Exploring a database with SQL

Matthew Bain

2024-03-22

Table of contents

Imports

```
import sqlite3
import pandas as pd
from IPython.display import Markdown, display
```

SQL

Let us imagine we have the following tables in our database:

1. **Sales**

- `sales_id` (INT)
- `customer_id` (INT)
- `product_id` (INT)
- `sale_date` (DATE)
- `quantity` (INT)
- `total_amount` (DECIMAL)

2. **Customers**

- `customer_id` (INT)
- `customer_name` (VARCHAR)
- `sales_region` (VARCHAR)

- `sign_up_date` (DATE)

3. **Products**

- `product_id` (INT)
- `product_name` (VARCHAR)
- `category` (VARCHAR)
- `price` (DECIMAL)

We can simulate this problem by creating a test database in Python and adding tables to it that match this description.

```python
# Create a database and cursor to query it
conn = sqlite3.connect("testdatabase.db", timeout=10.0)
cursor = conn.cursor()
```

```python
# Create `Sales` table
cursor.execute(
    """
    CREATE TABLE IF NOT EXISTS Sales
    (
        sales_id INTEGER PRIMARY KEY AUTOINCREMENT,
        customer_id INTEGER,
        product_id INTEGER,
        sales_date DATE,
        quantity INTEGER,
        total_amount DECIMAL(10, 2),
        FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
        FOREIGN KEY (product_id) REFERENCES Products(product_id)
    )
    """
)

# Create `Customers` table
cursor.execute(
    """
    CREATE TABLE IF NOT EXISTS Customers
    (
        customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
        customer_name VARCHAR(255) NOT NULL,
        sales_region VARCHAR(255),
        sign_up_date DATE
    )
    """
)

# Create `Products` table
cursor.execute(
    """
    CREATE TABLE IF NOT EXISTS Products
    (
        product_id INTEGER PRIMARY KEY AUTOINCREMENT,
        product_name VARCHAR(255) NOT NULL,
        category VARCHAR(255),
        price DECIMAL(10, 2)
    )
    """
)
```

```
<sqlite3.Cursor at 0x127d3b3c0>
```

```
# Insert record into `Customers` table
query = """
    INSERT INTO Customers
    (customer_name, sales_region, sign_up_date)
    VALUES (?, ?, ?)
"""
values = [
    ("John Doe", "West", "2023-09-25"),
    ("Jane Young", "South", "2024-09-25"),
    ("Chris Nguyen", "West", "2024-09-25"),
]
cursor.executemany(query, values)

# Insert record into `Products` table
query = """
    INSERT INTO Products
    (product_name, category, price)
    VALUES (?, ?, ?)
"""
values = [
    ("Washing machine", "Appliances", 1500.00),
    ("Laptop", "Electronics", 1000.00),
    ("Phone", "Electronics", 800.00),
]
cursor.executemany(query, values)

# Insert record into `Sales` table
query = """
    INSERT INTO Sales
    (customer_id, product_id, sales_date, quantity, total_amount)
    VALUES (?, ?, ?, ?, ?)
"""
values = [
    (1, 1, "2023-09-26", 2, values[0][2] * 2),
    (2, 1, "2023-01-15", 4, values[0][2] * 4),
    (2, 2, "2024-09-20", 3, values[1][2] * 3),
    (3, 3, "2024-09-22", 9, values[2][2] * 9),
    (1, 2, "2023-09-26", 5, values[1][2] * 5),
]
cursor.executemany(query, values)
```

```
<sqlite3.Cursor at 0x127d3b3c0>
```

```
# Query `Sales`
query = """
    SELECT *
    FROM Sales
    LIMIT 5;
"""
display(Markdown("**`Sales`**:"), pd.read_sql(query, conn))

# Query `Customers`
query = """
    SELECT *
    FROM Customers
    LIMIT 5;
"""
display(Markdown("**`Customers`**:"), pd.read_sql(query, conn))

# Query `Products`
query = """
    SELECT *
    FROM Products
    LIMIT 5;
"""
display(Markdown("**`Products`**:"), pd.read_sql(query, conn))
```

`Sales`:

	sales_id	customer_id	product_id	sales_date	quantity	total_amount
0	1	1	1	2023-09-26	2	3000
1	2	2	1	2023-01-15	4	6000
2	3	2	2	2024-09-20	3	3000
3	4	3	3	2024-09-22	9	7200
4	5	1	2	2023-09-26	5	5000

`Customers`:

	customer_id	customer_name	sales_region	sign_up_date
0	1	John Doe	West	2023-09-25
1	2	Jane Young	South	2024-09-25
2	3	Chris Nguyen	West	2024-09-25

Products:

	product_id	product_name	category	price
0	1	Washing machine	Appliances	1500
1	2	Laptop	Electronics	1000
2	3	Phone	Electronics	800

Q1

Write a query to return the customer_name, product_name, and total_amount for each sale in the last 30 days.

```
query = """
    SELECT
        Customers.customer_name,
        Products.product_name,
        Sales.total_amount
    FROM
        Sales
    LEFT JOIN Customers
        ON Sales.customer_id = Customers.customer_id
    LEFT JOIN Products
        ON Sales.product_id = Products.product_id
    WHERE
        Sales.sales_date >= DATE('now', '-30 days');
"""
pd.read_sql(query, conn)
```

	customer_name	product_name	total_amount
0	Jane Young	Laptop	3000
1	Chris Nguyen	Phone	7200

Q2

Write a query to find the total revenue generated by each product category in the last year. The output should include the product category and the total revenue for that category.

```
query = """
    SELECT
        Products.category,
        SUM(Sales.total_amount) AS total_revenue
    FROM
        Sales
    LEFT JOIN Products
        ON Sales.product_id = Products.product_id
    WHERE
        Sales.sales_date >= DATE('now', '-1 year')
    GROUP BY
        category;
"""
pd.read_sql(query, conn)
```

	category	total_revenue
0	Electronics	10200

Q3

Write a query to return all customers who made purchases in 2023 and are located in the "West" region.

```
query = """
    SELECT DISTINCT
        Customers.customer_name
    FROM
        Customers
    INNER JOIN Sales
        ON Customers.customer_id = Sales.customer_id
    WHERE
        strftime('%Y', Sales.sales_date) = '2023'
        AND Customers.sales_region = 'West'
"""
pd.read_sql(query, conn)
```

	customer_name
0	John Doe

Q4

Write a query to display the total number of sales, total quantity sold, and total revenue for each customer. The result should include the customer_name, total sales, total quantity, and total revenue.

```
query = """
    SELECT
        Customers.customer_name,
        COUNT(Sales.sales_id) AS total_sales,
        SUM(Sales.quantity) AS total_quantity,
        SUM(Sales.total_amount) AS total_revenue
    FROM
        Customers
    LEFT JOIN Sales
        ON Sales.customer_id = Customers.customer_id
    GROUP BY
        Customers.customer_id;
"""
pd.read_sql(query, conn)
```

	customer_name	total_sales	total_quantity	total_revenue
0	John Doe	2	7	8000
1	Jane Young	2	7	9000
2	Chris Nguyen	1	9	7200

Q5

Write a query to find the top 3 customers (by total revenue) in the year 2023.

```
query = """
    SELECT
        Customers.customer_name,
        SUM(Sales.total_amount) AS total_revenue
    FROM
        Customers
    LEFT JOIN Sales
        ON Customers.customer_id = Sales.customer_id
        AND strftime('%Y', Sales.sales_date) = '2023'
    GROUP BY
        Customers.customer_name
    ORDER BY
        total_revenue DESC;
"""
pd.read_sql(query, conn)
```

	customer_name	total_revenue
0	John Doe	8000.0
1	Jane Young	6000.0
2	Chris Nguyen	NaN

Q6

Write a query to rank products by their total sales quantity in 2023. The result should include the product_name, total quantity sold, and rank.

```
query = """
    SELECT
        Products.product_name,
        SUM(Sales.quantity) AS total_quantity,
        RANK() OVER
            (ORDER BY SUM(Sales.quantity) DESC) AS quantity_rank
    FROM
        Products
    LEFT JOIN Sales
        ON Products.product_id = Sales.product_id
        AND strftime('%Y', Sales.sales_date) = '2023'
    GROUP BY
        Products.product_name;
"""
pd.read_sql(query, conn)
```

	product_name	total_quantity	quantity_rank
0	Washing machine	6.0	1
1	Laptop	5.0	2
2	Phone	NaN	3

Q7

Write a query that categorizes customers into "New" (if they signed up in the last 6 months) or "Existing" based on their sign_up_date. Include the customer_name, region, and category in the result.

```
query = """
    SELECT DISTINCT
        Customers.customer_name,
        Customers.sales_region,
    CASE
        WHEN
            Customers.sign_up_date >= DATE('now', '-6 months')
        THEN
```

```
            'New'
        ELSE
            'Existing'
    END AS customer_status
    FROM
        Customers
    LEFT JOIN Sales
        ON Customers.customer_id = Sales.customer_id
"""
pd.read_sql(query, conn)
```

	customer_name	sales_region	customer_status
0	John Doe	West	Existing
1	Jane Young	South	New
2	Chris Nguyen	West	New

Q8

Write a query to return the month and year along with the total sales for each month for the last 12 months.

```

```

Wrapping up

```
cursor.close()
conn.close()
```