code-milestone

November 23, 2024

1 Building Web-based Applications with MERNStack

1.1 Milestone: Implementation using Python

1.1.1 Group 13

Team Members: - Harini Prasad Vasisht
- Phone: +1(984)374-4836
- Email: vasisht.h@northeastern.edu

- **Sushmitha Sudharsan**
  - Phone: +1(857)565-8800
  - Email: sudharsan.s@northeastern.edu

1.1.2 Contribution Breakdown:

- **Percentage of Effort Contributed by Student 1 (Harini Prasad Vasisht):** 50%
- **Percentage of Effort Contributed by Student 2 (Sushmitha Sudharsan):** 50%

1.1.3 Submission Date:

11/23/2024

[1]:
```
%pip install mysql-connector-python
```

```
Note: you may need to restart the kernel to use updated packages.Requirement
already satisfied: mysql-connector-python in c:\users\sushmitha sudharsan\appdat
a\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\l
ocal-packages\python311\site-packages (9.1.0)



[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: C:\Users\Sushmitha Sudharsan\AppData\Local\Microsoft\Wi
ndowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip
install --upgrade pip
```

[2]:
```
import mysql.connector
```

[3]:

```python
# Database connection details
host = "localhost"
port = 3306
user = "root"
password = "Sairam@sush2002"
database = "notes_management"

# Connect to the database
conn = mysql.connector.connect(
    host=host,
    port=port,
    user=user,
    password=password,
    database=database
)
cursor = conn.cursor()
```

[4]:

```
%pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\sushmitha sudharsan\appdata\lo
cal\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local
-packages\python311\site-packages (5.24.1)Note: you may need to restart the
kernel to use updated packages.

Requirement already satisfied: tenacity>=6.2.0 in c:\users\sushmitha sudharsan\a
ppdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localca
che\local-packages\python311\site-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in c:\users\sushmitha sudharsan\appdata
\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\lo
cal-packages\python311\site-packages (from plotly) (23.1)


[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: C:\Users\Sushmitha Sudharsan\AppData\Local\Microsoft\Wi
ndowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe -m pip
install --upgrade pip
```

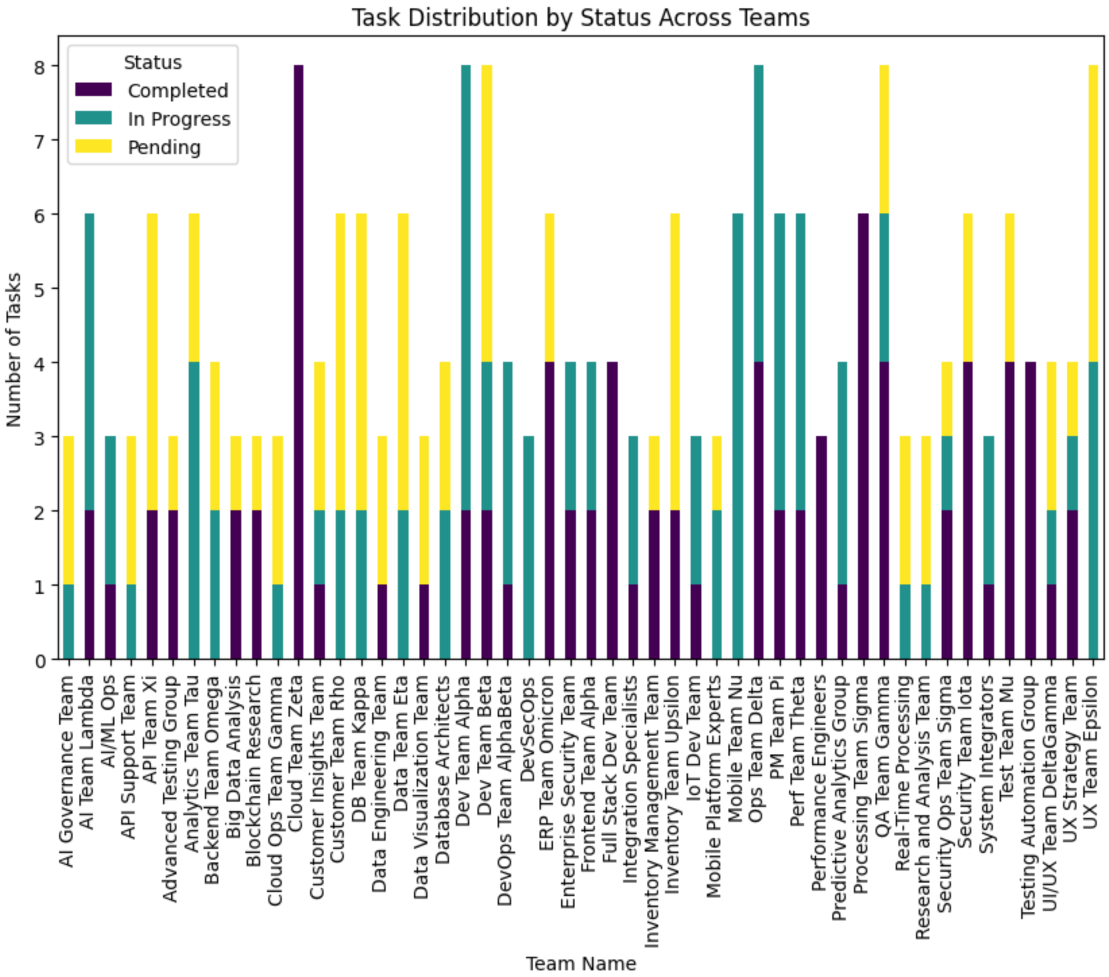
[5]:

```python
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
C:\Users\Sushmitha Sudharsan\AppData\Local\Temp\ipykernel_19444\2236774888.py:2:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

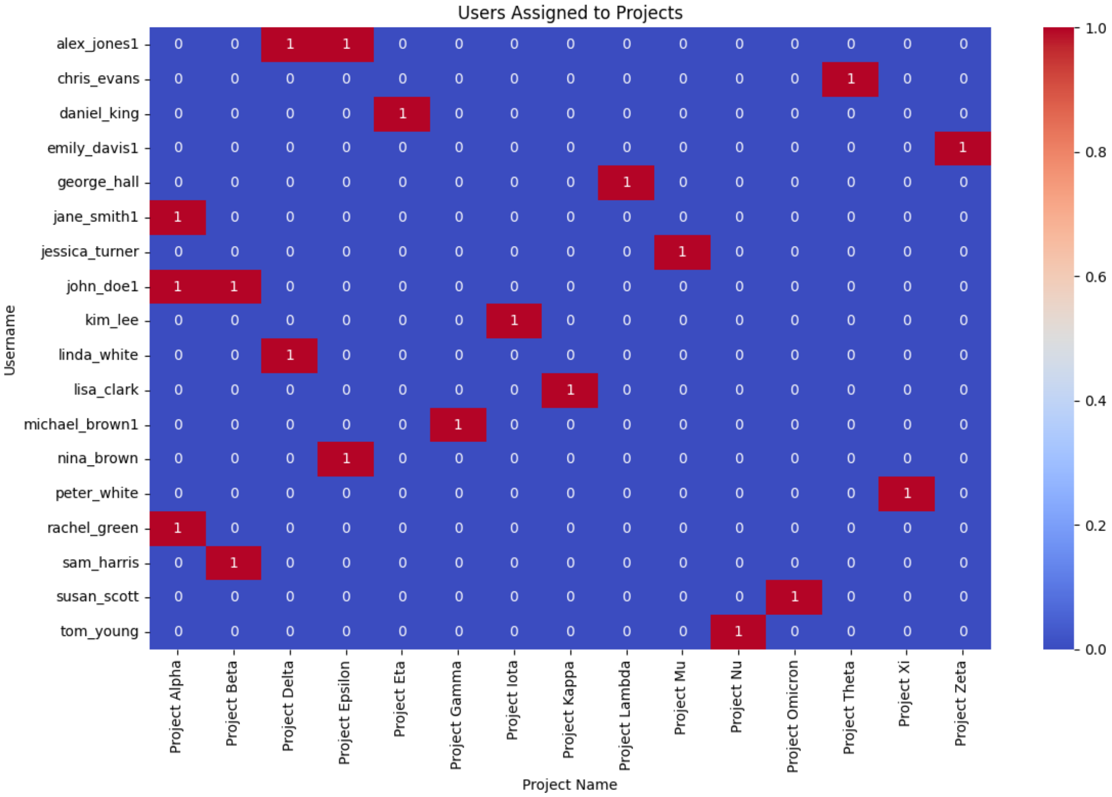
[6]:

```
query = """
SELECT t.team_name, ta.status, COUNT(ta.task_id) AS task_count
FROM Team t
JOIN Task ta ON t.project_id = ta.project_id
GROUP BY t.team_name, ta.status;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_team_tasks = pd.DataFrame(data, columns=['team_name', 'status',␣
 ↪'task_count'])

# Visualization: Stacked Bar Chart
df_pivot = df_team_tasks.pivot(index='team_name', columns='status',␣
 ↪values='task_count').fillna(0)
df_pivot.plot(kind='bar', stacked=True, figsize=(10, 6), colormap="viridis")
plt.title("Task Distribution by Status Across Teams")
plt.xlabel("Team Name")
plt.ylabel("Number of Tasks")
plt.legend(title="Status")
plt.show()
```



[7]:
```
query = """
SELECT u.username, p.project_name, COUNT(ut.task_id) AS task_count
FROM User u
JOIN User_Task ut ON u.user_id = ut.user_id
JOIN Task t ON ut.task_id = t.task_id
JOIN Project p ON t.project_id = p.project_id
GROUP BY u.username, p.project_name;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_user_projects = pd.DataFrame(data, columns=['username', 'project_name',␣
 ↪'task_count'])

# Visualization: Heatmap
```

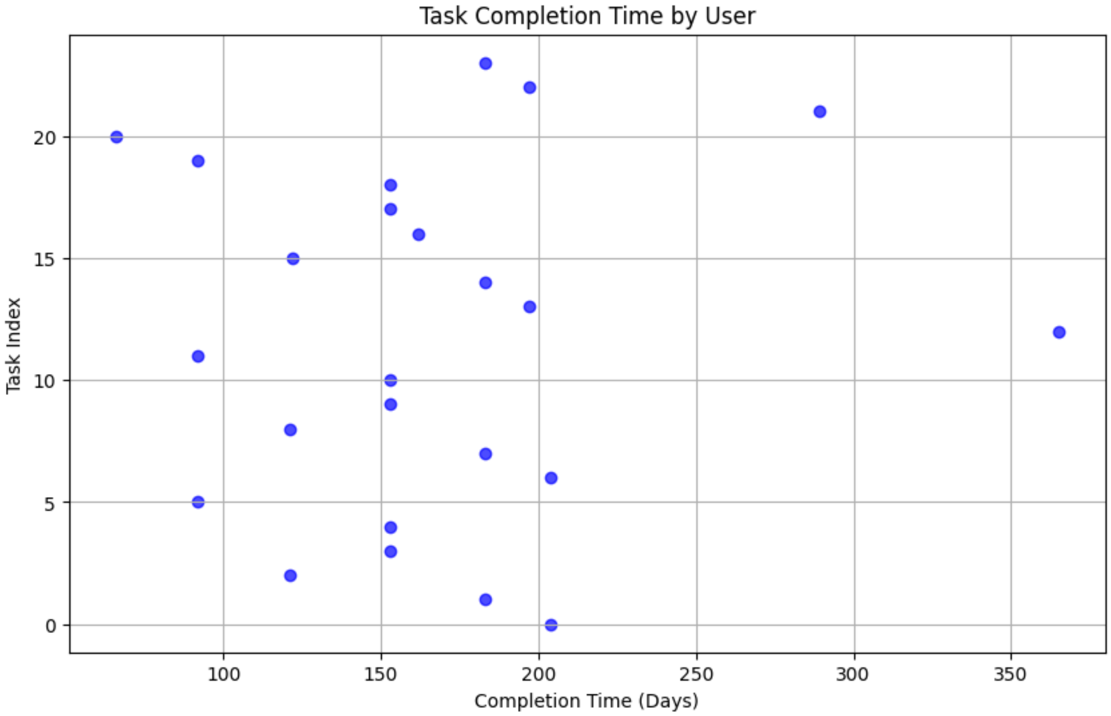
```
df_pivot = df_user_projects.pivot(index='username', columns='project_name',␣
 ↪values='task_count').fillna(0)
plt.figure(figsize=(12, 8))
sns.heatmap(df_pivot, annot=True, cmap="coolwarm", fmt=".0f")
plt.title("Users Assigned to Projects")
plt.xlabel("Project Name")
plt.ylabel("Username")
plt.tight_layout()
plt.show()
```



[8]:

```
query = """
SELECT t.title, t.status, TIMESTAMPDIFF(DAY, p.start_date, p.end_date) AS␣
 ↪completion_time, u.username
FROM Task t
JOIN User u ON t.user_id = u.user_id
JOIN Project p ON t.project_id = p.project_id
WHERE t.status = 'Completed';
"""
cursor.execute(query)
data = cursor.fetchall()
```

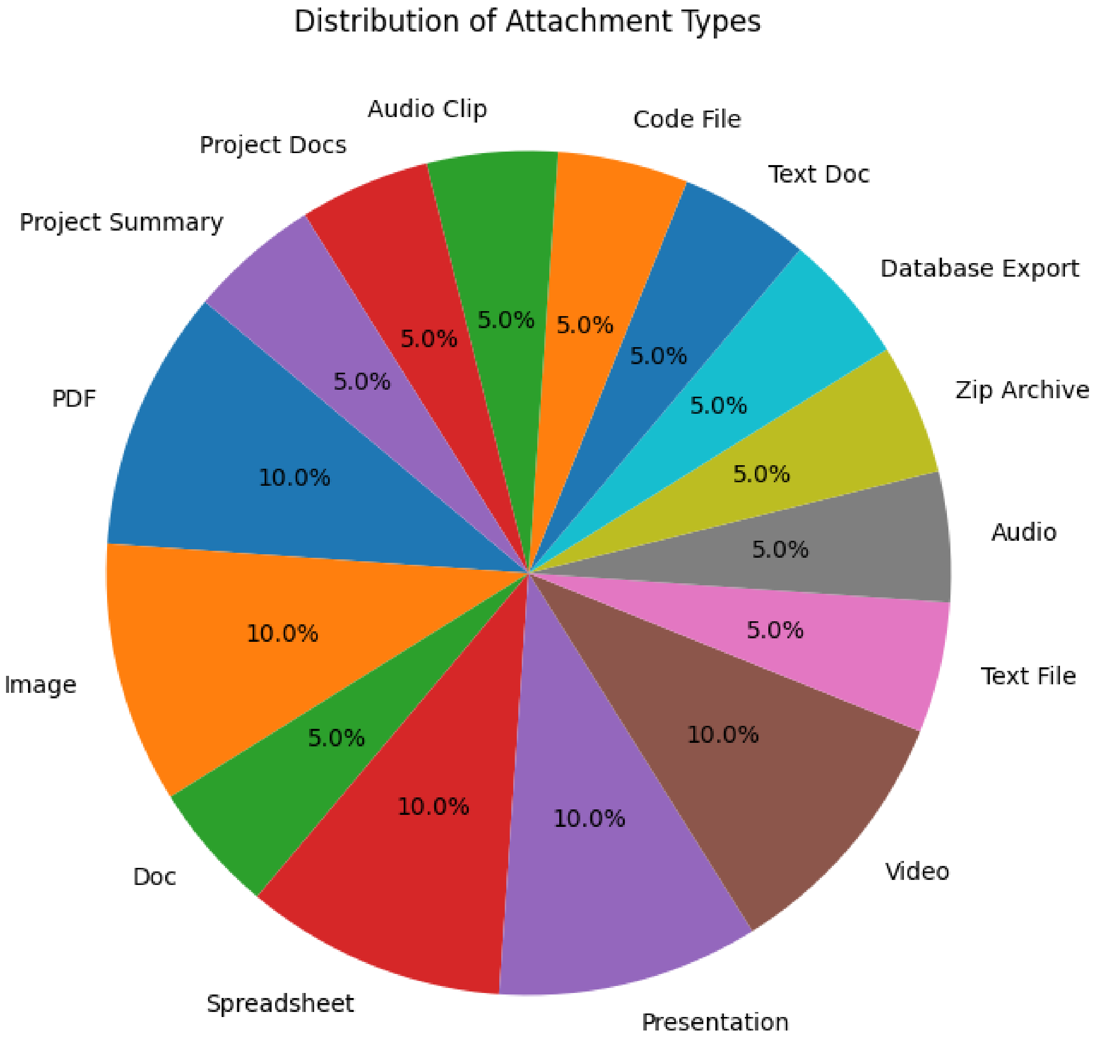
```
# Convert to DataFrame
df_completion = pd.DataFrame(data, columns=['title', 'status',␣
 ↪'completion_time', 'username'])

# Visualization: Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(df_completion['completion_time'], df_completion.index, alpha=0.7,␣
 ↪c='blue')
plt.title("Task Completion Time by User")
plt.xlabel("Completion Time (Days)")
plt.ylabel("Task Index")
plt.grid(True)
plt.show()
```



[9]:

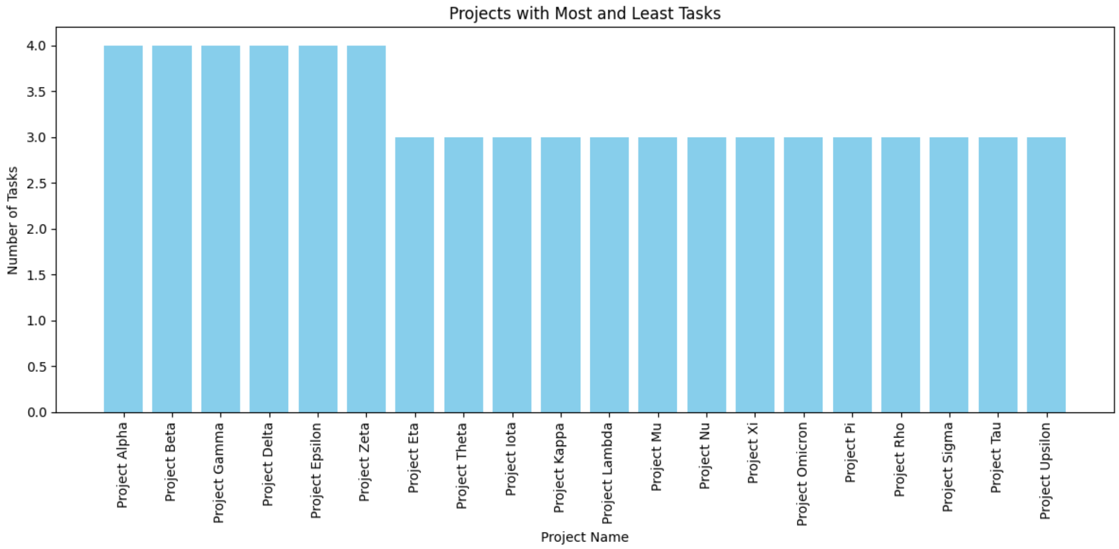
```
query = """
SELECT a.type, COUNT(a.attachment_id) AS num_attachments
FROM Attachment a
GROUP BY a.type;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
```

```
df_attachments = pd.DataFrame(data, columns=['type', 'num_attachments'])

# Visualization: Pie Chart
plt.figure(figsize=(8, 8))
plt.pie(df_attachments['num_attachments'], labels=df_attachments['type'],␣
  ↪autopct='%1.1f%%', startangle=140)
plt.title("Distribution of Attachment Types")
plt.show()
```



[27]:
```
query = """
SELECT p.project_name, COUNT(t.task_id) AS num_tasks
FROM Project p
LEFT JOIN Task t ON p.project_id = t.project_id
```

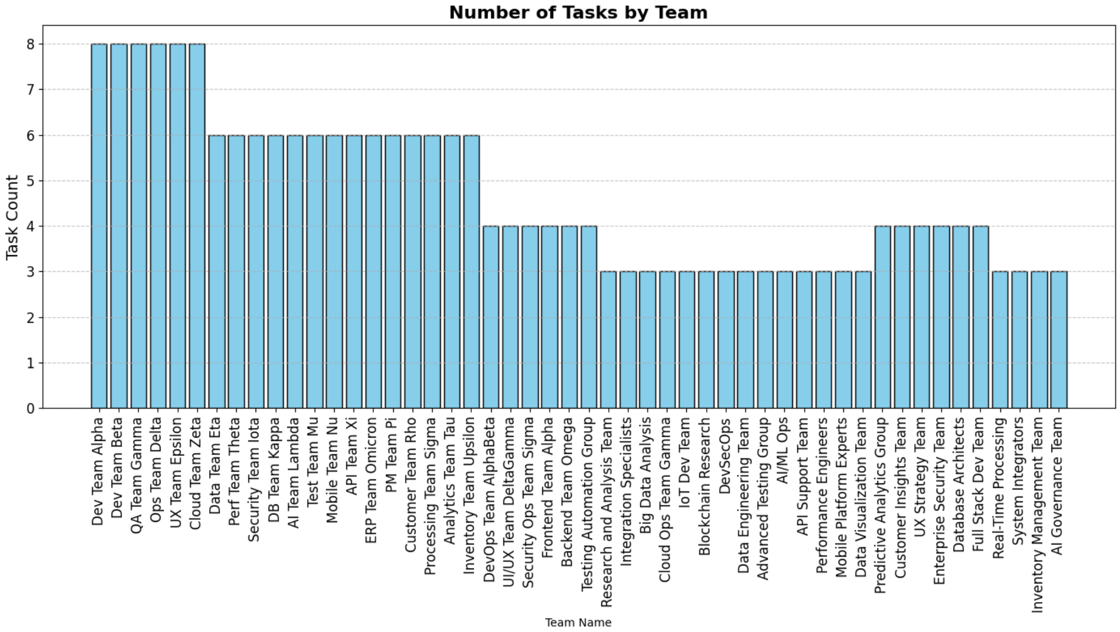
```
GROUP BY p.project_name
ORDER BY num_tasks DESC;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_tasks_per_project = pd.DataFrame(data, columns=['project_name', 'num_tasks'])

# Visualization: Bar Chart
plt.figure(figsize=(12, 6))
plt.bar(df_tasks_per_project['project_name'],␣
 ↪df_tasks_per_project['num_tasks'], color='skyblue')
plt.title("Projects with Most and Least Tasks")
plt.xlabel("Project Name")
plt.ylabel("Number of Tasks")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[21]:

```
query = """
SELECT tm.team_name, COUNT(t.task_id) AS task_count
FROM Team tm
INNER JOIN Task t ON tm.project_id = t.project_id
GROUP BY tm.team_name;
"""
cursor.execute(query)
data = cursor.fetchall()
```

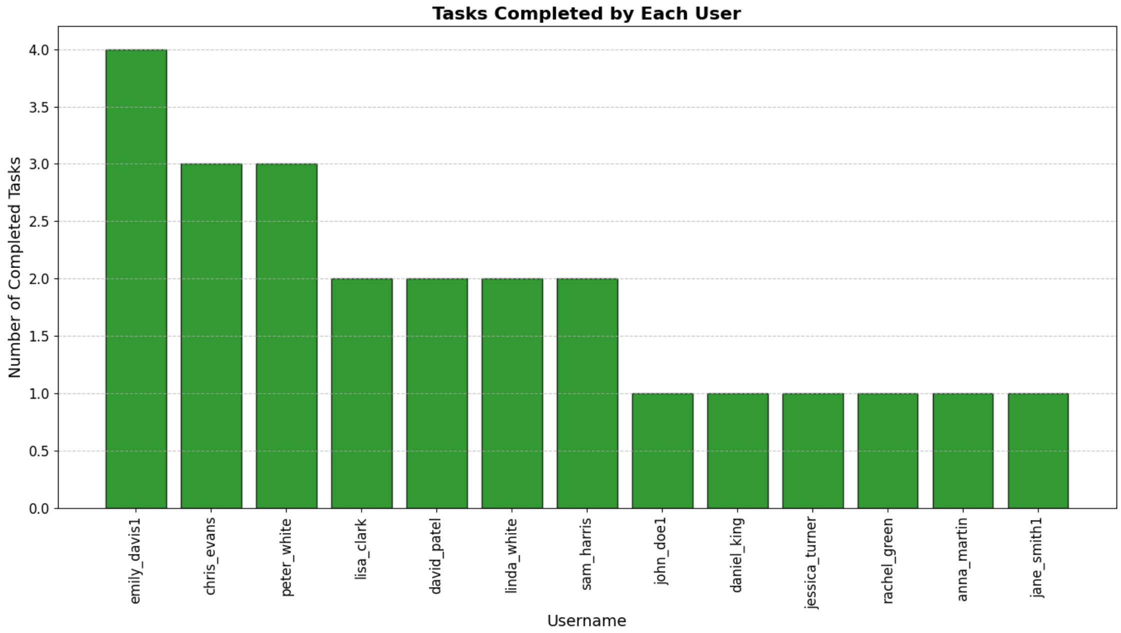
```
# Convert to DataFrame
df_team_tasks = pd.DataFrame(data, columns=['team_name', 'task_count'])

# Visualization: Bar Chart
plt.figure(figsize=(14, 8))
plt.bar(df_team_tasks['team_name'], df_team_tasks['task_count'],␣
 ↪color='skyblue', edgecolor='black')
plt.title("Number of Tasks by Team", fontsize=16, fontweight='bold')
plt.xlabel("Team Name", fontsize=10)
plt.ylabel("Task Count", fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



[28]:

```
# Execute the query
query = """
SELECT u.username, COUNT(t.task_id) AS completed_tasks
FROM User u
LEFT JOIN Task t ON u.user_id = t.user_id
WHERE t.status = 'Completed'
GROUP BY u.username
ORDER BY completed_tasks DESC;
"""
```

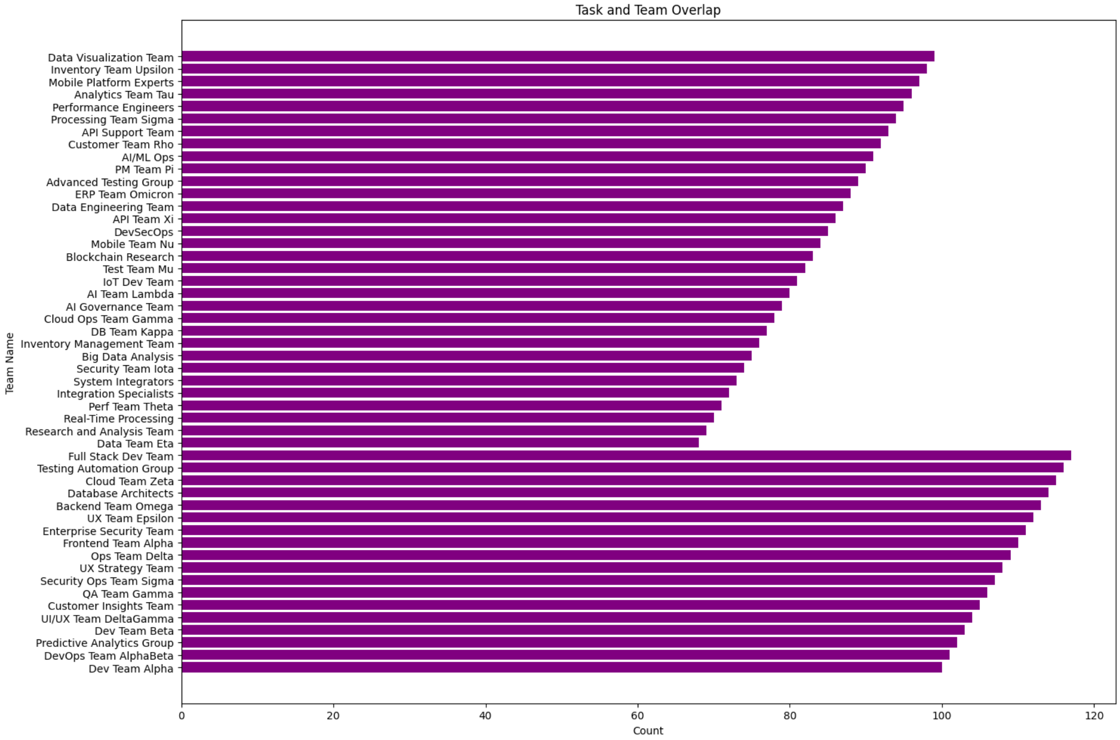
```
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_completed_tasks = pd.DataFrame(data, columns=['username', 'completed_tasks'])

# Visualization: Bar Chart
plt.figure(figsize=(14, 8))
plt.bar(df_completed_tasks['username'], df_completed_tasks['completed_tasks'],␣
  ↪color='green', edgecolor='black', alpha=0.8)
plt.title("Tasks Completed by Each User", fontsize=16, fontweight='bold')
plt.xlabel("Username", fontsize=14)
plt.ylabel("Number of Completed Tasks", fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Tasks Completed by Each User

[36]:
```
query = """
SELECT t.title, tm.team_name
FROM Task t
LEFT JOIN Team tm ON t.project_id = tm.project_id
UNION
SELECT t.title, tm.team_name
FROM Task t
```

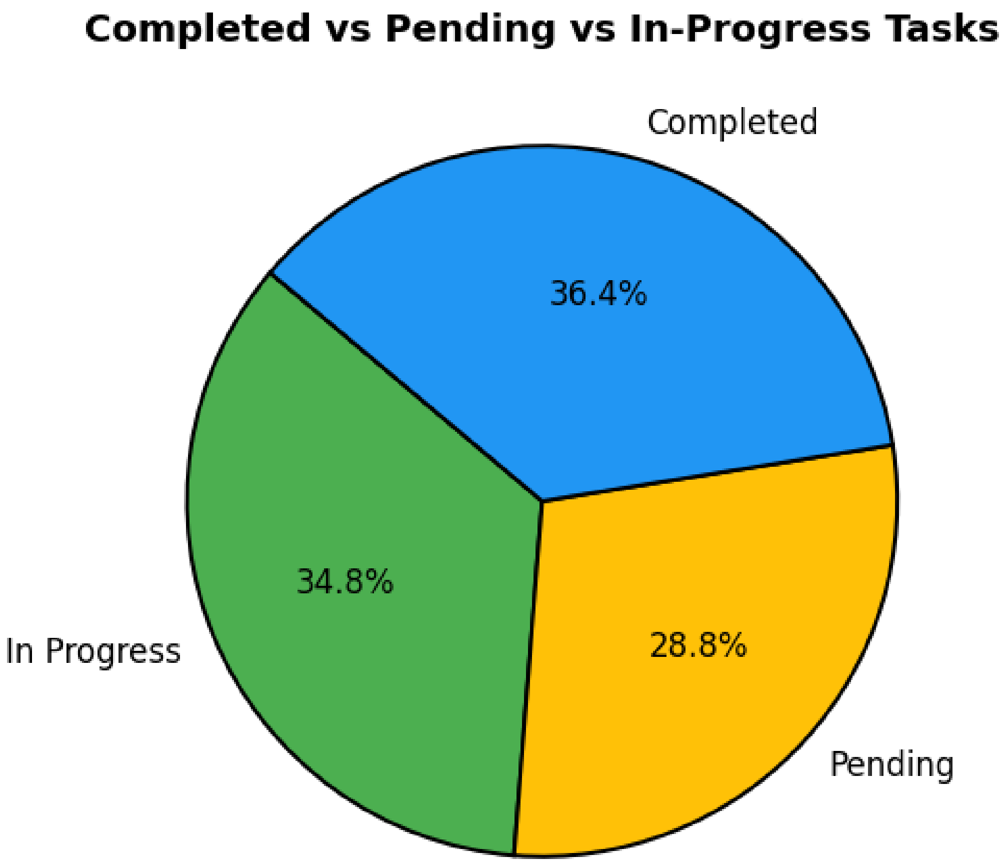
```
RIGHT JOIN Team tm ON t.project_id = tm.project_id;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_full_outer = pd.DataFrame(data, columns=['title', 'team_name'])

# Visualization: Horizontal Bar Chart
plt.figure(figsize=(15, 10))
plt.barh(df_full_outer['team_name'], range(len(df_full_outer)), color='purple')
plt.title("Task and Team Overlap")
plt.xlabel("Count")
plt.ylabel("Team Name")
plt.tight_layout()
plt.show()
```



[31]:

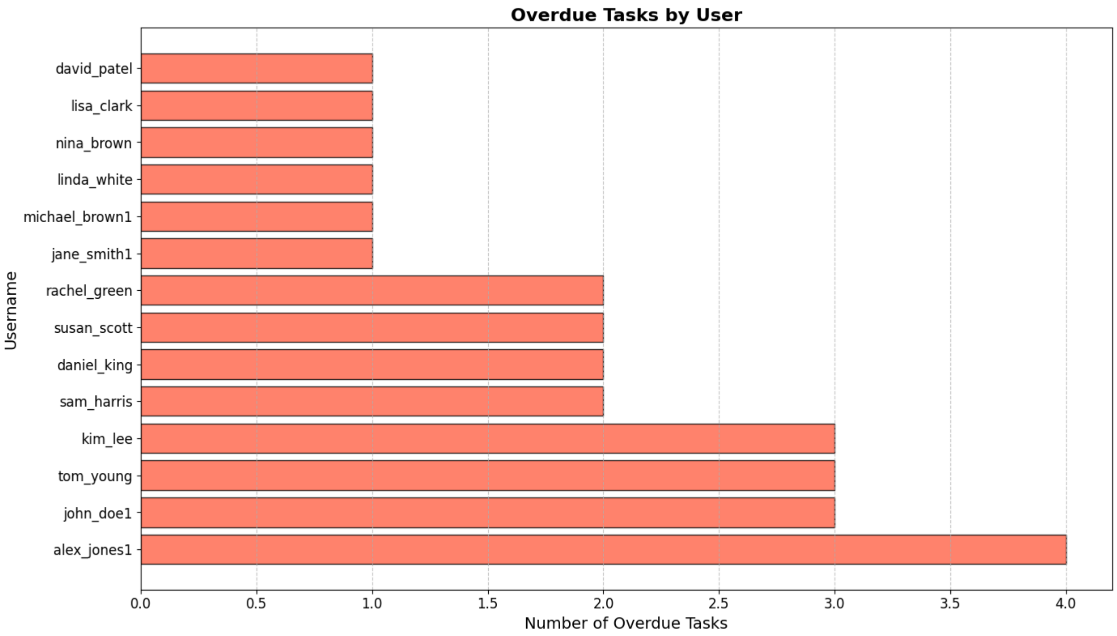
```
# Execute the query
query = """
SELECT t.status, COUNT(t.task_id) AS task_count
FROM Task t
GROUP BY t.status;
"""
```

```
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_task_status = pd.DataFrame(data, columns=['status', 'task_count'])

# Visualization: Pie Chart
plt.figure(figsize=(8, 6))
plt.pie(
    df_task_status['task_count'],
    labels=df_task_status['status'],
    autopct='%1.1f%%',
    startangle=140,
    colors=['#4CAF50', '#FFC107', '#2196F3'],  # Green, Orange, Blue
    textprops={'fontsize': 12},
    wedgeprops={'edgecolor': 'black', 'linewidth': 1.5}  # Adding separating lines
)
plt.title("Completed vs Pending vs In-Progress Tasks", fontsize=14, fontweight='bold')
plt.show()
```



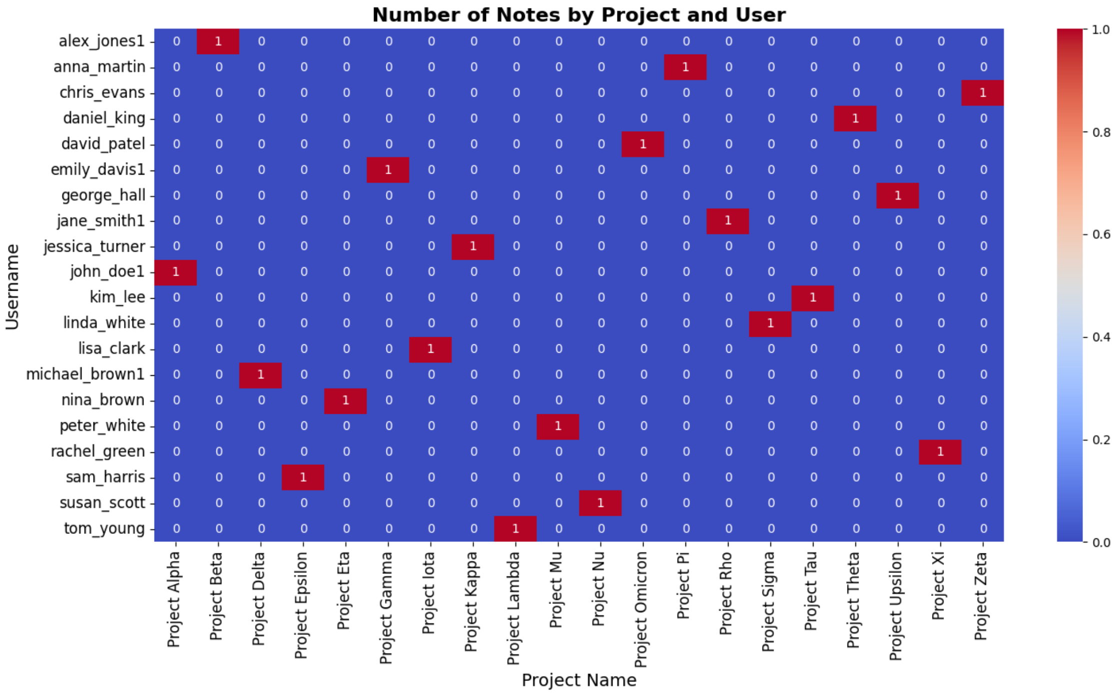
[34]:

```
# Execute the query
query = """
SELECT u.username, COUNT(t.task_id) AS overdue_tasks
FROM User u
JOIN Task t ON u.user_id = t.user_id
WHERE t.due_date < CURDATE() AND t.status != 'Completed'
GROUP BY u.username
ORDER BY overdue_tasks DESC;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_overdue_tasks = pd.DataFrame(data, columns=['username', 'overdue_tasks'])

# Visualization: Horizontal Bar Chart
plt.figure(figsize=(14, 8))
plt.barh(df_overdue_tasks['username'], df_overdue_tasks['overdue_tasks'],␣
    ↪color='tomato', edgecolor='black', alpha=0.8)
plt.title("Overdue Tasks by User", fontsize=16, fontweight='bold')
plt.xlabel("Number of Overdue Tasks", fontsize=14)
plt.ylabel("Username", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



[41]:
```
query = """
SELECT p.project_name, u.username, COUNT(n.note_id) AS note_count
FROM Note n
JOIN Task t ON n.task_id = t.task_id
JOIN User u ON t.user_id = u.user_id
JOIN Project p ON t.project_id = p.project_id
GROUP BY p.project_name, u.username;
"""
cursor.execute(query)
data = cursor.fetchall()

# Convert to DataFrame
df_notes = pd.DataFrame(data, columns=['project_name', 'username',␣
    ↪'note_count'])
```

```
# Pivot the data for a heatmap
df_pivot = df_notes.pivot(index='username', columns='project_name',␣
 ↪values='note_count').fillna(0)

# Visualization: Heatmap
plt.figure(figsize=(14, 8))
sns.heatmap(df_pivot, annot=True, fmt=".0f", cmap="coolwarm", cbar=True)
plt.title("Number of Notes by Project and User", fontsize=16, fontweight='bold')
plt.xlabel("Project Name", fontsize=14)
plt.ylabel("Username", fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```


Number of Notes by Project and User
Username
alex_jones1
anna_martin
chris_evans
daniel_king
david_patel
emily_davis1
george_hall
jane_smith1
jessica_turner
john_doe1
kim_lee
linda_white
lisa_clark
michael_brown1
nina_brown
peter_white
rachel_green
sam_harris
susan_scott
tom_young
Project Alpha
Project Beta
Project Delta
Project Epsilon
Project Eta
Project Gamma
Project Iota
Project Kappa
Project Lambda
Project Mu
Project Nu
Project Omicron
Project Pi
Project Rho
Project Sigma
Project Tau
Project Theta
Project Upsilon
Project Xi
Project Zeta
Project Name