SQLite Data Handling and Analysis using Python

What I Do:

This code snippet demonstrates how to: 1. Connect to a SQLite database. 2. Create a table called 'sales' if it doesn't already exist. 3. Insert multiple rows of sales data into the table. 4. Run a query to calculate total quantity and revenue grouped by product type.

How I Do It:

1. Import Necessary Libraries:

import sqlite3 import pandas as pd import matplotlib.pyplot as plt - sqlite3: To create and interact with the SQLite database. - pandas: For data manipulation (used later for data visualization or handling). - matplotlib.pyplot: For plotting graphs if needed.

2. Connect to SQLite Database:

conn = sqlite3.connect("sales_data.db") cursor = conn.cursor() - Establishes a connection to a SQLite database file named 'sales_data.db'. - Creates a cursor object to execute SQL commands.

3. Create the Sales Table:

cursor.execute(""" CREATE TABLE IF NOT EXISTS sales (id INTEGER PRIMARY KEY AUTOINCREMENT, product TEXT, quantity INTEGER, price REAL) """) - Ensures the 'sales' table is created with columns for ID, product name, quantity sold, and price.

4. Insert Data into the Table:

data = [('Laptop',5,50000), ..., ('Monitor',4,10100)] cursor.executemany("INSERT INTO sales (product, quantity, price) VALUES (?, ?, ?)", data) conn.commit() - Prepares a list of tuples with product sales records. - Uses 'executemany()' to batch insert all rows. - Commits the transaction to save changes in the database.

5. Query to Analyze Sales:

query = """ SELECT product, SUM(quantity) AS total_qty, SUM(quantity * price) AS revenue FROM sales GROUP BY product """ - Selects each unique product. - Calculates total quantity sold. - Calculates total revenue. - Groups results by product.

Conclusion:

This program is a simple but powerful way to manage and analyze sales data using SQLite with Python. It can be expanded further to include visualizations and advanced reporting using libraries like pandas and matplotlib.