










Lucky Draw ANALYSIS

Table of Content



	Introduction
	Performance Dashboard
	Ad-hoc Analysis
	Target Audience
	Key Tracking Metrics
	Potential Experiments
	Appendix

01

Introduction

About the Draw

The Lucky Draw is a game that selects certain payments made by debit card holders at random, and then reimburses the amount to the user after the random selection in the draw takes place.

This draw is done among every 1000 payments made using the debit card. The lucky user whose name gets drawn receives the refund amount directly in their account.

However, there are certain conditions the user needs to meet to be eligible for the draw. The payments that are not considered for the draw are ATM cash withdrawals and money transfers.

Further, the reward amount depends on the user's subscription type as well.

If the user chosen does not have a subscription, then only half the amount is reimbursed, not exceeding 50€.

- If a user has a Blue subscription, then they become eligible to win the entire amount of their payment in reimbursement for amounts upto 100€.
- If a user has Black subscription, then they become eligible to win twice the amount of their entire payment in reimbursement for amounts upto 200€.
- Additionally, if a user has Blue subscription for less than a month and has not won a reward through the draw, they become eligible for another draw which would choose among every 50 payments made among Blue subscribers less than a month belonging to the same country.

The winner of the draw is notified about their reward via their registered e-mail id.

02

Performance Dashboard

No. of Transactions

86,962,201

Avg. Amount Spent

227,759.54

Avg. Amount Won

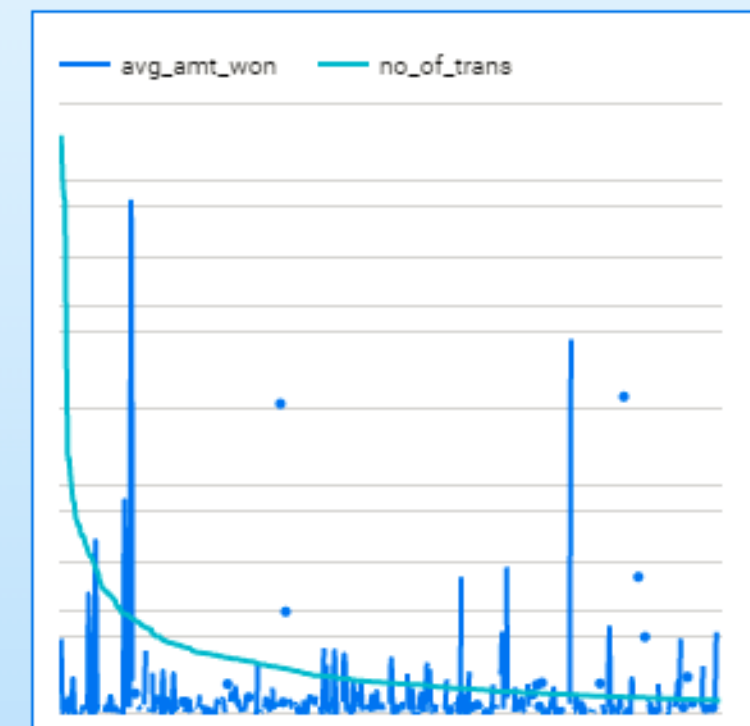
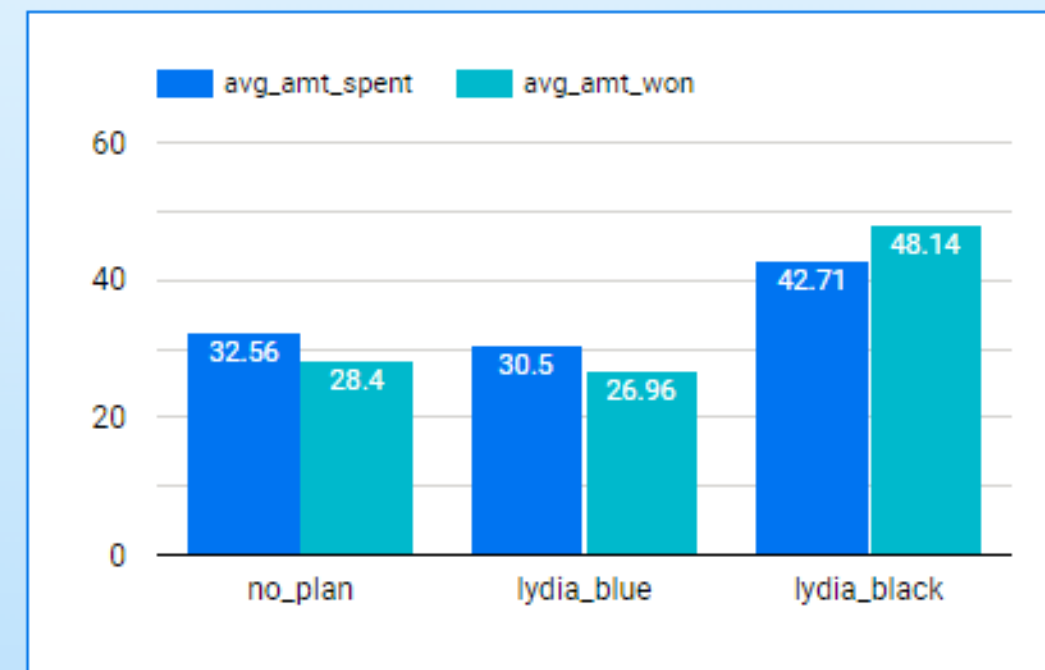
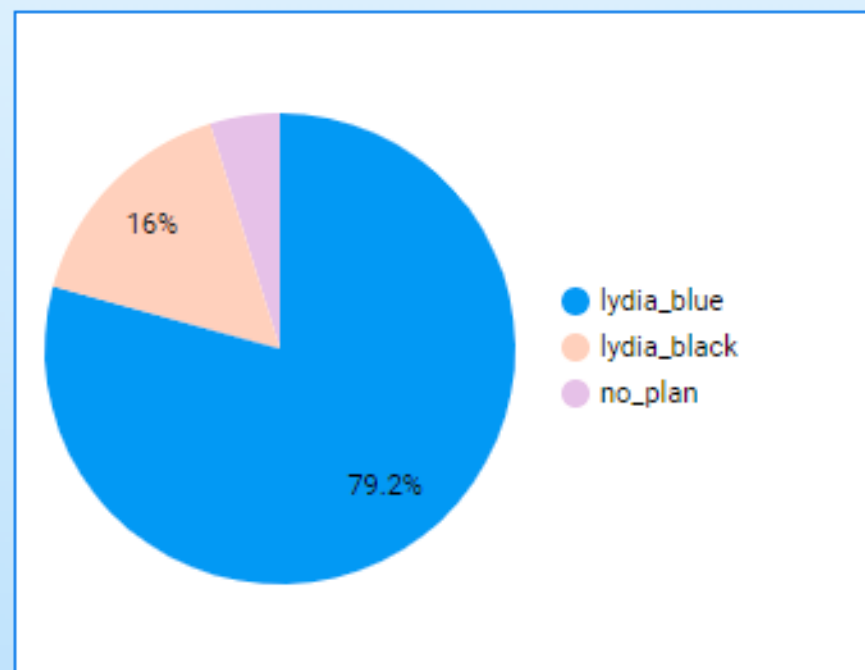
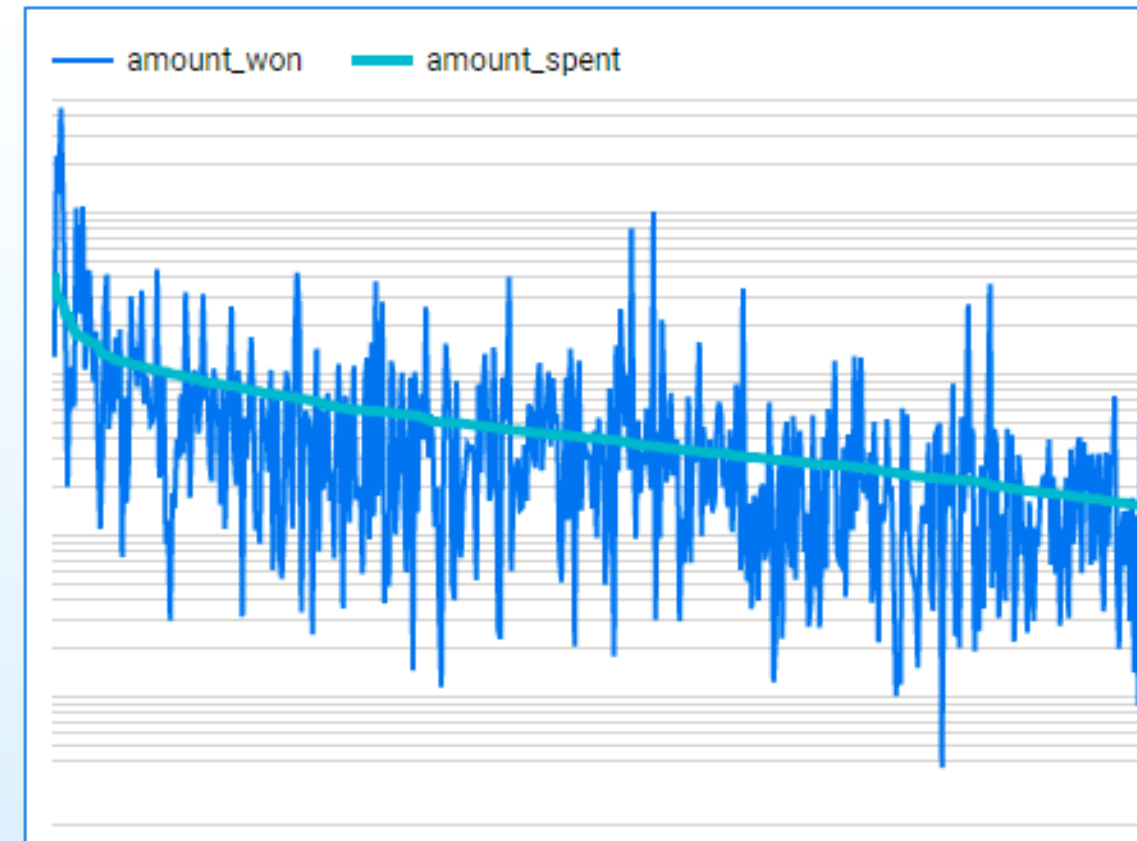
66,595.65

plan

Select date range

spender_id	no_of_trans ▾	avg_amt_spent	avg_amt_won
5500285	2,827,442	14.28	31.96
3841427	1,874,048	262.07	79.86
1222807	1,517,824	50.9	367
2782317	1,370,904	48.32	66.79
1503885	1,357,952	41.01	26
2415099	1,305,728	31.7	31.26
4824329	1,254,400	150.32	588.48
1065085	929,296	47.42	19.9
2785485	826,898	74.06	10
7694093	786,258	163.52	1,046.76
941807	783,752	66.16	7.92

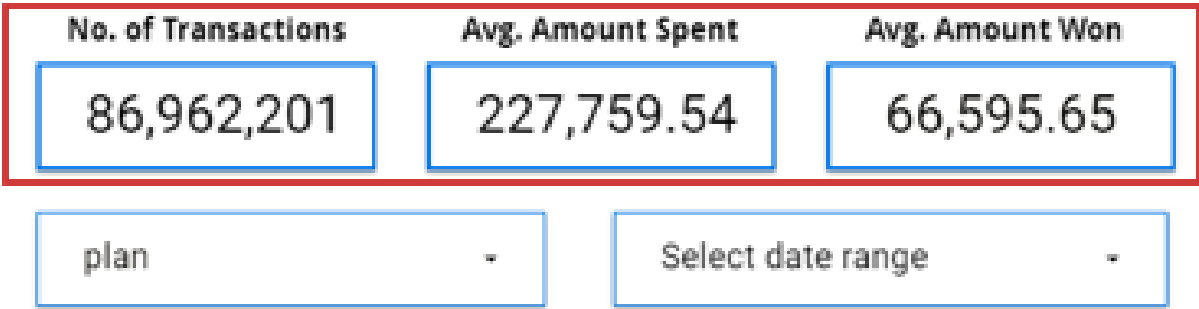
1 - 100 / 5226 < >



This dashboard is created using Google Data Studio.

> User Guidelines

Key Metrics

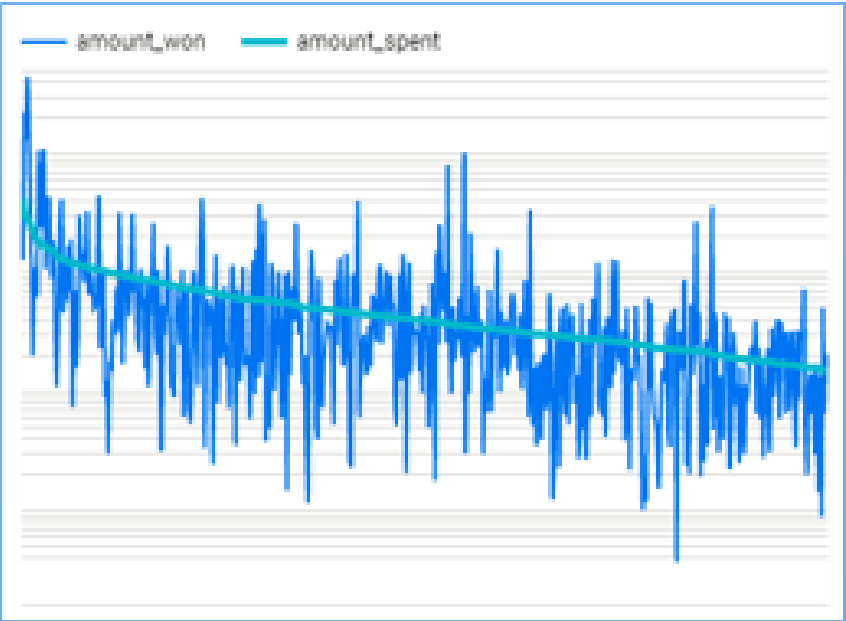


The Plan filter allows one to view and analyse the dashboard separately for each plan.

The Date filter allows one to study the dashboard for a customized duration of time.

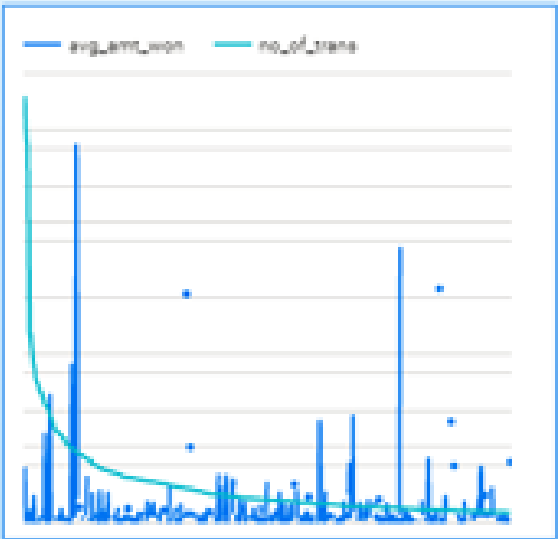
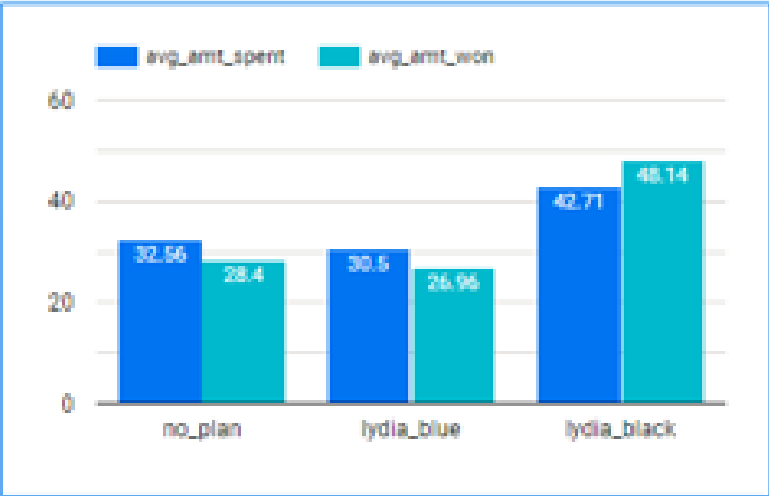
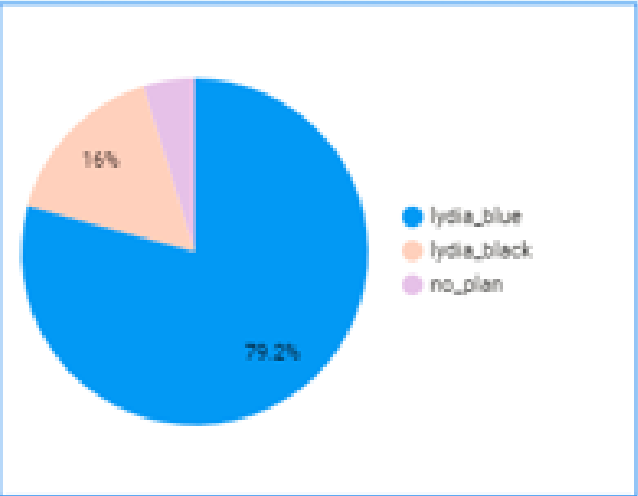
I. Tabular representation of data with a heatmap conditional formatting to provide an easy way to check data points for comparison.

spender_id	no_of_trans	avg_amt_spent	avg_amt_won
5500285	2,827,442	14.28	31.96
3841427	1,874,048	262.07	79.86
1222807	1,517,824	50.9	367
2782317	1,370,904	48.32	66.79
1503885	1,357,952	41.01	26
2415099	1,305,728	31.7	31.26
4824329	1,254,400	150.32	588.48
1065085	929,296	47.42	19.9
2785485	826,898	74.06	10
7694093	786,258	163.52	1,046.76
941807	783,760	66.16	7.90



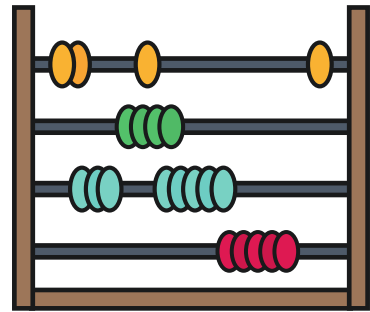
IV. Line chart plotting amount_spent vs. amount_won to track the trend.

III. Bar chart of avg_amount_spent vs avg_amount_won for each plan to track performance of each plan.



V. Line chart plotting avg_amount_won vs. no_of_transactions to check for correlation.

Key Metrics



No. of Transactions

The number of transactions made by each user.



Average amount spent by a user

The average amount spent by a user for the duration of the dataset.



Average amount won by a user

The average amount won by a user for the duration of the dataset.

03

Ad-hoc
Analysis

The Datasets

Roulette Winners Dataset

roulette_winners			
member_id	operation_id	date	amount
1	3	2021-01-01	10
2	4	2021-01-02	25

Card Transaction Dataset

card_transactions						
spender_id	operation_id	date	amount	status	plan	card_activation_date
1	3	2021-01-01	10	completed	blue	2020-01-01
2	4	2021-01-02	25	canceled	no_plan	2020-09-01

Methodology

1. After studying both the datasets, 'Roulette Winners' and 'Card Transactions', an inner join was used to merge them to first check for the number of distinct users.

JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS
Row	n_spender		n_unique_spender
1	127904		842

The two datasets have been merged on spender_id and member_id.

2. The two datasets are merged using inner join to create table 1 - 'Winner_analysis'. The new table is grouped by spender_id and contains the following columns that are calculated for each Lydia user -

- No. of transactions (count of operation_id)
- Average amount spent by a user
- Average amount won by a user
- Total amount spent by a user
- Total amount won by a user
- Plan subscribed to by a user

Note: Since inner join is performed, only users who have won the roulette atleast one or more times are included in the table.

Methodology

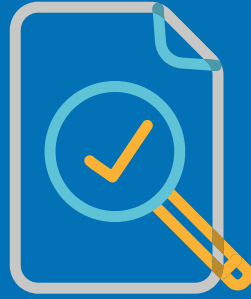


3. Table 2 - 'not_won_analysis', is created by performing a left outer join to merge both the datasets. As the left dataset is Card Transactions, the new table created contains users who have won and who have not. The table created is similar to table 1.

4. A third and final table, Table 3 is created by performing left outer join to merge the datasets. This table is not grouped by spender_id and accounts for each transaction of each user individually. This table has been created for exploratory data analysis on python.

Note: All the tables created so far have been made using BigQuery. Please find the queries in the appendix section.

Exploratory Data Analysis



Python

For table 3, contains
information on all users.

Duration of the dataset

Start dates:

The first transaction made by a user was on: 2020-10-01 00:00:00

The first amount won by a user was on: 2020-12-01 00:00:00

End dates:

The last transaction made by a user was on: 2021-03-29 00:00:00

The last amount won by a user was on: 2021-01-29 00:00:00

User Information

Total number of users - **5226**

Number of users who have won atleast once - **842**

Number of users who have never won - **4113**

Thus, **20.47%** of total users have won atleast once.

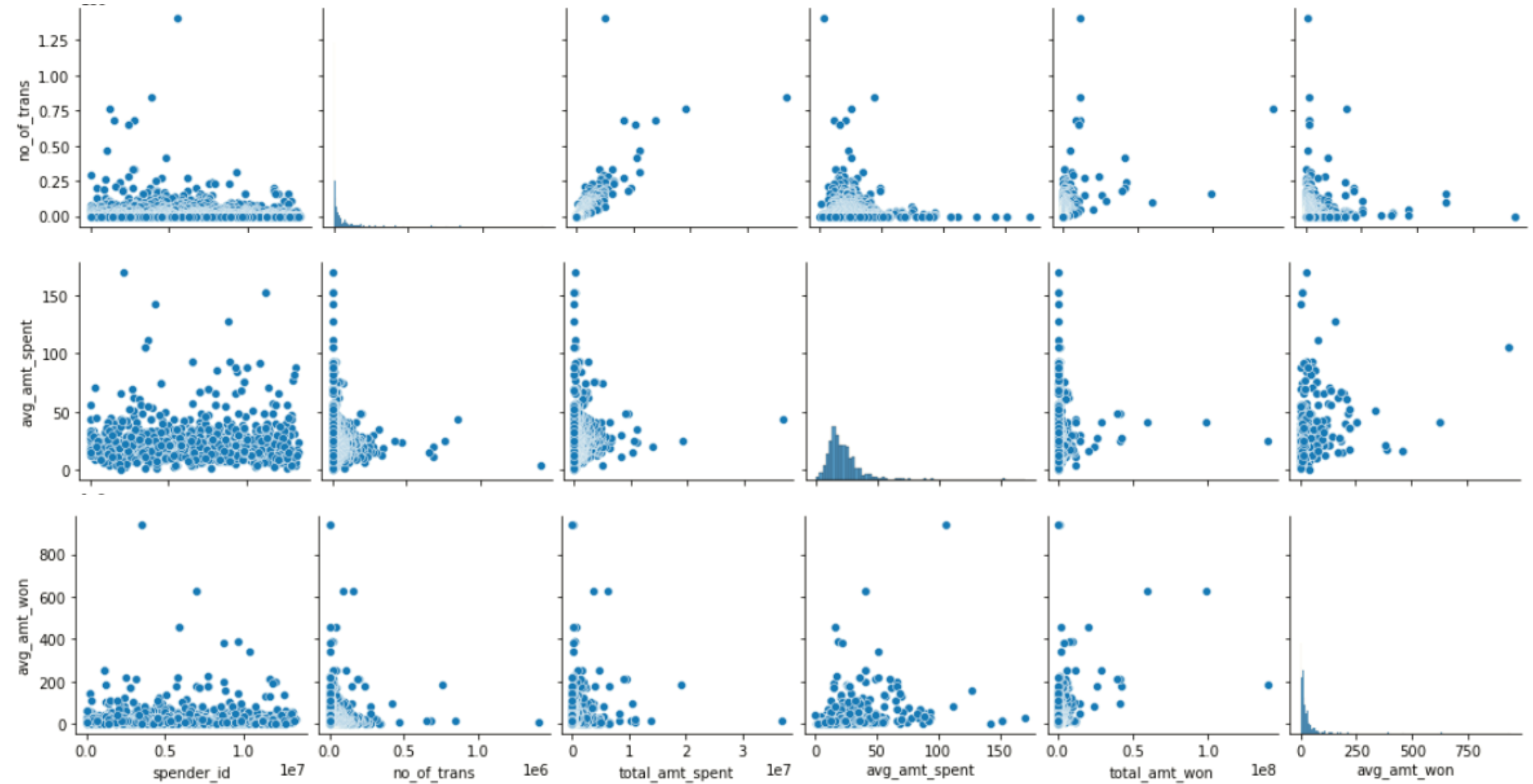
Types of Plans - 3 - Black, Blue, No Plan

Types of transaction Status - 3 - Pending, Completed, Cancelled

Analysis



Correlation between the key metrics



	spender_id	no_of_trans	total_amt_spent	avg_amt_spent	total_amt_won	avg_amt_won
spender_id	1.000000	-0.123766	-0.111098	0.037971	-0.046685	-0.021619
no_of_trans	-0.123766	1.000000	0.807182	-0.025623	0.441627	0.037325
total_amt_spent	-0.111098	0.807182	1.000000	0.110529	0.526067	0.108041
avg_amt_spent	0.037971	-0.025623	0.110529	1.000000	0.064357	0.288053
total_amt_won	-0.046685	0.441627	0.526067	0.064357	1.000000	0.421542
avg_amt_won	-0.021619	0.037325	0.108041	0.288053	0.421542	1.000000

The following observations are made from the previous slide :

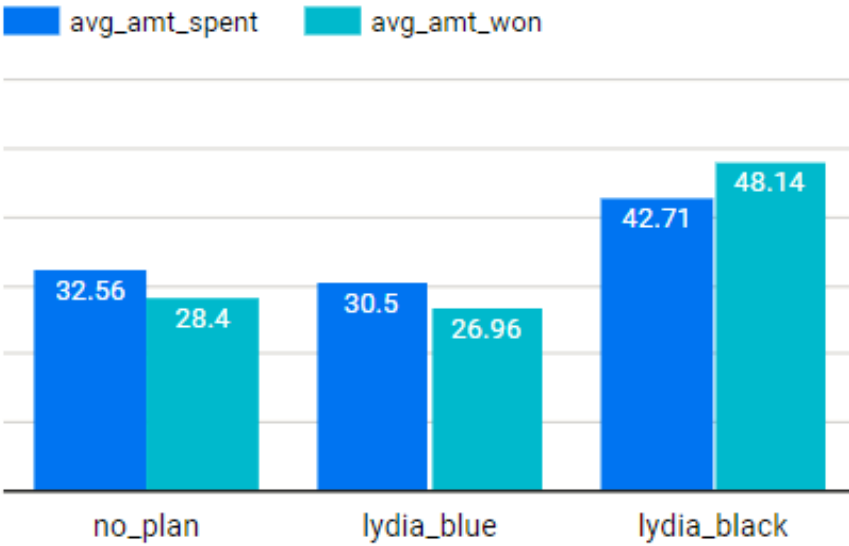
1. It is observed on the plot that the number of transactions made by the user is mildly and positively related to the amount won by a user thus meaning, the higher the intensity of usage, the higher the chances of winning as the graph is extending rightward.

This is confirmed by the correlation of 3.73%.

2. At the same time, it is interesting to note that while the total amount spent by a user shares a strong positive correlation with number of transactions a user makes (as one would expect), it is not the case with number of transactions vs. average amount spent. There exists a very minute negative correlation. However, this can be understood with the help of the graph and this correlation can be attributed to the presence of certain extreme outliers.
3. The average amount spent by a user shares a positive correlation of 28.8% with average amount a user is likely to win. The plots for both these variables are extending rightwards. While it can stand to be beneficial from a user point of view, for the company it could mean the opposite as it indicates that if this correlation continues to increase, the company will be giving away higher amounts of money in the roulette.



Further Observations :



	spender_id	no_of_trans	total_amt_spent	avg_amt_spent	total_amt_won	avg_amt_won
plan						
lydia_black	6.918366e+06	49908.875000	1.324508e+06	29.359668	3.391685e+06	48.142125
lydia_blue	6.230781e+06	39772.902405	8.622325e+05	23.368987	1.155974e+06	26.955186
no_plan	6.628614e+06	4853.909396	1.037795e+05	21.842365	1.380359e+05	28.400845

- Black users spend highly and win the most amount. The high amount won is explained by the black benefit of earning 2x time the transaction made.
- Blue users spend the most amount but win the least.
- Users with no subscription plan are noted to win higher amounts (as no. of transactions is low) than Lydia blue users.

	spender_id	member_id	amount_spent	amount_won
status				
cancelled	6.342136e+06	5.910332e+06	-11.212602	40.931968
completed	6.049670e+06	5.843042e+06	26.947449	31.731634
pending	5.669338e+06	5.481148e+06	25.315611	33.403988

- High amount won by users having 'cancelled' transaction status.
- Amount won by both those having the status, 'pending' or 'completed', is higher than the amount spent.
- Users having 'completed' transaction status have the least amount comparatively.

Recommendations



It would be recommended to the company to keep a check on how many times a particular user is winning over a certain period of time. Putting a limit to the wins in a certain time period will ensure that the results are not biased and that all users get a chance. This will generate a widespread customer satisfaction over the user base and will help retaining customers.



The current analysis shows that the average amount won by the group of users who are not subscribed to a plan is higher than that of Blue. This indicates that it could be a situation where cash outflow is greater than cash inflow. It would be advisable to focus marketing efforts on this group of customers to convert them into Blue members.

As there is already a cap on the amount of money that can be won by a user without a plan, an additional limit can be applied to the number of times a user without a plan can win.



The analysis further shows that users who have a higher rate of 'completed' transaction status are overall winning the least amount. Transaction statuses having 'cancelled' and 'pending' have won higher amounts (in total). It would be recommended to the company to keep track of a user's successful transactions vs. cancelled transactions and understand how it affecting the company's profits.



04

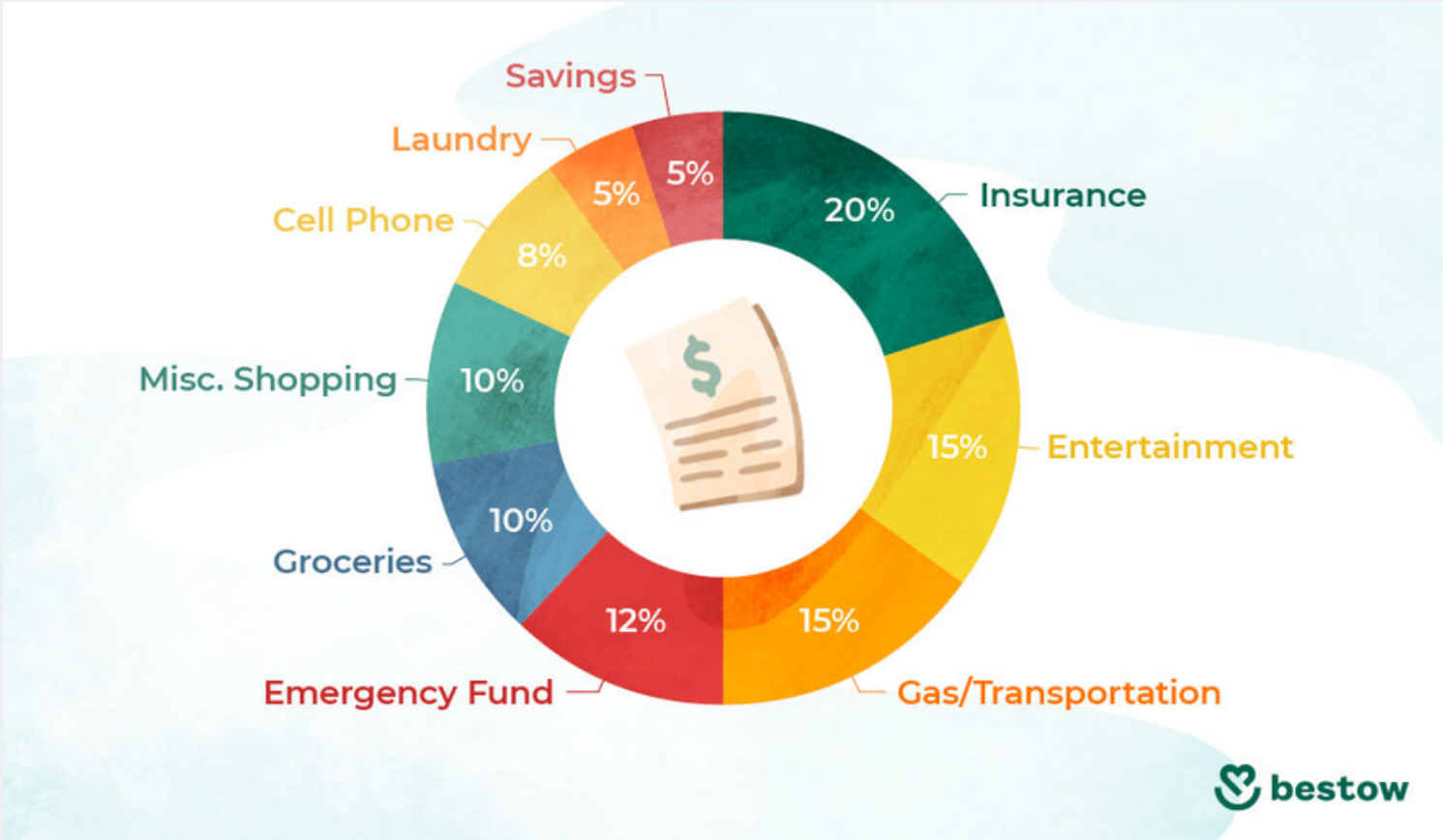
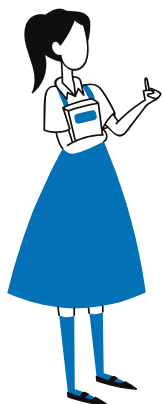
Target
Audience

Demographic

Age & Income

Target - Students & Young working population.

This is a huge customer group. Their spending frequency is high but average amount spent is low. This helps Lydia tap into a huge consumer base and provide the attractive offer of the chance to win back money. To a student, winning 10 Euros generates more satisfaction than a person with a high paying job. Students also greatly aid word-of-mouth advertising.

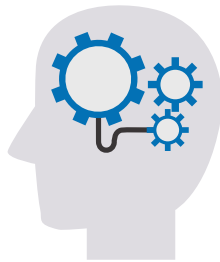


Stop Buying in Bulk

It ends up wasting food and money. Shop more frequently instead.

You should go to the grocery store every day. Several of them. Seriously.

Why Buying in Bulk Doesn't Always Save You Money



Psychographic

Lifestyle & Personality

It would be beneficial for the company to analyse and understand the spending patterns of consumers and group them as those who make several frequent low amount transactions versus those who make less frequent high amount transactions. For example, many individuals prefer buying fresh groceries on daily basis whereas some people bulk buy for the week.

05

Key Tracking Metrics

Key Success Metrics

It is important to track the total incoming amount and total outgoing amount and the difference between them.

In this case, the total incoming amount would be generated from the card transactions made by the user as well as from their subscription plan. The total outgoing amount is amount paid as price money to the winner.

It is important to ensure that at no point cash outflow is greater than cash inflow as this will have harm the company's net profit.

Secondary Goals

A secondary goal would be to track the performance for each plan and understand which plan is the most profitable one for the company.


Further, to track, the winning chances of each plan and their subsequent consequences on the company's profit.



06

Potential Experiments

> Important Parameters for an Experimental Setup



An experimental setup can be done for different age groups, where each group is presented with curated messages and creatives for them, to identify the user age group which is most engaged with Roulette.

The best performing test can be identified by studying the best outcomes from the following parameters -

1. Landing Page - how comprehensive is it ? ease of use ?
2. UX - self explanatory structure & navigation
3. UI - appealing visual hierarchy
4. Relevance of creatives
5. Click through rate to the next step
6. Bounce rate (should be low)
7. Level of delight on winning
8. Level of dissatisfaction on losing
9. Likelihood of using Lydia card again
10. Likelihood of the user telling a non-user about Lydia and encouraging them to use it.

A weight can be assigned to each of the parameters based on their importance. Then a final weighted average can be calculated depending on their performance score to identify the best performing test.

07

Appendix



SQL QUERIES



I. Trying to view a temporary table


```
#standardSQL
CREATE VIEW IF NOT EXISTS `data_analyst_case.winer_analysis`
OPTIONS(
  description = ""
) AS
SELECT
  card_tans.spender_id,
  card_tans.amount as amount_spent,
  card_tans.operation_id,
  card_tans.plan,
  card_tans.status,
  user_wins.member_id,
  user_wins.amount as amount_won
FROM `i-hiring.data_analyst_case.card_transactions` as card_tans
INNER JOIN `i-hiring.data_analyst_case.roulette_winners` as user_wins
ON card_tans.spender_id = user_wins.member_id
```



II. Checking if there are distinct users

```
WITH winer_analysis AS
(
  SELECT
    card_tans.spender_id,
    card_tans.amount as amount_spent,
    card_tans.operation_id,
    card_tans.plan,
    card_tans.status,
    user_wins.member_id,
    user_wins.amount as amount_won
  FROM `__hiring.data_analyst_case.card_transactions` as card_tans
  INNER JOIN `__hiring.data_analyst_case.roulette_winners` as user_wins
  ON card_tans.spender_id = user_wins.member_id)

SELECT
  COUNT(spender_id) as n_spender,
  COUNT(DISTINCT spender_id) as n_unique_spender
FROM winer_analysis;
```





III. Merged the two datasets to create a new table 'Winner_analysis' (Table I)

```
WITH winner_analysis AS
(
    SELECT
        card_trans.spender_id,
        card_trans.amount as amount_spent,
        card_trans.operation_id,
        card_trans.plan,
        card_trans.status,
        user_wins.member_id,
        user_wins.amount as amount_won
    FROM `i-hiring.data_analyst_case.card_transactions` as card_trans
    INNER JOIN `i-hiring.data_analyst_case.roulette_winners` as user_wins
    ON card_trans.spender_id = user_wins.member_id)

SELECT winner_analysis.spender_id,
COUNT(winner_analysis.operation_id) as no_of_trans,
SUM(amount_spent) as total_amt_spent,
AVG(amount_spent) as avg_amt_spent,
SUM(amount_won) as total_amt_won,
AVG(amount_won) as avg_amt_won,
card_trans.plan
FROM winner_analysis
INNER JOIN `i-hiring.data_analyst_case.card_transactions` as card_trans
ON winner_analysis.spender_id = card_trans.spender_id
GROUP BY spender_id, plan
ORDER BY no_of_trans DESC;
```



IV. Merged the two datasets to create a new table with all users (Table 2)

```
WITH not_won_analysis AS
(
  SELECT
    card_trans.spender_id,
    card_trans.amount as amount_spent,
    card_trans.operation_id,
    card_trans.plan,
    card_trans.status,
    user_wins.member_id,
    user_wins.amount as amount_won
  FROM `    -hiring.data_analyst_case.card_transactions` as card_trans
  LEFT OUTER JOIN `    -hiring.data_analyst_case.roulette_winners` as user_wins
  ON card_trans.spender_id = user_wins.member_id)

SELECT not_won_analysis.spender_id,
COUNT(not_won_analysis.operation_id) as no_of_trans,
SUM(amount_spent) as total_amt_spent,
AVG(amount_spent) as avg_amt_spent,
SUM(amount_won) as total_amt_won,
AVG(amount_won) as avg_amt_won,
card_trans.plan
FROM not_won_analysis
INNER JOIN `    -hiring.data_analyst_case.card_transactions` as card_trans
ON not_won_analysis.spender_id = card_trans.spender_id
GROUP BY spender_id, plan
ORDER BY total_amt_won ASC;
```



V. Merged the two datasets to create a new table with all users (Table 3)

```
SELECT
    card_trans.spender_id,
    user_wins.member_id,
    card_trans.operation_id as transaction_id,
    card_trans.amount as amount_spent,
    user_wins.amount as amount_won,
    card_trans.plan,
    card_trans.status,
    card_trans.date as transaction_date,
    user_wins.date as reimburse_date
FROM `[-hiring.data_analyst_case.card_transactions` as card_trans
LEFT OUTER JOIN `[-hiring.data_analyst_case.roulette_winners` as user_wins
ON card_trans.spender_id = user_wins.member_id
```



PYTHON SCRIPTS

```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# importing exported CSV files

df = pd.read_csv(r'C:\Users\HiyaBanerjee\OneDrive\Desktop\Winner_analysis.csv')
df1 = pd.read_csv(r'C:\Users\HiyaBanerjee\OneDrive\Desktop\Left Outer Join - Full dataset.csv')
```

```
df.head()
```

	spender_id	no_of_trans	total_amt_spent	avg_amt_spent	total_amt_won	avg_amt_won	plan
0	5500285	1406587	5022118.92	3.570429	1.123863e+07	7.990000	lydia_blue
1	3841427	846032	36952580.16	43.677521	1.126069e+07	13.310000	lydia_blue
2	1222807	758912	19314408.96	25.450130	1.392604e+08	183.500000	lydia_blue
3	2782317	679716	8211196.80	12.080335	1.134899e+07	16.696667	lydia_blue
4	1503885	678976	13924001.44	20.507354	8.826688e+06	13.000000	lydia_blue

```
df1.head()
```

	spender_id	member_id	transaction_id	amount_spent	amount_won	plan	status	transaction_date	reimburse_date
0	2172161	2172161.0	c815b8bb-aca5-4447-a209-f6d4b0ebf6fa	5.49	75.2	lydia_black	pending	2021-03-29	2021-01-04
1	2172161	2172161.0	052d125b-38e2-4707-a359-6e6fd38bc5d7	10.99	75.2	lydia_black	pending	2021-03-29	2021-01-04
2	6891777	NaN	0f1baa2c-068a-4f73-a21b-ae13fad02418	120.83	NaN	lydia_blue	pending	2021-03-27	NaN
3	12792577	NaN	11cf846a-7c26-42d7-9f14-187f7c916799	26.84	NaN	lydia_black	pending	2021-03-27	NaN
4	12792577	NaN	bdca0e15-c7a9-4a98-9e08-c4b335b495c3	4.66	NaN	lydia_black	pending	2021-03-29	NaN

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1125 entries, 0 to 1124
Data columns (total 7 columns):
#   Column          Non-Null Count  Dtype
---  -
0   spender_id      1125 non-null   int64
1   no_of_trans     1125 non-null   int64
2   total_amt_spent  1125 non-null   float64
3   avg_amt_spent   1125 non-null   float64
4   total_amt_won   1125 non-null   float64
5   avg_amt_won     1125 non-null   float64
6   plan            1125 non-null   object
dtypes: float64(4), int64(2), object(1)
memory usage: 61.6+ KB
```

df1.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250781 entries, 0 to 250780
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   spender_id      250781 non-null int64
1   member_id       127904 non-null float64
2   transaction_id  250781 non-null object
3   amount_spent    250781 non-null float64
4   amount_won      127904 non-null float64
5   plan            250781 non-null object
6   status          250781 non-null object
7   transaction_date 250781 non-null object
8   reimburse_date  127904 non-null object
dtypes: float64(3), int64(1), object(5)
memory usage: 17.2+ MB
```

df.describe()

	spender_id	no_of_trans	total_amt_spent	avg_amt_spent	total_amt_won	avg_amt_won
count	1.125000e+03	1.125000e+03	1.125000e+03	1125.000000	1.125000e+03	1125.000000
mean	6.409505e+06	3.160442e+04	7.106360e+05	23.603608	1.124810e+06	29.598065
std	3.650186e+06	7.926994e+04	1.804529e+06	17.415853	6.217817e+06	64.063744
min	7.815000e+03	1.000000e+00	1.420000e-14	0.000000	8.330000e+00	0.530000
25%	3.481883e+06	1.521000e+03	2.780271e+04	13.961884	1.422135e+04	5.980000
50%	6.406663e+06	7.056000e+03	1.520439e+05	19.869784	9.096684e+04	13.360000
75%	9.433351e+06	3.027600e+04	7.073501e+05	27.908000	5.229000e+05	29.800000
max	1.332366e+07	1.406587e+06	3.695258e+07	169.816857	1.392604e+08	937.200000

df1.describe()

	spender_id	member_id	amount_spent	amount_won
count	2.507810e+05	1.279040e+05	250781.000000	127904.000000
mean	6.061011e+06	5.842575e+06	24.623488	32.307722
std	3.668987e+06	3.523906e+06	66.999328	68.617795
min	1.550000e+02	7.815000e+03	-4998.000000	0.530000
25%	2.839865e+06	2.782317e+06	3.000000	6.250000
50%	5.912401e+06	5.667343e+06	10.090000	13.360000
75%	8.878577e+06	8.660053e+06	23.670000	30.000000
max	1.414198e+07	1.332366e+07	3992.540000	937.200000

```
df.shape
```

```
(127904, 8)
```

```
df1.shape
```

```
(250781, 9)
```

```
# How many distinct plans are available?
```

```
num_plans = df['plan'].nunique()
```

```
num_plans
```

```
3
```

```
list(df['plan'].unique())
```

```
['lydia_black', 'lydia_blue', 'no_plan']
```

```
# What are the different statuses?
```

```
n_status = df1['status'].nunique()
```

```
n_status
```

```
3
```

```
list(df1['status'].unique())
```

```
['pending', 'cancelled', 'completed']
```

```
# Duration of the dataset
```

```
# Start date
```

```
a = df1['transaction_date'].astype('datetime64[ns]').min()
```

```
b = df1['reimburse_date'].astype('datetime64[ns]').min()
```

```
print("The first transaction made by a user was on: ",a)
```

```
print("The first amount won by a user was on: ",b)
```

```
The first transaction made by a user was on: 2020-10-01 00:00:00
```

```
The first amount won by a user was on: 2020-12-01 00:00:00
```

```
# End Dates
```

```
a = df1['transaction_date'].astype('datetime64[ns]').max()
```

```
b = df1['reimburse_date'].astype('datetime64[ns]').max()
```

```
print("The last transaction made by a user was on: ", a)
```

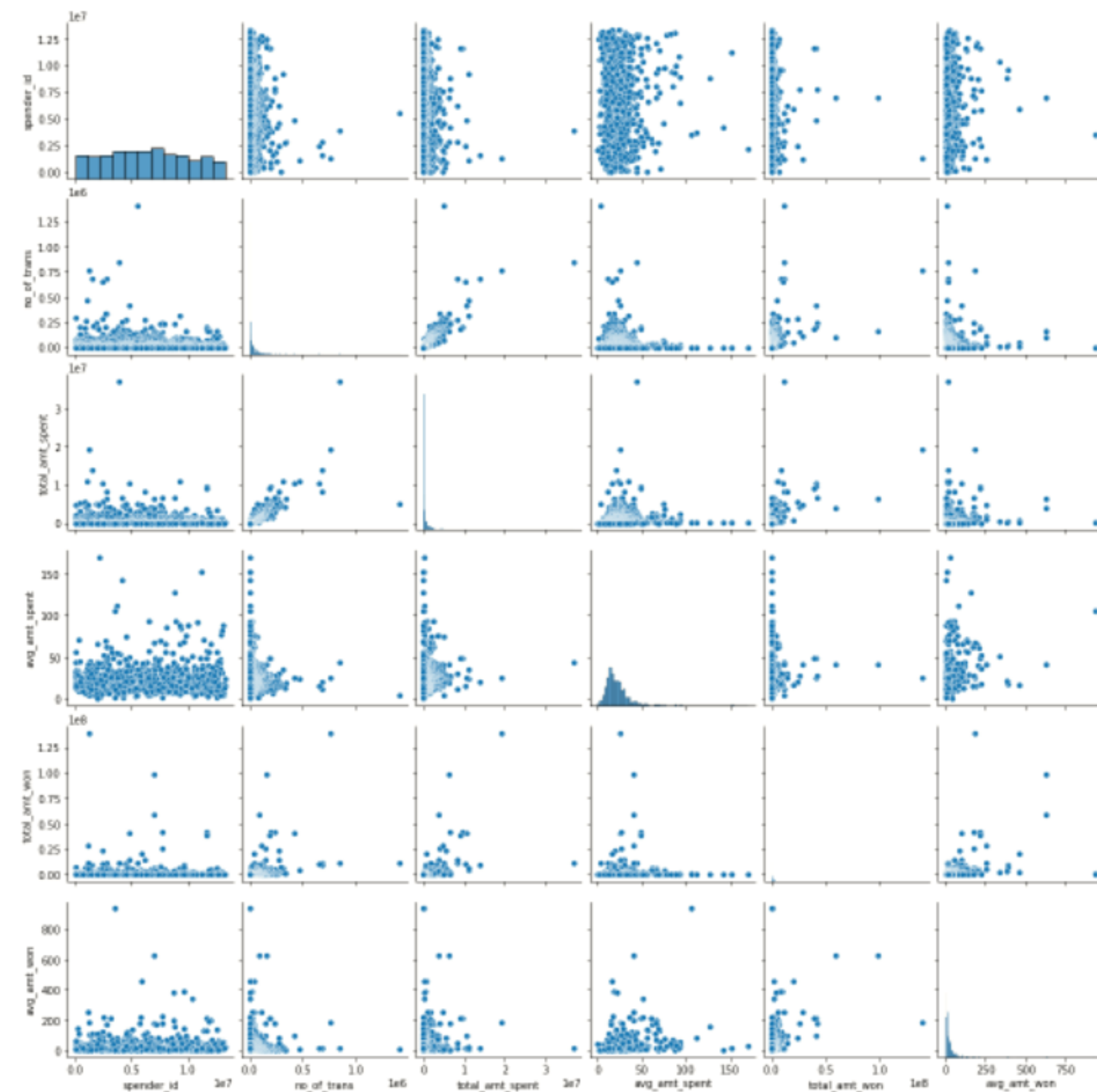
```
print("The last amount won by a user was on: ",b)
```

```
The last transaction made by a user was on: 2021-03-29 00:00:00
```

```
The last amount won by a user was on: 2021-01-29 00:00:00
```

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x1d723b6b490>
```



```

x_values = df['avg_amt_spent']
y_values = df['avg_amt_won']

correlation_matrix = np.corrcoef(x_values, y_values)
correlation_xy = correlation_matrix[0,1]
r_squared = correlation_xy**2

print(r_squared)

```

0.08297438963409612

```
df.corr()
```

	spender_id	no_of_trans	total_amt_spent	avg_amt_spent	total_amt_won	avg_amt_won
spender_id	1.000000	-0.123766	-0.111098	0.037971	-0.046685	-0.021619
no_of_trans	-0.123766	1.000000	0.807182	-0.025623	0.441627	0.037325
total_amt_spent	-0.111098	0.807182	1.000000	0.110529	0.526067	0.108041
avg_amt_spent	0.037971	-0.025623	0.110529	1.000000	0.064357	0.288053
total_amt_won	-0.046685	0.441627	0.526067	0.064357	1.000000	0.421542
avg_amt_won	-0.021619	0.037325	0.108041	0.288053	0.421542	1.000000

```
df1.corr()
```

	spender_id	member_id	amount_spent	amount_won
spender_id	1.000000	1.000000	0.018613	-0.002711
member_id	1.000000	1.000000	0.006222	-0.002711
amount_spent	0.018613	0.006222	1.000000	0.061337
amount_won	-0.002711	-0.002711	0.061337	1.000000

```
df2 = df1.groupby(['spender_id']).sum()
df2
```

	member_id	amount_spent	amount_won
spender_id			
155	0.0	23.29	0.00
177	0.0	143.80	0.00
355	0.0	1523.14	0.00
525	0.0	264.52	0.00
7815	46890.0	91.46	50.34
...
14074325	0.0	11.47	0.00
14107525	0.0	199.40	0.00
14113377	0.0	53.62	0.00
14122355	0.0	140.34	0.00
14141977	0.0	21.37	0.00

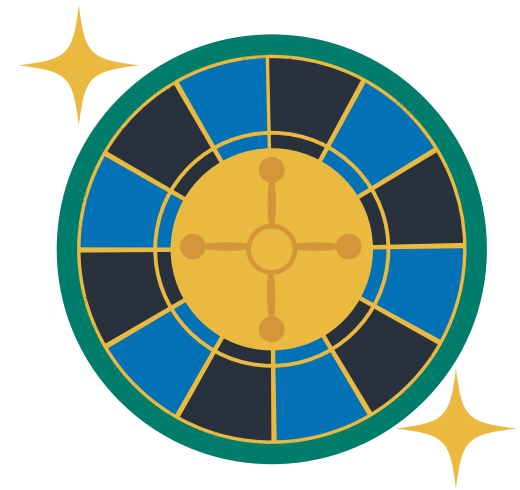
5226 rows × 3 columns

```
n_winners = df['spender_id'].nunique()
n_winners
```

842

```
n_losers = df2.loc[df2['amount_won'] == 0, 'amount_spent'].nunique()
n_losers
```

4113



Thank You!

