**Experiment-1**

**Aim**

Creation, altering and dropping of tables and inserting rows into a table, examples using SELECT command.

**Description**

The SQL Command Line is a text-based interface used to interact with relational database management systems (RDBMS) like Oracle, MySQL, PostgreSQL, and SQL Server. It allows users to execute SQL commands directly, manage databases, and query data. It is a fundamental tool for database administrators, developers, and users to perform database operations.

**Features of SQL Command Line**

1. **Interactive Environment:**
   * Users can type SQL commands and see the results immediately.
   * Ideal for testing and troubleshooting database queries.
2. **Database Connectivity:**
   * Connect to a database server using authentication credentials (e.g., username and password).
3. **Execution of SQL Commands:**
   * Execute SELECT, INSERT, UPDATE, DELETE, and other SQL commands to interact with database tables.

**Procedure**

**Connecting to a database**

SQL > connect

SQL> username : sys as sysdba

SQL> password: sys as sysdba

SQL> connected

**Creating Employee Table and perform insert, select, alter, update , rename, delete operations**

**Creating Table**

**Syntax:** create table tablename ( column1 datatype , column2 datatype , column3 datatype , ... );

* **Create Table**: The SQL command to create a new table.
* **table\_Name**: The name of the table to be created.
* **column1, column2, ...**: The columns that the table will have.
* **datatype**: The data type of the column (e.g., INT, VARCHAR, DATE, etc.).

**Example:**

SQL> create table employees ( empid int, empname VARCHAR(50), empsalary int );

**DESC (DESCRIBE) command** : It is used in SQL to display the structure of a table. It provides information about the table’s columns, their data types, and constraints.

**Syntax :**  DESC table\_name; (or) DESCRIBE table\_name;

**Example:**

SQL> DESC employees;

**Inserting values into table**

The INSERT INTO statement is used to add new rows of data into a table. There are two primary ways to insert values into a table: specifying the columns explicitly or inserting values into all columns.

1. Inserting Values into All Columns

**Syntax** : INSERT INTO table\_name VALUES (value1, value2, value3, ...);

**Example** : INSERT INTO employees VALUES (123, 'John Doe', 50000);

1. Inserting Values into Specific Columns

**Syntax** : INSERT INTO table\_name (column1, column2, ...) VALUES (value1, value2, ...);

**Example** : INSERT INTO employees (empid, empsalary) VALUES (456, 85000);

**Select command :** The SELECT command is one of the most frequently used SQL statements. It is used to retrieve data from one or more tables in a database. The retrieved data is displayed in a result set.

1.Retrive all rows and columns in a table

**Syntax** : select \* from tablename;

**Example :** select \* from employees;

2.Retrive specific columns

**Syntax** : select columnname1,columnname2 from tablename;

**Example** : select empid,empsalary from employee;

**Adding a Column**

To add a new column to an existing table, you use the ALTER TABLE command with the ADD COLUMN clause. This allows you to modify the structure of the table by inserting a new column.

**Syntax** : ALTER TABLE table\_name ADD column\_name datatype;

* table\_name: The name of the table to which you want to add a column.
* column\_name: The name of the new column.
* datatype: The data type of the new column (e.g., INT, VARCHAR, DATE, etc.).

**Example** : ALTER TABLE employees ADD department VARCHAR(25);

**UPDATE**

The UPDATE statement in SQL is used to modify existing records in a table. You can update one or more columns for one or more rows in the table based on a specified condition.

**Syntax** : UPDATE table\_name SET column1 = value1, column2 = value2, ...

WHERE condition;

* **table\_name**: The name of the table where you want to update the data.
* **column1, column2, ...**: The columns to be updated.
* **value1, value2, ...**: The new values you want to assign to the columns.
* **WHERE condition**: Specifies which rows should be updated.

**Example** : UPDATE employees SET depatment = ‘cse’ ,empphone=8976546778 WHERE empid = 123;

**1.Update All Rows**

If you want to update a column for all rows (without a WHERE clause),For example, to give all employees a salary increase of 10%.

**Example :** UPDATE employees SET empsalary = empsalary \* 1.10;

**2.Using Expressions in UPDATE**

You can use expressions to calculate the new values. For example, to increase the salary by 5000 for all employees who earn less than 60000:

**Example :** UPDATE employees SET empsalary = empsalary + 5000

WHERE empsalary < 60000;

**Renaming a Column**

Renaming a column in an SQL table is accomplished using the ALTER TABLE command.

**Syntax** : ALTER TABLE table\_name RENAME COLUMN old\_column\_name TO new\_column\_name;

* table\_name: The name of the table containing the column.
* old\_column\_name: The current name of the column to be renamed.
* new\_column\_name: The new name for the column.

**Example** : ALTER TABLE employees RENAME COLUMN department TO dept;

**Renaming a Table**

Renaming a table in SQL can be accomplished using the RENAME or ALTER TABLE command, depending on the database system.

**Syntax :** ALTER TABLE old\_table\_name RENAME TO new\_table\_name;

**Example :** ALTER TABLE employees RENAME TO staff;

**Truncate Table**

TRUNCATE TABLE is an efficient way to clear all rows from a table while preserving the structure.

**Syntax** : TRUNCATE TABLE table\_name;

**Example :** TRUNCATE TABLE staff;

**Deleting a Table**

To delete a table from a database, you can use the DROP TABLE statement. This command removes the table and all of its data, schema, and any associated constraints permanently.

**Syntax :** DROP TABLE table\_name;

**Example** : drop table staff;