Name: Shreya Bansal
PRN: 23070521144
Batch: B2

Practical Number 2

Aim: Write and execute basic SQL query- create, alter, insert, update and delete.

Introduction:
SQL is a standard language for storing, manipulating and retrieving data in databases.

What is SQL?
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views
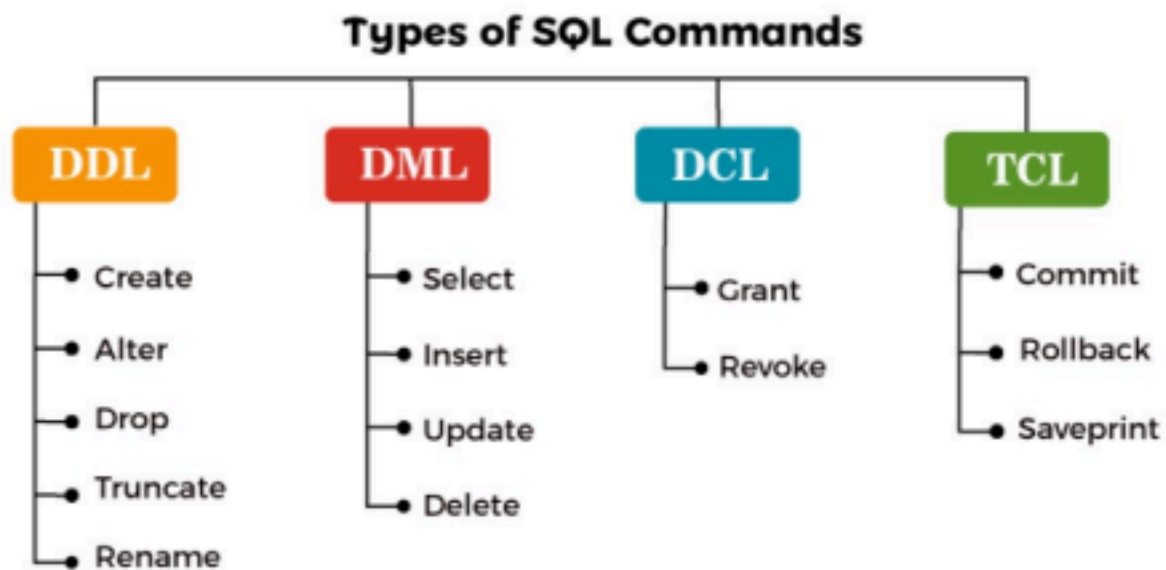
Semicolon after SQL Statements?
Some database systems require a semicolon at the end of each SQL statement.
Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of The Most Important SQL Commands
- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)

- DROP INDEX - deletes an index

Types of SQL Commands

DDL	DML	DCL	TCL
• Create	• Select	• Grant	• Commit
• Alter	• Insert	• Revoke	• Rollback
• Drop	• Update		• Saveprint
• Truncate	• Delete		
• Rename			

SQL Commands
SQLPlus is a command-line interface for interacting with Oracle databases. These commands are specific to SQLPlus and are not standard SQL.

- **SHOW**: Displays database or session settings.
  Example:

  ```
  SHOW USER;
  ```

- **DESCRIBE**: Displays the structure of a table.
  Example:

  ```
  DESCRIBE employees;
  ```

- **EXIT**: Closes the SQL*Plus session.
  Example:

  ```
  EXIT;
  ```

- **SPOOL**: Saves query output to a file.
  Example:

  ```
  SPOOL output.txt;
  SELECT * FROM employees;
  SPOOL OFF;
  ```

DDL Commands (Data Definition Language)
These commands define the structure of a database, including creating, altering, and deleting schema objects.

- **CREATE**: Creates database objects like tables, views, indexes, etc.
  Example:

```
CREATE TABLE employees (
    id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    hire_date DATE
);
```

- **ALTER**: Modifies the structure of existing objects.
  Example:

```
ALTER TABLE employees ADD salary NUMBER;
```

- **DROP**: Deletes database objects.
  Example:

```
DROP TABLE employees;
```

- **TRUNCATE**: Removes all rows from a table, resetting it to an empty state.
  Example:

```
TRUNCATE TABLE employees;
```

- **RENAME**: Changes the name of a database object.
  Example:

```
RENAME employees TO staff;
```

- **COMMENT**: Adds comments to database objects.
  Example:

```
COMMENT ON TABLE employees IS 'Stores employee details';
```

DML Commands (Data Manipulation Language)
These commands manipulate data stored in the database.

- **INSERT**: Adds new rows to a table.
  Example:
```
INSERT INTO employees (id, name, hire_date)
VALUES (1, 'John Doe', SYSDATE);
```

- **UPDATE**: Modifies existing data.
  Example:
```
UPDATE employees
SET salary = 50000
WHERE id = 1;
```

- **DELETE**: Removes rows from a table.
  Example:
```
DELETE FROM employees
WHERE id = 1;
```

- **MERGE**: Combines INSERT and UPDATE functionality.
  Example:
```
MERGE INTO employees e
USING (SELECT 1 AS id, 'Jane Doe' AS name FROM dual) src
ON (e.id = src.id)
```

```
WHEN MATCHED THEN
    UPDATE SET e.name = src.name
WHEN NOT MATCHED THEN
    INSERT (id, name) VALUES (src.id, src.name);
```

- **SELECT**: Retrieves data from the database.
  Example:
  ```
  SELECT * FROM employees;
  ```

To add multiple rows of data into a table, you can use either:
1. **Multiple INSERT statements**
2. **A single INSERT statement with VALUES for multiple rows** (for databases that support this syntax)
3. **INSERT INTO combined with SELECT**

Option 1: Multiple INSERT Statements
This approach is supported by all databases.

```
INSERT INTO employees (id, name, hire_date, salary)
VALUES (1, 'John Doe', TO_DATE('2025-01-01', 'YYYY-MM-DD'), 50000);

INSERT INTO employees (id, name, hire_date, salary)
VALUES (2, 'Jane Smith', TO_DATE('2025-01-02', 'YYYY-MM-DD'), 55000);

INSERT INTO employees (id, name, hire_date, salary)
VALUES (3, 'Alice Brown', TO_DATE('2025-01-03', 'YYYY-MM-DD'), 60000);
```

Option 2: Single INSERT with Multiple VALUES
This approach is supported by some databases like MySQL, PostgreSQL, and SQL Server.
Note: Oracle does not support this directly.

```
INSERT INTO employees (id, name, hire_date, salary)
VALUES
    (1, 'John Doe', TO_DATE('2025-01-01', 'YYYY-MM-DD'), 50000),
    (2, 'Jane Smith', TO_DATE('2025-01-02', 'YYYY-MM-DD'), 55000),
    (3, 'Alice Brown', TO_DATE('2025-01-03', 'YYYY-MM-DD'), 60000);
```

Option 3: INSERT INTO with SELECT
This approach inserts rows by selecting data from another table or using the DUAL table (Oracle-specific).
Insert using SELECT from DUAL:
```
INSERT INTO employees (id, name, hire_date, salary)
SELECT 1, 'John Doe', TO_DATE('2025-01-01', 'YYYY-MM-DD'), 50000 FROM dual
UNION ALL
SELECT 2, 'Jane Smith', TO_DATE('2025-01-02', 'YYYY-MM-DD'), 55000 FROM
dual UNION ALL
SELECT 3, 'Alice Brown', TO_DATE('2025-01-03', 'YYYY-MM-DD'), 60000 FROM
dual;
```
Insert by selecting from another table:
```
INSERT INTO employees (id, name, hire_date, salary)
SELECT id, name, hire_date, salary
FROM temp_employees;
```

Notes:
- Use **TO_DATE** in Oracle for inserting date values in the correct format.

Practice Example 1: Insert Multiple Rows Using Multiple INSERT Statements
Problem:

Insert the following rows into a table named products:

Product_ID	Product_Name	Category	Price
101	Laptop	Electronics	1000
102	Smartphone	Electronics	700
103	Coffee Maker	Appliances	80

Solution:
```
INSERT INTO products (Product_ID, Product_Name, Category, Price)
VALUES (101, 'Laptop', 'Electronics', 1000);

INSERT INTO products (Product_ID, Product_Name, Category, Price)
VALUES (102, 'Smartphone', 'Electronics', 700);

INSERT INTO products (Product_ID, Product_Name, Category, Price)
VALUES (103, 'Coffee Maker', 'Appliances', 80);
```

Practice Example 2: Insert Using Single INSERT with Multiple VALUES
Problem:

Insert the following data into a table named departments:

Department_ID	Department_Name	Location
1	Sales	New York
2	HR	Chicago
3	IT	San Francisco

Solution (for MySQL, PostgreSQL, or SQL Server):
```
INSERT INTO departments (Department_ID, Department_Name, Location)
VALUES
    (1, 'Sales', 'New York'),
    (2, 'HR', 'Chicago'),
    (3, 'IT', 'San Francisco');
```

Practice Example 3: Insert Using SELECT from DUAL (Oracle)
Problem:

Insert the following rows into a table named students:

Student_ID	Name	Enrollment_Date
1001	Alice Johnson	2025-01-05
1002	Bob Smith	2025-01-06
1003	Charlie Brown	2025-01-07

Solution:
```
INSERT INTO students (Student_ID, Name, Enrollment_Date)
SELECT 1001, 'Alice Johnson', TO_DATE('2025-01-05', 'YYYY-MM-DD') FROM dual
UNION ALL
SELECT 1002, 'Bob Smith', TO_DATE('2025-01-06', 'YYYY-MM-DD') FROM dual
UNION ALL
SELECT 1003, 'Charlie Brown', TO_DATE('2025-01-07', 'YYYY-MM-DD') FROM
dual;
```

Practice Example 4: Insert Data Selected from Another Table

Problem:

You have a table `backup_employees` with the following data:

Employee_ID	Full_Name	Hire_Date	Salary
201	Mary Adams	2024-05-10	50000
202	John Carter	2024-06-15	55000

Insert this data into the `employees` table.

Solution:

```
INSERT INTO employees (Employee_ID, Name, Hire_Date, Salary)
SELECT Employee_ID, Full_Name, Hire_Date, Salary
FROM backup_employees;
```

Practice Example 5: Insert into Table with Auto-Increment or Sequence

Problem:

Insert the following rows into a table `orders` with an auto-increment column `Order_ID`:

Customer_Name	Order_Date	Total_Amount
Alice	2025-01-03	150.75
Bob	2025-01-04	200.00

Solution (MySQL):

```
INSERT INTO orders (Customer_Name, Order_Date, Total_Amount)
VALUES
    ('Alice', '2025-01-03', 150.75),
    ('Bob', '2025-01-04', 200.00);
```

Solution (Oracle with Sequence):

```
INSERT INTO orders (Order_ID, Customer_Name, Order_Date, Total_Amount)
VALUES (order_seq.NEXTVAL, 'Alice', TO_DATE('2025-01-03', 'YYYY-MM-DD'),
150.75);

INSERT INTO orders (Order_ID, Customer_Name, Order_Date, Total_Amount)
VALUES (order_seq.NEXTVAL, 'Bob', TO_DATE('2025-01-04', 'YYYY-MM-DD'),
200.00);
```

Practice Example 6: Insert Using Subquery and Calculations

Problem:

You have a table `sales_data` with columns `Product_ID` and `Quantity_Sold`. Insert data into `sales_summary` where `Total_Revenue` is calculated as `Quantity_Sold` * 20 (price per product).

Solution:

```
INSERT INTO sales_summary (Product_ID, Total_Revenue)
SELECT Product_ID, Quantity_Sold * 20
FROM sales_data;
```

TASK

Question 1: Insert Data into a Table

You have a table `books` with the following structure:

Column Name	Data Type
Book_ID	NUMBER
Title	VARCHAR2(100)
Author	VARCHAR2(100)
Price	NUMBER

Insert the following rows into the `books` table:

Book_ID	Title	Author	Price
1	The Great Gatsby	F. Scott	300
2	To Kill a Mockingbird	Harper Lee	350

Question 2: Insert Multiple Rows Using SELECT

Create a new table `employees_backup` with the same structure as the `employees` table. Insert all data from the `employees` table into `employees_backup` using the SELECT statement.

Question 3: Update Existing Data

Given a table `students` with the following columns:

Column Name	Data Type
Student_ID	NUMBER
Name	VARCHAR2(100)
Grade	NUMBER

Update the grade of the student with `Student_ID = 101` to `90`.

Question 4: Delete Specific Rows

In a table `products`, delete all rows where the `Price` is greater than `500`.

Question 5: Retrieve and Insert Data

Given two tables, `employees` and `departments`:
- The `employees` table has columns: `Employee_ID`, `Name`, `Department_ID`.
- The `departments` table has columns: `Department_ID`, `Department_Name`.

Insert data into a new table `employee_departments` (with the same columns as `employees` and `departments`) by combining data from both tables using a SELECT statement.

```
SQL> SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') AS
  2  formatted_date FROM dual;

FORMATTED_DATE
-------------------
2025-02-06 14:21:56

SQL> SELECT TO_CHAR(12345.67, 'L99,999.99') AS formatted_currency
  2  FROM dual;

FORMATTED_CURRENCY
-------------------
        $12,345.67

SQL> SELECT TO_DATE('2025-01-29', 'YYYY-MM-DD') AS converted_date
  2  FROM dual;

CONVERTED
---------
29-JAN-25

SQL> SELECT TO_DATE('29-01-2025', 'DD-MM-YYYY') FROM dual;

TO_DATE('
---------
29-JAN-25

SQL> SELECT TO_NUMBER('12345.67') AS number_value FROM dual;

NUMBER_VALUE
------------
    12345.67

SQL> SELECT CAST(123.45 AS VARCHAR2(10)) AS string_value FROM
  2  dual;

STRING_VAL
----------
123.45

SQL> SELECT CAST(TO_DATE('2025-01-29', 'YYYY-MM-DD') AS DATE)
  2  FROM dual;

CAST(TO_D
---------
29-JAN-25

SQL> SELECT emp_id, TO_CHAR(salary, 'L99,999.99') AS
  2
SQL> ;
  1* SELECT emp_id, TO_CHAR(salary, 'L99,999.99') AS
```

```
SQL>
SQL> CREATE TABLE products (
  2      product_id INT PRIMARY KEY,
  3      product_name VARCHAR2(255),
  4      price_usd DECIMAL(10, 2)
  5  );

Table created.

SQL>
SQL> CREATE TABLE transactions (
  2      transaction_id INT PRIMARY KEY,
  3      account_id INT,
  4      amount DECIMAL(10, 2),
  5      transaction_time TIMESTAMP
  6  );

Table created.

SQL>
SQL> CREATE TABLE sensor_logs (
  2      sensor_id INT PRIMARY KEY,
  3      unix_timestamp INT
  4  );

Table created.

SQL>
SQL> INSERT INTO products (product_id, product_name, price_usd) VALUES (101, 'iPhone 15', 1199.99);

1 row created.

SQL> INSERT INTO products (product_id, product_name, price_usd) VALUES (202, 'MacBook Pro', 2649.99);

1 row created.

SQL>
SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (89234, 123456, 5000, TO_TIMESTAMP('2025-02-06 02:30:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (97345, 789012, 25000, TO_TIMESTAMP('2025-02-06 03:15:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO sensor_logs (sensor_id, unix_timestamp) VALUES (1, 1706505600);

1 row created.
```

Created with cam scan

```
1 row created.

SQL> INSERT INTO transactions (transaction_id, account_id, amount, transaction_time)
  2  VALUES (973345, 789012, 25000, TO_TIMESTAMP('2025-02-06 03:15:00', 'YYYY-MM-DD HH24:MI:SS'));

1 row created.

SQL>
SQL> INSERT INTO sensor_logs (sensor_id, unix_timestamp) VALUES (1, 1706585600);

1 row created.

SQL> INSERT INTO sensor_logs (sensor_id, unix_timestamp) VALUES (2, 1706589200);

1 row created.

SQL>
SQL> SELECT
  2      product_id,
  3      product_name,
  4      TO_CHAR(price_usd * 83.50, 'L99,999.99') AS price_inr
  5  FROM products;

PRODUCT_ID
-----------
PRODUCT_NAME
---------------------------------------------------------------------------------
PRICE_INR
-----------------------
       101
iPhone 15
####################

       202
MacBook Pro
####################

PRODUCT_ID
-----------
PRODUCT_NAME
---------------------------------------------------------------------------------
PRICE_INR
-----------------------

SQL>
SQL> SELECT
  2      transaction_id,
  3      account_id,
  4      amount,
```

Created with cam scan

```
SQL>
SQL> SELECT
  2      transaction_id,
  3      account_id,
  4      amount,
  5      TO_CHAR(transaction_time, 'HH24:MI') AS transaction_hour
  6  FROM transactions
  7  WHERE EXTRACT(HOUR FROM transaction_time) BETWEEN 0 AND 4;

TRANSACTION_ID ACCOUNT_ID     AMOUNT TRANS
-------------- ---------- ---------- -----
         89234     123456       5000 02:30
        973345     789012      25000 03:15

SQL>
SQL> SELECT
  2      sensor_id,
  3      FROM_TZ(TO_TIMESTAMP(unix_timestamp), 'UTC') AS reading_time
  4  FROM sensor_logs;
    FROM_TZ(TO_TIMESTAMP(unix_timestamp), 'UTC') AS reading_time
                        *
ERROR at line 3:
ORA-00932: inconsistent datatypes: expected - got NUMBER


SQL>     sensor_id,
SP2-0042: unknown command "sensor_id," - rest of line ignored.
SQL>     FROM_TZ(TO_TIMESTAMP('1970-01-01', 'YYYY-MM-DD') + (unix_timestamp / 86400), 'UTC') AS reading_time
SP2-0734: unknown command beginning "FROM_TZ(TO..." - rest of line ignored.
SQL> FROM sensor_logs;
SP2-0734: unknown command beginning "FROM senso..." - rest of line ignored.
SQL> SELECT
  2      sensor_id,
  3      FROM_TZ(TO_TIMESTAMP('1970-01-01', 'YYYY-MM-DD') + (unix_timestamp / 86400), 'UTC') AS reading_time
  4  FROM sensor_logs;
    FROM_TZ(TO_TIMESTAMP('1970-01-01', 'YYYY-MM-DD') + (unix_timestamp / 86400), 'UTC') AS reading_time
                        *
ERROR at line 3:
ORA-00932: inconsistent datatypes: expected TIMESTAMP got DATE


SQL> DROP TABLE purchases;
DROP TABLE purchases
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Created with cam scan

```
SQL>
SQL> CREATE TABLE purchases (
  2      customer_id INT,
  3      purchase_date DATE
  4  );

Table created.

SQL>
SQL> INSERT INTO purchases (customer_id, purchase_date) VALUES (1, TO_DATE('2025-01-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO purchases (customer_id, purchase_date) VALUES (2, TO_DATE('2025-02-03', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> SELECT
  2      customer_id,
  3      purchase_date,
  4      TO_CHAR(purchase_date, 'Month') AS purchase_month,
  5      TO_CHAR(purchase_date, 'YYYY') AS purchase_year
  6  FROM purchases;

CUSTOMER_ID PURCHASE_ PURCHASE_ PURC
----------- --------- --------- ----
          1 15-JAN-25 January   2025
          2 03-FEB-25 February  2025
```

Created with cam scan

```
1 row created.

SQL>
SQL> SELECT TO_DATE('29/01/2025', 'DD/MM/YYYY') AS formatted_date FROM dual;

FORMATTED
---------
29-JAN-25

SQL>
SQL> SELECT
  2      order_id,
  3      distance_km,
  4      ROUND(distance_km / 60, 2) AS estimated_hours
  5  FROM deliveries;

  ORDER_ID DISTANCE_KM ESTIMATED_HOURS
---------- ----------- ---------------
      1001         120               2

SQL>
SQL> SELECT
  2      post_id,
  3      TO_CHAR(post_date, 'Month DD, YYYY HH24:MI') AS formatted_date
  4  FROM posts;

   POST_ID FORMATTED_DATE
---------- --------------------------
       101 February  06, 2025 12:30
       102 February  06, 2025 14:00

SQL> DROP TABLE sensor_logs;

Table dropped.

SQL>
SQL> CREATE TABLE sensor_logs (
  2      sensor_id INT PRIMARY KEY,
  3      unix_timestamp INT
  4  );

Table created.

SQL>
SQL> INSERT INTO sensor_logs (sensor_id, unix_timestamp) VALUES (1, 1706505600);

1 row created.

SQL> INSERT INTO sensor_logs (sensor_id, unix_timestamp) VALUES (2, 1706509200);

1 row created.
```

Created with cam scan

```
SQL>
SQL> CREATE TABLE purchases (
  2      customer_id INT,
  3      purchase_date DATE
  4  );

Table created.

SQL>
SQL> CREATE TABLE deliveries (
  2      order_id INT,
  3      distance_km DECIMAL(10, 2)
  4  );

Table created.

SQL>
SQL> CREATE TABLE posts (
  2      post_id INT,
  3      post_date DATE
  4  );

Table created.

SQL>
SQL> INSERT INTO purchases (customer_id, purchase_date) VALUES (1, TO_DATE('2025-01-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO purchases (customer_id, purchase_date) VALUES (2, TO_DATE('2025-02-03', 'YYYY-MM-DD'));

1 row created.

SQL>
SQL> INSERT INTO deliveries (order_id, distance_km) VALUES (1001, 120);

1 row created.

SQL>
SQL> INSERT INTO posts (post_id, post_date) VALUES (101, TO_DATE('2025-02-06 12:30', 'YYYY-MM-DD HH24:MI'));

1 row created.

SQL> INSERT INTO posts (post_id, post_date) VALUES (102, TO_DATE('2025-02-06 14:00', 'YYYY-MM-DD HH24:MI'));

1 row created.

SQL>
SQL> SELECT TO_DATE('29/01/2025', 'DD/MM/YYYY') AS formatted_date FROM dual;

FORMATTED
```

Created with cam scan