IE 6700 Data Management for Analytics

Taste Genius- Data Driven Recipe Generator

Milestone: Python application

Group 16

Student1 Pranav Kuramkote Sudhir

Student2 Sancia Saldanha

857-465-9377 (Pranav K S)

617-352-1569 (Sancia Saldanha)
kuramkotesudhir.p@northeastern.edu
saldanha.s@northeastern.edu

Percentage of Effort Contributed by Student1: 50%
Percentage of Effort Contributed by Student2: 50%

Signature of Student 1: [illegible]

Signature of Student 2: [illegible]

Submission Date: November 26 2023

In [106]:

```python
# Creating a class tp handle MySQL connections
import mysql.connector
from sqlalchemy import create_engine

class MySQLConnector:
    def __init__(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database
        self.connection = None

    def connect(self):
        try:
            self.connection = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                database=self.database
            )
            print("Connected to MySQL!")
        except mysql.connector.Error as err:
            print(f"Error: {err}")

    def disconnect(self):
        if self.connection.is_connected():
            self.connection.close()
            print("Disconnected from MySQL!")

    def execute_query(self, query):
        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            result = cursor.fetchall()
            print("Query executed successfully!")
            return result
        except mysql.connector.Error as err:
            print(f"Error: {err}")
            return None

    def import_dataframe_to_table(self, dataframe, table_name):
        try:
            engine = create_engine(f"mysql+mysqlconnector://{self.user}:
```

```
                                    {self.password}@{self.host}/{self.database}")
            dataframe.to_sql(name=table_name, con=engine,
                                 if_exists='append', index=False)
            print(f"DataFrame imported into '{table_name}' table successfully!")
        except Exception as e:
            import traceback
            print(f"Error: {e}")
            traceback.print_exc()


```

In [107]:

```
connector = MySQLConnector('127.0.0.1', 'root','','Recipe')
```

In [108]:

```
connector.connect()
```

```
Connected to MySQL!
```

In [109]:

```
# trying to retrieve the first name, last name, dietary preference,
# and allergy information for a user with the User_ID of 5,
result=connector.execute_query('SELECT up.First_name, up.Last_Name, dp.preference, a.allergy \
FROM User_Profile up \
JOIN User_Preference upr ON up.preference_ID = upr.Preference_ID \
LEFT JOIN Is_Restricted_By irb ON upr.Preference_ID = irb.Preference_ID \
LEFT JOIN Dietary_Preference dp ON irb.Dietary_preference_ID = dp.Dietary_preference_ID \
LEFT JOIN Allergies a ON irb.Allergy_ID = a.Allergy_ID \
WHERE up.User_ID =5;')
print(result)
```

```
Query executed successfully!
[('William', 'Martin', 'Veg', 'role'), ('William', 'Martin', 'Vegan', 'it')]
```

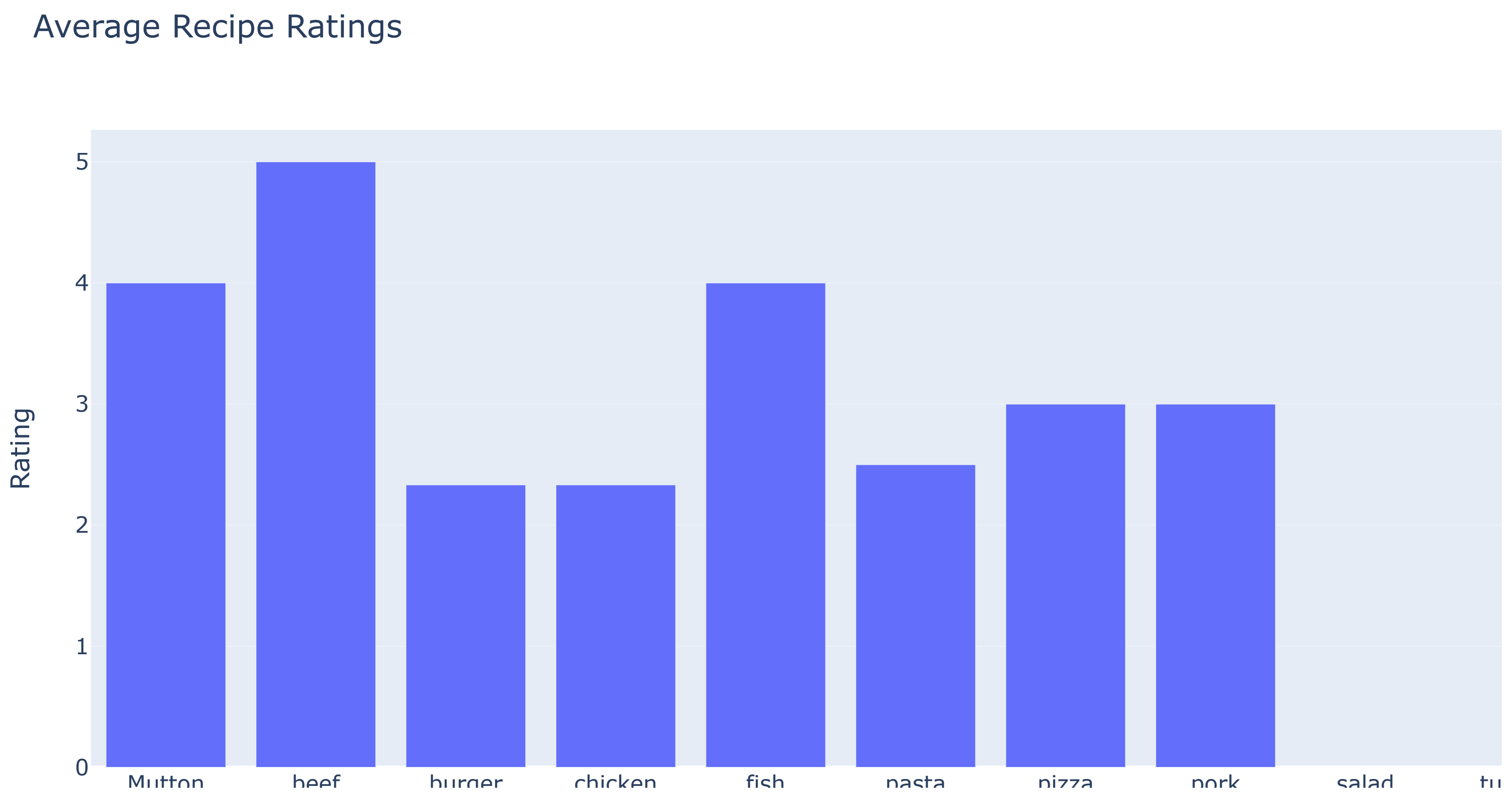
In [110]:
```
# retrieves the Recipe ID and the associated Rating, if available.
result = connector.execute_query('SELECT r.Recipe_name, s.Rating \
FROM Recipe r \
LEFT JOIN Searches sr ON r.Recipe_ID = sr.Recipe_ID \
LEFT JOIN Suggestion s ON sr.Suggestion_ID = s.Suggestion_ID;')
print(result)
```

```
Query executed successfully!
[('chicken', 2), ('chicken', 4), ('chicken', 1), ('fish', 4), ('fish', 4), ('fish', 4), ('fish', 4),
('Mutton', 4), ('beef', 5), ('pizza', 4), ('pizza', 5), ('pizza', 2), ('pizza', 1), ('burger', 1),
('burger', 2), ('burger', 4), ('pasta', 4), ('pasta', 1), ('salad', None), ('turkey', None), ('por
k', 1), ('pork', 5)]
```

In [111]:
```python
import pandas as pd
import plotly.express as px
df = pd.DataFrame(result, columns=[ 'Recipe_name', 'Rating'])

average_ratings = df.groupby('Recipe_name')['Rating'].mean().reset_index()

# Plot the average ratings using Plotly Express
fig = px.bar(average_ratings, x='Recipe_name', y='Rating', title='Average Recipe Ratings')
fig.show()
```



In [112]:
```
# Lists distinct ingredients that are restricted due to allergies.
result = connector.execute_query("SELECT DISTINCT ii.Name \
FROM Ingredient_inventory ii \
JOIN Contains c ON ii.Ingredient_ID = c.Ingredient_ID \
JOIN Is_restricted_by irb ON c.Recipe_ID = irb.Preference_ID \
WHERE irb.Allergy_ID IS NOT NULL; ")
print(result)

```

```
Query executed successfully!
[('Peanut',), ('Cheese',), ('Sugar',), ('Garlic',), ('Milk',), ('Beef',), ('Spinich',), ('Onion',)]
```

In [113]:
```
# Retrieves distinct recipes that match a specific dietary preference (assuming
result = connector.execute_query("SELECT DISTINCT r.Recipe_ID, r.Recipe_name \
FROM Recipe r \
JOIN Is_restricted_by irb ON r.Recipe_ID = irb.Preference_ID \
WHERE irb.Dietary_preference_ID = 2;")
print(result)
```

```
Query executed successfully!
[(9, 'turkey')]
```

In [114]:
```
# Counts the number of recipes that have both allergies and dietary preferences.
result = connector.execute_query("SELECT irb.Allergy_ID, irb.Dietary_preference_ID, \
COUNT(r.Recipe_ID) AS Recipe_Count \
FROM Recipe r \
JOIN Is_restricted_by irb ON r.Recipe_ID = irb.Preference_ID \
WHERE irb.Allergy_ID IS NOT NULL AND irb.Dietary_preference_ID IS NOT NULL \
GROUP BY irb.Allergy_ID, irb.Dietary_preference_ID;")
print(result)
```

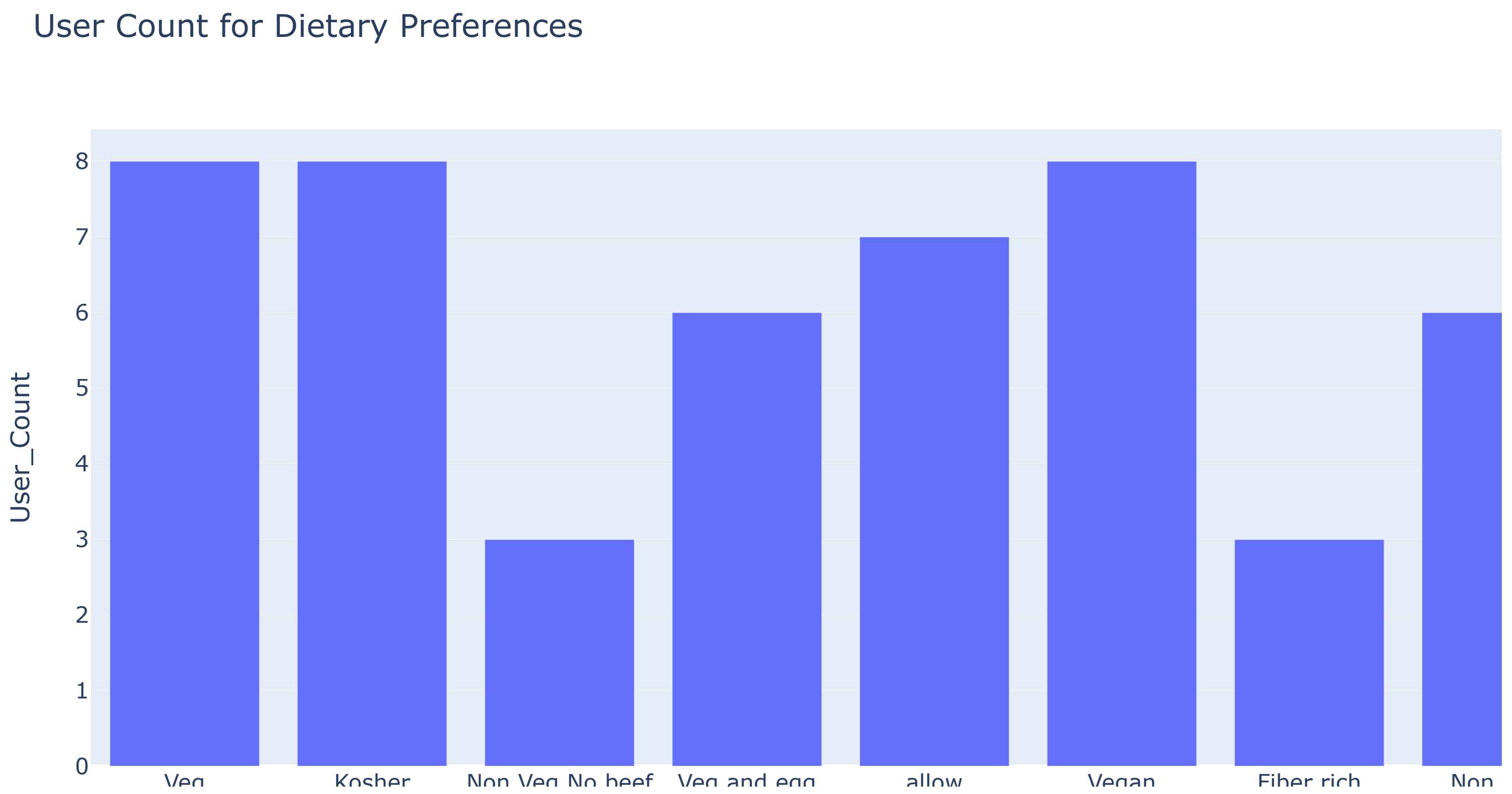
```
Query executed successfully!
[(2, 1, 1), (5, 6, 1), (6, 4, 1), (3, 7, 1), (4, 8, 1), (1, 3, 1), (7, 6, 1), (4, 9, 1), (9, 2, 1),
(10, 7, 1)]
```

In [115]:
```
# retrieve the count of users for each dietary preference
result = connector.execute_query("SELECT dp.preference, COUNT(up.User_ID) AS User_Count \
FROM User_profile up \
JOIN Is_restricted_by irb ON up.preference_ID = irb.Preference_ID \
JOIN Dietary_preference dp ON irb.Dietary_preference_ID = dp.Dietary_preference_ID \
GROUP BY dp.preference;")
print(result)
```

```
Query executed successfully!
[('Veg', 8), ('Kosher', 8), ('Non Veg No beef', 3), ('Veg and egg', 6), ('allow', 7), ('Vegan', 8),
('Fiber rich', 3), ('Non Veg', 6)]
```

In [116]:
```python
df = pd.DataFrame(result, columns=['preference', 'User_Count'])

# Plot the data using Plotly Express
fig = px.bar(df, x='preference', y='User_Count', title='User Count for Dietary Preferences')
fig.show()
```



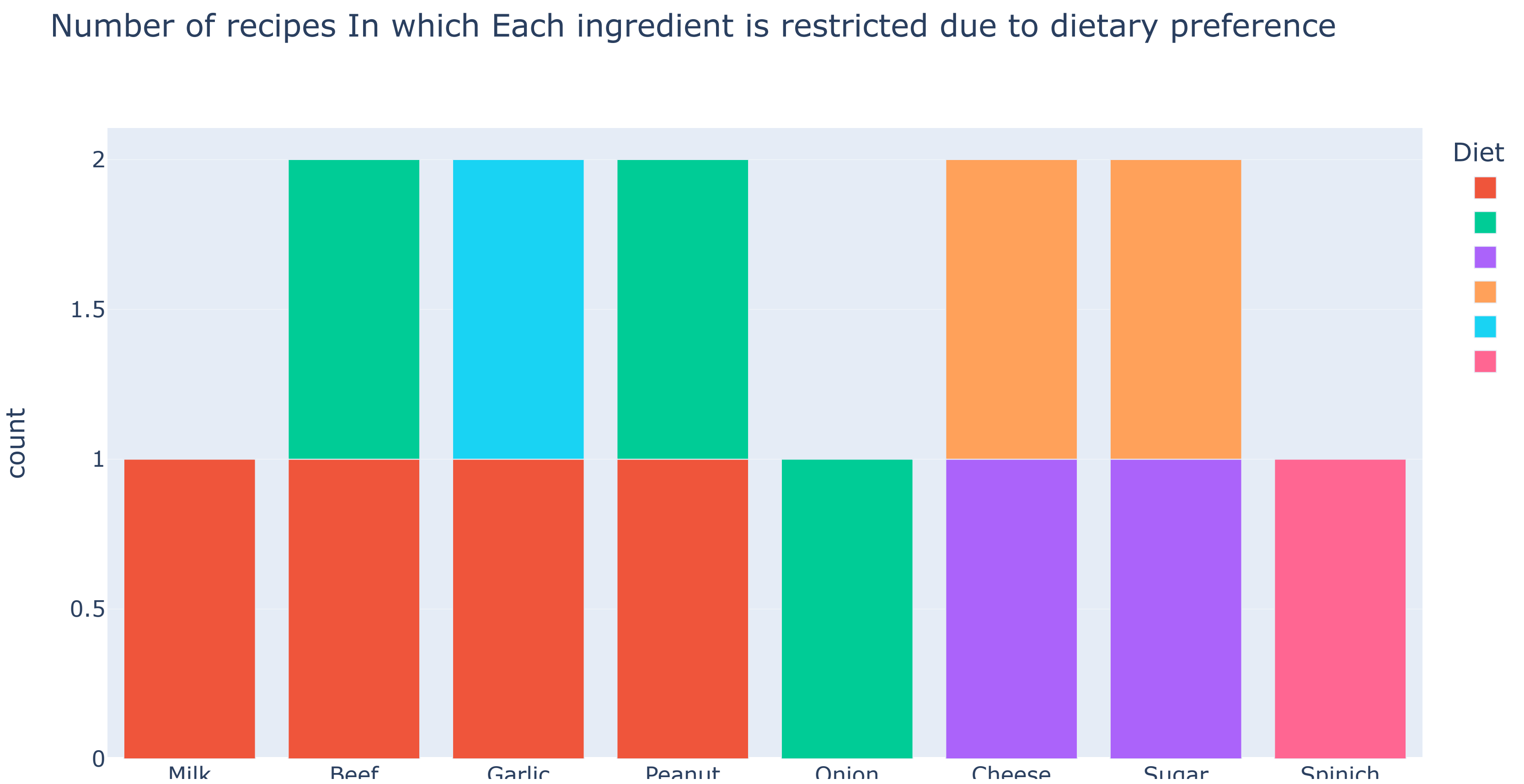
In [117]:

```python
# recipes In which Each ingredient is restricted due to dietary preference
result = connector.execute_query("SELECT r.Recipe_ID, ii.Name AS Ingredient_Name, "
"dp.preference AS Dietary_Preference "
"FROM Recipe r "
"JOIN Contains c ON r.Recipe_ID = c.Recipe_ID "
"JOIN Ingredient_inventory ii ON c.Ingredient_ID = ii.Ingredient_ID "
"LEFT JOIN Is_restricted_by irb ON r.Recipe_ID = irb.Preference_ID "
"LEFT JOIN Dietary_preference dp ON irb.Dietary_preference_ID = dp.Dietary_preference_ID;")
print(result)
```

```
Query executed successfully!
[(1, 'Potato', None), (1, 'Sugar', None), (1, 'Ginger', None), (2, 'Beef', None), (2, 'Spinich', Non
e), (2, 'Spinich', None), (2, 'Spinich', None), (2, 'Potato', None), (2, 'Spinich', None), (3, 'Mil
k', 'Kosher'), (3, 'Beef', 'Kosher'), (3, 'Garlic', 'Kosher'), (4, 'Peanut', 'Kosher'), (4, 'Peanu
t', 'Veg and egg'), (5, 'Beef', 'Veg and egg'), (5, 'Onion', 'Veg and egg'), (7, 'Cheese', 'Non Veg
No beef'), (7, 'Sugar', 'Non Veg No beef'), (7, 'Cheese', 'Fiber rich'), (7, 'Sugar', 'Fiber rich'),
(8, 'Garlic', 'allow'), (9, 'Spinich', 'Non Veg'), (10, 'Egg', None)]
```

In [118]:
```python
df = pd.DataFrame(result, columns=['Recipe_ID', 'Ingredient_Name', 'Dietary_Preference'])

# Plot the data using Plotly Express
fig = px.bar(df, x='Ingredient_Name', color='Dietary_Preference', title=
             'Number of recipes In which Each ingredient is restricted due to dietary preference'
            )
fig.show()
```


Number of recipes In which Each ingredient is restricted due to dietary preference
count
Milk
Beef
Garlic
Peanut
Onion
Cheese
Sugar
Spinich
Diet