**20. Queries using SQLite**

import sqlite3  # a. Import SQLite library

# b. Connect to an in-memory database (temporary)

conn = sqlite3.connect(':memory:')  # This database exists only while the program runs

cursor = conn.cursor()

# c. Create a table

cursor.execute('''

    CREATE TABLE IF NOT EXISTS users (

        id INTEGER PRIMARY KEY,

        name TEXT,

        age INTEGER   )”)

 conn.commit()

print("Table created successfully.")

# d. Insert records

cursor.execute("INSERT INTO users (name, age) VALUES ('Alice', 25)")

cursor.execute("INSERT INTO users (name, age) VALUES ('Bob', 35)")

cursor.execute("INSERT INTO users (name, age) VALUES ('Charlie', 45)")

conn.commit()

print("Records inserted.")

# Display records

cursor.execute("SELECT \* FROM users")

records = cursor.fetchall()

print("\nDisplaying records:")

for record in records:

 print(record)

# e. Add a column

cursor.execute("ALTER TABLE users ADD COLUMN email TEXT")

conn.commit()

print("\nColumn 'email' added.")

# f. Update a column

cursor.execute("UPDATE users SET email = 'alice@example.com' WHERE name = 'Alice'")

cursor.execute("UPDATE users SET email = 'bob@example.com' WHERE name = 'Bob'")

conn.commit()

print("Records updated.")

# Delete a column (SQLite does not support direct column deletion, so we recreate the table)

cursor.execute('''

 CREATE TABLE users\_new (

 id INTEGER PRIMARY KEY,

 name TEXT,

 age INTEGER)”)

cursor.execute("INSERT INTO users\_new SELECT id, name, age FROM users")

cursor.execute("DROP TABLE users")

cursor.execute("ALTER TABLE users\_new RENAME TO users")

conn.commit()

print("\nColumn 'email' deleted.")

# g. Drop the table and close connection

cursor.execute("DROP TABLE users")

conn.commit()

print("\nTable dropped.")

conn.close()

print("Database connection closed.")

**Output:**

Table created successfully.

Records inserted.

Displaying records:

(1, 'Alice', 25)

(2, 'Bob', 35)

(3, 'Charlie', 45)

Column 'email' added.

Records updated.

Column 'email' deleted.

Table dropped.

Database connection closed.

**21.data transformation in python using pandas**

**import pandas as pd**

import numpy as np

# Sample dataset

data = pd.DataFrame({

    'ID': [1, 2, 2, 3, 4, 4, 5],

    'Name': ['Alice', 'Bob', 'Bob', 'Charlie', 'David', 'David', 'Eve'],

    'Age': [30, 35, 40, 45, 50, 55, 60],

    'Salary': [50000, 60000, 60000, 70000, 80000, 80000, 90000]})

print("Original Data:\n", data)

# 1. Removing duplicates

data = data.drop\_duplicates()

print("\nData after removing duplicates:\n", data)

# 2. Adding a new column

data['Experience'] = [2, 5, 7, 10, 12]

print("\nData after adding a column:\n", data)

# 3. Replacing values

data['Name'] = data['Name'].replace({'Alice': 'Alicia', 'Bob': 'Robert'})

print("\nData after replacing values:\n", data)

# 4. Renaming axis/index

data = data.rename(columns={'Salary': 'Income'})

print("\nData after renaming a column:\n", data)

# 5. Discretization and binning

data['Age Group'] = pd.cut(data['Age'], bins=[20, 30, 40, 50], labels=['Young', 'Mid-age', 'Senior'])

print("\nData after discretization and binning:\n", data)

# 6. Detecting and filtering outliers (Using IQR method)

Q1 = data['Income'].quantile(0.25)

Q3 = data['Income'].quantile(0.75)

IQR = Q3 - Q1

filtered\_data = data[(data['Income'] >= (Q1 - 1.5 \* IQR)) & (data['Income'] <= (Q3 + 1.5 \* IQR))]

print("\nData after filtering outliers:\n", filtered\_data)

# 7. Permutation and random sampling

sampled\_data = data.sample(frac=0.5, random\_state=42)

print("\nRandom sample of data:\n", sampled\_data)

**Output:** Original Data:

    ID     Name  Age  Salary

0   1    Alice   25   50000

1   2      Bob   30   60000

2   2      Bob   30   60000

3   3  Charlie   35   70000

4   4    David   40   80000

5   4    David   40   80000

6   5      Eve   45   90000

Data after removing duplicates:

    ID     Name  Age  Salary

0   1    Alice   25   50000

1   2      Bob   30   60000

3   3  Charlie   35   70000

4   4    David   40   80000

6   5      Eve   45   90000

Data after adding a column:

    ID     Name  Age  Salary  Experience

0   1    Alice   25   50000           2

1   2      Bob   30   60000           5

3   3  Charlie   35   70000           7

4   4    David   40   80000          10

6   5      Eve   45   90000          12

Data after replacing values:

    ID     Name  Age  Salary  Experience

0   1   Alicia   25   50000           2

1   2   Robert   30   60000           5

3   3  Charlie   35   70000           7

4   4    David   40   80000          10

6   5      Eve   45   90000          12

Data after renaming a column:

    ID     Name  Age  Income  Experience

0   1   Alicia   25   50000           2

1   2   Robert   30   60000           5

3   3  Charlie   35   70000           7

4   4    David   40   80000          10

6   5      Eve   45   90000          12

Data after discretization and binning:

    ID     Name  Age  Income  Experience Age Group

0   1   Alicia   25   50000           2     Young

1   2   Robert   30   60000           5     Young

3   3  Charlie   35   70000           7   Mid-age

4   4    David   40   80000          10   Mid-age

6   5      Eve   45   90000          12    Senior

Data after filtering outliers:

    ID     Name  Age  Income  Experience Age Group

0   1   Alicia   25   50000           2     Young

1   2   Robert   30   60000           5     Young

3   3  Charlie   35   70000           7   Mid-age

4   4    David   40   80000          10   Mid-age

6   5      Eve   45   90000          12    Senior

Random sample of data:

    ID    Name  Age  Income  Experience Age Group

1   2  Robert   30   60000           5     Young

6   5     Eve   45   90000          12    Senior