {'name': 'Lovish',  
    'lang': 'Java',  
    'member_since': 4  
  }  
])
```

### 4. Search in a MongoDB Database

```
db.comments.find({'lang': 'Python'})
```

### 5. Update a row

```
db.comments.updateOne({'name': 'Shubham'},  
  {'$set': {'name': 'Harry',  
    'lang': 'JavaScript',  
    'member_since': 51  
  }}, {'upsert': true})
```

### 6. Mongodb Rename Operator

```
db.comments.update({'name': 'Rohan'},  
  {'$rename': {  
    member_since: 'member'  
  }})
```

### 7 .Delete Row

```
db.comments.remove({'name': 'Harry'})
```

## VIVA QUESTIONS WITH ANSWERS

### 1) What is Database?

An organized collection of structured information or data.

### 2) What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

### 3) What is a Database system?

The database and DBMS software together is called as Database system.

### 4) Advantages of DBMS?

- Redundancy is controlled.
- Unauthorized access is restricted.
- Providing multiple user interfaces.
- Enforcing integrity constraints.
- Providing backup and recovery.

### 5) Disadvantage in File Processing System?

- Data redundancy & inconsistency.
- Difficult in accessing data.
- Data Isolation.
- Data Integrity.
- Concurrent access is not possible.
- Security Problems.

### 6) Describe the three levels of data Abstraction?

Three levels of abstraction:

**Physical level:** The lowest level of abstraction describes how data are stored.

**Logical level:** The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

**View level:** The highest level of abstraction describes only part of entire database.

### 7) Define the "integrity rules"

There are two Integrity rules.

**Entity Integrity:** States that Primary key cannot have NULL value

**Referential Integrity:** States that Foreign Key can be either a NULL value or should be Primary Key value of other relation.

### 8) What is extension and intension?

**Extension:** It is the number of tuples present in a table at any instance. This is time dependent.

**Intension:** It is a constant value that gives the name, structure of table and the constraints laid on it.

### 9) What is Data Independence?

**Data independence** means that “The application is independent of the storage structure and access strategy of data”. In other words, the ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

### 10) Two types of Data Independence?

**Physical Data Independence:** Modification in physical level should not affect the logical level.

**Logical Data Independence:** Modification in logical level should affect the view level.

### 11) What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views..

### 12) What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

### 13) What is E-R Model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

### 14) What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are

grouped together into classes.

**15) What is an Entity?**

It is a 'thing' in the real world with an independent existence.

**16) What is an Entity type?**

It is a collection (set) of entities that have same attributes.

**17) What is an Entity set?**

It is a collection of all entities of particular entity type in the database.

**18) What is an Extension of entity type?**

The collections of entities of a particular entity type are grouped together into an entity set.

**19) What is Weak Entity set?**

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

**20) What is an attribute?**

It is a particular property, which describes the entity.

**21) What is a Relation?**

A relation is defined as a set of tuples.

**22) What is degree of a Relation?**

It is the number of attribute of its relation schema.

**23) What is Relationship?**

It is an association among two or more entities.

**24) What is Relationship set?**

The collection (or set) of similar relationships.

**25) What is Relationship type?**

Relationship type defines a set of associations or a relationship set among a given set of entity types.

**26) What is degree of Relationship type?**

It is the number of entity type participating.

**27) What is DDL (Data Definition Language)?**

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

**28) What is VDL (View Definition Language)?**

It specifies user views and their mappings to the conceptual schema.

**29) What is DML (Data Manipulation Language)?**

This language that enable user to access or manipulate data as organized by appropriate data model.

**30) What is DML Compiler?**

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

**31) What is Query evaluation engine?**

It executes low-level instruction generated by compiler.

**32) What is DDL Interpreter?**

It interprets DDL statements and records them in tables containing metadata.

**33) What is a query?**

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

**34) What do you mean by Correlated subquery?**

A correlated sub query can be easily identified if it contains any references to the parent sub query columns in its WHERE clause. Columns from the sub query cannot be referenced anywhere else in the parent query.

**35) Are the resulting relations of PRODUCT and JOIN operation the same?**

**No.**

**PRODUCT:** Concatenation of every row in one relation with every row in another.

**JOIN:** Concatenation of rows from one relation and related rows from another.

**36) What is database Trigger?**

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can be defined to execute once for the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database

procedures that are also written in PL/SQL.

**37) Define super key and give example to illustrate the super key?**

Set of one or more attributes taken collectively, allowing to identify uniquely an entity in the entity set. Eg1. {SSN} and {SSN, Cust\_name} of customer table are super keys.

{Branch name} and {Branch name, Branch city} of Branch table re super keys.

**38) What is Primary key?**

A key chosen to act as the means by which to identify tuples in a relation.

**39) What is foreign key?**

A foreign key of relation R is a set of its attributes intended to be used (by each tuple in R) for identifying/referring to a tuples in some relation S. (R is called the referencing relation and S the referenced relation.) For this to make sense, the set of attributes of R forming the foreign key should "correspond to" some superkey of S. Indeed, by definition we require this superkey to be the primary key of S.

**40) What is a Cursor?**

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

**41) What is Functional Dependency?**

A Functional dependency is a relationship between two sets of attributes in a database table.

**42) What is 1 NF (Normal Form)?**

The domain of attribute must include only atomic (simple, indivisible) values.

**43) What is Fully Functional dependency?**

It is based on concept of full functional dependency. A functional dependency  $X \rightarrow Y$  is fully functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

**44) What is 2NF?**

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

**45) What is 3NF?**

A relation schema R is in 3NF if it is in 2NF and for every FD  $X \rightarrow A$  either of the following is true X

is a Super-key of R.

**46) What is BCNF (Boyce-Codd Normal Form)?**

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD  $X \rightarrow A$ , X must be a candidate key.

**47) What is 4 NF?**

A relation schema R is said to be in 4NF if for every multivalued dependency  $X \twoheadrightarrow Y$  that holds over R, one of following is true X is subset or equal to (or)  $XY = R$ . X is a superkey.

**48) What is dependency preservation?**

Dependency Preservation Property enables us to enforce a constraint on the original relation from corresponding instances in the smaller relations.

**49) What is a SQL.**

SQL stands for Structured Query Language. It is a tool for organizing, managing and retrieving archived data from a computer database.

**50) What is a NoSQL**

An approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.