Problem Statement 1

1. API Data Retrieval and Storage: You are tasked with fetching data from an external REST API, storing it in a local SQLite database, and displaying the retrieved data. The API provides a list of books in JSON format with attributes like title, author, and publication year.

   External API used: Open Library REST API

   Tool used to Visualize: DB Browser Sqlite

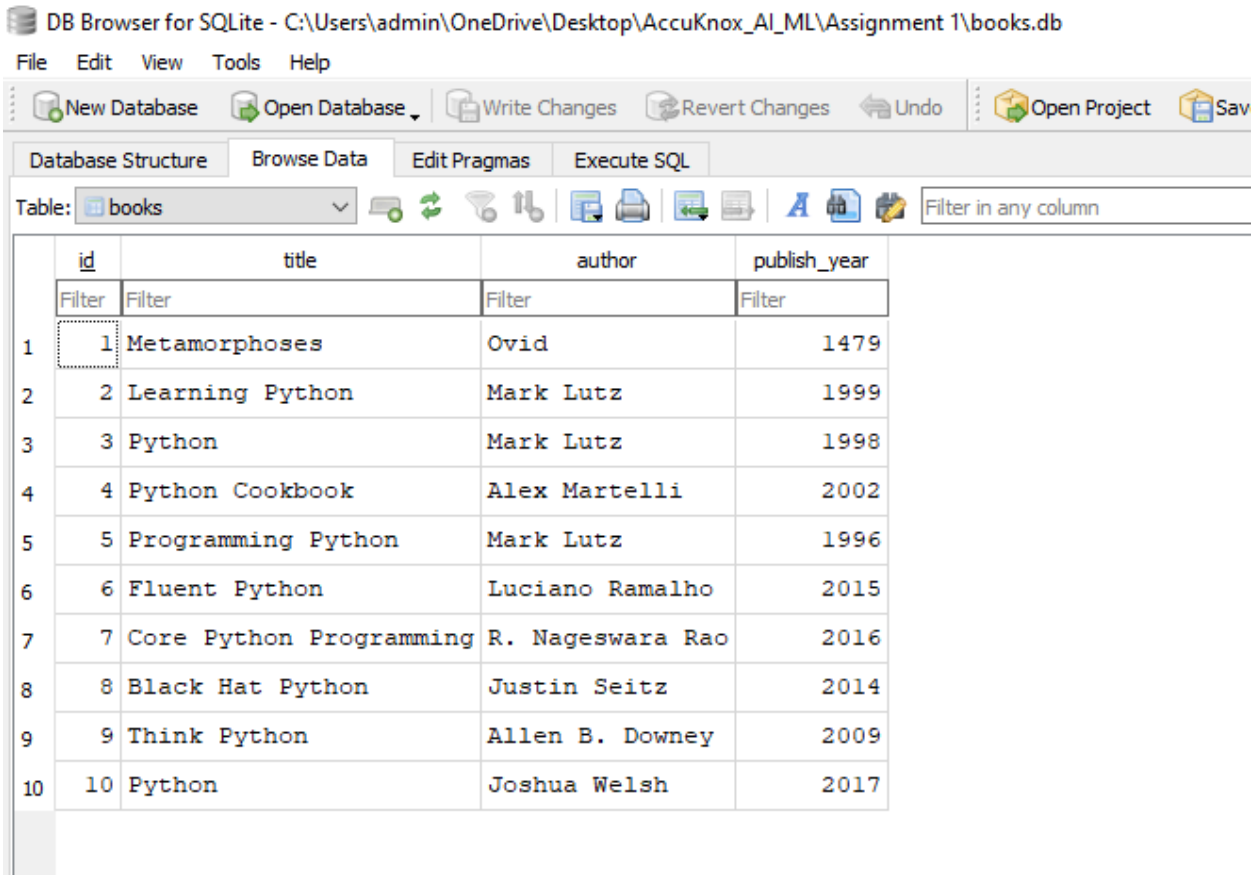
   Book_api_to_sqlite.py

```
import requests

]import sqlite3



# 1. Fetch data from Open Library API

url = "https://openlibrary.org/search.json?q=python"

response = requests.get(url)

data = response.json()



books = data["docs"][:10]



# 2. Connect to SQLite

conn = sqlite3.connect("books.db")

cursor = conn.cursor()



# 3. Create table

cursor.execute("""

CREATE TABLE IF NOT EXISTS books (

    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    title TEXT,
    author TEXT,
    publish_year INTEGER
)
""")

# 4. Insert data
for book in books:
    title = book.get("title")
    author = book.get("author_name", ["Unknown"])[0]
    year = book.get("first_publish_year")

    cursor.execute(
   "INSERT INTO books (title, author, publish_year) VALUES (?, ?, ?)",
        (title, author, year)
    )

conn.commit()

# 5. Read & display data
cursor.execute("SELECT title, author, publish_year FROM books")
rows = cursor.fetchall()

for row in rows:
    print(row)
```

```
conn.close()
```

Result:



	id	title	author	publish_year
	Filter	Filter	Filter	Filter
1	1	Metamorphoses	Ovid	1479
2	2	Learning Python	Mark Lutz	1999
3	3	Python	Mark Lutz	1998
4	4	Python Cookbook	Alex Martelli	2002
5	5	Programming Python	Mark Lutz	1996
6	6	Fluent Python	Luciano Ramalho	2015
7	7	Core Python Programming	R. Nageswara Rao	2016
8	8	Black Hat Python	Justin Seitz	2014
9	9	Think Python	Allen B. Downey	2009
10	10	Python	Joshua Welsh	2017

2. Data Processing and Visualization: Given a dataset containing information about students' test scores, fetch the data from an API, calculate the average score, and create a bar chart to visualize the data.

   External API used: https://github.com/jgillson/student-exam-scores-api

   Tool used to Visualize: Matplotlib library

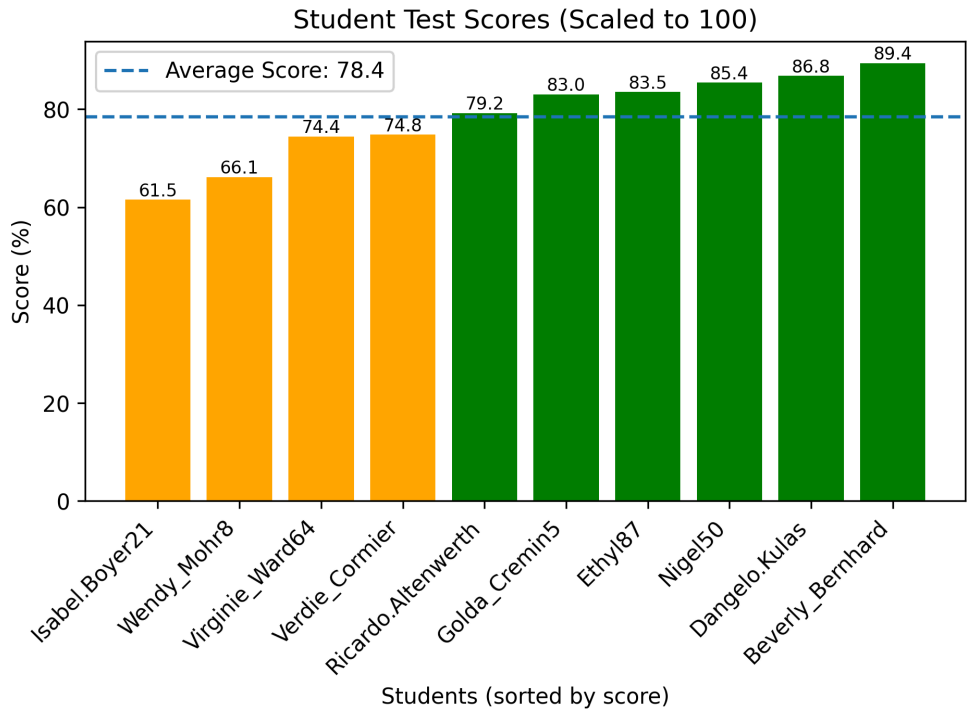
Data_processing_visualization.py

```
import requests

import json

import matplotlib.pyplot as plt


URL = "http://live-test-scores.herokuapp.com/scores/"


students = []

scores = []


response = requests.get(URL, stream=True)


for line in response.iter_lines():

    if line:

        decoded = line.decode("utf-8")

        if decoded.startswith("data:"):

            record = json.loads(decoded.replace("data: ", ""))

            students.append(record["studentId"])

            scores.append(record["score"])


        if len(scores) == 10:

            break


# Convert to percentage scale

scores = [round(score * 100, 1) for score in scores]
```

```
# Sort by score
sorted_data = sorted(zip(students, scores), key=lambda x: x[1])
students, scores = zip(*sorted_data)

average_score = round(sum(scores) / len(scores), 1)
print("Average Score:", average_score)

# Color logic
colors = []
for score in scores:
    if score >= 75:
        colors.append("green")
    elif score >= 60:
        colors.append("orange")
    else:
        colors.append("red")

plt.bar(students, scores, color=colors)

plt.axhline(
    average_score,
    linestyle='--',
    label=f"Average Score: {average_score}"
)
```

```
plt.legend()

# Value labels

for i, score in enumerate(scores):

    plt.text(i, score, f"{score}", ha="center", va="bottom", fontsize=8)

plt.title("Student Test Scores (Scaled to 100)")

plt.xlabel("Students (sorted by score)")

plt.ylabel("Score (%)")

plt.xticks(rotation=45, ha="right")

plt.tight_layout()

plt.savefig("student_scores.png", dpi=300, bbox_inches="tight")

plt.show()
```

Result:



3. CSV Data Import to a Database: Write a Python script that reads data from a CSV file containing user information (e.g., name, email) and inserts it into a SQLite database.

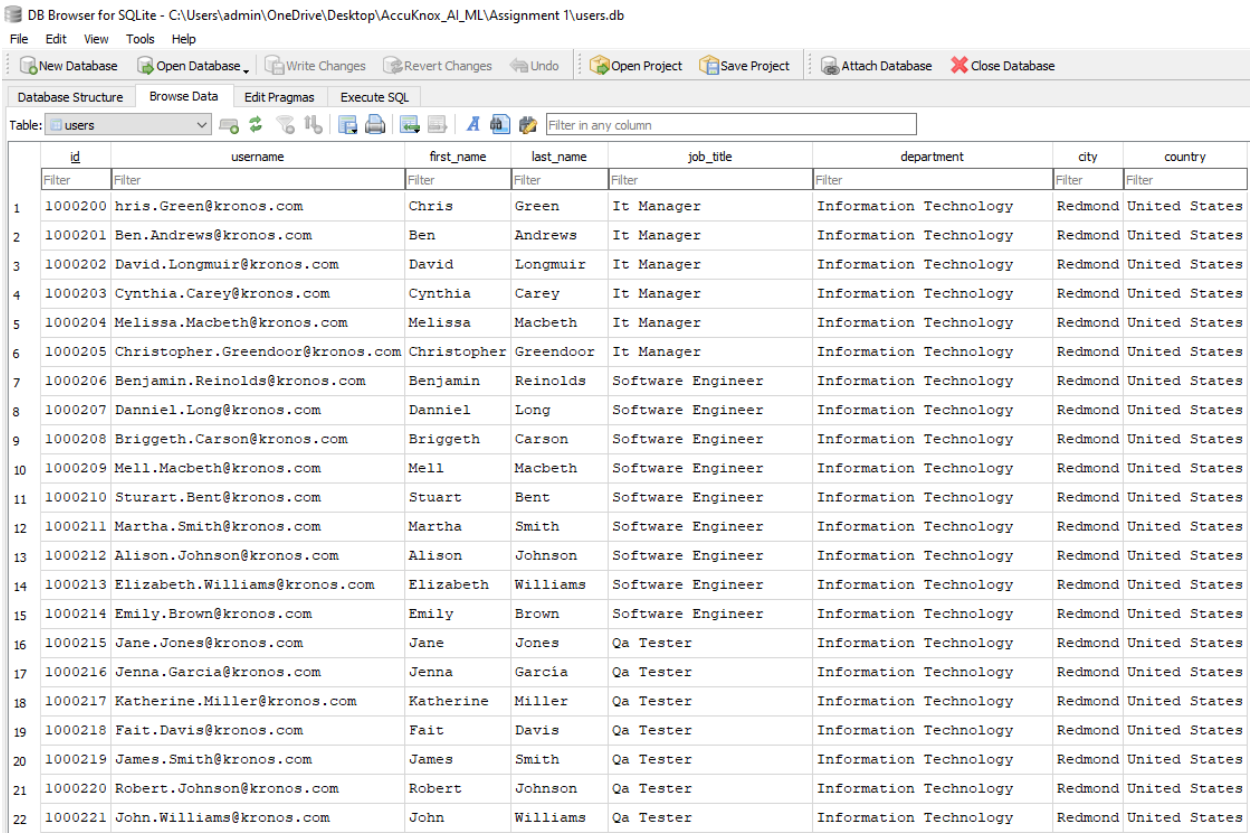
   Tool used to Visualize: DB Browser Sqlite

   Csv_to_sqlite.py

```
import csv

import sqlite3



conn = sqlite3.connect("users.db")

cursor = conn.cursor()



# Create table

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

    id INTEGER PRIMARY KEY,

    username TEXT,

    first_name TEXT,

    last_name TEXT,

    job_title TEXT,

    department TEXT,

    city TEXT,

    country TEXT

)

""")
```

```
with open("user_info.csv", newline="", encoding="utf-8") as file:
    reader = csv.DictReader(file)

    for row in reader:
        try:
            # Skip broken/incomplete rows
            if not row["Id"] or not row["Username"]:
                continue

            cursor.execute("""
                INSERT INTO users (
                    id, username, first_name, last_name,
                    job_title, department, city, country
                )
                VALUES (?, ?, ?, ?, ?, ?, ?, ?)
            """, (
                int(row["Id"]),
                row["Username"],
                row["Firstname"],
                row["Lastname"],
                row["Jobtitle"],
                row["Department"],
                row["City"],
                row["Country Or Region"]
```

```
            ))


        except Exception as e:
            print(f"Skipping row due to error: {e}")


conn.commit()
conn.close()


print("User data inserted successfully.")
```



DB Browser for SQLite - C:\Users\admin\OneDrive\Desktop\AccuKnox_AI_ML\Assignment 1\users.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: users | Filter in any column

	id	username	first_name	last_name	job_title	department	city	country
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1000200	hris.Green@kronos.com	Chris	Green	It Manager	Information Technology	Redmond	United States
2	1000201	Ben.Andrews@kronos.com	Ben	Andrews	It Manager	Information Technology	Redmond	United States
3	1000202	David.Longmuir@kronos.com	David	Longmuir	It Manager	Information Technology	Redmond	United States
4	1000203	Cynthia.Carey@kronos.com	Cynthia	Carey	It Manager	Information Technology	Redmond	United States
5	1000204	Melissa.Macbeth@kronos.com	Melissa	Macbeth	It Manager	Information Technology	Redmond	United States
6	1000205	Christopher.Greendoor@kronos.com	Christopher	Greendoor	It Manager	Information Technology	Redmond	United States
7	1000206	Benjamin.Reinolds@kronos.com	Benjamin	Reinolds	Software Engineer	Information Technology	Redmond	United States
8	1000207	Danniel.Long@kronos.com	Danniel	Long	Software Engineer	Information Technology	Redmond	United States
9	1000208	Briggeth.Carson@kronos.com	Briggeth	Carson	Software Engineer	Information Technology	Redmond	United States
10	1000209	Mell.Macbeth@kronos.com	Mell	Macbeth	Software Engineer	Information Technology	Redmond	United States
11	1000210	Sturart.Bent@kronos.com	Stuart	Bent	Software Engineer	Information Technology	Redmond	United States
12	1000211	Martha.Smith@kronos.com	Martha	Smith	Software Engineer	Information Technology	Redmond	United States
13	1000212	Alison.Johnson@kronos.com	Alison	Johnson	Software Engineer	Information Technology	Redmond	United States
14	1000213	Elizabeth.Williams@kronos.com	Elizabeth	Williams	Software Engineer	Information Technology	Redmond	United States
15	1000214	Emily.Brown@kronos.com	Emily	Brown	Software Engineer	Information Technology	Redmond	United States
16	1000215	Jane.Jones@kronos.com	Jane	Jones	Qa Tester	Information Technology	Redmond	United States
17	1000216	Jenna.Garcia@kronos.com	Jenna	Garcia	Qa Tester	Information Technology	Redmond	United States
18	1000217	Katherine.Miller@kronos.com	Katherine	Miller	Qa Tester	Information Technology	Redmond	United States
19	1000218	Fait.Davis@kronos.com	Fait	Davis	Qa Tester	Information Technology	Redmond	United States
20	1000219	James.Smith@kronos.com	James	Smith	Qa Tester	Information Technology	Redmond	United States
21	1000220	Robert.Johnson@kronos.com	Robert	Johnson	Qa Tester	Information Technology	Redmond	United States
22	1000221	John.Williams@kronos.com	John	Williams	Qa Tester	Information Technology	Redmond	United States

4. Send a link to the most complex Python code you have written

https://github.com/ZainaPasha/Resume-Analyzer/blob/main/server/routes/ask_question.py

5. Send a link to the most complex database code you have written

   https://github.com/ZainaPasha/Resume-Analyzer/blob/main/server/modules/load_vectorstore.py