Python SQL

June 17, 2024

[44]:
```python
import pandas as pd

def execute_queries_to_df():
    try:
        # Establish connection to MySQL
        cnx = mysql.connector.connect(**config)

        # List of SQL queries
        queries = [
            """
            -- 1. What devices do my customers use to reach me?
            SELECT
                Device_Type,
                COUNT(Device_Type) AS Count
            FROM
                retail
            GROUP BY
                Device_Type;
            """,
            """
            -- 2. What product categories am I selling?
            SELECT
                DISTINCT Product_Category,
                SUM(Quantity) AS Total_Quantity
            FROM
                retail
            GROUP BY
                Product_Category;
            """,
            """
            -- 3. Which product categories do I sell to whom?
            SELECT
                Product_Category,
                Gender,
                COUNT(Gender) AS Count
            FROM
                retail
            GROUP BY
                Product_Category, Gender;
            """,
            """
            -- 4. Which login type do my customers prefer when shopping?
            SELECT
                Customer_Login_type,
                COUNT(Customer_Login_type) AS Count
            FROM
                retail
            GROUP BY
                Customer_Login_type;
            """,
            """
            -- 5. How does the date and time affect my sales? (Total sales by␣
↪month)
            SELECT
                Order_Date,
                SUM(Sales) AS Total_Sales
            FROM
                retail
            GROUP BY
                Order_Date
            ORDER BY
                SUM(Sales) DESC
            limit 10;
            """,
            """
            -- 6. Top month quantity sales
            SELECT
                MONTH(Order_Date) AS Month,
                SUM(Sales) AS Total_Sales
            FROM
                retail
            GROUP BY
                MONTH(Order_Date)
            ORDER BY
                SUM(Sales) DESC;
            """,
            """
            -- 7. Top week quantity sales
            SELECT
                CONCAT(DATE_FORMAT(MIN(Order_Date), "%Y-%m-%d"), " to ",␣
↪DATE_FORMAT(DATE_ADD(MIN(Order_Date), INTERVAL 6 DAY), "%Y-%m-%d")) AS␣
↪Week_Dates,
                SUM(Sales) AS Total_Sales
            FROM
```

```
        retail
    GROUP BY
        WEEK(Order_Date)
    ORDER BY
        SUM(Sales) DESC;
    """,
    """
    -- 8. Average delivery time by hour of day
    SELECT
        HOUR(Time) AS Hour_of_Day,
        AVG(Aging) AS Avg_Delivery_Time
    FROM
        retail
    GROUP BY
        HOUR(Time)
    ORDER BY
        HOUR(Time);
    """,
    """
    -- 9. From which product do I earn the most profit per unit?
    SELECT
        Product,
        SUM(Profit) / SUM(Quantity) AS Total_Profit_per_Unit
    FROM
        retail
    GROUP BY
        Product
    ORDER BY
        Total_Profit_per_Unit DESC;
    """,
    """
    -- 10. Total profit per product
    SELECT
        Product,
        SUM(Profit) AS Total_Profit
    FROM
        retail
    GROUP BY
        Product
    ORDER BY
        Total_Profit DESC;
    """,
    """
    -- 11. Average delivery time by month and order priority
    SELECT
        Order_Priority,
        MONTH(Order_Date) AS Month,
                AVG(Aging) AS Avg_Delivery_Time
            FROM
                retail
            GROUP BY
                MONTH(Order_Date), Order_Priority;
            """
        ]

        # Execute each query and store results in DataFrames
        dfs = []
        for index, query in enumerate(queries, start=1):
            print(f"Executing Query {index}:")
            print(query.strip())  # Print the query for visibility

            # Execute query and fetch results into DataFrame
            df = pd.read_sql_query(query, cnx)
            dfs.append(df)
            # Print DataFrame using tabulate for nice formatting
            print(tabulate(df, headers='keys', tablefmt='fancy_grid'))
            print("\n")  # Separate queries for clarity

        return dfs

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        # Close connection
        if 'cnx' in locals() and cnx:
            cnx.close()

# Execute the queries and get DataFrames
if __name__ == "__main__":
    data_frames = execute_queries_to_df()

    # Access individual DataFrames from the list 'data_frames'
    for idx, df in enumerate(data_frames, start=1):
        print("\n")
```

```
Executing Query 1:
-- 1. What devices do my customers use to reach me?
            SELECT
                Device_Type,
                COUNT(Device_Type) AS Count
            FROM
                retail
            GROUP BY
                Device_Type;
```

```
      Device_Type      Count

   0  Web              47626

   1  Mobile            3658



Executing Query 2:
-- 2. What product categories am I selling?
            SELECT
                DISTINCT Product_Category,
                SUM(Quantity) AS Total_Quantity
            FROM
                retail
            GROUP BY
                Product_Category;
C:\Users\natha\AppData\Local\Temp\ipykernel_1720\599513356.py:144: UserWarning:
pandas only supports SQLAlchemy connectable (engine/connection) or database
string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested.
Please consider using SQLAlchemy.
  df = pd.read_sql_query(query, cnx)

      Product_Category      Total_Quantity

   0  Auto & Accessories             17582

   1  Fashion                        66639

   2  Electronic                      5951

   3  Home & Furniture               38190



Executing Query 3:
-- 3. Which product categories do I sell to whom?
            SELECT
                Product_Category,
                Gender,
                COUNT(Gender) AS Count
            FROM
                retail
            GROUP BY
                Product_Category, Gender;
```

```
    Product_Category    Gender       Count

 0  Auto & Accessories  Female        2322

 1  Auto & Accessories  Male          5177

 2  Fashion             Female       11365

 3  Fashion             Male         14281

 4  Electronic          Female        1484

 5  Electronic          Male          1217

 6  Home & Furniture    Male          7458

 7  Home & Furniture    Female        7980



Executing Query 4:
-- 4. Which login type do my customers prefer when shopping?
            SELECT
                Customer_Login_type,
                COUNT(Customer_Login_type) AS Count
            FROM
                retail
            GROUP BY
                Customer_Login_type;

    Customer_Login_type     Count

 0  Member                  49091

 1  Guest                    1993

 2  New                        27

 3  First SignUp              173



Executing Query 5:
-- 5. How does the date and time affect my sales? (Total sales by month)
            SELECT
                Order_Date,
                SUM(Sales) AS Total_Sales
            FROM
                retail
            GROUP BY
                Order_Date
            ORDER BY
                SUM(Sales) DESC
            limit 10;

       Order_Date      Total_Sales

    0  2018-04-24            94531

    1  2018-07-30            72191

    2  2018-06-30            49882

    3  2018-04-25            43157

    4  2018-05-16            43036

    5  2018-11-07            41837

    6  2018-11-02            40918

    7  2018-11-16            40255

    8  2018-11-06            40246

    9  2018-05-14            39390



    Executing Query 6:
    -- 6. Top month quantity sales
            SELECT
                MONTH(Order_Date) AS Month,
                SUM(Sales) AS Total_Sales
            FROM
                retail
            GROUP BY
                MONTH(Order_Date)
            ORDER BY
                SUM(Sales) DESC;

         Month    Total_Sales

    0       11         877881

    1        5         824362
```

```
   2         7          810205

   3        12          767147

   4        10          743387

   5         9          738303

   6         8          664245

   7         6          642501

   8         4          596990

   9         3          435502

  10         1          379627

  11         2          332495
```

```
Executing Query 7:
-- 7. Top week quantity sales
            SELECT
                CONCAT(DATE_FORMAT(MIN(Order_Date), "%Y-%m-%d"), " to ",
DATE_FORMAT(DATE_ADD(MIN(Order_Date), INTERVAL 6 DAY), "%Y-%m-%d")) AS
Week_Dates,
                SUM(Sales) AS Total_Sales
            FROM
                retail
            GROUP BY
                WEEK(Order_Date)
            ORDER BY
                SUM(Sales) DESC;
```

```
      Week_Dates                  Total_Sales

   0  2018-04-22 to 2018-04-28         273418

   1  2018-11-04 to 2018-11-10         250264

   2  2018-11-11 to 2018-11-17         230012

   3  2018-05-13 to 2018-05-19         219888

   4  2018-05-06 to 2018-05-12         217716
```

5	2018-07-15 to 2018-07-21	199171
6	2018-07-22 to 2018-07-28	199163
7	2018-04-29 to 2018-05-05	196935
8	2018-10-07 to 2018-10-13	195966
9	2018-12-16 to 2018-12-22	192053
10	2018-09-23 to 2018-09-29	185985
11	2018-11-18 to 2018-11-24	185789
12	2018-10-28 to 2018-11-03	184853
13	2018-07-29 to 2018-08-04	183975
14	2018-09-30 to 2018-10-06	175324
15	2018-09-02 to 2018-09-08	172820
16	2018-09-16 to 2018-09-22	172471
17	2018-08-05 to 2018-08-11	172362
18	2018-06-24 to 2018-06-30	170122
19	2018-12-02 to 2018-12-08	169786
20	2018-12-23 to 2018-12-29	169124
21	2018-12-09 to 2018-12-15	168526
22	2018-10-14 to 2018-10-20	167865
23	2018-08-19 to 2018-08-25	165749
24	2018-10-21 to 2018-10-27	160403
25	2018-08-12 to 2018-08-18	159763
26	2018-07-01 to 2018-07-07	159564
27	2018-09-09 to 2018-09-15	159070
28	2018-06-17 to 2018-06-23	158150

29	2018-06-10 to 2018-06-16	155454
30	2018-07-08 to 2018-07-14	154002
31	2018-05-20 to 2018-05-26	143191
32	2018-06-03 to 2018-06-09	136049
33	2018-03-04 to 2018-03-10	129871
34	2018-11-25 to 2018-12-01	124592
35	2018-02-25 to 2018-03-03	119345
36	2018-05-27 to 2018-06-02	115674
37	2018-04-15 to 2018-04-21	111539
38	2018-03-11 to 2018-03-17	110262
39	2018-08-26 to 2018-09-01	103338
40	2018-01-07 to 2018-01-13	93913
41	2018-02-11 to 2018-02-17	93481
42	2018-04-08 to 2018-04-14	91395
43	2018-01-14 to 2018-01-20	84400
44	2018-01-01 to 2018-01-07	80936
45	2018-01-21 to 2018-01-27	80611
46	2018-02-18 to 2018-02-24	77089
47	2018-04-01 to 2018-04-07	74322
48	2018-03-25 to 2018-03-31	71649
49	2018-01-28 to 2018-02-03	69563
50	2018-03-18 to 2018-03-24	69267
51	2018-02-04 to 2018-02-10	67237
52	2018-12-30 to 2019-01-05	39178

```
Executing Query 8:
-- 8. Average delivery time by hour of day
            SELECT
                HOUR(Time) AS Hour_of_Day,
                AVG(Aging) AS Avg_Delivery_Time
            FROM
                retail
            GROUP BY
                HOUR(Time)
            ORDER BY
                HOUR(Time);
```

```
        Hour_of_Day    Avg_Delivery_Time

  0               0              5.23788

  1               1              5.16949

  2               2              5.29482

  3               3              4.92647

  4               4              5.22642

  5               5              5.33333

  6               6              5.28689

  7               7              5

  8               8              5.27103

  9               9              5.27431

 10              10              5.25085

 11              11              5.21592

 12              12              5.2873

 13              13              5.22584

 14              14              5.20285

 15              15              5.28571
 16            16             5.16953

 17            17             5.32151

 18            18             5.3686

 19            19             5.32071

 20            20             5.21283

 21            21             5.28421

 22            22             5.23622

 23            23             5.28102
```

```
Executing Query 9:
-- 9. From which product do I earn the most profit per unit?
            SELECT
                Product,
                SUM(Profit) / SUM(Quantity) AS Total_Profit_per_Unit
            FROM
                retail
            GROUP BY
                Product
            ORDER BY
                Total_Profit_per_Unit DESC;

      Product                    Total_Profit_per_Unit

   0  Apple Laptop                            67.124

   1  Tyre                                    65.5511

   2  Iron                                    57.2136

   3  T - Shirts                              56.9196

   4  Car Pillow & Neck Rest                  53.5939

   5  Samsung Mobile                          53.0311

   6  Towels                                  52.0572

   7  Running Shoes                           47.6744
```

8	Titak watch	47.4445
9	Jeans	46.5618
10	LED	46.4667
11	Sofa Covers	46.4487
12	Car Speakers	46.2254
13	Tablet	45.5361
14	Bed Sheets	45.1068
15	Formal Shoes	43.1184
16	Shirts	38.2698
17	Fossil Watch	25.0036
18	Fans	23.0358
19	Car Media Players	21.4352
20	Speakers	17.0554
21	Dinner Crockery	16.7415
22	Bike Tyres	14.0069
23	Beds	13.7135
24	Umbrellas	13.4822
25	Shoe Rack	13.2967
26	LCD	12.4192
27	Sofas	12.2509
28	Casula Shoes	11.9129
29	Car & Bike Care	11.7072
30	Dinning Tables	11.1838
31	Sneakers	11.0466

```
 32   Car Mat                                10.8526

 33   Car Body Covers                        10.6029

 34   Car Seat Covers                        10.5522

 35   Mixer/Juicer                           10.4425

 36   Mouse                                   9.67182

 37   Suits                                   7.47687

 38   Watch                                   6.88454

 39   Keyboard                                6.76057

 40   Sports Wear                             6.39816

 41   Curtains                                6.25492
```

```
Executing Query 10:
-- 10. Total profit per product
            SELECT
                Product,
                SUM(Profit) AS Total_Profit
            FROM
                retail
            GROUP BY
                Product
            ORDER BY
                Total_Profit DESC;

      Product                  Total_Profit

  0   T - Shirts                   340721

  1   Titak watch                  296718

  2   Running Shoes                289098

  3   Jeans                        276856

  4   Formal Shoes                 265351

  5   Shirts                       230078
```

6	Towels	196828
7	Sofa Covers	178921
8	Bed Sheets	172263
9	Fossil Watch	151272
10	Tyre	132020
11	Car Pillow & Neck Rest	107885
12	Car Speakers	92034.8
13	Casula Shoes	71894.5
14	Sneakers	66820.7
15	Dinner Crockery	64287.5
16	Beds	53592.5
17	Shoe Rack	50886.5
18	Umbrellas	49789.6
19	Sofas	46283.9
20	Suits	44831.3
21	Dinning Tables	43326
22	Car Media Players	39933.8
23	Sports Wear	38984
24	Apple Laptop	33025
25	Iron	26833.2
26	Bike Tyres	26767.2
27	Samsung Mobile	26568.6
28	Curtains	23881.3
29	Car & Bike Care	22700.3

```
 30   Tablet                          22312.7

 31   Car Body Covers                 21629.9

 32   LED                             20910

 33   Car Mat                         20782.8

 34   Car Seat Covers                 20006.9

 35   Fans                            12047.7

 36   Speakers                         9909.2

 37   LCD                              6197.2

 38   Mixer/Juicer                     5033.3

 39   Mouse                            4632.8

 40   Watch                            3428.5

 41   Keyboard                         3292.4
```

```
Executing Query 11:
-- 11. Average delivery time by month and order priority
            SELECT
                Order_Priority,
                MONTH(Order_Date) AS Month,
                AVG(Aging) AS Avg_Delivery_Time
            FROM
                retail
            GROUP BY
                MONTH(Order_Date), Order_Priority;

      Order_Priority       Month     Avg_Delivery_Time

   0  Medium                   1               5.37568

   1  Medium                   7               5.32444

   2  Critical                11               4.60163

   3  High                     4               5.15346
```

4	Critical	8	4.64832
5	Critical	7	4.69643
6	High	5	5.08793
7	Critical	6	4.76807
8	Critical	5	4.6988
9	High	10	5.19919
10	High	12	5.15712
11	Medium	10	5.35697
12	High	3	5.27528
13	Critical	1	4.75253
14	Critical	2	4.48077
15	Critical	12	4.87432
16	High	8	5.05762
17	High	6	5.15431
18	Critical	10	4.59649
19	Medium	8	5.3769
20	High	11	5.12278
21	Critical	9	4.92
22	Medium	3	5.42831
23	Critical	4	4.72026
24	Medium	4	5.43089
25	Medium	9	5.33035
26	High	7	5.23566
27	High	2	5.16797

28	High	9	5.12042
29	High	1	5.19407
30	Critical	3	4.6036
31	Medium	5	5.34116
32	Medium	6	5.40629
33	Medium	12	5.43907
34	Medium	11	5.29146
35	Medium	2	5.32441
36		10	2
37		7	4
38	Low	6	5.34783
39	Low	1	6
40	Low	10	5.13821
41	Low	7	5.43426
42	Low	9	5.64583
43	Low	2	5.35849
44	Low	5	5.46058
45	Low	12	5.46552
46	Low	4	5.86364
47	Low	8	5.44643
48	Low	3	5.3209
49	Low	11	5.24615

```
[ ]:
```