21BEC2508 MAREPALLI VISHNU VARDHAN

HITWICKET ASSIGNMENT

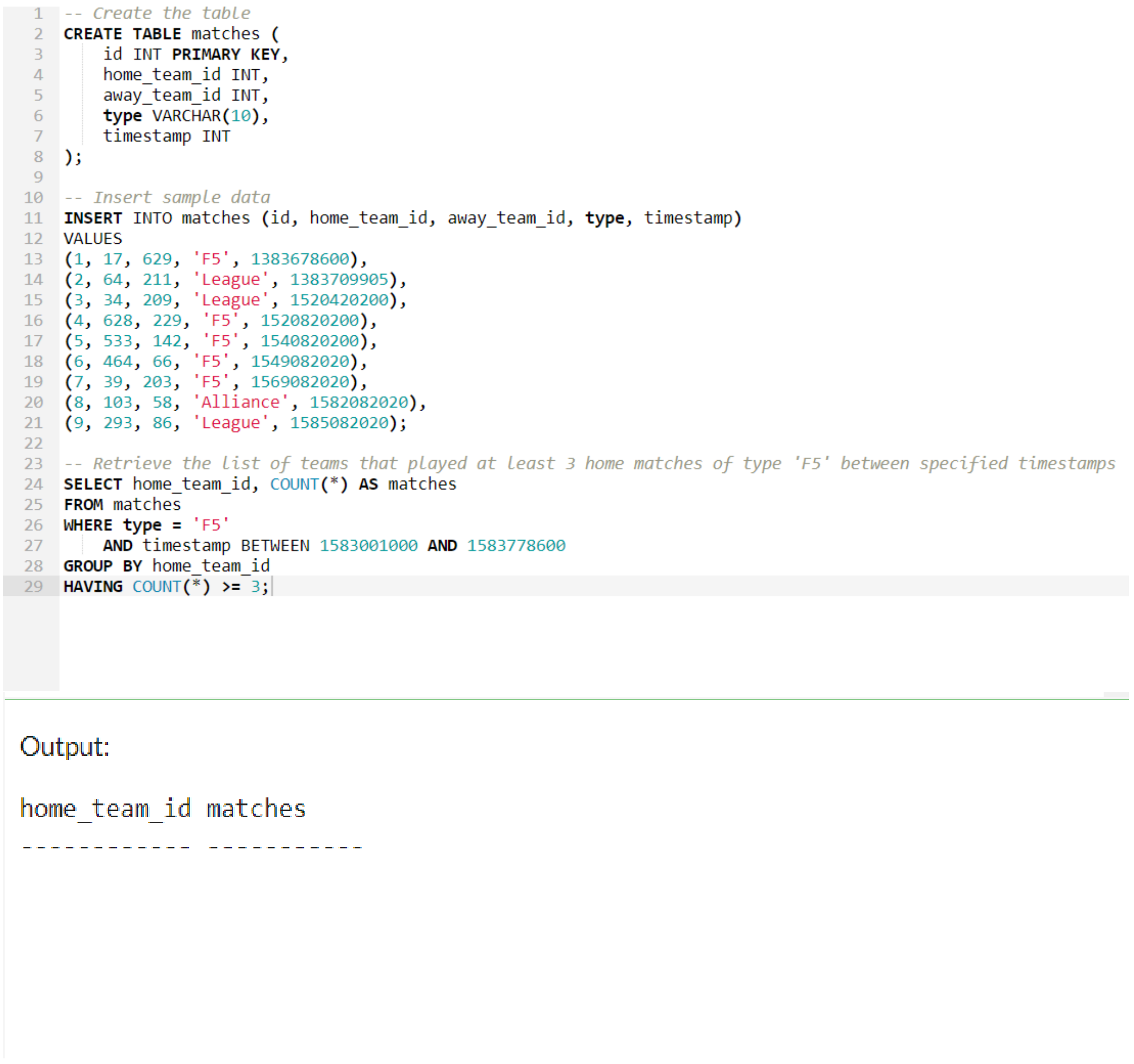
AUGUST 28, 2024

HITWICKET

[Company address]

SECTION 1: SQL

1

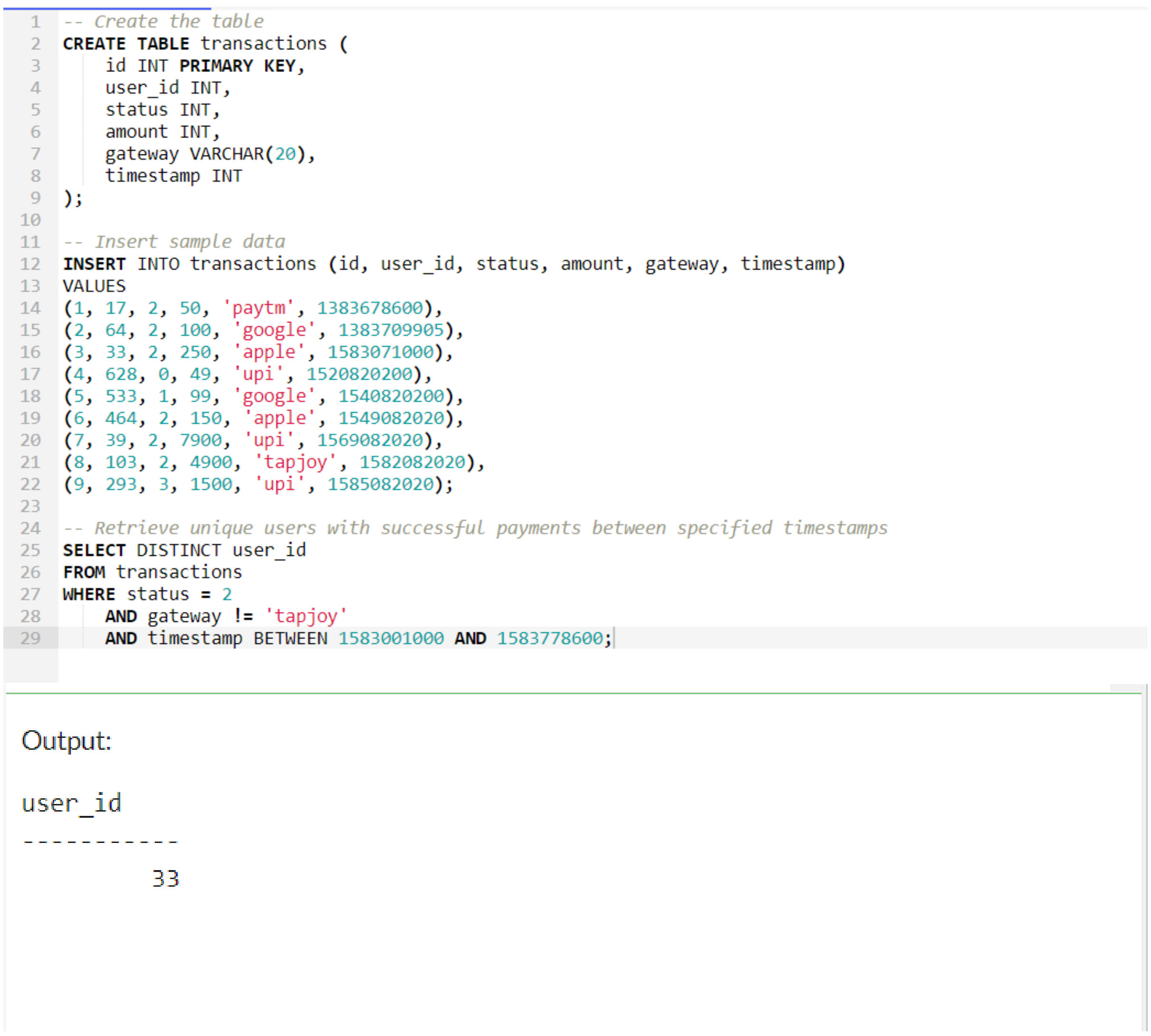


```sql
-- Create the table
CREATE TABLE matches (
    id INT PRIMARY KEY,
    home_team_id INT,
    away_team_id INT,
    type VARCHAR(10),
    timestamp INT
);

-- Insert sample data
INSERT INTO matches (id, home_team_id, away_team_id, type, timestamp)
VALUES
(1, 17, 629, 'F5', 1383678600),
(2, 64, 211, 'League', 1383709905),
(3, 34, 209, 'League', 1520420200),
(4, 628, 229, 'F5', 1520820200),
(5, 533, 142, 'F5', 1540820200),
(6, 464, 66, 'F5', 1549082020),
(7, 39, 203, 'F5', 1569082020),
(8, 103, 58, 'Alliance', 1582082020),
(9, 293, 86, 'League', 1585082020);

-- Retrieve the list of teams that played at least 3 home matches of type 'F5' between specified timestamps
SELECT home_team_id, COUNT(*) AS matches
FROM matches
WHERE type = 'F5'
    AND timestamp BETWEEN 1583001000 AND 1583778600
GROUP BY home_team_id
HAVING COUNT(*) >= 3;
```

Output:

```
home_team_id matches
------------ -----------
```

2

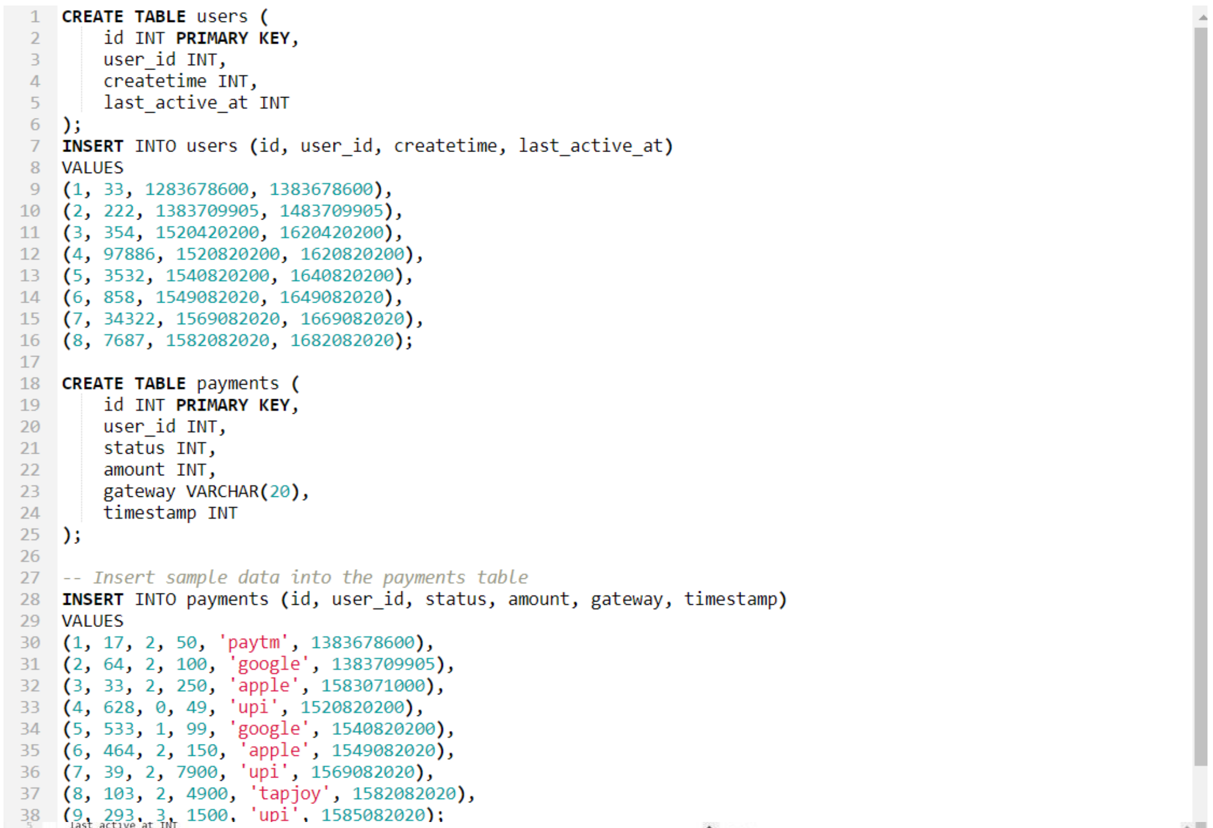


```sql
-- Create the table
CREATE TABLE transactions (
    id INT PRIMARY KEY,
    user_id INT,
    status INT,
    amount INT,
    gateway VARCHAR(20),
    timestamp INT
);

-- Insert sample data
INSERT INTO transactions (id, user_id, status, amount, gateway, timestamp)
VALUES
(1, 17, 2, 50, 'paytm', 1383678600),
(2, 64, 2, 100, 'google', 1383709905),
(3, 33, 2, 250, 'apple', 1583071000),
(4, 628, 0, 49, 'upi', 1520820200),
(5, 533, 1, 99, 'google', 1540820200),
(6, 464, 2, 150, 'apple', 1549082020),
(7, 39, 2, 7900, 'upi', 1569082020),
(8, 103, 2, 4900, 'tapjoy', 1582082020),
(9, 293, 3, 1500, 'upi', 1585082020);

-- Retrieve unique users with successful payments between specified timestamps
SELECT DISTINCT user_id
FROM transactions
WHERE status = 2
    AND gateway != 'tapjoy'
    AND timestamp BETWEEN 1583001000 AND 1583778600;
```

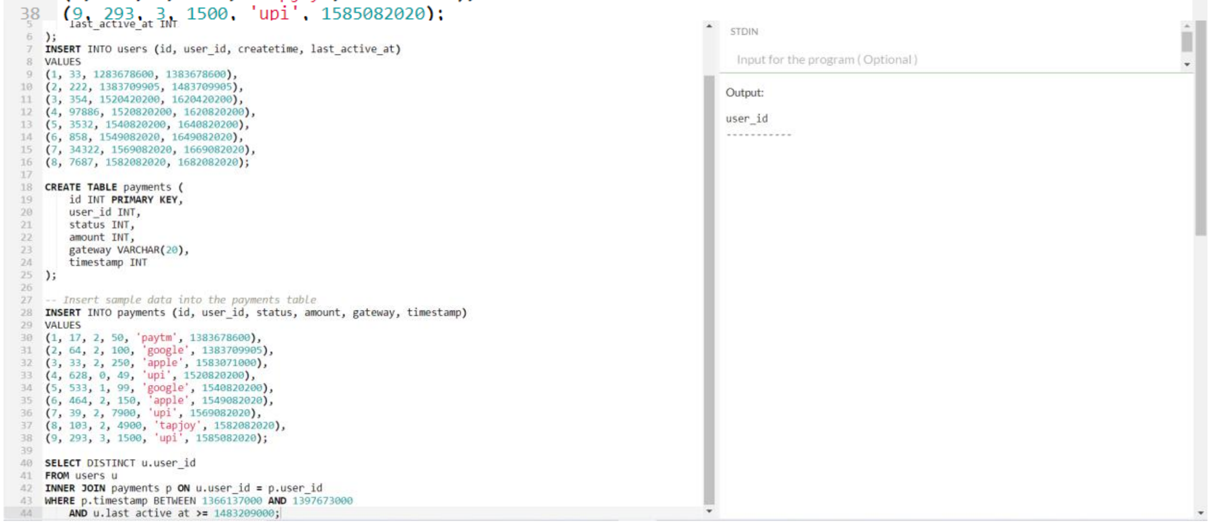
Output:

```
user_id
-----------
         33
```

3



```sql
CREATE TABLE users (
    id INT PRIMARY KEY,
    user_id INT,
    createtime INT,
    last_active_at INT
);
INSERT INTO users (id, user_id, createtime, last_active_at)
VALUES
(1, 33, 1283678600, 1383678600),
(2, 222, 1383709905, 1483709905),
(3, 354, 1520420200, 1620420200),
(4, 97886, 1520820200, 1620820200),
(5, 3532, 1540820200, 1640820200),
(6, 858, 1549082020, 1649082020),
(7, 34322, 1569082020, 1669082020),
(8, 7687, 1582082020, 1682082020);

CREATE TABLE payments (
    id INT PRIMARY KEY,
    user_id INT,
    status INT,
    amount INT,
    gateway VARCHAR(20),
    timestamp INT
);

-- Insert sample data into the payments table
INSERT INTO payments (id, user_id, status, amount, gateway, timestamp)
VALUES
(1, 17, 2, 50, 'paytm', 1383678600),
(2, 64, 2, 100, 'google', 1383709905),
(3, 33, 2, 250, 'apple', 1583071000),
(4, 628, 0, 49, 'upi', 1520820200),
(5, 533, 1, 99, 'google', 1540820200),
(6, 464, 2, 150, 'apple', 1549082020),
(7, 39, 2, 7900, 'upi', 1569082020),
(8, 103, 2, 4900, 'tapjoy', 1582082020),
(9, 293, 3, 1500, 'upi', 1585082020);
```



```sql
    last_active_at INT
);
INSERT INTO users (id, user_id, createtime, last_active_at)
VALUES
(1, 33, 1283678600, 1383678600),
(2, 222, 1383709905, 1483709905),
(3, 354, 1520420200, 1620420200),
(4, 97886, 1520820200, 1620820200),
(5, 3532, 1540820200, 1640820200),
(6, 858, 1549082020, 1649082020),
(7, 34322, 1569082020, 1669082020),
(8, 7687, 1582082020, 1682082020);

CREATE TABLE payments (
    id INT PRIMARY KEY,
    user_id INT,
    status INT,
    amount INT,
    gateway VARCHAR(20),
    timestamp INT
);

-- Insert sample data into the payments table
INSERT INTO payments (id, user_id, status, amount, gateway, timestamp)
VALUES
(1, 17, 2, 50, 'paytm', 1383678600),
(2, 64, 2, 100, 'google', 1383709905),
(3, 33, 2, 250, 'apple', 1583071000),
(4, 628, 0, 49, 'upi', 1520820200),
(5, 533, 1, 99, 'google', 1540820200),
(6, 464, 2, 150, 'apple', 1549082020),
(7, 39, 2, 7900, 'upi', 1569082020),
(8, 103, 2, 4900, 'tapjoy', 1582082020),
(9, 293, 3, 1500, 'upi', 1585082020);
```

```sql
SELECT DISTINCT u.user_id
FROM users u
INNER JOIN payments p ON u.user_id = p.user_id
WHERE p.timestamp BETWEEN 1366137000 AND 1397673000
    AND u.last_active_at >= 1483209000;
```

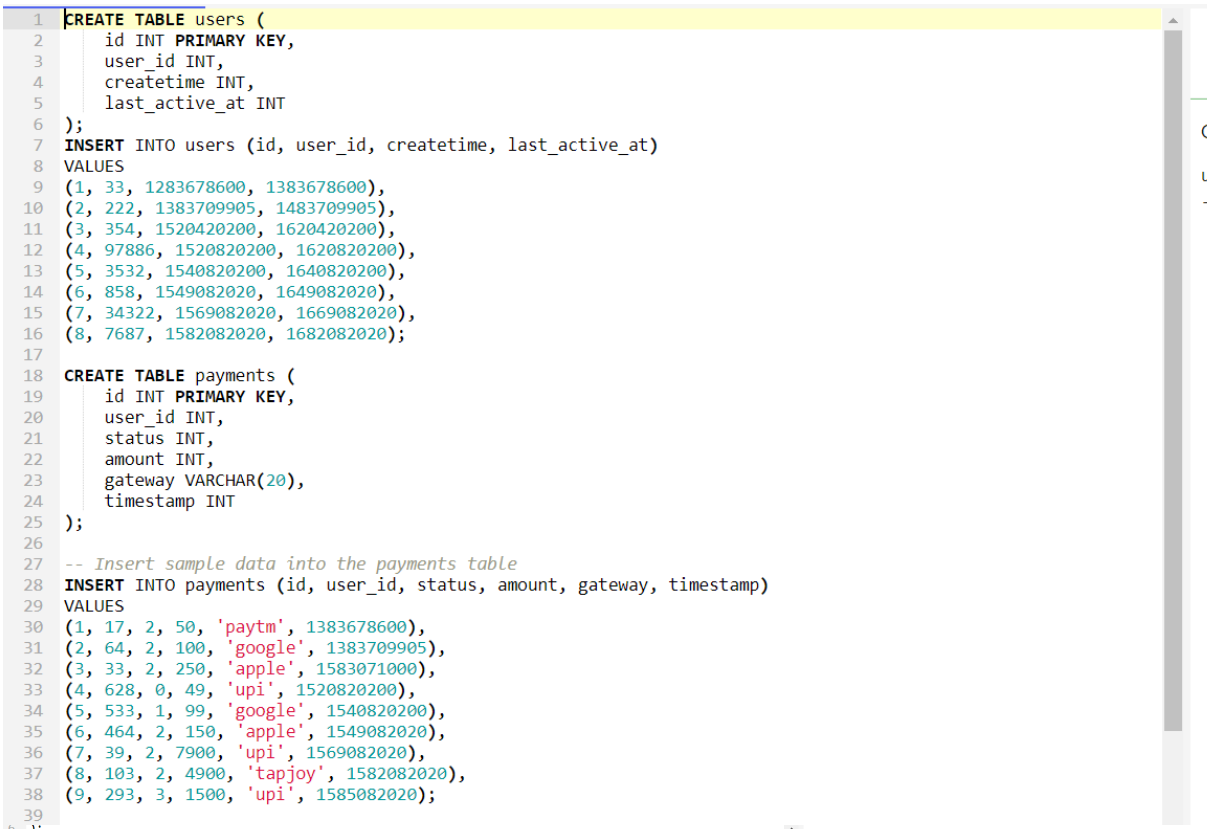
STDIN

Input for the program (Optional)

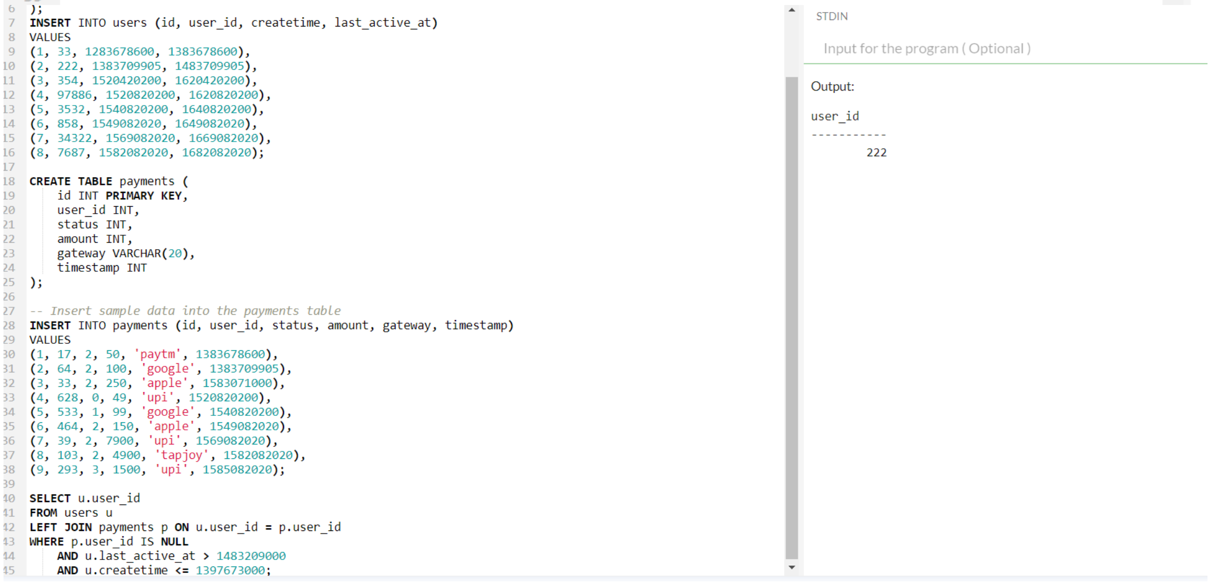
Output:

```
user_id
-----------
```

4



```sql
CREATE TABLE users (
    id INT PRIMARY KEY,
    user_id INT,
    createtime INT,
    last_active_at INT
);
INSERT INTO users (id, user_id, createtime, last_active_at)
VALUES
(1, 33, 1283678600, 1383678600),
(2, 222, 1383709905, 1483709905),
(3, 354, 1520420200, 1620420200),
(4, 97886, 1520820200, 1620820200),
(5, 3532, 1540820200, 1640820200),
(6, 858, 1549082020, 1649082020),
(7, 34322, 1569082020, 1669082020),
(8, 7687, 1582082020, 1682082020);

CREATE TABLE payments (
    id INT PRIMARY KEY,
    user_id INT,
    status INT,
    amount INT,
    gateway VARCHAR(20),
    timestamp INT
);

-- Insert sample data into the payments table
INSERT INTO payments (id, user_id, status, amount, gateway, timestamp)
VALUES
(1, 17, 2, 50, 'paytm', 1383678600),
(2, 64, 2, 100, 'google', 1383709905),
(3, 33, 2, 250, 'apple', 1583071000),
(4, 628, 0, 49, 'upi', 1520820200),
(5, 533, 1, 99, 'google', 1540820200),
(6, 464, 2, 150, 'apple', 1549082020),
(7, 39, 2, 7900, 'upi', 1569082020),
(8, 103, 2, 4900, 'tapjoy', 1582082020),
(9, 293, 3, 1500, 'upi', 1585082020);
```



```sql
);
INSERT INTO users (id, user_id, createtime, last_active_at)
VALUES
(1, 33, 1283678600, 1383678600),
(2, 222, 1383709905, 1483709905),
(3, 354, 1520420200, 1620420200),
(4, 97886, 1520820200, 1620820200),
(5, 3532, 1540820200, 1640820200),
(6, 858, 1549082020, 1649082020),
(7, 34322, 1569082020, 1669082020),
(8, 7687, 1582082020, 1682082020);

CREATE TABLE payments (
    id INT PRIMARY KEY,
    user_id INT,
    status INT,
    amount INT,
    gateway VARCHAR(20),
    timestamp INT
);

-- Insert sample data into the payments table
INSERT INTO payments (id, user_id, status, amount, gateway, timestamp)
VALUES
(1, 17, 2, 50, 'paytm', 1383678600),
(2, 64, 2, 100, 'google', 1383709905),
(3, 33, 2, 250, 'apple', 1583071000),
(4, 628, 0, 49, 'upi', 1520820200),
(5, 533, 1, 99, 'google', 1540820200),
(6, 464, 2, 150, 'apple', 1549082020),
(7, 39, 2, 7900, 'upi', 1569082020),
(8, 103, 2, 4900, 'tapjoy', 1582082020),
(9, 293, 3, 1500, 'upi', 1585082020);
```

```sql
SELECT u.user_id
FROM users u
LEFT JOIN payments p ON u.user_id = p.user_id
WHERE p.user_id IS NULL
    AND u.last_active_at > 1483209000
    AND u.createtime <= 1397673000;
```

STDIN

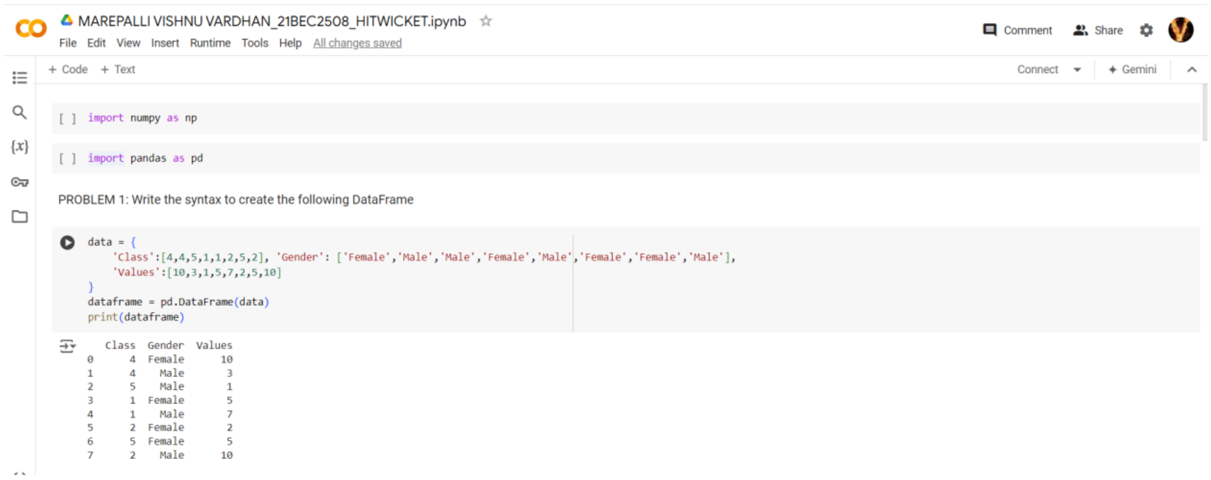
Input for the program (Optional)

Output:

```
user_id
-----------
        222
```

PYTHON:

1



MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

```
import numpy as np
```

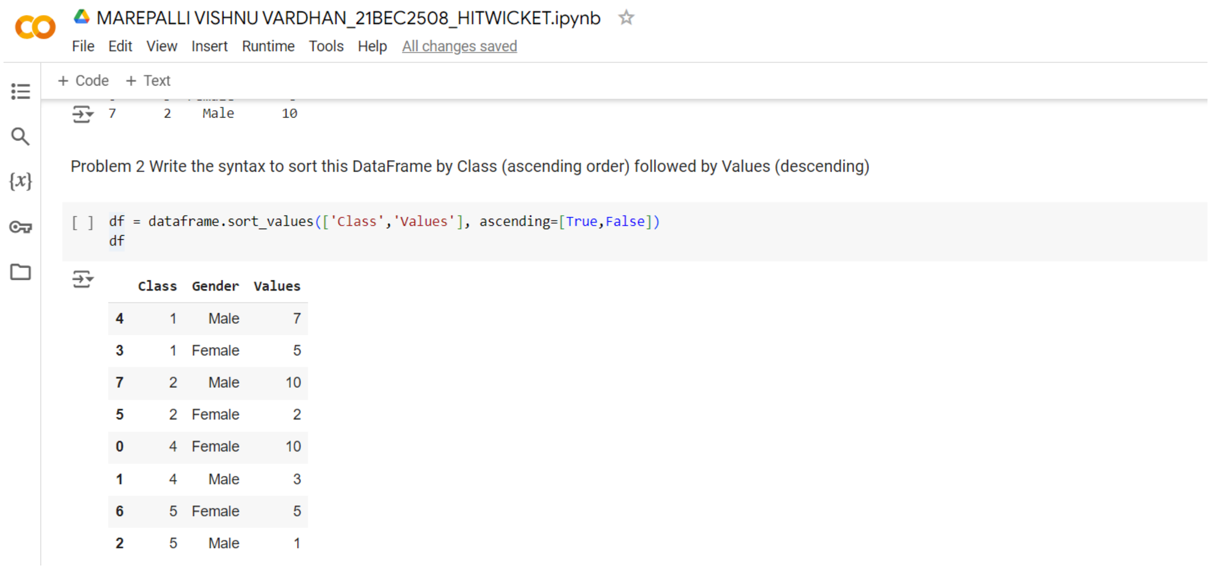
```
import pandas as pd
```

PROBLEM 1: Write the syntax to create the following DataFrame

```
data = {
    'Class':[4,4,5,1,1,2,5,2], 'Gender': ['Female','Male','Male','Female','Male','Female','Female','Male'],
    'Values':[10,3,1,5,7,2,5,10]
}
dataframe = pd.DataFrame(data)
print(dataframe)
```

```
   Class  Gender  Values
0      4  Female      10
1      4    Male       3
2      5    Male       1
3      1  Female       5
4      1    Male       7
5      2  Female       2
6      5  Female       5
7      2    Male      10
```

2



MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

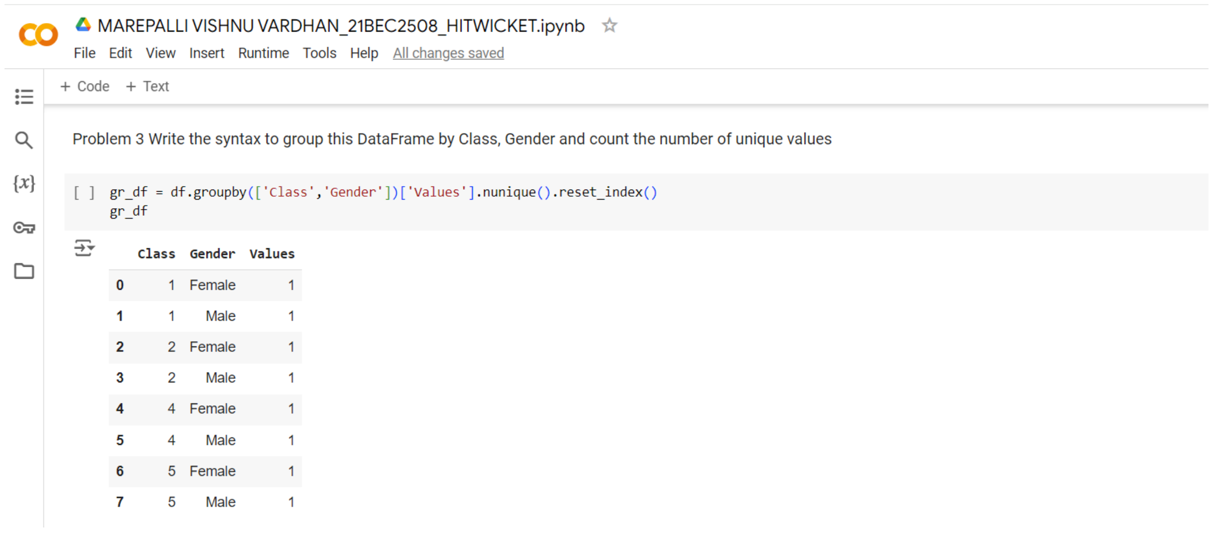
```
7      2    Male      10
```

Problem 2 Write the syntax to sort this DataFrame by Class (ascending order) followed by Values (descending)

```
df = dataframe.sort_values(['Class','Values'], ascending=[True,False])
df
```

	Class	Gender	Values
4	1	Male	7
3	1	Female	5
7	2	Male	10
5	2	Female	2
0	4	Female	10
1	4	Male	3
6	5	Female	5
2	5	Male	1

3

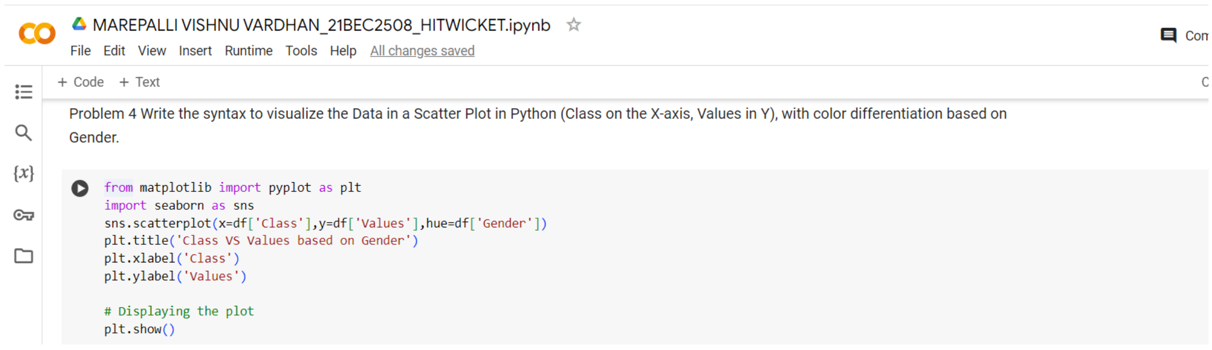

MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

Problem 3 Write the syntax to group this DataFrame by Class, Gender and count the number of unique values

```
gr_df = df.groupby(['Class','Gender'])['Values'].nunique().reset_index()
gr_df
```

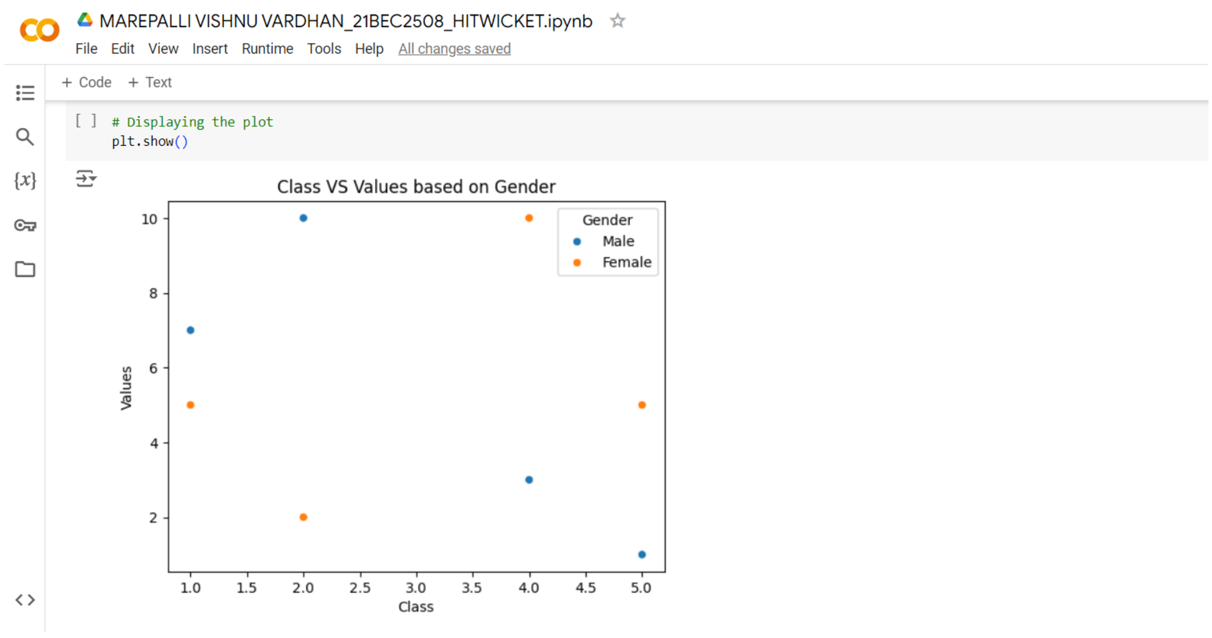
	Class	Gender	Values
0	1	Female	1
1	1	Male	1
2	2	Female	1
3	2	Male	1
4	4	Female	1
5	4	Male	1
6	5	Female	1
7	5	Male	1

4


MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

Problem 4 Write the syntax to visualize the Data in a Scatter Plot in Python (Class on the X-axis, Values in Y), with color differentiation based on Gender.

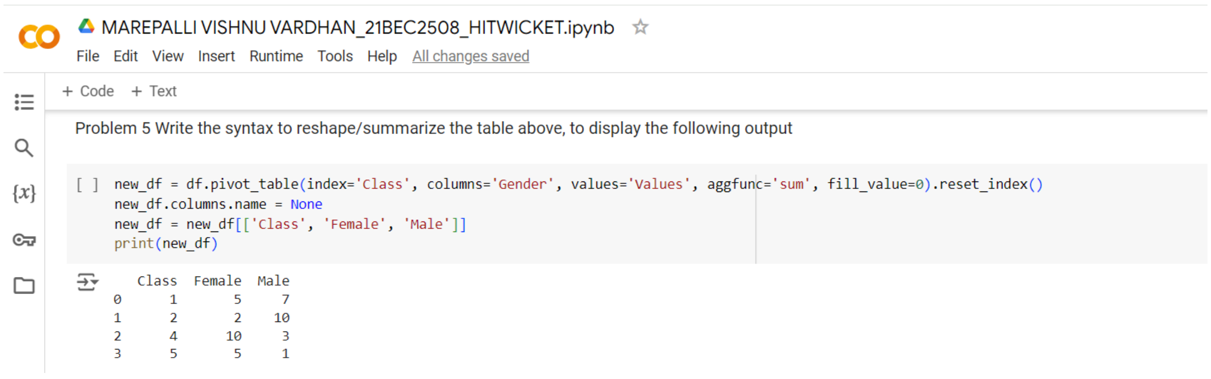
```
from matplotlib import pyplot as plt
import seaborn as sns
sns.scatterplot(x=df['Class'],y=df['Values'],hue=df['Gender'])
plt.title('Class VS Values based on Gender')
plt.xlabel('Class')
plt.ylabel('Values')

# Displaying the plot
plt.show()
```


MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
# Displaying the plot
plt.show()
```

Class VS Values based on Gender
Gender
● Male
● Female
Values
Class
2
4
6
8
10
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

5

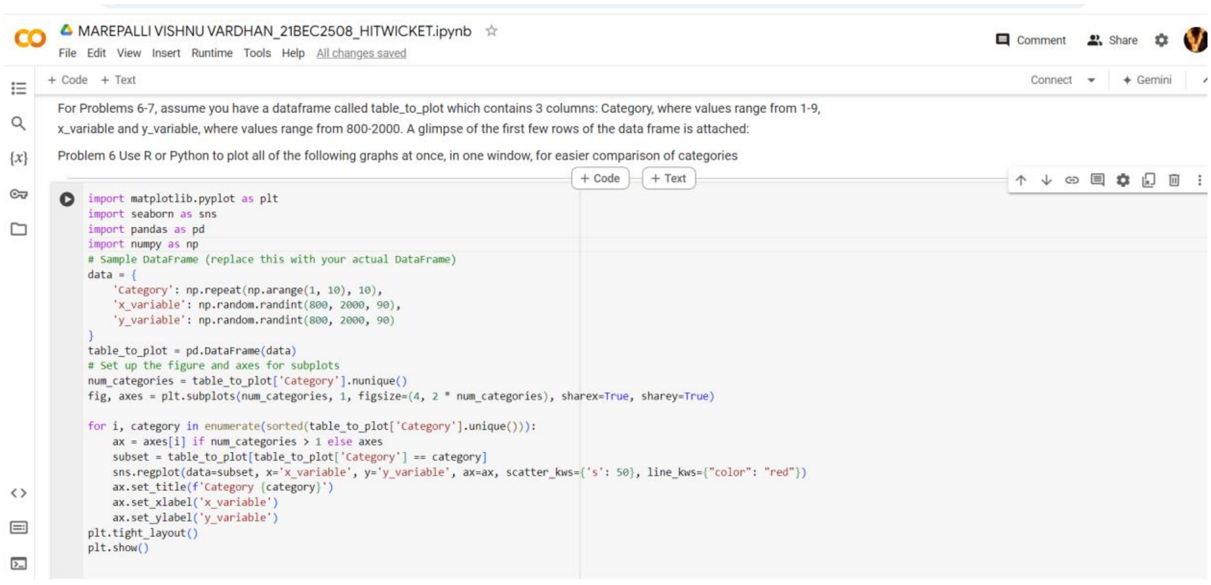

MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

Problem 5 Write the syntax to reshape/summarize the table above, to display the following output

```
new_df = df.pivot_table(index='Class', columns='Gender', values='Values', aggfunc='sum', fill_value=0).reset_index()
new_df.columns.name = None
new_df = new_df[['Class', 'Female', 'Male']]
print(new_df)
```

```
   Class  Female  Male
0      1       5     7
1      2       2    10
2      4      10     3
3      5       5     1
```

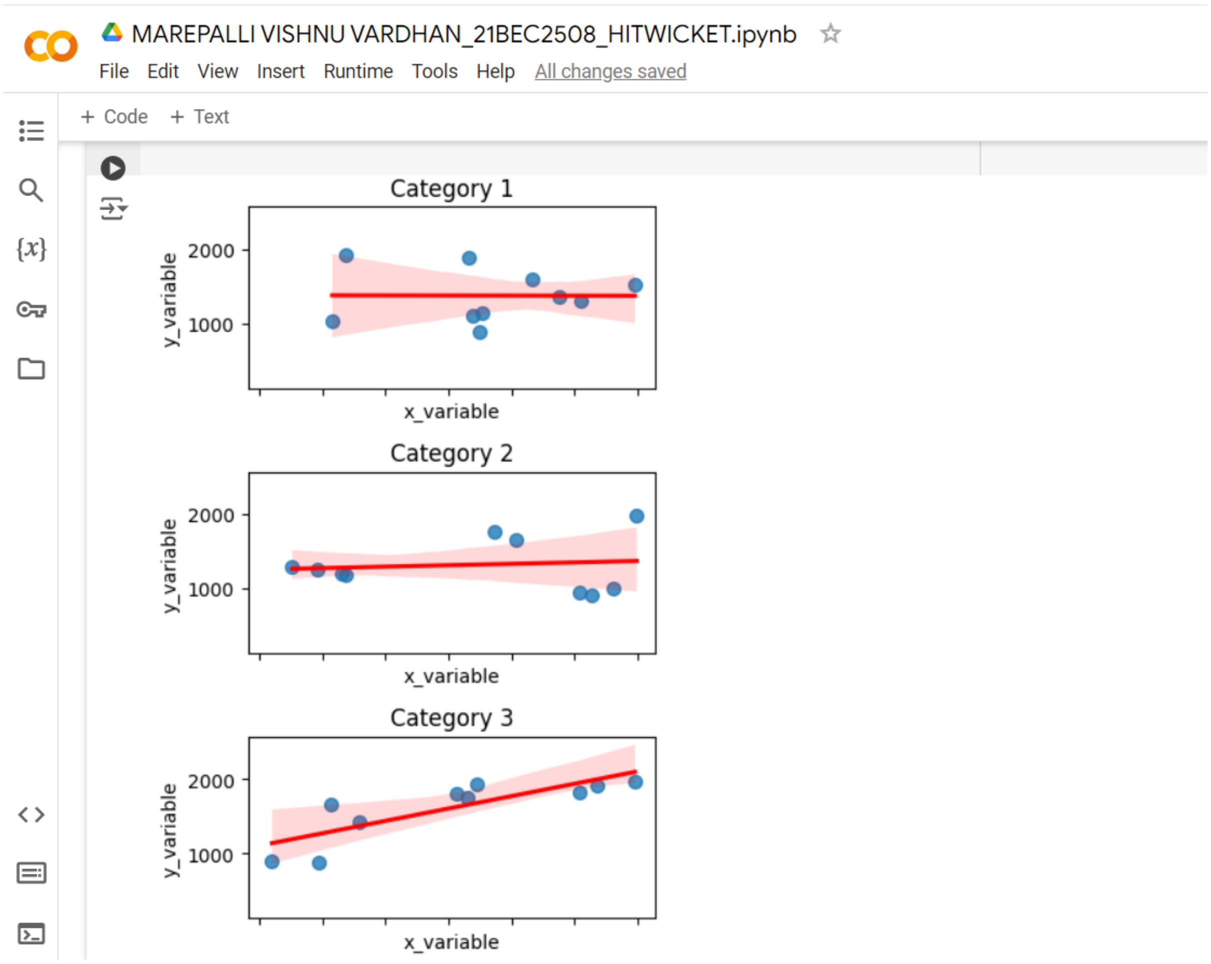
6


MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

For Problems 6-7, assume you have a dataframe called table_to_plot which contains 3 columns: Category, where values range from 1-9, x_variable and y_variable, where values range from 800-2000. A glimpse of the first few rows of the data frame is attached:

Problem 6 Use R or Python to plot all of the following graphs at once, in one window, for easier comparison of categories

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
# Sample DataFrame (replace this with your actual DataFrame)
data = {
    'Category': np.repeat(np.arange(1, 10), 10),
    'x_variable': np.random.randint(800, 2000, 90),
    'y_variable': np.random.randint(800, 2000, 90)
}
table_to_plot = pd.DataFrame(data)
# Set up the figure and axes for subplots
num_categories = table_to_plot['Category'].nunique()
fig, axes = plt.subplots(num_categories, 1, figsize=(4, 2 * num_categories), sharex=True, sharey=True)

for i, category in enumerate(sorted(table_to_plot['Category'].unique())):
    ax = axes[i] if num_categories > 1 else axes
    subset = table_to_plot[table_to_plot['Category'] == category]
    sns.regplot(data=subset, x='x_variable', y='y_variable', ax=ax, scatter_kws={'s': 50}, line_kws={"color": "red"})
    ax.set_title(f'Category {category}')
    ax.set_xlabel('x_variable')
    ax.set_ylabel('y_variable')
plt.tight_layout()
plt.show()
```


MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

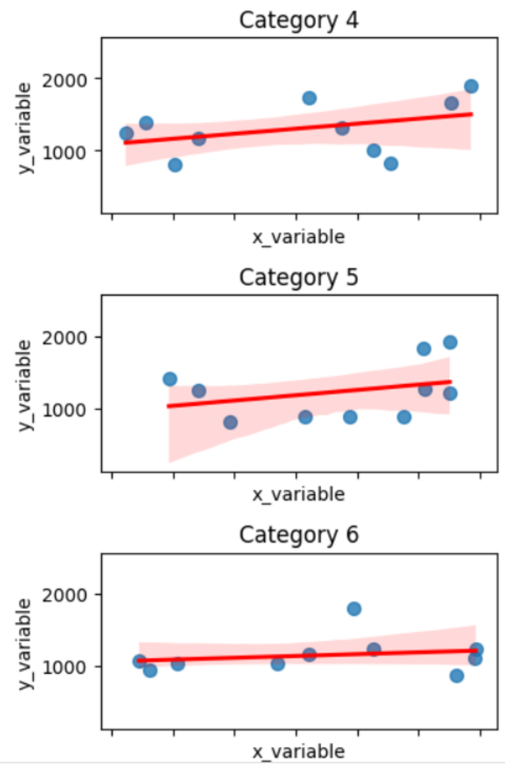
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

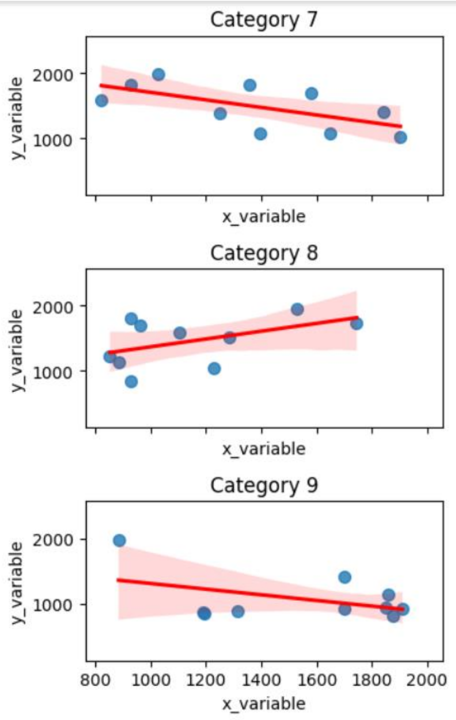
MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb

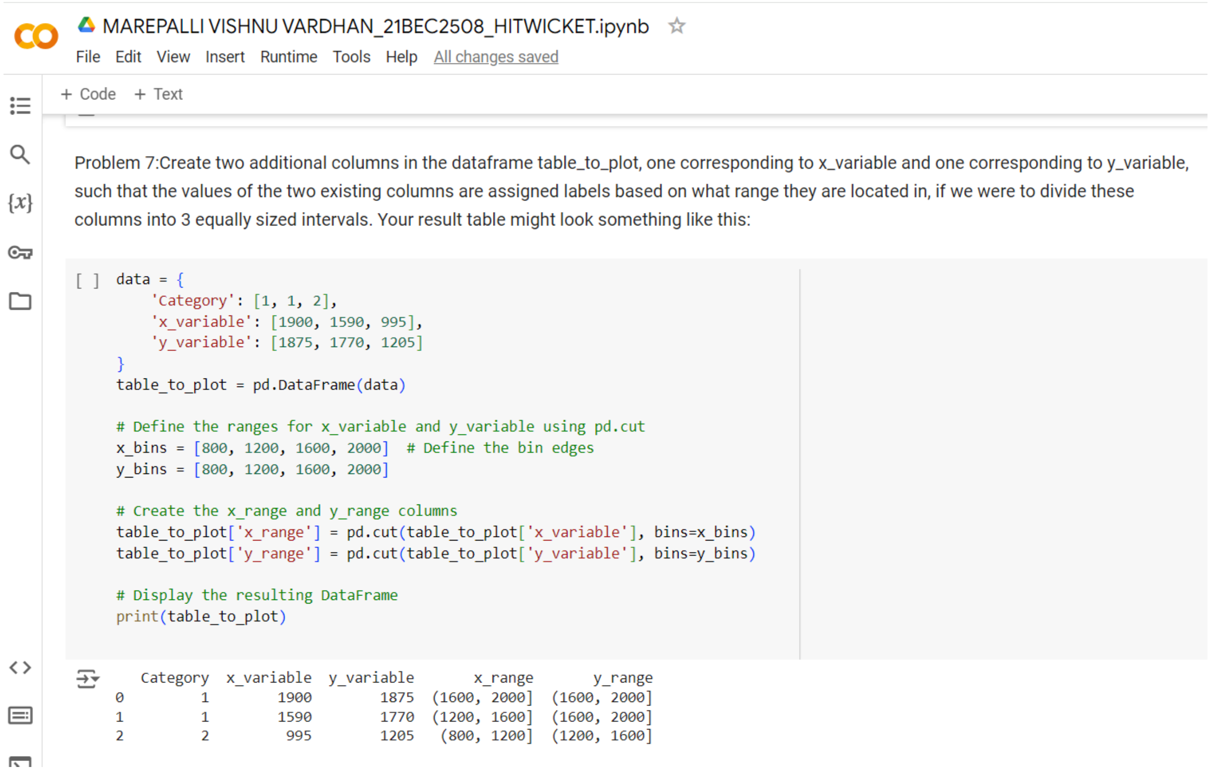
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text



+ Code + Text




MAREPALLI VISHNU VARDHAN_21BEC2508_HITWICKET.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

Problem 7:Create two additional columns in the dataframe table_to_plot, one corresponding to x_variable and one corresponding to y_variable, such that the values of the two existing columns are assigned labels based on what range they are located in, if we were to divide these columns into 3 equally sized intervals. Your result table might look something like this:

```
data = {
    'Category': [1, 1, 2],
    'x_variable': [1900, 1590, 995],
    'y_variable': [1875, 1770, 1205]
}
table_to_plot = pd.DataFrame(data)

# Define the ranges for x_variable and y_variable using pd.cut
x_bins = [800, 1200, 1600, 2000]  # Define the bin edges
y_bins = [800, 1200, 1600, 2000]

# Create the x_range and y_range columns
table_to_plot['x_range'] = pd.cut(table_to_plot['x_variable'], bins=x_bins)
table_to_plot['y_range'] = pd.cut(table_to_plot['y_variable'], bins=y_bins)

# Display the resulting DataFrame
print(table_to_plot)
```

```
   Category  x_variable  y_variable       x_range       y_range
0         1        1900        1875  (1600, 2000]  (1600, 2000]
1         1        1590        1770  (1200, 1600]  (1600, 2000]
2         2         995        1205   (800, 1200]  (1200, 1600]
```

Section 3: Problem Solving Approach

2.Feature Improvement

Assuming you've explored Hitwicket and based on other mobile game experiences, identify one game feature that you believe has room for improvement.

a. What specific data or user feedback would you collect to establish that the feature has scope for improvement.

b. List down your top 3 suggestions/ideas of improving the feature including the pros and cons of all the ideas.

c. Suggest an experiment and the success metrics to validate your best idea.

d. Explain the reason behind your proposed experiment and possible drawbacks if any where the experiment could fail.

Feature Improvement: Balanced Matchmaking System

Context:
I've been playing Hitwicket for about 710 days now, and while I've seen my team grow—starting with just 1 star player and now having 4 (3 batsmen and 1 bowler)—I've noticed a significant issue. At my current level (Silver 2, Level 31), I'm often matched against players who are much higher in level (e.g., Levels 45, 46, 39) and have teams stacked with 7 or more star players. These matches feel onesided and frustrating, as I'm often losing before the game even starts. The lack of interest comes from knowing that, unless I somehow acquire more star players, I'm unlikely to compete effectively. Also, it feels like only the star players make an impact, while the rest of my team underperforms.

Data and User Feedback to Collect:

1. Match Outcomes: Collect data on the outcomes of matches for players at different levels, specifically looking at win/loss ratios when lowerlevel players face significantly higherlevel opponents.
2. Player Feedback on Match Fairness: Survey players at various levels to gather feedback on their perception of match fairness, particularly focusing on those with fewer star players.
3. Star Player Distribution: Analyze the distribution of star players across different player levels to understand if there's a disparity affecting match outcomes.
4. DropOff Rates: Track if there's a higher dropoff rate among players who frequently lose to much higherlevel opponents.
5. Performance Metrics of NonStar Players: Evaluate the performance metrics of nonstar players in these matches to see if they truly are underperforming, contributing to the imbalance.

Top 3 Suggestions for Improving the Feature:

1. LevelBased Matchmaking Adjustment:
Idea: Implement stricter matchmaking rules that prioritize pairing players against opponents of a similar level and star player count.
Pros: Creates a more balanced and competitive environment, reducing frustration and increasing player retention.
Cons: Could lead to longer wait times for matches if the pool of similarlyleveled players is small, potentially frustrating users.

2. Dynamic Star Player Allocation:
Idea: Introduce a system where players temporarily receive additional star players or boosts when matched against significantly higherlevel opponents.

Pros: Levels the playing field in uneven matchups, making games more competitive and enjoyable.
Cons: Might reduce the perceived value of earning star players naturally, and could feel artificial or unfair to higherlevel players.

3. Enhanced Performance of NonStar Players:
Idea: Improve the performance of nonstar players in matches, particularly when they're up against a stronger team, through temporary stat boosts or improved AI.
Pros: Ensures that every player contributes to the match, making games feel less dependent on star players alone.
Cons: Balancing this without making nonstar players overpowered could be challenging, potentially disrupting the game's competitive integrity.

Experiment to Validate the Best Idea:

Best Idea:
LevelBased Matchmaking Adjustment

Experiment:
Test Enhanced Matchmaking Algorithm:
Setup: Roll out an updated matchmaking algorithm to a subset of players that prioritizes matching them with opponents who are within a close level range and have a similar number of star players.
Control Group: Players continue with the current matchmaking system.
Test Group: Players experience the new, levelbased matchmaking.

Success Metrics:
Match Outcome Balance: Track the win/loss ratios to see if matches are more balanced.
Player Satisfaction: Survey players in the test group on their enjoyment and perceived fairness of the matches.
Retention Rates: Compare the retention rates between the control and test groups to see if the new matchmaking improves player engagement.

Reasoning:
By ensuring that players face opponents of similar strength, the game becomes more enjoyable and competitive, addressing the frustration of losing before the match even begins.

Possible Drawbacks:
Increased Wait Times: If the player pool at certain levels is small, matching players with similar opponents might take longer, potentially causing frustration.
Perceived Limitations: Highlevel players might feel constrained by facing only

similar opponents, potentially reducing the excitement of challenging stronger teams.

This approach directly addresses the issue of unbalanced matches, making the game more enjoyable and competitive for all players.