exercise-module-43

October 24, 2024

1 Module 43: Advanced Topics on SQLite with Python

Author: Juliho David Castillo Colmenares

In this activity, you are required to propose a set of tables that you can relate to each other, similar to what was demonstrated in this module. Preferably, these tables should be related to your area of work.

The goal is to create multiple relational databases in Python using CSV files and SQLite, allowing you to perform relational queries between them.

1.1 Step-by-Step Instructions:

1.1.1 1. Create a Relational Database:

- Generate a relational database that contains the proposed tables.
- Explain the purpose of each table and describe the meaning of each field (column) in the table.

1.1.2 2. Provide Two Query Examples Using `INNER JOIN` and `LEFT JOIN`:

- Write two specific SQL queries where you apply the `INNER JOIN` and `LEFT JOIN` operations.
- Explain the logic behind each query.

1.1.3 3. Develop a Query Using `CASE WHEN`:

- Create a query that uses the `CASE WHEN` statement.
- Explain the problem this query resolves.

1.1.4 4. Provide Three Subquery Examples:

- Develop three query examples using subqueries (e.g., Semi-Join, Anti-Join, or any other subquery of your choice).
- Explain the logic behind each subquery.

1.2 Final Submission

1.2.1 How to Submit Your Work:

- You should submit a **Jupyter Notebook (.ipynb file)** containing:
  - The code developed for creating the database and running the queries.
  - Explanations for each query, describing its purpose and logic.

- Include the **CSV files** used in the project.
- Provide a **PDF** with screenshots showing:
  - The code execution.
  - The output of the queries.
  - Explanations and comments related to the specific questions asked.

Note: Submissions that do not include these elements will not be evaluated.

1.2.2 Estimated Time for Completion:

- This task should take approximately **40 minutes**.

Submit your files to the tutor via the EBAC platform.

[62]:
```
import sqlite3
import csv
import time

# Connect to the SQLite database (it will be created if it doesn't exist)
now = str(int(time.time()))
conn = sqlite3.connect(f"my_database_{now}.db")
cursor = conn.cursor()


# Function to load data from a CSV file into a table
def load_data_from_csv(csv_file, table_name, columns):
    with open(csv_file, "r") as file:
        dict_reader = csv.DictReader(file)
        to_db = [tuple(row[col] for col in columns) for row in dict_reader]

    cursor.execute(f'CREATE TABLE IF NOT EXISTS {table_name} ({", ".
 ↪join(columns)});')
    cursor.executemany(
        f'INSERT INTO {table_name} ({", ".join(columns)}) VALUES ({", ".join(["?
 ↪" for _ in columns])});',
        to_db,
    )
    conn.commit()


# Define the columns and load data from Customers and Orders
customers_columns = ["customer_id", "customer_name", "address", "city",␣
 ↪"country"]
orders_columns = ["order_id", "customer_id", "order_date", "total_purchase"]

load_data_from_csv("Customers1.csv", "Customers", customers_columns)
load_data_from_csv("Orders.csv", "Orders", orders_columns)
```

[63]:
```
# Query to perform INNER JOIN between Customers and Orders
inner_join_query = """
SELECT Customers.customer_name, Orders.order_date, Orders.total_purchase
FROM Customers
INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;
"""
cursor.execute(inner_join_query)
inner_join_results = cursor.fetchall()

print("INNER JOIN Results:")
# for row in inner_join_results:
#     print(row)
inner_join_results[:5]
```

```
INNER JOIN Results:
```

[63]:
```
[('Juan Perez', '2023-01-15', '250'),
 ('Juan Perez', '2024-08-05', '50'),
 ('Maria Lopez', '2023-02-10', '100'),
 ('Maria Lopez', '2024-09-20', '400'),
 ('Carlos Sanchez', '2023-03-05', '75')]
```

[64]:
```
# Query to perform LEFT JOIN between Customers and Orders
left_join_query = """
SELECT Customers.customer_name, Orders.order_date, Orders.total_purchase
FROM Customers
LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;
"""
cursor.execute(left_join_query)
left_join_results = cursor.fetchall()

print("\nLEFT JOIN Results:")
# for row in left_join_results:
#     print(row)
left_join_results[:5]
```

```

LEFT JOIN Results:
```

[64]:
```
[('Juan Perez', '2023-01-15', '250'),
 ('Juan Perez', '2024-08-05', '50'),
 ('Maria Lopez', '2023-02-10', '100'),
 ('Maria Lopez', '2024-09-20', '400'),
 ('Carlos Sanchez', '2023-03-05', '75')]
```

[65]:
```
# Query to categorize total purchases into Low, Medium, and High categories␣
 ↪using CASE WHEN
case_when_query = """
```

```
SELECT Orders.customer_id,
       CASE
           WHEN Orders.total_purchase < 100 THEN 'Low'
           WHEN Orders.total_purchase BETWEEN 100 AND 500 THEN 'Medium'
           ELSE 'High'
       END AS purchase_category
FROM Orders;
"""
cursor.execute(case_when_query)
case_when_results = cursor.fetchall()

print("\nCASE WHEN Results:")
# for row in case_when_results:
#     print(row)
case_when_results[:5]
```

```
CASE WHEN Results:
```

[65]:
```
[('1', 'High'), ('2', 'High'), ('3', 'High'), ('4', 'High'), ('5', 'High')]
```

[66]:
```
# Query to return customers who have placed at least one order (Semi-Join)
semi_join_query = """
SELECT customer_name
FROM Customers
WHERE EXISTS (SELECT 1 FROM Orders WHERE Customers.customer_id = Orders.
  ↪customer_id);
"""
cursor.execute(semi_join_query)
semi_join_results = cursor.fetchall()

print("\nSemi-Join Results (Customers with orders):")
# for row in semi_join_results:
#     print(row)
semi_join_results[:5]
```

```
Semi-Join Results (Customers with orders):
```

[66]:
```
[('Juan Perez',),
 ('Maria Lopez',),
 ('Carlos Sanchez',),
 ('Ana Gomez',),
 ('Pedro Rodriguez',)]
```

[67]:
```
# Query to return customers who have not placed any orders (Anti-Join)
anti_join_query = """
SELECT customer_name
```

```
FROM Customers
WHERE NOT EXISTS (SELECT 1 FROM Orders WHERE Customers.customer_id = Orders.
 ↪customer_id);
"""
cursor.execute(anti_join_query)
anti_join_results = cursor.fetchall()

print("\nAnti-Join Results:")
# for row in anti_join_results:
#     print(row)
anti_join_results[:5]
```

```
Anti-Join Results:
```

[67]:
```
[]
```

[68]:
```
# Close the SQLite connection
conn.close()
```