Project: Python Implementation

Group 2

Aniruthan SA

Aravind
Anbazhagan

617-238-8857 (Tel of Aniruthan)
629-899-0339 (Tel of Aravind)

swaminathanarulmur.a@northeastern.edu
anbazhagan.a@northeastern.edu

Percentage of Effort Contributed by Studentl: 50%

Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: Aniruthan SA

Signature of Student 2: Aravind Anbazhagan

Submission Date: 11/26/2023

Python Implementation:

The provided Python script is designed to connect to a MySQL database and generate various visualizations using Matplotlib, a plotting library. The script performs the following tasks:

1. Database Connection:

  - It establishes a connection to a MySQL database named 'dma_project' using `mysql.connector`.

  - Connection parameters like `host`, `user`, `password`, and `database` are specified in `db_config`.

  - If the connection is successful, it prints a confirmation message; otherwise, it displays an error and exits the script.

2. Query Execution Function (`execute_query`):

  - This function takes an SQL query as an input and executes it.

  - It returns the fetched results from the database or prints an error message if the query execution fails.

3. Visualization Function (`query_and_plot`):

  - This function executes a provided SQL query to fetch data from the `Tickets` table, grouped by a specified column (`case_status`, `issue_type`, or `resolution_code`).

  - It then plots the results as a bar graph using Matplotlib, showing the distribution of tickets based on the specified column.

  - The function is called three times to create three separate plots for `case_status`, `issue_type`, and `resolution_code`.

4. Employee Distribution Plot (`plot_employee_distribution`):

  - This function queries the distribution of employees across different assignment groups, joining the `Employees` and `Team_PDL` tables.

  - It visualizes the results as a bar graph, showing the number of employees in each assignment group.

5. Closing the Database Connection:

  - After executing all functions and plotting the graphs, the script closes the database connection.

This script effectively demonstrates how to integrate SQL querying with data visualization in Python. It's a useful approach for analyzing and presenting data stored in a database in a more accessible and understandable format.

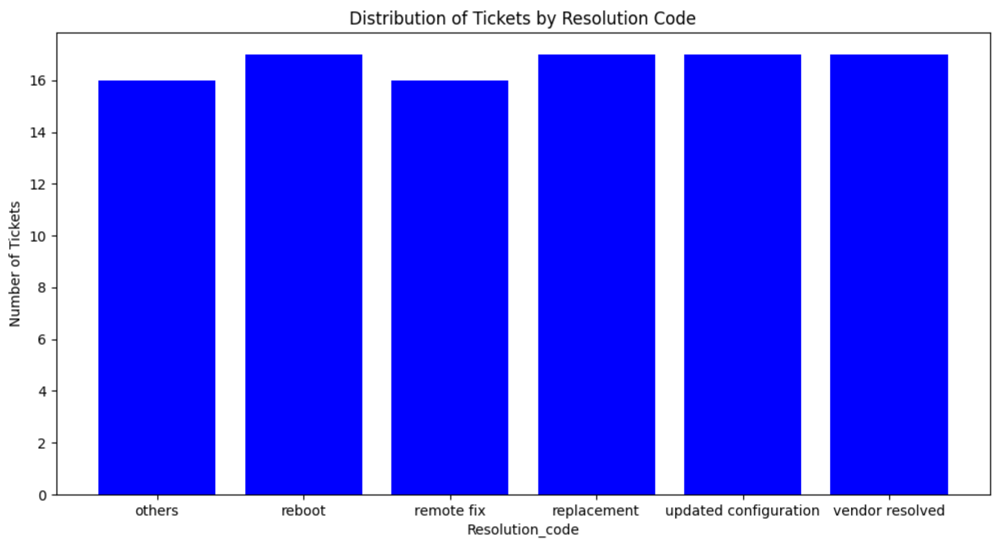
Script:

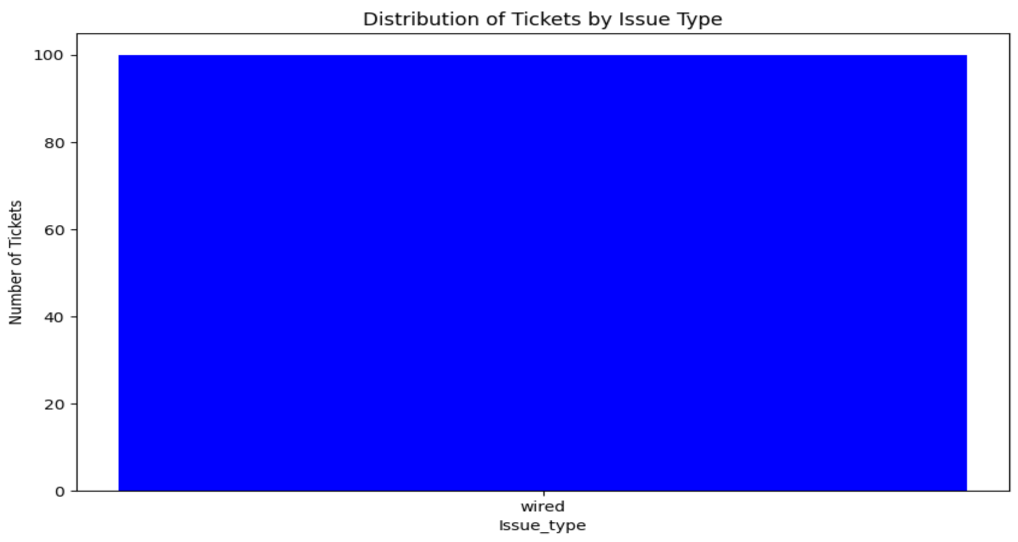
```
import mysql.connector
import matplotlib.pyplot as plt
```

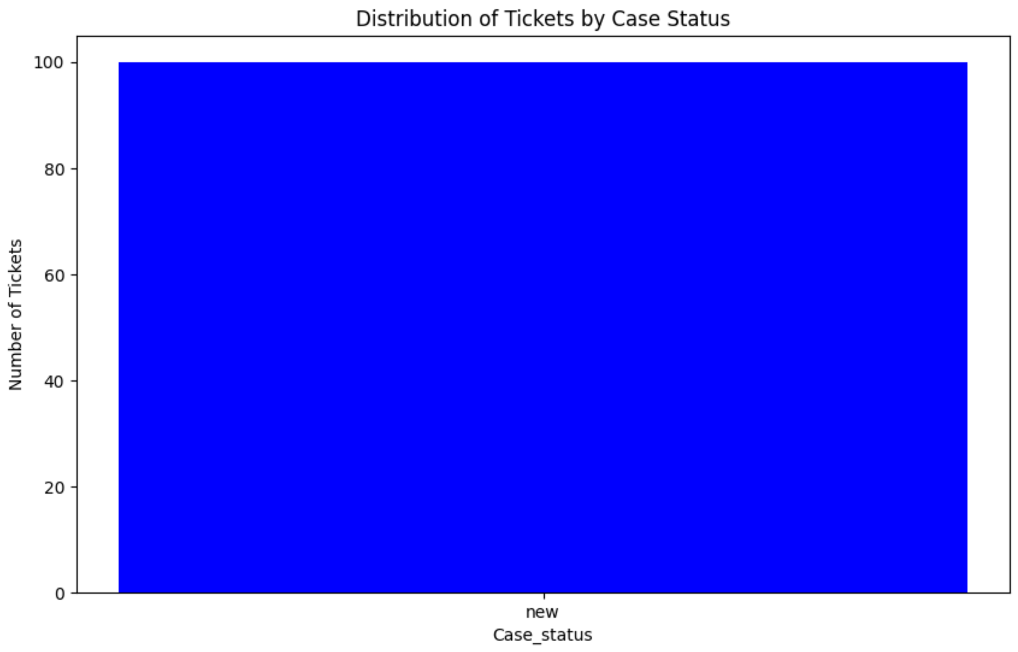
```python
# Database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'admin123',
    'database': 'dma_project'
}

# Connect to the MySQL database
try:
    db_connection = mysql.connector.connect(**db_config)
    cursor = db_connection.cursor()
    print("Successfully connected to the database.")
except Exception as e:
    print(f"Error connecting to the database: {e}")
    exit()

# Function to execute a query and return results
def execute_query(query):
    try:
        cursor.execute(query)
        return cursor.fetchall()
    except Exception as e:
        print(f"Error executing the query: {e}")
        return None

# Query and plot function for a specific column
def query_and_plot(column_name, plot_title):
    query = f"""
    SELECT {column_name}, COUNT(*) AS count
    FROM Tickets
    GROUP BY {column_name};
    """
    results = execute_query(query)
    if results:
        labels = [row[0] for row in results]
        counts = [row[1] for row in results]

        plt.figure(figsize=(10, 6))
        plt.bar(labels, counts, color='blue')
        plt.xlabel(column_name.capitalize())
        plt.ylabel('Number of Tickets')
        plt.title(plot_title)
        plt.show()

# Plotting distribution of tickets based on case_status
query_and_plot("case_status", "Distribution of Tickets by Case Status")

# Plotting distribution of tickets based on issue_type
query_and_plot("issue_type", "Distribution of Tickets by Issue Type")

# Plotting distribution of tickets based on resolution_code
query_and_plot("resolution_code", "Distribution of Tickets by Resolution
Code")

# Query and plot for the distribution of employees in assignment groups
def plot_employee_distribution():
    query = """
    SELECT Team_PDL.assignment_group, COUNT(Employees.emp_id) AS
employee_count
```

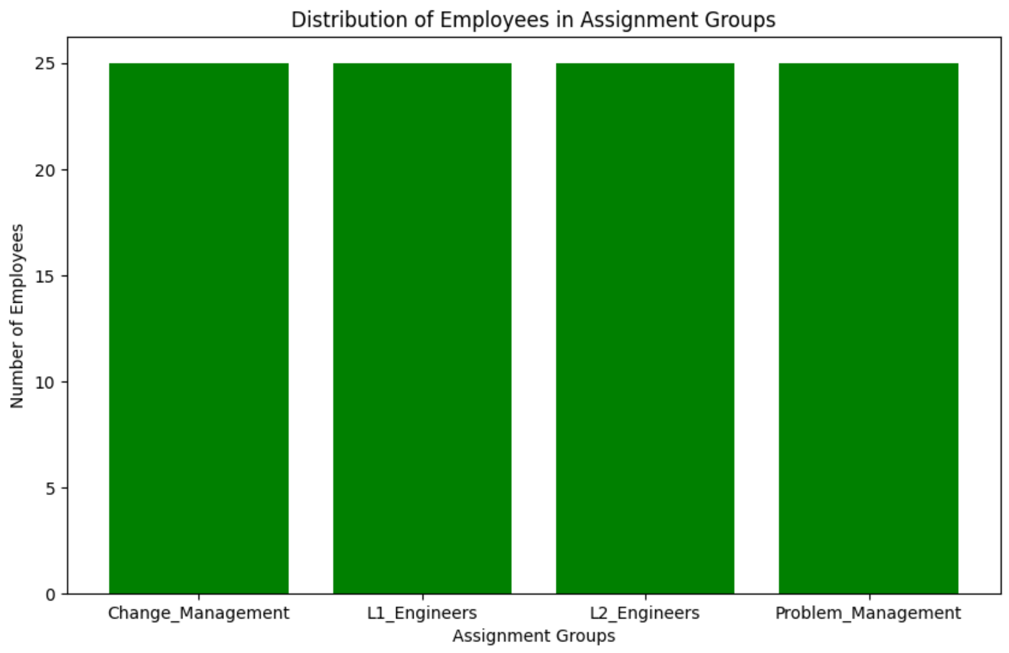
```
    FROM Employees
    JOIN Team_PDL ON Employees.team_email = Team_PDL.team_email
    GROUP BY Team_PDL.assignment_group;
    """
    results = execute_query(query)
    if results:
        groups = [row[0] for row in results]
        employee_counts = [row[1] for row in results]

        plt.figure(figsize=(10, 6))
        plt.bar(groups, employee_counts, color='green')
        plt.xlabel('Assignment Groups')
        plt.ylabel('Number of Employees')
        plt.title('Distribution of Employees in Assignment Groups')
        plt.show()

plot_employee_distribution()

# Close the database connection
db_connection.close()
```

Graphs




Distribution of Tickets by Issue Type
Number of Tickets
0
20
40
60
80
100
wired
Issue_type


Distribution of Tickets by Case Status
Number of Tickets
0
20
40
60
80
100
new
Case_status


Distribution of Employees in Assignment Groups
Number of Employees
25
20
15
10
5
0
Change_Management
L1_Engineers
L2_Engineers
Problem_Management
Assignment Groups