PROGRAM -1

AIM: Introduction to Structure Query Language/SQL and its Commands

1.Introduction:

SQL is a domain-specific language used for managing and manipulating relational databases. It allows users to define, manipulate, and control data stored in a relational database management system.

OR

SQL is a language to communicate with the database. SQL commands help you to store , process , analyze and manipulate databases.

2.Features of SQL:

1. **Data Definition Language (DDL):** SQL provides a set of commands to define and modify the structure of a database, including creating tables, modifying table structure, and dropping tables.
2. **Data Manipulation Language (DML):** SQL provides a set of commands to manipulate data within a database, including adding, modifying, and deleting data.
3. **Query Language:** SQL provides a rich set of commands for querying a database to retrieve data, including the ability to filter, sort, group, and join data from multiple tables.
4. **Transaction Control:** SQL supports transaction processing, which allows users to group a set of database operations into a single transaction that can be rolled back in case of failure.
5. **Data Integrity:** SQL includes features to enforce data integrity, such as the ability to specify constraints on the values that can be inserted or updated in a table, and to enforce referential integrity between tables
6. **User Access Control:** SQL provides mechanisms to control user access to a database, including the ability to grant and revoke privileges to perform certain operations on the database.
7. **Portability:** SQL is a standardized language, meaning that SQL code written for one database management system can be used on another system with minimal modification.

3.SQL Languages:

1.Data Definition Language (DDL): SQL provides a set of commands to define and modify the structure of a database, including creating tables, modifying table structure, and dropping tables.

>CREATE:

Creates a new table with specified columns and data types.

SYNTAX-

create table

tablename(attribute 1 datatype….attribute n datatype);

>ALTER:

Modifies the structure of an existing table, such as adding a new column.

SYNTAX-

alter table

tablename add(new column1 datatype.....new column x datatype);

>DROP:

Deletes an existing table and its data.

SYNTAX-

drop table tablename;

>TRUNCATE:

Removes all records from a table, but retains the table structure for future use.

>COMMENT:

Adds comments or annotations to a database object, providing additional information.

>RENAME:

Changes the name of a database object, like renaming a table or column.

SYNTAX-

rename old tablename to new tablename;

2.Data Manipulation Language (DML): SQL provides a set of commands to manipulate data within a database, including adding, modifying, and deleting data.

>SELECT:

Retrieves data from one or more tables based on specified conditions.

> INSERT:

Adds new records or rows to a table.

>UPDATE:

Modifies existing records in a table based on specified conditions.

>DELETE:

Removes records from a table based on specified conditions.

3.Data Control Language (DCL): A Data Control Language is a syntax similar to a computer programming language used to control access to data stored in a database (Authorization).

>GRANT:

Provides specific permissions to users on database objects.

>REVOKE:

Revokes previously granted permissions from users on database objects.

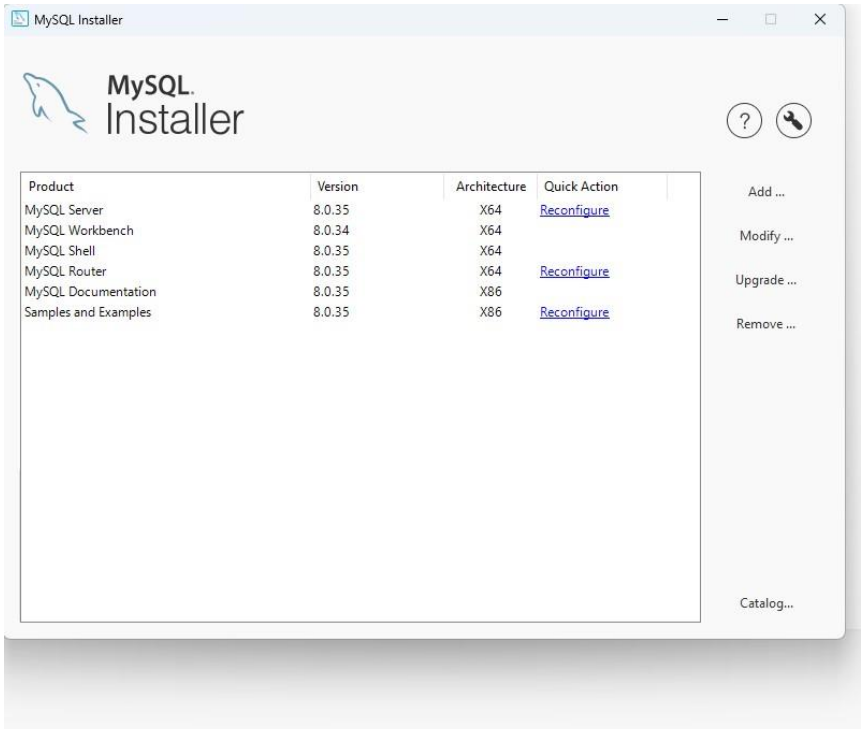
PROGRAM-2

AIM: Installing MySQL software for practicing DDL& DML Command

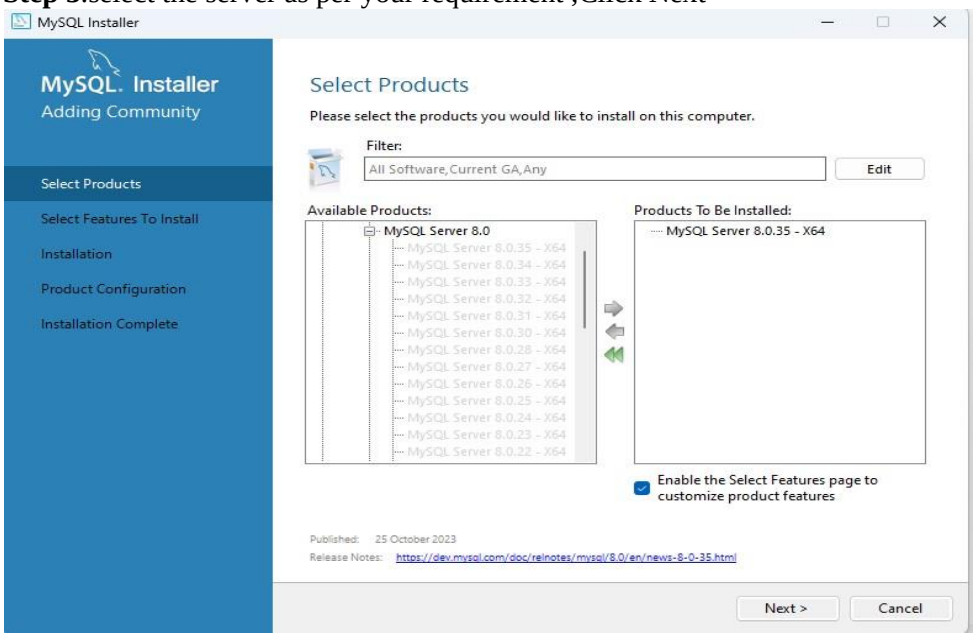
Step 1: We need to download the MySQL software from Google Chrome.

{https://dev.mysql.com/downloads/installer/}

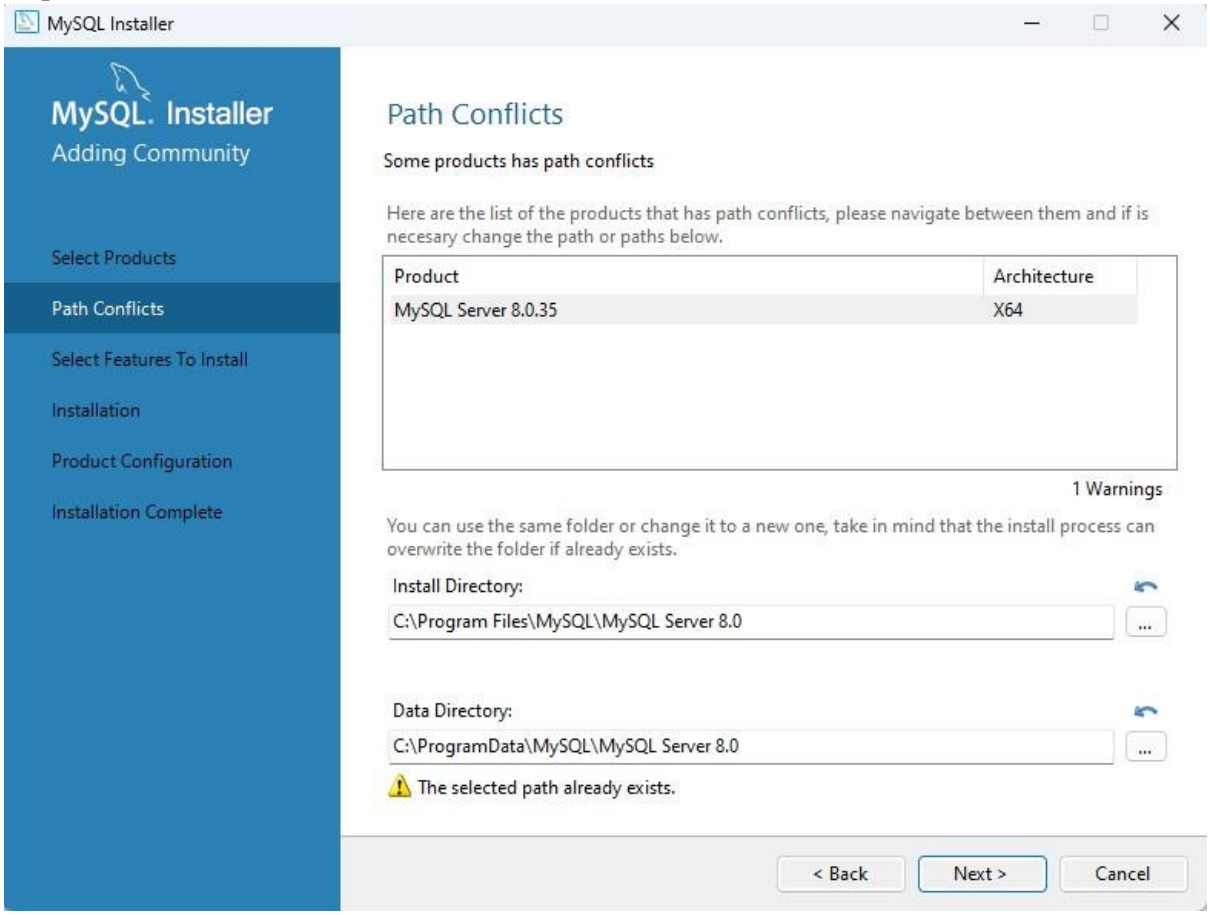
Step 2. Double click on the software to install it on our machine. Follow the instructions to complete the installation.Click on Add



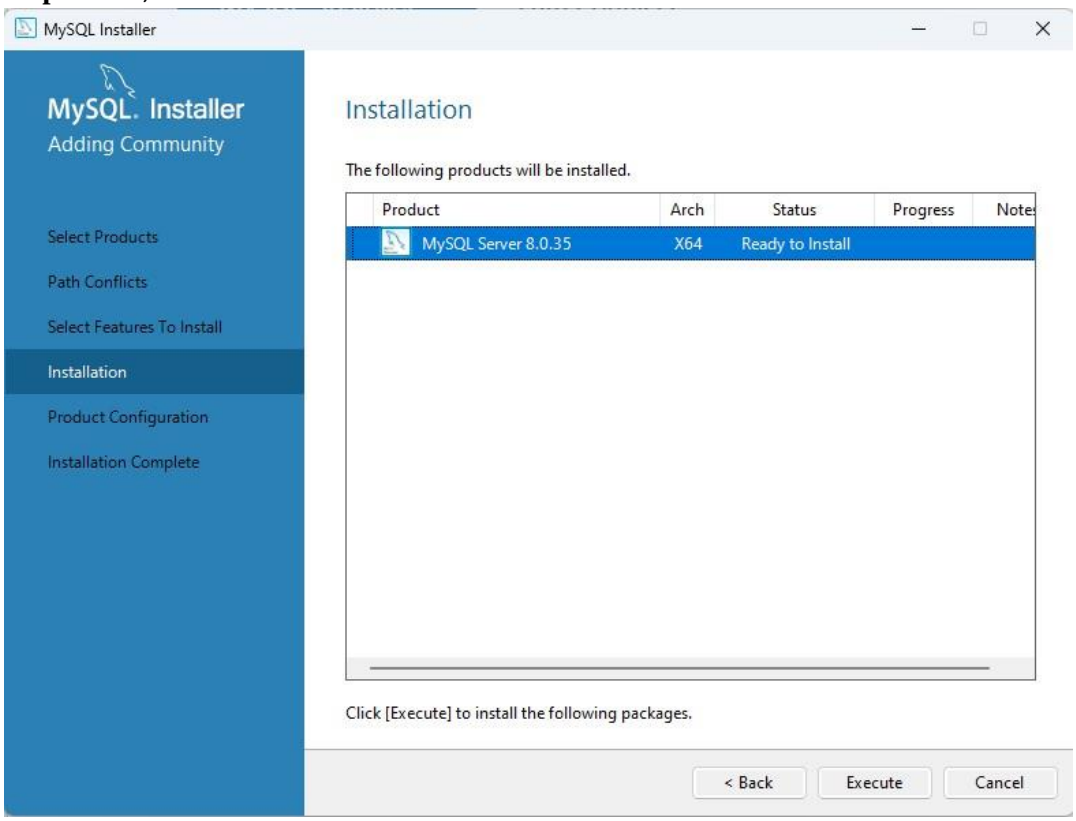
Step 3.select the server as per your requirement ,Click Next



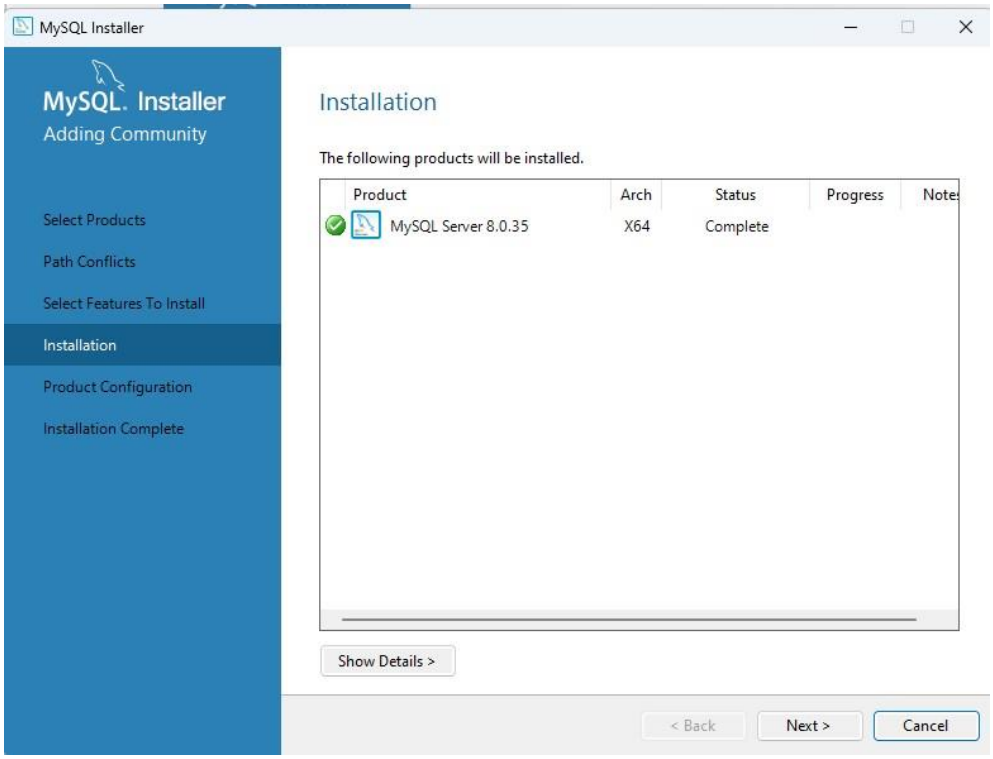
Step 4. Select the directory and click next.



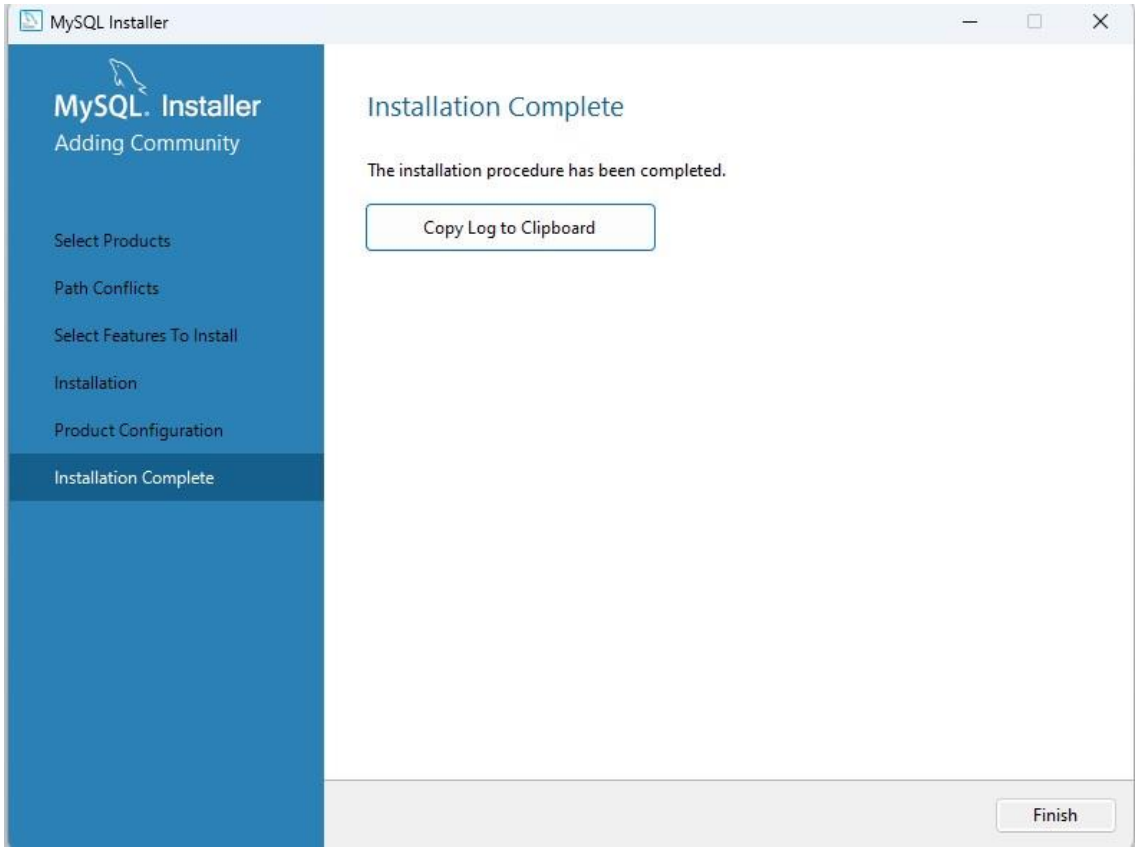
Step 5.Now,Press Execute for the installation.



Step 6. Now installation of MySQL 8.0.35 has been completed ,Click on next.



Step 7.Installation Complete.



PROGRAM-3

AIM: CREATE, ALTER, DROP, TRUNCATE commands and Constraints

Program:

It is used to communicate with database. DDL is used to:

- Create an object
- Alter the structure of an object
- To drop the object created.

CREATE:

Creates a new table in the database.

Syntax: CREATE TABLE table_name (

column1 datatype,

column2 datatype,

column3 datatype

PRIMARY KEY (one or more columns)

);

ALTER:

Modifies an existing table structure. It can add or delete columns.

Syntax to add a new column:

ALTER TABLE tablename

ADD column_name datatype;

Syntax to delete a column:

ALTER TABLE tablename

DROP COLUMN column_name;

DROP:

Removes an entire table from the database.

Syntax:

DROP TABLE tablename;

TRUNCATE:

Deletes all data from an existing table, keeping the table structure intact.

Syntax:

TRUNCATE TABLE tablename;

Queries:

1) **Task is to create a table and insert some values to it so the query is:**

```
CREATE TABLE Student_information(
  Student_Name VARCHAR(20),
  Student_Sem INT,
  Student_Ph INT
);
INSERT INTO Student_information (Student_Name, Student_Sem, Student_Ph)VALUES
  ('Priya', 3, 87632),
  ('Vikram', 5, 98123),
  ('Neha', 3, 56789),
  ('Rajesh', 6, 23456),
  ('Pooja', 4, 78901),
  ('Sandeep', 5, 12345);
```

Output:

Student_information

Student_Name	Student_Sem	Student_Ph
Priya	3	87632
Vikram	5	98123
Neha	3	56789
Rajesh	6	23456
Pooja	4	78901
Sandeep	5	12345

2)Alter command execution:

Query:

```
ALTER TABLE Student_information
ADD Student_Grade VARCHAR(2);
```

Output:

Student_information

Student_Name	Student_Sem	Student_Ph	Student_Grade
Priya	3	87632	
Vikram	5	98123	
Neha	3	56789	
Rajesh	6	23456	
Pooja	4	78901	
Sandeep	5	12345	

3) **Execution of TURNCATE**

Query:

TRUNCATE TABLE Student_information;
SELECT * FROM Student_information;

Output:

Student_information

Student_Name	Student_Sem	Student_Ph
empty		

4) **Execution of DROP Command:**

Query:

DROP TABLE Student_information;

Output:

SQL query successfully executed. However, the result set is empty.

PROGRAM-4

AIM:DML commands: Introduction about the INSERT, SELECT, UPDATE & DELETE command

1. INSERT:

Purpose: The INSERT command is used to add new records (rows) to a table.
Syntax:

INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

Example:
Query: INSERT INTO Sharda_University (customer_id, first_name, last_name, email)
VALUES
(1, 'Rohan', 'Rai', 'Rohan.rai@example.com'),
(2, 'Seema', 'Kaur', 'Seema.kaur@example.com');

Output:

Sharda_University

customer_id	first_name	last_name	email
1	Rohan	Rai	Rohan.rai@example.com
2	Seema	Kaur	Seema.kaur@example.com

2 SELECT:

Purpose: The SELECT command is used to retrieve data from one or more tables.

Syntax:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Query:

SELECT * FROM Sharda_University;

Output

customer_id	first_name	last_name	email
1	Rohan	Rai	Rohan.rai@example.com
2	Seema	Kaur	Seema.kaur@example.com

3 UPDATE:

Purpose: The UPDATE command is used to modify existing records in a table.

Syntax:UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Query:

UPDATE Sharda_University

SET email = 'kaur.seema@example.com'

WHERE first_name = 'Seema';

Output:

Sharda_University

customer_id	first_name	last_name	email
1	Rohan	Rai	Rohan.rai@example.com
2	Seema	Kaur	kaur.seema@example.com

DELETE:

Purpose: The DELETE command is used to remove records from a table.

Syntax:

DELETE FROM table_name

WHERE condition;

Query:

DELETE FROM Sharda_University

WHERE first_name = 'Seema';

Output:

Sharda_University

customer_id	first_name	last_name	email
1	Rohan	Rai	Rohan.rai@example.com

PROGRAM-5

AIM: To implement SQL Constraints(NOT NULL, UNIQUE, PRIMARY KEY, CHECK, DEFAULT)

1. **Not Null Constraint:**
   Ensures that a column cannot have a NULL value.
   Syntax:
   CREATE TABLE your_table (
   column1 datatype NOT NULL,
   column2 datatype,
   -- other columns
   );
   Query:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
INSERT INTO Persons (ID, LastName, FirstName, Age)
VALUES
    (1, 'Sharma', 'Akhil', 20),
    (2, 'Patil', 'Gopesh', 21);
```

Output:

Persons

ID	LastName	FirstName	Age
1	Sharma	Akhil	20
2	Patil	Gopesh	21

2. **UNIQUE Constraint:**
   Ensures that all values in a column are unique.

   Syntax: CREATE TABLE your_table (
   column1 datatype,
   column2 datatype,
   -- other columns
   CONSTRAINT constraint_name UNIQUE (column1, column2)
   );

   Query:
   CREATE TABLE Unique_Table (
   ID int NOT NULL UNIQUE,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255) NOT NULL,
   Age int
   );

INSERT INTO Unique_Table (ID, LastName, FirstName, Age)
VALUES
(1, 'Kumar', 'Rahul', 25),
(2, 'Gupta', 'Pooja', 30);

Output:

Unique_Table

ID	LastName	FirstName	Age
1	Kumar	Rahul	25
2	Gupta	Pooja	30

3. **PRIMARY KEY Constraint:**
Combines the NOT NULL and UNIQUE constraints. It ensures that a column (or a set of columns) has unique values and does not contain NULLs.

Syntax: CREATE TABLE YourTable (
Column1 datatype,
Column2 datatype,
-- other columns
PRIMARY KEY (Column1)
);
Query: CREATE TABLE primary_Table (
EmployID int PRIMARY KEY,
FirstName varchar(255) NOT NULL,
LastName varchar(255) NOT NULL,
JoiningYear int
);

INSERT INTO primary_Table (EmployID, FirstName, LastName, JoiningYear)
VALUES
(101, 'Amit', 'Kumar', 2018),
(102, 'Priya', 'Sharma', 2019),
(103, 'Raj', 'Patel', 2020);

Output:

Primary_Table

EmployID	FirstName	LastName	JoiningYear
101	Amit	Kumar	2018
102	Priya	Sharma	2019
103	Raj	Patel	2020

4. **CHECK Constraint:**
Ensures that the values in a column satisfy a specific condition.

Syntax:

```
CREATE TABLE YourTable (
   Column1 datatype,
   Column2 datatype,
   -- other columns
   CONSTRAINT constraint_name CHECK (condition)
);
```

Query:

```
CREATE TABLE Syntax_Table (
   EmployeeID int PRIMARY KEY,
   FirstName varchar(255) NOT NULL,
   LastName varchar(255) NOT NULL,
   Age int CHECK (Age >= 18)
);
INSERT INTO Syntax_Table (EmployeeID, FirstName, LastName, Age)
VALUES
   (101, 'Amit', 'Kumar', 25),
   (102, 'Priya', 'Sharma', 30),
   (103, 'Raj', 'Patel', 22);
```

Output:

CHECK_Table

EmployeeID	FirstName	LastName	Age
101	Amit	Kumar	25
102	Priya	Sharma	30
103	Raj	Patel	22

5. **DEFAULT Constraint:**
   Specifies a default value for a column.

   Syntax:

```
CREATE TABLE YourTable (
   Column1 datatype DEFAULT default_value,
   Column2 datatype DEFAULT default_value,
   -- other columns
);
```

   Query:

```
CREATE TABLE Default_Table (
   ID int PRIMARY KEY,
   Name varchar(255) DEFAULT 'AKHIL'
);
INSERT INTO Default_Table (ID, Name)
VALUES
   (1, 'AKHIL'),
   (2, 'GOPESH'),
   (3, 'RAJ');
```

Output:

Default_Table

ID	Name
1	AKHIL
2	GOPESH
3	RAJ

PROGRAM-6

AIM: Introduction of Arithmetic Operations and Implement Operation (+, -, *, %)

In Structured Query Language, the arithmetic operators are used to perform mathematical operations on the numerical values stored in the database tables.

Initial Table Query:

```
CREATE TABLE Staff
(
Staff_ID INT AUTO_INCREMENT PRIMARY KEY,
Staff_Name VARCHAR(100),
Staff_City VARCHAR(50),
Staff_Salary INT NOT NULL,
Staff_Bonus INT NOT NULL
);
INSERT INTO Staff (Staff_ID, Staff_Name, Staff_City, Staff_Salary, Staff_Bonus) VALUES
(101, 'Anuj', 'Ghaziabad', 25000, 2000),
(102, 'Tushar', 'Lucknow', 29000, 1000),
(103, 'Vivek', 'Kolkata', 35000, 2500),
(104, 'Shivam', 'Goa', 22000, 3000);
SELECT * FROM Staff;
```

Initial Table Output:

Staff_ID	Staff_Name	Staff_City	Staff_Salary	Staff_Bonus
101	Anuj	Ghaziabad	25000	2000
102	Tushar	Lucknow	29000	1000
103	Vivek	Kolkata	35000	2500
104	Shivam	Goa	22000	3000

1. **SQL Addition Operator (+) -** The SQL Addition Operator performs the addition on the numerical columns in the table.

   Syntax:

   ```
   SELECT Column_Name_1 Addition_Operator Column_Name2 FROM Table_Name;
   ```

Example:

SELECT Staff_Name, Staff_Salary, Staff_Bonus, (Staff_Salary + Staff_Bonus) AS Total_Earnings

FROM Staff;

Staff_Name	Staff_Salary	Staff_Bonus	Total_Earnings
Anuj	25000	2000	27000
Tushar	29000	1000	30000
Vivek	35000	2500	37500
Shivam	22000	3000	25000

2. **SQL Subtraction Operator (-) -** The SQL Subtraction Operator performs the Subtraction on the numerical columns in the table.

   Syntax:

   SELECT Column_Name_1 Subtraction_Operator Column_Name2 **FROM** Table_Name;

   Example:

   SELECT Staff_Name, Staff_Salary, Staff_Bonus, (Staff_Salary - Staff_Bonus) AS Salary_Minus_Bonus

   FROM Staff;

Staff_Name	Staff_Salary	Staff_Bonus	Salary_Minus_Bonus
Anuj	25000	2000	23000
Tushar	29000	1000	28000
Vivek	35000	2500	32500
Shivam	22000	3000	19000

3. **SQL Multiplication Operator (*) -** The SQL Addition Operator performs the Multiplication on the numerical columns in the table.

   Syntax:

   SELECT Column_Name_1 Multiplication_Operator Column_Name2 **FROM** Table_Name;

   Example:

   SELECT Staff_Name, Staff_Salary, Staff_Bonus, (Staff_Salary * Staff_Bonus) AS Salary_Times_Bonus

   FROM Staff;

Staff_Name	Staff_Salary	Staff_Bonus	Salary_Times_Bonus
Anuj	25000	2000	50000000
Tushar	29000	1000	29000000
Vivek	35000	2500	87500000
Shivam	22000	3000	66000000

4. **SQL Divison Operator (/) -** The SQL Addition Operator performs the divison on the numerical columns in the table.

Syntax:

SELECT Column_Name_1 Division_Operator Column_Name2 **FROM** Table_Name;

Example:

SELECT Staff_Name, Staff_Salary, Staff_Bonus, (Staff_Salary / Staff_Bonus) AS Salary_Divided_By_Bonus

FROM Staff;

Staff_Name	Staff_Salary	Staff_Bonus	Salary_Divided_By_Bonus
Anuj	25000	2000	12
Tushar	29000	1000	29
Vivek	35000	2500	14
Shivam	22000	3000	7

PROGRAM-7

AIM: Implement Aggregate Functions(Sum,Count,Avg,Max,Min) in Sql

Scalar Functions:

Scalar functions in SQL operate on individual values, transforming or manipulating data within a single row.

Concatenation:

Example: CONCAT(first_name, ' ', last_name) combines the first_name and last_name columns into a single full_name.

String Length:

Example: LENGTH(first_name) calculates the length of the first_name column.

Aggregate Functions:

Aggregate functions perform a calculation on a set of values and return a single value summarizing that set.

Sum:

Example: SUM(amount) calculates the total sum of the amount column.

Count:

Example: COUNT(*) counts the total number of rows in a table.

Average (Avg):

Example: AVG(salary) computes the average salary from the salary column.

Maximum (Max):

Example: MAX(price) retrieves the maximum value from the price column.

Minimum (Min):

Example: MIN(quantity) retrieves the minimum value from the quantity column.

Query:

```
CREATE TABLE staff_salary (
   employee_id INT PRIMARY KEY,
   name VARCHAR(50),
   salary INT
);
```

```
INSERT INTO staff_salary VALUES
    (1, 'Akhil', 60000),
    (2, 'Ayush', 55000),
    (3, 'Karan', 70000),
    (4, 'Niharika', 65000),
    (5, 'Rishita', 72000);
```

Output:

Staff_salary

employee_id	name	salary
1	Akhil	60000
2	Ayush	55000
3	Karan	70000
4	Niharika	65000
5	Rishita	72000

Implementation of Aggregate Functions:

1 Count Function()

Query:

```
SELECT COUNT(*) AS total_employees
FROM staff_salary;
```

Output

total_employees
5

2 Average Function()

Query:

```
SELECT AVG(salary) AS average_salary
FROM staff_salary;
```

Output:

average_salary
64400

3 Sum Function()

Query:

SELECT SUM(salary) AS total_salary

FROM staff_salary;

Output:

total_salary
322000

4 Maximum Function()

Query:

SELECT MAX(salary) AS max_salary

FROM staff_salary;

Output:

max_salary
72000

5 Minimum Function()

Query:

SELECT MIN(salary) AS min_salary

FROM staff_salary;

Output:

min_salary
55000

PROGRAM-8

AIM: To Implement different operators Like Operation(%, _) Having Clause,Orderby,Group by

>LIKE Operator:

- The LIKE operator in SQL is used to filter results based on a specified pattern within a column.
- It enables searching for partial matches or specific patterns using wildcard characters such as % (for any number of characters) or _ (for a single character).

>HAVING Clause:

- The HAVING clause in SQL is similar to the WHERE clause but is used specifically with GROUP BY.
- It filters the results of a grouped query based on aggregate conditions, allowing you to specify criteria for aggregated values.

>ORDER BY:

- The ORDER BY clause in SQL is used to sort the result set of a query in ascending or descending order based on one or more columns.
- It helps in organizing data presentation and is often used with SELECT statements to arrange the output according to specific requirements.

>GROUP BY:

- The GROUP BY clause in SQL is used to group rows that have the same values in specified columns.
- It is typically used with aggregate functions like COUNT, SUM, AVG, etc., to perform operations on each group separately and is useful when analyzing data at a higher, summarized level.

Query:

```
CREATE TABLE Data_Information (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    age INT,
    email VARCHAR(255)
);
```

```
INSERT INTO Data_Information VALUES
    (1, 'Akhil Sharma', 20, 'sharmaakhil@gmail.com'),
    (2, 'Ayush Tripathi', 19, 'Tripathiayush@gmail.com'),
    (3, 'Mohit Sethiya', 20, 'SethiyaCAmohit@gmail.com');
```

Output:

Data_Information

id	name	age	email
1	Akhil Sharma	20	sharmaakhil@gmail.com
2	Ayush Tripathi	19	Tripathiayush@gmail.com
3	Mohit Sethiya	20	SethiyaCAmohit@gmail.com

Like Operation Query:

```
SELECT * FROM Data_Information WHERE name LIKE '%Akhil%';
SELECT * FROM Data_Information WHERE email LIKE '%gmail.com%';
```

Output:

id	name	age	email
1	Akhil Sharma	20	sharmaakhil@gmail.com

id	name	age	email
1	Akhil Sharma	20	sharmaakhil@gmail.com
2	Ayush Tripathi	19	Tripathiayush@gmail.com
3	Mohit Sethiya	20	SethiyaCAmohit@gmail.com

Having Clause Query:

```
SELECT age, COUNT(*) as count_of_people
FROM Data_Information
GROUP BY age
HAVING COUNT(*) > 1;
```

Output:

age	count_of_people
20	2

Orderby Query:

SELECT * FROM Data_Information

ORDER BY name ASC;

Output:

age	count_of_people
20	2

Groupby Query:

SELECT age, COUNT(*) as count_of_people

FROM Data_Information

GROUP BY age*;*

Output:

age	count_of_people
19	1
20	2