SQL Definition :

SQL, which stands for Structured Query Language, is a standard programming language used to manage and manipulate relational databases. It's used to communicate with a database to perform tasks such as retrieving data, updating data, inserting data, and deleting data.

SQL Use:

SQL is used for managing relational databases. Its primary uses include retrieving, inserting, updating, and deleting data, defining database structures, and controlling data access and integrity.

Types of SQL Commands

1. DDL (Data Definition Language) – Used to create, alter, or delete database structures (CREATE, ALTER, DROP).
2. DML (Data Manipulation Language) – Used to insert, update, or delete data in tables (INSERT, UPDATE, DELETE).
3. DQL (Data Query Language) – Used to retrieve data from database tables (SELECT).
4. DCL (Data Control Language) – Used to grant or revoke database access permissions (GRANT, REVOKE).
5. TCL (Transaction Control Language) – Used to manage transactions in a database (COMMIT, ROLLBACK, SAVEPOINT).

Data Types in SQL

1. INT / BIGINT – Whole numbers; used for IDs, age, counts
2. DECIMAL / FLOAT – Decimal numbers; used for prices, salary, marks
3. CHAR / VARCHAR / TEXT – Text data; used for names, emails, comments
4. DATE / TIME / TIMESTAMP – Date & time; used for DOB, login time, logs
5. BOOLEAN – TRUE / FALSE; used for status like is_active
6. BLOB / BINARY – Files & images; used for documents, photos
7. JSON – JSON formatted data; used for semi-structured API data

CREATE TABLE

CREATE TABLE is used to make a new table in the database by defining its column names and their data types.

INSERT INTO

INSERT INTO is used to add new data (rows) into an existing table.

```
-- CREATE TABLE example
CREATE TABLE Products (
    ProductID INTEGER PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2),
    StockQuantity INTEGER
);

-- INSERT INTO example
INSERT INTO Products (ProductID, ProductName, Price, StockQuantity)
VALUES (101, 'Laptop', 1200.00, 50);

INSERT INTO Products (ProductID, ProductName, Price, StockQuantity)
VALUES (102, 'Mouse', 25.50, 200);

INSERT INTO Products (ProductID, ProductName, Price, StockQuantity)
VALUES (103, 'Keyboard', 75.00, 150);
```

Explanation:

- The `CREATE TABLE` statement defines a new table named `Products` with columns for `ProductID`, `ProductName`, `Price`, and `StockQuantity`, along with their respective data types and constraints.
- `PRIMARY KEY` makes `ProductID` a unique identifier for each row.

- `NOT NULL` ensures `ProductName` cannot be empty.
- The `INSERT INTO` statements add three distinct product records into the `Products` table.

```
import sqlite3
import pandas as pd

# Connect to in-memory database
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Create table
cursor.execute("""
CREATE TABLE Courses (
    CourseID INTEGER PRIMARY KEY,
    CourseName TEXT,
    Instructor TEXT,
    Credits INTEGER
);
""")

# Insert data (including NEW course)
cursor.executemany("""
INSERT INTO Courses VALUES (?, ?, ?, ?);
""", [
    (101, 'Introduction to Python', 'Dr. Smith', 3),
    (102, 'Database Management', 'Prof. Lee', 4),
    (103, 'Web Development Basics', 'Ms. Jones', 3),
    (104, 'Data Science Fundamentals', 'Dr. Kumar', 4) # new data
])

conn.commit()

# Display the full table
print("Full Courses Table:")
df = pd.read_sql("SELECT * FROM Courses", conn)
print(df)

# Add query using clauses
print("\nCourses grouped by Credits, filtered and ordered:")
query_with_clauses = """
SELECT Credits, COUNT(*) AS total_courses
FROM Courses
WHERE Credits >= 3
GROUP BY Credits
HAVING COUNT(*) > 0
ORDER BY Credits DESC
LIMIT 10;
"""
df_clauses = pd.read_sql(query_with_clauses, conn)
print(df_clauses)

conn.close()
```

```
Full Courses Table:
   CourseID                CourseName Instructor  Credits
0       101     Introduction to Python  Dr. Smith        3
1       102        Database Management  Prof. Lee        4
2       103     Web Development Basics  Ms. Jones        3
3       104  Data Science Fundamentals  Dr. Kumar        4

Courses grouped by Credits, filtered and ordered:
   Credits  total_courses
0        4              2
1        3              2
```

SQL Clauses

1. SELECT – Chooses the columns to display
2. FROM – Specifies the table to fetch data from
3. WHERE – Filters rows based on condition
4. GROUP BY – Groups rows with same values
5. HAVING – Filters groups after aggregation
6. ORDER BY – Sorts the result (ASC / DESC)

7. DISTINCT – Removes duplicate rows
8. LIMIT / OFFSET – Restricts number of rows returned
9. JOIN (ON) – Combines rows from multiple tables

```
import sqlite3
import pandas as pd

# Connect to the in-memory database
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Create the Courses table and insert data (as in cell iNRDOzt8ptTn)
cursor.execute("""
CREATE TABLE Courses (
    CourseID INTEGER PRIMARY KEY,
    CourseName TEXT,
    Instructor TEXT,
    Credits INTEGER
);
""")

cursor.executemany("""
INSERT INTO Courses VALUES (?, ?, ?, ?);
""", [
    (101, 'Introduction to Python', 'Dr. Smith', 3),
    (102, 'Database Management', 'Prof. Lee', 4),
    (103, 'Web Development Basics', 'Ms. Jones', 3),
    (104, 'Data Science Fundamentals', 'Dr. Kumar', 4)
])

conn.commit()

# Execute the SQL query using pd.read_sql
query_to_execute = """
SELECT Credits, COUNT(*) AS TotalCourses
FROM Courses
WHERE Credits >= 3
GROUP BY Credits
HAVING COUNT(*) > 0
ORDER BY Credits DESC
LIMIT 10;
"""

df_result = pd.read_sql(query_to_execute, conn)
print(df_result)

conn.close()
```

```
   Credits  TotalCourses
0        4             2
1        3             2
```

Aggregate Function

1. COUNT(*) – Counts total rows
2. COUNT(column) – Counts non-NULL values
3. SUM() – Adds all values
4. AVG() – Calculates average
5. MIN() – Finds minimum value
6. MAX() – Finds maximum value

```
import sqlite3
import pandas as pd

# Connect to an in-memory SQLite database
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Create the Courses table
cursor.execute("""
```

```
CREATE TABLE Courses (
    CourseID INTEGER PRIMARY KEY,
    CourseName TEXT,
    Instructor TEXT,
    Credits INTEGER
);
""")

# Insert data into the Courses table
cursor.executemany("""
INSERT INTO Courses VALUES (?, ?, ?, ?);
""", [
    (101, 'Introduction to Python', 'Dr. Smith', 3),
    (102, 'Database Management', 'Prof. Lee', 4),
    (103, 'Web Development Basics', 'Ms. Jones', 3),
    (104, 'Data Science Fundamentals', 'Dr. Kumar', 4)
])

conn.commit()

print("--- Aggregate Functions Example ---")
query_aggregates = """
SELECT
    COUNT(*) AS total_courses,
    SUM(Credits) AS total_credits,
    AVG(Credits) AS avg_credits,
    MIN(Credits) AS min_credits,
    MAX(Credits) AS max_credits
FROM Courses;
"""
df_aggregates = pd.read_sql(query_aggregates, conn)
print(df_aggregates)

# Close the database connection
conn.close()
```

```
--- Aggregate Functions Example ---
   total_courses  total_credits  avg_credits  min_credits  max_credits
0              4             14          3.5            3            4
```

JOIN

1. INNER JOIN – Returns only matching rows from both tables
2. LEFT JOIN – Returns all rows from left table and matching rows from right
3. RIGHT JOIN – Returns all rows from right table and matching rows from left
4. FULL JOIN – Returns all rows from both tables
5. CROSS JOIN – Returns all possible row combinations
6. SELF JOIN – A table joined with itself

```
import sqlite3
import pandas as pd

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Create the Courses table with user's specified data
cursor.execute("""
CREATE TABLE Courses (
    CourseID INTEGER PRIMARY KEY,
    CourseName TEXT,
    Instructor TEXT,
    Credits INTEGER,
    DepartmentID INTEGER
);
""")

cursor.executemany("""
INSERT INTO Courses VALUES (?, ?, ?, ?, ?);
""", [
    (101, 'Introduction to Python', 'Dr. Smith', 3, 1),
    (102, 'Database Management', 'Prof. Lee', 4, 2),
    (103, 'Web Development Basics', 'Ms. Jones', 3, 1),
```

```
    (104, 'Data Science Fundamentals', 'Dr. Kumar', 4, 3)
])

# Create a Departments table for joining
cursor.execute("""
CREATE TABLE Departments (
    DepartmentID INTEGER PRIMARY KEY,
    DepartmentName TEXT
);
""")

cursor.executemany("""
INSERT INTO Departments VALUES (?, ?);
""", [
    (1, 'Computer Science'),
    (2, 'Information Systems'),
    (3, 'Data Science'),
    (4, 'Mathematics') # Department with no courses yet
])

conn.commit()

print("--- INNER JOIN: Courses with their Departments ---")
query_inner_join = """
SELECT
    C.CourseName,
    C.Instructor,
    D.DepartmentName
FROM
    Courses AS C
INNER JOIN
    Departments AS D ON C.DepartmentID = D.DepartmentID;
"""
df_inner_join = pd.read_sql(query_inner_join, conn)
print(df_inner_join)

print('\n') # Added space
print("--- LEFT JOIN: All Departments and their Courses (if any) ---")
query_left_join = """
SELECT
    D.DepartmentName,
    C.CourseName,
    C.Instructor
FROM
    Departments AS D
LEFT JOIN
    Courses AS C ON D.DepartmentID = C.DepartmentID
ORDER BY D.DepartmentName, C.CourseName;
"""
df_left_join = pd.read_sql(query_left_join, conn)
print(df_left_join)

conn.close()
```

```
--- INNER JOIN: Courses with their Departments ---
                  CourseName Instructor       DepartmentName
0     Introduction to Python  Dr. Smith     Computer Science
1        Database Management  Prof. Lee  Information Systems
2     Web Development Basics  Ms. Jones     Computer Science
3  Data Science Fundamentals  Dr. Kumar         Data Science



--- LEFT JOIN: All Departments and their Courses (if any) ---
        DepartmentName                CourseName Instructor
0     Computer Science     Introduction to Python  Dr. Smith
1     Computer Science     Web Development Basics  Ms. Jones
2         Data Science  Data Science Fundamentals  Dr. Kumar
3  Information Systems        Database Management  Prof. Lee
4          Mathematics                       None       None
```