1. **Setup and Imports**

from sqlalchemy import create\_engine, text

import pandas as pd

**create\_engine**: Establishes a connection to the database.

**text**: Used to write raw SQL queries.

**pandas**: Library for data manipulation and analysis, used here to handle query results and generate data frames.

1. **Database Connection**

# URL-encode the password

password = "joshin%402016"

# Establish connection to the database

engine = create\_engine(f"mysql+mysqlconnector://root:{password}@localhost/joshinmasaiproject")

**create\_engine**: Connects to the MySQL database using SQLAlchemy. The URL is encoded to handle special characters in the password.

**3. Data Extraction Functions**

Each function performs a specific SQL query to extract and analyze data.

**a. Total Purchases**

def calculate\_total\_purchases(connection):

    query = text("SELECT COUNT(DISTINCT TransactionID) AS total\_purchases FROM transactions;")

    result = connection.execute(query)

    total\_purchases = result.scalar()

    return total\_purchases

**SQL Query**: Counts distinct transaction IDs.

**result.scalar()**: Retrieves a single scalar result from the query

b. **Total Revenue**

def calculate\_total\_revenue(connection):

    query = text("SELECT SUM(PurchaseQuantity \* PurchasePrice) AS total\_revenue FROM transactions;")

    result = connection.execute(query)

    total\_revenue = result.scalar()

    return total\_revenue

**SQL Query**: Calculates total revenue by summing the product of purchase quantity and price

c. **Average Purchase Value**

def calculate\_average\_purchase\_value(total\_revenue, total\_purchases):

    return total\_revenue / total\_purchases

**Calculation**: Computes the average purchase value by dividing total revenue by the total number of purchases.

d. **Top Customers**

def get\_top\_customers(connection):

    query = text("""

        SELECT c.CustomerID, c.CustomerName, SUM(t.PurchaseQuantity \* t.PurchasePrice) AS total\_revenue

        FROM transactions t

        JOIN customers c ON t.CustomerID = c.CustomerID

        GROUP BY c.CustomerID, c.CustomerName

        ORDER BY total\_revenue DESC

        LIMIT 10;

    """)

    result = connection.execute(query)

    top\_customers = pd.DataFrame(result.fetchall(), columns=['CustomerID', 'CustomerName', 'total\_revenue'])

    return top\_customers

**SQL Query**: Retrieves the top 10 customers based on total revenue.

**pd.DataFrame**: Converts query results into a DataFrame for easier manipulation and viewing

e. **Monthly Trends**

def get\_monthly\_trends(connection):

    query = text("""

        SELECT YEAR(PurchaseDate) AS year, MONTH(PurchaseDate) AS month,

               SUM(PurchaseQuantity \* PurchasePrice) AS total\_revenue

        FROM transactions

        GROUP BY year, month

        ORDER BY year, month;

    """)

    result = connection.execute(query)

    monthly\_trends = pd.DataFrame(result.fetchall(), columns=['year', 'month', 'total\_revenue'])

    return monthly\_trends

SQL Query: Analyzes purchase trends on a monthly basis.

pd.DataFrame: Converts query results into a DataFrame.

f. **Quarterly Trends**

def get\_quarterly\_trends(connection):

    query = text("""

        SELECT YEAR(PurchaseDate) AS year, QUARTER(PurchaseDate) AS quarter,

               SUM(PurchaseQuantity \* PurchasePrice) AS total\_revenue

        FROM transactions

        GROUP BY year, quarter

        ORDER BY year, quarter;

    """)

    result = connection.execute(query)

    quarterly\_trends = pd.DataFrame(result.fetchall(), columns=['year', 'quarter', 'total\_revenue'])

    return quarterly\_trends

SQL Query: Analyzes purchase trends on a quarterly basis.

pd.DataFrame: Converts query results into a DataFrame.

g. **Yearly Trends**

def get\_yearly\_trends(connection):

    query = text("""

        SELECT YEAR(PurchaseDate) AS year,

               SUM(PurchaseQuantity \* PurchasePrice) AS total\_revenue

        FROM transactions

        GROUP BY year

        ORDER BY year;

    """)

    result = connection.execute(query)

    yearly\_trends = pd.DataFrame(result.fetchall(), columns=['year', 'total\_revenue'])

    return yearly\_trends

SQL Query: Analyzes purchase trends on a yearly basis.

pd.DataFrame: Converts query results into a DataFrame

h. **Top Product Categories**

def get\_top\_categories(connection):

    query = text("""

        SELECT p.ProductCategory, SUM(t.PurchaseQuantity \* t.PurchasePrice) AS total\_revenue

        FROM transactions t

        JOIN products p ON t.ProductID = p.ProductID

        GROUP BY p.ProductCategory

        ORDER BY total\_revenue DESC

        LIMIT 10;

    """)

    result = connection.execute(query)

    top\_categories = pd.DataFrame(result.fetchall(), columns=['ProductCategory', 'total\_revenue'])

    return top\_categories

**4.Main Script**

# Main script

def main():

    with engine.connect() as connection:

        total\_purchases = calculate\_total\_purchases(connection)

        total\_revenue = calculate\_total\_revenue(connection)

        average\_purchase\_value = calculate\_average\_purchase\_value(total\_revenue, total\_purchases)

        print(f"Total Purchases: {total\_purchases}")

        print(f"Total Revenue: {total\_revenue}")

        print(f"Average Purchase Value: {average\_purchase\_value}")

        top\_customers = get\_top\_customers(connection)

        print("Top Customers and Their Purchasing Behavior:")

        print(top\_customers)

        monthly\_trends = get\_monthly\_trends(connection)

        print("Monthly Trends:")

        print(monthly\_trends)

        quarterly\_trends = get\_quarterly\_trends(connection)

        print("Quarterly Trends:")

        print(quarterly\_trends)

        yearly\_trends = get\_yearly\_trends(connection)

        print("Yearly Trends:")

        print(yearly\_trends)

        top\_categories = get\_top\_categories(connection)

        print("Top-Performing Product Categories:")

        print(top\_categories)

        generate\_summary\_report(total\_purchases, total\_revenue, average\_purchase\_value, top\_customers, monthly\_trends, quarterly\_trends, yearly\_trends, top\_categories)

main(): Connects to the database, runs all the analysis functions, and prints the results.

generate\_summary\_report(): Provides a comprehensive summary report based on the extracted data.

5. **Summary Report Function**

# Summary Report with Key Insights

def generate\_summary\_report(total\_purchases, total\_revenue, average\_purchase\_value, top\_customers, monthly\_trends, quarterly\_trends, yearly\_trends, top\_categories):

    summary\_report = {

        "Total Purchases": total\_purchases,

        "Total Revenue": total\_revenue,

        "Average Purchase Value": average\_purchase\_value,

        "Top Customers": top\_customers,

        "Monthly Trends": monthly\_trends,

        "Quarterly Trends": quarterly\_trends,

        "Yearly Trends": yearly\_trends,

        "Top-Performing Product Categories": top\_categories

    }

    print("Summary Report with Key Insights:")

    for key, value in summary\_report.items():

        print(f"{key}:")

        print(value)

        print("\n")

generate\_summary\_report(): Organizes the results from all analysis functions into a summary report and prints it.

Output:

Total Purchases: 1000

Total Revenue: 1485760.55

Average Purchase Value: 1485.76055

Top Customers and Their Purchasing Behavior:

CustomerID CustomerName total\_revenue

0 467 Ashley Parrish 18054.83

1 108 Eddie Mueller 11734.37

2 335 Sean Clay 11682.13

3 328 Nicole Good 11632.91

4 168 Justin Arnold 9437.78

5 635 David Thompson 9046.70

6 762 Robert Walker 8904.06

7 529 Taylor Small 8477.97

8 808 Michelle Martin 8357.65

9 699 Samuel Ruiz 8295.40

Monthly Trends:

year month total\_revenue

0 2023 6 8735.67

1 2023 7 140528.32

2 2023 8 98276.78

3 2023 9 143595.40

4 2023 10 124279.18

5 2023 11 122373.42

6 2023 12 155534.30

7 2024 1 134083.39

8 2024 2 128590.54

9 2024 3 113325.83

10 2024 4 102830.12

11 2024 5 133782.95

12 2024 6 79824.65

Quarterly Trends:

year quarter total\_revenue

0 2023 2 8735.67

1 2023 3 382400.50

2 2023 4 402186.90

3 2024 1 375999.76

4 2024 2 316437.72

Yearly Trends:

year total\_revenue

0 2023 793323.07

1 2024 692437.48

Top-Performing Product Categories:

ProductCategory total\_revenue

0 Electronics 769154.14

1 Home Appliances 716606.41

Summary Report with Key Insights:

Total Purchases:

1000

Total Revenue:

1485760.55

Average Purchase Value:

1485.76055

Top Customers:

CustomerID CustomerName total\_revenue

0 467 Ashley Parrish 18054.83

1 108 Eddie Mueller 11734.37

2 335 Sean Clay 11682.13

3 328 Nicole Good 11632.91

4 168 Justin Arnold 9437.78

5 635 David Thompson 9046.70

6 762 Robert Walker 8904.06

7 529 Taylor Small 8477.97

8 808 Michelle Martin 8357.65

9 699 Samuel Ruiz 8295.40

Monthly Trends:

year month total\_revenue

0 2023 6 8735.67

1 2023 7 140528.32

2 2023 8 98276.78

3 2023 9 143595.40

4 2023 10 124279.18

5 2023 11 122373.42

6 2023 12 155534.30

7 2024 1 134083.39

8 2024 2 128590.54

9 2024 3 113325.83

10 2024 4 102830.12

11 2024 5 133782.95

12 2024 6 79824.65

Quarterly Trends:

year quarter total\_revenue

0 2023 2 8735.67

1 2023 3 382400.50

2 2023 4 402186.90

3 2024 1 375999.76

4 2024 2 316437.72

Yearly Trends:

year total\_revenue

0 2023 793323.07

1 2024 692437.48

Top-Performing Product Categories:

ProductCategory total\_revenue

0 Electronics 769154.14

1 Home Appliances 716606.41