Sales_Data_Analysis

April 25, 2025

1 Sales Data Analysis

This project analyzes sales transaction data to uncover business insights and patterns. Using Python and MySQL, I explored order records to identify top-performing products, geographic sales trends, and key financial metrics. The analysis includes data cleaning and SQL queries for aggregations to support data-driven decision-making.

Key Insights:
- Calculated total revenue and profit margins
- Identified best-selling products and cities
- Analyzed sales distributions and order trends
- Verified data quality through null checks

Tools Used: Python, MySQL, Pandas, SQL Queries

1.1 About the Dataset

The dataset contains **6 columns** representing attributes of product purchases:

- **Order ID** - Unique identifier for each order
- **Product** - Name of the purchased item
- **Quantity Ordered** - Number of units ordered
- **Price Each** - Price per unit of product
- **Sales** - Calculated (Quantity Ordered × Price Each)
- **City** - Extracted from Purchase Address

[4]:
```
#import library

import mysql.connector
from mysql.connector import Error
import pandas as pd
```

[6]:
```
# ============================================
# Database Connection and Query Functions
# ============================================
```

```python
def create_server_connection(host_name, user_name, user_password):
    """Create a connection to MySQL server instance (without specific database).
    """
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")  # Handle connection errors gracefully
    return connection

def create_database(connection, query):
    """Create a new database using provided SQL query."""
    cursor = connection.cursor()
    try:
        cursor.execute(query)  # Execute CREATE DATABASE statement
        print("Database created successfully")
    except Error as err:
        print(f"Error: '{err}'")  # Handle database creation errors

def create_db_connection(host_name, user_name, user_password, db_name):
    """Create connection to a SPECIFIC MySQL database."""
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name  # Connect to specified database
        )
        print(f"MySQL connection to database '{db_name}' successful")
    except Error as err:
        print(f"Error: '{err}'")
    return connection

def execute_query(connection, query):
    """Execute WRITE operations (INSERT, UPDATE, DELETE) and persist changes."""
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()  # Commit transaction for data modifications
        print("Query executed successfully")
    except Error as err:
```

```
        print(f"Error: '{err}'")  # Rollback happens automatically on error

def read_query(connection, query):
    """Execute READ operations (SELECT) and return results."""
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()  # Get all records from query
        return result
    except Error as err:
        print(f"Error: '{err}'")  # Handle query execution errors
```

[8]:
```
# ================================================
# Main Script Execution
# ================================================
if __name__ == "__main__":
    # Database credentials
    pw = "******"  # MySQL password (replace with your actual password)
    db = "sales"     # Name of the database we'll work with

    # Step 1: Connect to MySQL server (no specific database selected yet)
    connection = create_server_connection("localhost", "root", pw)

    # Step 2: Create new database if it doesn't already exist
    create_database_query = "CREATE DATABASE IF NOT EXISTS sales"
    create_database(connection, create_database_query)

    # Step 3: Reconnect to specifically use the 'sales' database
    connection = create_db_connection("localhost", "root", pw, "sales")
```

```
MySQL Database connection successful
Database created successfully
MySQL connection to database 'sales' successful
```

[10]:
```
try:
    # ==================== DATABASE INFO ====================
    # Show all databases on the server
    print("\n=== Databases ===")
    databases = read_query(connection, "SHOW DATABASES;")
    for db in databases:
        print(db[0])  # Extract database name from tuple

    # Show tables in current database
    print("\n=== Tables ===")
    tables = read_query(connection, "SHOW TABLES;")
    for table in tables:
        print(table[0])  # Extract table name from tuple
```

```
    # ==================== TABLE STRUCTURE ====================
    # Display column details for sales_info table
    print("\n=== Table Structure ===")
    describe = read_query(connection, "DESCRIBE sales_info;")
    df_describe = pd.DataFrame(describe, columns=["Field", "Type", "Null",␣
↪"Key", "Default", "Extra"])
    print(df_describe)


    # ==================== DATA QUALITY ====================
    # Check for missing values
    print("\n=== Null Values Check ===")
    null_check = read_query(connection, """
        SELECT *
        FROM sales_info
        WHERE `Order ID` IS NULL
           OR Product IS NULL
           OR `Quantity Ordered` IS NULL
           OR `Price Each` IS NULL
           OR Sales IS NULL
           OR City IS NULL;
    """)
    df_null_check = pd.DataFrame(null_check, columns=["Order ID", "Product",␣
↪"Quantity Ordered", "Price Each", "Sales", "City"])
    print(df_null_check)

    # ==================== DATA EXPLORATION ====================
    # Get all records from sales_info table
    print("\n=== All Data ===")
    all_data = read_query(connection, "SELECT * FROM sales_info;")
    df_all = pd.DataFrame(all_data, columns=["Order ID", "Product", "Quantity␣
↪Ordered", "Price Each", "Sales", "City"])
    print(df_all)

    # ==================== SALES ANALYSIS ====================
    # Show products with total sales
    print("\n=== Total Sales by Product ===")
    total_sales = read_query(connection, """
        SELECT product, ROUND(SUM(sales), 2) AS total_sales
        FROM sales_info
        GROUP BY product
    """)
    df_total_sales = pd.DataFrame(total_sales, columns=["Product", "Total␣
↪Sales"])
    print(df_total_sales)
```

```
    # ==================== GEOGRAPHICAL ANALYSIS ====================
    # Get unique cities from sales data
    print("\n=== Distinct Cities ===")
    cities = read_query(connection, "SELECT DISTINCT city FROM sales_info;")
    for city in cities:
        print(city[0])

    # Count products sold per city
    print("\n=== Product Count by City ===")
    product_count = read_query(connection, """
        SELECT city, COUNT(product)
        FROM sales_info
        GROUP BY city
        ORDER BY COUNT(product) DESC;
    """)
    df_product_count = pd.DataFrame(product_count, columns=["City", "Product␣
↪Count"])
    print(df_product_count)

    # ==================== INVENTORY ANALYSIS ====================
    # Total quantities sold per product
    print("\n=== Total Item Sales by Product ===")
    total_item_sales = read_query(connection, """
        SELECT product, SUM(`Quantity Ordered`) AS total_item_sales
        FROM sales_info
        GROUP BY product
        ORDER BY total_item_sales DESC;
    """)
    df_total_item_sales = pd.DataFrame(total_item_sales, columns=["Product",␣
↪"Total Item Sales"])
    print(df_total_item_sales)

    # Top 5 best-selling products
    print("\n=== Top 5 Products by Quantity ===")
    top_products = read_query(connection, """
        SELECT Product, SUM(`Quantity Ordered`) AS TotalQuantity
        FROM sales_info
        GROUP BY Product
        ORDER BY TotalQuantity DESC
        LIMIT 5;
    """)
    df_top_products = pd.DataFrame(top_products, columns=["Product", "Total␣
↪Quantity"])
    print(df_top_products)

    # ==================== ADVANCED ANALYTICS ====================
    # Window functions for ranking sales performance
```

```
    print("\n=== Window Functions ===")
    window_query = """
        SELECT
            `Order ID`,
            `Product`,
            `Quantity Ordered`,
            `Price Each`,
            `Sales`,
            ROW_NUMBER() OVER (PARTITION BY `Product` ORDER BY `Quantity␣
↪Ordered` DESC) AS RowNum,
            RANK() OVER (PARTITION BY `Product` ORDER BY `Quantity Ordered`␣
↪DESC) AS Qrank,
            FIRST_VALUE(`Quantity Ordered`) OVER (PARTITION BY `Product` ORDER␣
↪BY `Quantity Ordered` DESC) AS FirstQuant
        FROM sales_info;
    """
    window_data = read_query(connection, window_query)
    df_window = pd.DataFrame(window_data, columns=["Order ID", "Product",␣
↪"Quantity Ordered", "Price Each", "Sales", "RowNum", "Qrank", "FirstQuant"])
    print(df_window.head())  # Show sample of ranked data

    # ==================== CITY-SPECIFIC ANALYSIS ====================
    # Filter sales for New York City
    print("\n=== Sales in New York City ===")
    ny_sales = read_query(connection, "SELECT * FROM sales_info WHERE City =␣
↪'New York City';")
    df_ny_sales = pd.DataFrame(ny_sales, columns=["Order ID", "Product",␣
↪"Quantity Ordered", "Price Each", "Sales", "City"])
    print(df_ny_sales)

    # Sales totals per city
    print("\n=== Total Sales by City ===")
    total_sales_by_city = read_query(connection, """
        SELECT City, ROUND(SUM(Sales), 2) AS TotalSales
        FROM sales_info
        GROUP BY City
        ORDER BY TotalSales DESC;
    """)
    df_total_sales_by_city = pd.DataFrame(total_sales_by_city, columns=["City",␣
↪"Total Sales"])
    print(df_total_sales_by_city)

    # ==================== FINANCIAL ANALYSIS ====================
    # Calculate profit metrics
    print("\n=== Profit Calculation ===")
    profit_query = """
```

```
        SELECT
            `Order ID`,
            Product,
            Sales,
            ROUND((Sales - Sales * 0.85), 2) AS Profit,
            ROUND(((Sales - Sales * 0.85) / Sales), 2) * 100 AS ProfitMargin
        FROM sales_info;
    """
    profit_data = read_query(connection, profit_query)
    df_profit = pd.DataFrame(profit_data, columns=["Order ID", "Product",␣
↪"Sales", "Profit", "Profit Margin"])
    print(df_profit)

    # ==================== ORDER ANALYSIS ====================
    # Count orders per city
    print("\n=== Order Count by City ===")
    order_count = read_query(connection, """
        SELECT City, COUNT(*) AS OrderCount
        FROM sales_info
        GROUP BY City
        ORDER BY OrderCount DESC;
    """)
    df_order_count = pd.DataFrame(order_count, columns=["City", "Order Count"])
    print(df_order_count)

    # ==================== SUMMARY METRICS ====================
    # Calculate total revenue
    print("\n=== Total Revenue ===")
    total_revenue = read_query(connection, "SELECT ROUND(SUM(Sales), 2) AS␣
↪TotalRevenue FROM sales_info;")
    print(f"Total Revenue: ${total_revenue[0][0]}")

    # Find top-selling products per city
    print("\n=== Maximum Quantity Sold by City ===")
    max_quantity_by_city = read_query(connection, """
        SELECT
            s1.City,
            s1.Product,
            s1.TotalQuantity
        FROM (
            SELECT
                City,
                Product,
                SUM(`Quantity Ordered`) AS TotalQuantity
            FROM sales_info
            GROUP BY City, Product
        ) AS s1
```

```
        JOIN (
            SELECT
                City,
                MAX(TotalQuantity) AS MaxQuantity
            FROM (
                SELECT
                    City,
                    Product,
                    SUM(`Quantity Ordered`) AS TotalQuantity
                FROM sales_info
                GROUP BY City, Product
            ) AS sub
            GROUP BY City
        ) AS s2
        ON s1.City = s2.City AND s1.TotalQuantity = s2.MaxQuantity;
    """)
    df_max_quantity_by_city = pd.DataFrame(max_quantity_by_city,␣
 ↪columns=["City", "Product", "Total Quantity"])
    print(df_max_quantity_by_city)

except Error as e:
    print(f"Error during execution: {e}")

finally:
    # Cleanup: Close database connection
    if connection:
        connection.close()
        print("\nMySQL connection closed.")
```

```
=== Databases ===
classicmodels
information_schema
mysql
mysql_python
performance_schema
sales
sql_intro
sql_iq
sql_joins
sys
test
triggers
world

=== Tables ===
sales_info
```

```
=== Table Structure ===
              Field    Type Null Key Default Extra
0          Order ID     int  YES        None
1           Product    text  YES        None
2  Quantity Ordered     int  YES        None
3        Price Each  double  YES        None
4             Sales  double  YES        None
5              City    text  YES        None

=== Null Values Check ===
Empty DataFrame
Columns: [Order ID, Product, Quantity Ordered, Price Each, Sales, City]
Index: []

=== All Data ===
        Order ID                   Product  Quantity Ordered  Price Each  \
0         295665        Macbook Pro Laptop                 1     1700.00
1         295666        LG Washing Machine                 1      600.00
2         295667      USB-C Charging Cable                 1       11.95
3         295668          27in FHD Monitor                 1      149.99
4         295669      USB-C Charging Cable                 1       11.95
…           …                         …                …          …
159072    223845           ThinkPad Laptop                 1      999.99
159073    223846  Lightning Charging Cable                 1       14.95
159074    223847  Lightning Charging Cable                 1       14.95
159075    223848    AAA Batteries (4-pack)                 1        2.99
159076    223849          Wired Headphones                 2       11.99

          Sales            City
0       1700.00   New York City
1        600.00   New York City
2         11.95   New York City
3        149.99   San Francisco
4         11.95         Atlanta
…         …              …
159072   999.99     Los Angeles
159073    14.95     Los Angeles
159074    14.95     Los Angeles
159075     2.99   San Francisco
159076    23.98          Austin

[159077 rows x 6 columns]

=== Total Sales by Product ===
                      Product  Total Sales
0          Macbook Pro Laptop   6835700.00
1          LG Washing Machine    338400.00
2        USB-C Charging Cable    245978.80
```

```
3             27in FHD Monitor    963535.76
4        AA Batteries (4-pack)     90647.04
5   Bose SoundSport Headphones   1149285.06
6       AAA Batteries (4-pack)     79755.26
7              ThinkPad Laptop   3514964.85
8     Lightning Charging Cable    296742.55
9                 Google Phone   2851200.00
10            Wired Headphones    211143.90
11    Apple Airpods Headphones   2004300.00
12             Vareebadd Phone    711600.00
13                      iPhone   4088000.00
14                20in Monitor    385184.98
15      34in Ultrawide Monitor   2016986.92
16                Flatscreen TV   1224000.00
17      27in 4K Gaming Monitor   2081766.62
18                    LG Dryer    333000.00

=== Distinct Cities ===
 New York City
 San Francisco
 Atlanta
 Portland
 Dallas
 Los Angeles
 Boston
 Austin
 Seattle

=== Product Count by City ===
             City  Product Count
0   San Francisco          38125
1     Los Angeles          25342
2   New York City          21288
3          Boston          17111
4          Dallas          12725
5         Atlanta          12702
6         Seattle          12643
7        Portland          10672
8          Austin           8469

=== Total Item Sales by Product ===
                      Product Total Item Sales
0       AAA Batteries (4-pack)            26674
1        AA Batteries (4-pack)            23606
2         USB-C Charging Cable            20584
3     Lightning Charging Cable            19849
4             Wired Headphones            17610
5     Apple Airpods Headphones            13362
```

```
6   Bose SoundSport Headphones            11494
7              27in FHD Monitor             6424
8                        iPhone             5840
9       27in 4K Gaming Monitor              5338
10      34in Ultrawide Monitor              5308
11                Google Phone              4752
12               Flatscreen TV              4080
13          Macbook Pro Laptop              4021
14             ThinkPad Laptop              3515
15                20in Monitor              3502
16             Vareebadd Phone              1779
17          LG Washing Machine               564
18                     LG Dryer               555

=== Top 5 Products by Quantity ===
                    Product Total Quantity
0    AAA Batteries (4-pack)          26674
1     AA Batteries (4-pack)          23606
2      USB-C Charging Cable          20584
3  Lightning Charging Cable          19849
4          Wired Headphones          17610

=== Window Functions ===
   Order ID       Product  Quantity Ordered  Price Each   Sales  RowNum  \
0    255337  20in Monitor                 2      109.99  219.98       1
1    252442  20in Monitor                 2      109.99  219.98       2
2    252998  20in Monitor                 2      109.99  219.98       3
3    142940  20in Monitor                 2      109.99  219.98       4
4    141926  20in Monitor                 2      109.99  219.98       5

   Qrank  FirstQuant
0      1           2
1      1           2
2      1           2
3      1           2
4      1           2

=== Sales in New York City ===
Empty DataFrame
Columns: [Order ID, Product, Quantity Ordered, Price Each, Sales, City]
Index: []

=== Total Sales by City ===
             City  Total Sales
0   San Francisco   7035133.34
1     Los Angeles   4633807.92
2   New York City   4003120.22
3          Boston   3135613.36
```

```
4          Dallas   2381587.61
5         Atlanta   2376893.19
6         Seattle   2336167.94
7        Portland   1986621.63
8          Austin   1533246.53

=== Profit Calculation ===
        Order ID                    Product    Sales  Profit  Profit Margin
0         295665         Macbook Pro Laptop  1700.00  255.00           15.0
1         295666         LG Washing Machine   600.00   90.00           15.0
2         295667       USB-C Charging Cable    11.95    1.79           15.0
3         295668           27in FHD Monitor   149.99   22.50           15.0
4         295669       USB-C Charging Cable    11.95    1.79           15.0
…           …                          …      …      …             …
159072    223845            ThinkPad Laptop   999.99  150.00           15.0
159073    223846  Lightning Charging Cable    14.95    2.24           15.0
159074    223847  Lightning Charging Cable    14.95    2.24           15.0
159075    223848    AAA Batteries (4-pack)     2.99    0.45           15.0
159076    223849           Wired Headphones    23.98    3.60           15.0

[159077 rows x 5 columns]

=== Order Count by City ===
             City  Order Count
0   San Francisco        38125
1     Los Angeles        25342
2   New York City        21288
3          Boston        17111
4          Dallas        12725
5         Atlanta        12702
6         Seattle        12643
7        Portland        10672
8          Austin         8469

=== Total Revenue ===
Total Revenue: $29422191.74

=== Maximum Quantity Sold by City ===
             City                 Product Total Quantity
0          Dallas  AAA Batteries (4-pack)           2201
1   San Francisco  AAA Batteries (4-pack)           6314
2     Los Angeles  AAA Batteries (4-pack)           4308
3         Atlanta  AAA Batteries (4-pack)           2049
4        Portland  AAA Batteries (4-pack)           1753
5   New York City  AAA Batteries (4-pack)           3547
6          Austin  AAA Batteries (4-pack)           1448
7          Boston  AAA Batteries (4-pack)           2964
8         Seattle  AAA Batteries (4-pack)           2090
```

```
MySQL connection closed.
```

1.2 Reference

1. Beekiran. (n.d.). Sales data analysis. Kaggle. Retrieved October 2023, from https://www.kaggle.com/datasets/beekiran/sales-data-analysis

[11]:
```
!jupyter nbconvert --to pdf Sales_Data_Analysis.ipynb
```

```
[NbConvertApp] Converting notebook Sales_Data_Analysis.ipynb to pdf
[NbConvertApp] Writing 59180 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | b had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 67364 bytes to Sales_Data_Analysis.pdf
```

[ ]:
```

```