Provide Insights to the Product Strategy Team in the Banking Domain

Problem Statement

Mitron Bank is a legacy financial institution headquartered in Hyderabad. They want to introduce a new line of credit cards, aiming to broaden its product offerings and reach in the financial market.
AtliQ Data Services came to know about this through an internal link and approached Mitron Bank with a proposal to implement this project. However, strategy director of Mitron Bank, Mr.Bashnir Rover is skeptical and asked them to do a pilot project with the sample data before handing them the full project. They provided a sample dataset of 4000 customers across five cities on their online spend and other details.

Peter Pandey is a data analyst at AtliQ Data Services and asked by his manager to take over this project. His role is to analyse the provided sample data and report key findings to the strategy team of Mitron Bank. This analysis is expected to guide them in tailoring the credit cards to customer needs and market trends.

The successful acquisition of this project depends on Peter's ability to provide actionable, data-driven recommendations and impress Mr. Bashnir Rover & his team. Peter requested support from his manager Tony Sharma, and he provided him with some ideas to generate insights based on the data provided.
Task: Imagine yourself as Peter Pandey and perform the following task:

1. Use “Insight Ideas from Tony.pdf”. Create metrics and visuals accordingly.
2. Design a dashboard with your metrics and analysis. The end users of this dashboard are top-level management and product strategy team - hence the dashboard should be self-explanatory and easy to understand.
   codebasics.io
3. Present your insights to Mr.Bashnir Rover & team. Be creative and concise with your presentation. Use your dashboard in the presentation along with the deck.
4. Use additional data based on your own research to support your recommendations.
   Note:
5. We recommend you create a video presentation of ideally 15 minutes or less for the business stakeholders. Additionally, make a LinkedIn post that includes relevant links, your video presentation, and a reflection on your experience while working on this challenge.
6. You can check out this example presentation to gain some inspiration:
   Sample Presentation Link

7. Submit your post link on the resume project challenge page of codebasics. (https://codebasics.io/challenge/codebasics-resume-project-challenge)

Insight Ideas from Tony

Demographic classification:
Classify the customers based on available demography such as age group, gender, occupation etc. and provide insights based on them.

Avg income utilisation: Find the average income utilisation % of customers (avg_spends/avg_income). This will be your key metric. The higher the average income utilisation %, the more is their likelihood to use credit cards.

Spending Insights: Where do people spend money the most? Does it have any impact due to occupation, gender, city, age etc.? This can help you to add relevant credit card features for specific target groups.

Key Customer Segments: By doing above, you should be able to identify and profile key customer segments that are likely to be the highest-value users of the new credit cards. This includes understanding their demographics, spending behaviours, and financial preferences.

Credit Card Feature Recommendations: Provide recommendations on what key features should be included in the credit card which will improve the likelihood of credit card usage. This should be backed by the insights from data provided and also some secondary research on the internet for this.

Additional Thoughts: I added above insights based on my initial thoughts. However, you may get more valuable insights when you delve deep into the data.

Note: If you find any discrepancies in the data, include that in your presentation as a consideration.

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

```
customer_data = pd.read_csv('https://raw.githubusercontent.com/Yash-Kavaiya/codebasics/
```

- `customer_id`: The unique identifier for each customer.
- `age_group`: The age group of the customer (e.g., 25-34, 35-45, etc.).
- `city`: The city where the customer lives.
- `occupation`: The customer's occupation.
- `gender`: The customer's gender.

- `marital status`: The customer's marital status.
- `avg_income`: The customer's average income.

```
customer_data
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523
1	ATQCUS0809	25-34	Hyderabad	Salaried Other Employees	Male	Married	39922
2	ATQCUS0663	25-34	Chennai	Salaried Other Employees	Male	Married	37702
3	ATQCUS0452	25-34	Delhi NCR	Government Employees	Male	Married	54090
4	ATQCUS3350	21-24	Bengaluru	Freelancers	Male	Single	28376
...	...	...	...	...	...	...	...
3995	ATQCUS3035	45+	Delhi NCR	Business Owners	Female	Married	72805
3996	ATQCUS2585	35-45	Mumbai	Salaried Other Employees	Female	Married	41343
3997	ATQCUS1229	35-45	Bengaluru	Salaried IT Employees	Male	Married	65948
3998	ATQCUS0581	25-34	Bengaluru	Government Employees	Male	Married	52589
3999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541

4000 rows × 7 columns

```
customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   customer_id     4000 non-null   object
 1   age_group       4000 non-null   object
 2   city            4000 non-null   object
 3   occupation      4000 non-null   object
 4   gender          4000 non-null   object
 5   marital status  4000 non-null   object
 6   avg_income      4000 non-null   int64 
dtypes: int64(1), object(6)
memory usage: 218.9+ KB
```

```
for column in customer_data.columns:
    print(f"{column}: {customer_data[column].unique()}")
```

```
customer_id: ['ATQCUS1825' 'ATQCUS0809' 'ATQCUS0663' ... 'ATQCUS1229' 'ATQCUS0581'
 'ATQCUS3477']
age_group: ['45+' '25-34' '21-24' '35-45']
city: ['Bengaluru' 'Hyderabad' 'Chennai' 'Delhi NCR' 'Mumbai']
```

```
occupation: ['Salaried IT Employees' 'Salaried Other Employees' 'Government Employees'
 'Freelancers' 'Business Owners']
gender: ['Male' 'Female']
marital status: ['Married' 'Single']
avg_income: [73523 39922 37702 ... 65948 52589 73541]
```

```
customer_data.age_group.value_counts()
```

```
25-34    1498
35-45    1273
21-24     691
45+       538
Name: age_group, dtype: int64
```

```
!pip install Plotly
```

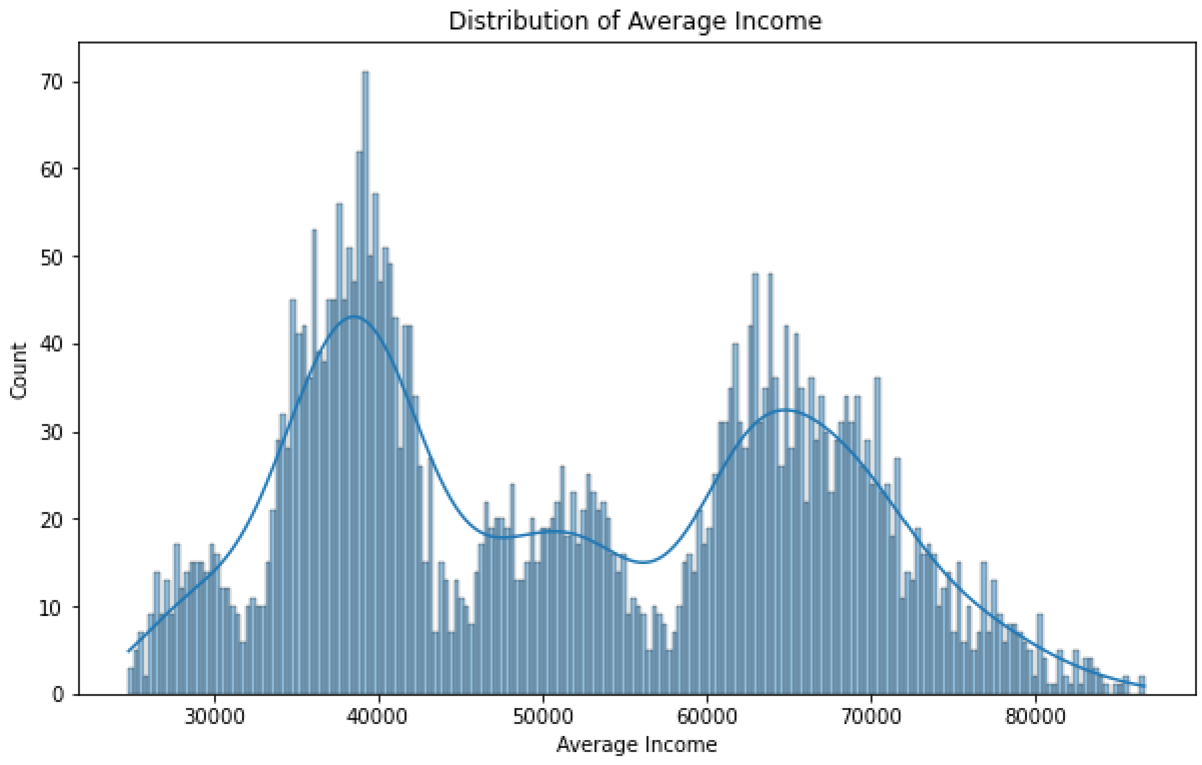
```
Requirement already satisfied: Plotly in /opt/conda/lib/python3.9/site-packages
(5.18.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.9/site-
packages (from Plotly) (8.2.3)
Requirement already satisfied: packaging in /opt/conda/lib/python3.9/site-packages
(from Plotly) (21.2)
Requirement already satisfied: pyparsing<3,>=2.0.2 in /opt/conda/lib/python3.9/site-
packages (from packaging->Plotly) (2.4.7)
```

```
import plotly.express as px
fig = px.pie(customer_data['age_group'].value_counts(), values=customer_data['age_group
fig.update_traces(hovertemplate='Age Group: %{label}<br>Count: %{value}<br>Percentage:
fig.show()
```

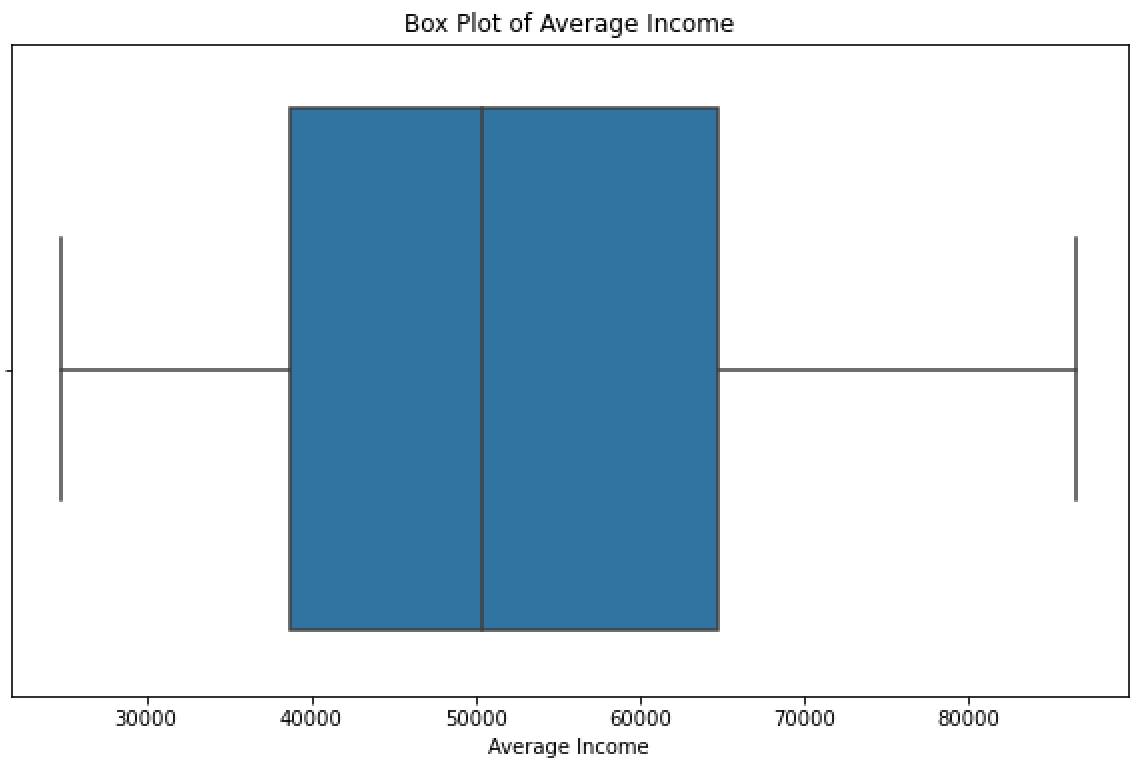
```
fig = px.pie(customer_data['city'].value_counts(), values=customer_data['city'].value_c
fig.update_traces(hovertemplate='city : %{label}<br>Count: %{value}<br>Percentage: %{pe
fig.show()
```

```
fig = px.pie(customer_data['occupation'].value_counts(), values=customer_data['occupati
fig.update_traces(hovertemplate='occupation : %{label}<br>Count: %{value}<br>Percentage
fig.show()
```

```
plt.figure(figsize=(10,6))
sns.histplot(customer_data['avg_income'], bins=200, kde=True)
plt.title('Distribution of Average Income')
plt.xlabel('Average Income')
plt.ylabel('Count')
plt.show()
```



```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a box plot
plt.figure(figsize=(10,6))
sns.boxplot(x=customer_data['avg_income'])
plt.title('Box Plot of Average Income')
plt.xlabel('Average Income')
plt.show()
```



```

gender_counts = customer_data['gender'].value_counts()
fig = px.bar(gender_counts, x=gender_counts.index, y=gender_counts.values, labels={'x':
fig.show()
```

```
marital_status_counts = customer_data['marital status'].value_counts()
fig = px.bar(marital_status_counts, x=marital_status_counts.index, y=marital_status_cou
fig.show()
```

```
fact_spends = pd.read_csv('https://raw.githubusercontent.com/Yash-Kavaiya/codebasics/ma
```

There are five columns in the dataset:

- customer_id: The unique identifier for each customer.
- month: The month in which the transaction occurred.
- category: The category of the spending (e.g., groceries, health & wellness, electronics).
- payment_type: The method used for payment (e.g., credit card, debit card, UPI).
- spend: The amount spent in the transaction.

```
fact_spends
```

	customer_id	month	category	payment_type	spend
0	ATQCUS1371	July	Health & Wellness	Credit Card	1114
1	ATQCUS0368	October	Groceries	Credit Card	1466

	customer_id	month	category	payment_type	spend
2	ATQCUS0595	May	Health & Wellness	Credit Card	387
3	ATQCUS0667	October	Electronics	Credit Card	1137
4	ATQCUS3477	September	Bills	UPI	2102
...	...	...	...	...	...
863995	ATQCUS1993	June	Bills	Debit Card	897
863996	ATQCUS1063	September	Bills	Credit Card	2680
863997	ATQCUS0416	August	Others	Credit Card	270
863998	ATQCUS3361	September	Bills	UPI	446
863999	ATQCUS1736	September	Apparel	UPI	242

864000 rows × 5 columns

```
fact_spends.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864000 entries, 0 to 863999
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   customer_id   864000 non-null  object
 1   month         864000 non-null  object
 2   category      864000 non-null  object
 3   payment_type  864000 non-null  object
 4   spend         864000 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 33.0+ MB
```

```
fig = px.pie(fact_spends['category'].value_counts(), values=fact_spends['category'].val
fig.update_traces(hovertemplate='category : %{label}<br>Count: %{value}<br>Percentage:
fig.show()
```

```
fig = px.pie(fact_spends['payment_type'].value_counts(), values=fact_spends['payment_ty
fig.update_traces(hovertemplate='payment_type : %{label}<br>Count: %{value}<br>Percenta
fig.show()
```

```
fact_spends.customer_id.value_counts()
```

```
ATQCUS1371     216
ATQCUS2876     216
ATQCUS0139     216
ATQCUS3635     216
ATQCUS3171     216
              ...
ATQCUS1464     216
```

```
ATQCUS3980    216
ATQCUS3340    216
ATQCUS2075    216
ATQCUS0890    216
Name: customer_id, Length: 4000, dtype: int64
```

```
df = pd.merge(customer_data, fact_spends, on='customer_id')
```

```
df
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income	month	category	paym
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	May	Electronics	Net
1	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	May	Groceries	D
2	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	June	Bills	Cr
3	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	September	Apparel	D
4	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	May	Food	D
...	...	...	...	...	...	...	...	...	...	
863995	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	May	Bills	Net
863996	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	October	Apparel	
863997	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	September	Food	D
863998	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	June	Apparel	Net
863999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	September	Health & Wellness	Cr

864000 rows × 11 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 864000 entries, 0 to 863999
Data columns (total 11 columns):
 #   Column          Non-Null Count   Dtype 
---  ------          --------------   ----- 
 0   customer_id     864000 non-null  object
 1   age_group       864000 non-null  object
 2   city            864000 non-null  object
 3   occupation      864000 non-null  object
 4   gender          864000 non-null  object
 5   marital status  864000 non-null  object
 6   avg_income      864000 non-null  int64
 7   month           864000 non-null  object
 8   category        864000 non-null  object
 9   payment_type    864000 non-null  object
 10  spend           864000 non-null  int64
dtypes: int64(2), object(9)
memory usage: 79.1+ MB
```

```
spending_insights = df.groupby(['customer_id'])['spend'].mean().reset_index()
spending_insights.columns = ['customer_id', 'avg_spends']
```

```
spending_insights
```

	customer_id	avg_spends
0	ATQCUS0001	789.560185
1	ATQCUS0002	780.157407
2	ATQCUS0003	729.324074
3	ATQCUS0004	753.032407
4	ATQCUS0005	728.222222
...	...	...
3995	ATQCUS3996	461.060185
3996	ATQCUS3997	260.481481
3997	ATQCUS3998	276.666667
3998	ATQCUS3999	310.842593
3999	ATQCUS4000	247.912037

4000 rows × 2 columns

```
customer_data
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523
1	ATQCUS0809	25-34	Hyderabad	Salaried Other Employees	Male	Married	39922
2	ATQCUS0663	25-34	Chennai	Salaried Other Employees	Male	Married	37702
3	ATQCUS0452	25-34	Delhi NCR	Government Employees	Male	Married	54090
4	ATQCUS3350	21-24	Bengaluru	Freelancers	Male	Single	28376
...	...	...	...	...	...	...	...
3995	ATQCUS3035	45+	Delhi NCR	Business Owners	Female	Married	72805
3996	ATQCUS2585	35-45	Mumbai	Salaried Other Employees	Female	Married	41343
3997	ATQCUS1229	35-45	Bengaluru	Salaried IT Employees	Male	Married	65948
3998	ATQCUS0581	25-34	Bengaluru	Government Employees	Male	Married	52589

	customer_id	age_group	city	occupation	gender	marital status	avg_income
3999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541

4000 rows × 7 columns

```
new_df = pd.merge(customer_data, spending_insights, on='customer_id')
```

```
new_df
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income	avg_spends
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519
1	ATQCUS0809	25-34	Hyderabad	Salaried Other Employees	Male	Married	39922	458.851852
2	ATQCUS0663	25-34	Chennai	Salaried Other Employees	Male	Married	37702	319.916667
3	ATQCUS0452	25-34	Delhi NCR	Government Employees	Male	Married	54090	566.245370
4	ATQCUS3350	21-24	Bengaluru	Freelancers	Male	Single	28376	340.305556
...	...	...	...	...	...	...	...	...
3995	ATQCUS3035	45+	Delhi NCR	Business Owners	Female	Married	72805	545.800926
3996	ATQCUS2585	35-45	Mumbai	Salaried Other Employees	Female	Married	41343	582.277778
3997	ATQCUS1229	35-45	Bengaluru	Salaried IT Employees	Male	Married	65948	1144.861111
3998	ATQCUS0581	25-34	Bengaluru	Government Employees	Male	Married	52589	497.009259
3999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852

4000 rows × 8 columns

```
new_df['income_utilization'] = new_df['avg_spends']*100 / new_df['avg_income']
```

```
new_df.describe()
```

	avg_income	avg_spends	income_utilization
count	4000.000000	4000.000000	4000.000000
mean	51657.032250	614.464994	1.193001
std	14690.140645	254.574848	0.342039
min	24816.000000	163.263889	0.396965
25%	38701.000000	420.989583	0.933414
50%	50422.000000	557.372685	1.163884
75%	64773.250000	755.150463	1.434777
max	86600.000000	1459.263889	2.149305

```
df = pd.merge(new_df, fact_spends, on='customer_id')
```

```
df
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income	avg_spends	income_utilizatio
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
1	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
2	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
3	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
4	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
...	...	...	...	...	...	...	...	...	.
863995	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863996	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863997	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863998	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233

864000 rows × 13 columns

Top 500 Customer

```
grouped_df = df.groupby('customer_id').agg({'spend': 'sum', 'avg_income': 'mean'})
top_spend_customers = grouped_df['spend'].nlargest(4000)
top_income_customers = grouped_df['avg_income'].nlargest(4000)
top_spend_customers_df = top_spend_customers.to_frame().reset_index()
top_income_customers_df = top_income_customers.to_frame().reset_index()
```

```
top_spend_customers_df
```

	customer_id	spend
0	ATQCUS0918	315201
1	ATQCUS0914	309425
2	ATQCUS0922	306975
3	ATQCUS0944	304288
4	ATQCUS0943	300422
...	...	...
3995	ATQCUS3887	40570
3996	ATQCUS3885	40500

	customer_id	spend
3997	ATQCUS3883	37650
3998	ATQCUS2066	36410
3999	ATQCUS2067	35265

4000 rows × 2 columns

```
top_income_customers_df
```

	customer_id	avg_income
0	ATQCUS2990	86600.0
1	ATQCUS1982	86327.0
2	ATQCUS1985	85593.0
3	ATQCUS1694	85416.0
4	ATQCUS1864	85082.0
...	...	...
3995	ATQCUS3224	25289.0
3996	ATQCUS0102	25159.0
3997	ATQCUS2053	24995.0
3998	ATQCUS3345	24888.0
3999	ATQCUS3392	24816.0

4000 rows × 2 columns

Demographic classification:
Classify the customers based on available
demography such as age group, gender, occupation etc. and provide insights
based on them.

```
import plotly.graph_objects as go
```

```

fig1 = go.Figure(data=go.Bar(x=grouped_df.index, y=grouped_df['spend'], name='Total Spe
fig1.update_layout(title='Total Spend by Occupation', xaxis_title='Occupation', yaxis_t
fig1.show()

```

```
# Create a bar chart for average income
fig2 = go.Figure(data=go.Bar(x=grouped_df.index, y=grouped_df['avg_income'], name='Aver
fig2.update_layout(title='Average Income by Occupation', xaxis_title='Occupation', yaxi
fig2.show()
```

```
grouped_df = df.groupby(['occupation', 'gender', 'marital status'])['spend'].sum().rese

# Create a sunburst chart
```

```
fig = px.sunburst(grouped_df, path=['occupation', 'gender', 'marital status'], values='
fig.show()
```

```
# Classify the customers based on age group, gender, and occupation
grouped_df = df.groupby(['age_group', 'gender', 'occupation']).size().reset_index(name=
```

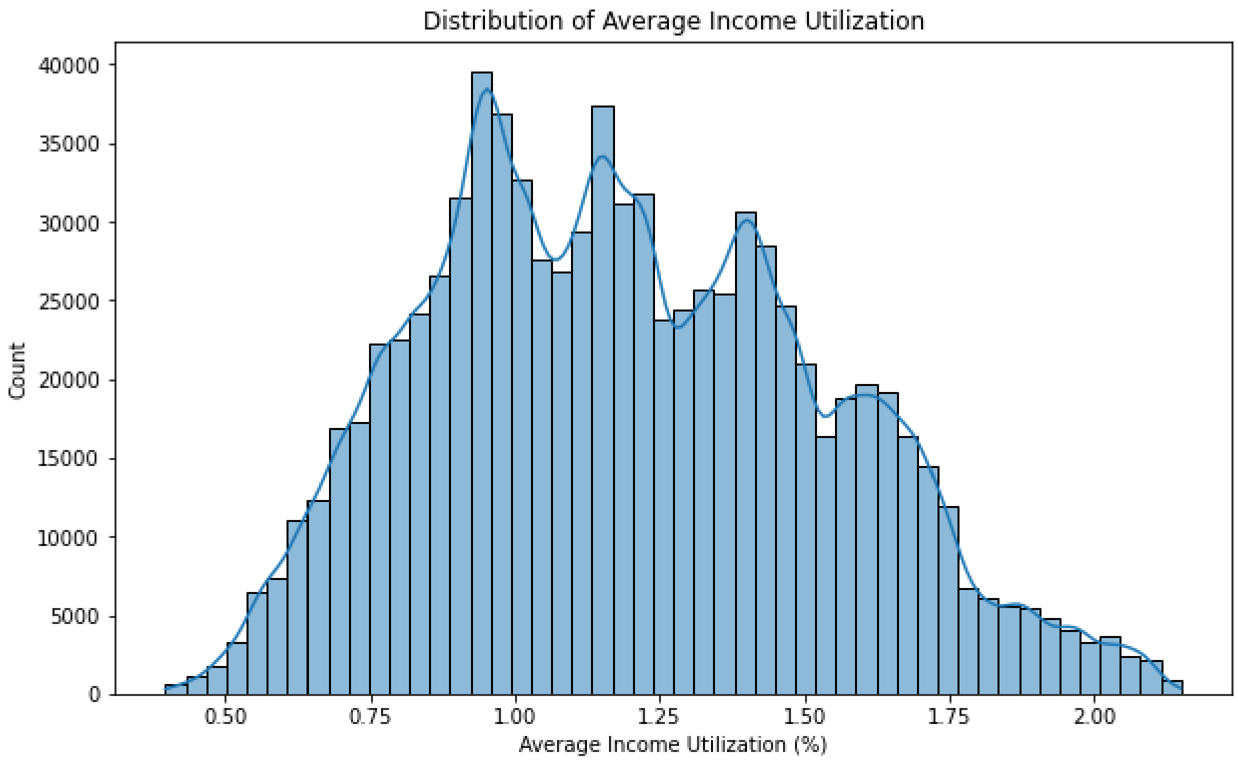
```
# Calculate the mean average income and total spend for each demographic group
grouped_df = df.groupby(['age_group', 'gender', 'occupation']).agg({'avg_income': 'mear
```

```
fig1 = px.bar(grouped_df, x='occupation', y='avg_income', color='age_group', barmode='g
fig1.show()
```

```
# Create a bar chart for total spend
fig2 = px.bar(grouped_df, x='occupation', y='spend', color='age_group', barmode='group'
fig2.show()
```

Avg income utilisation: Find the average income utilisation % of customers (avg_spends/avg_income). This will be your key metric. The higher the average income utilisation %, the more is their likelihood to use credit cards.

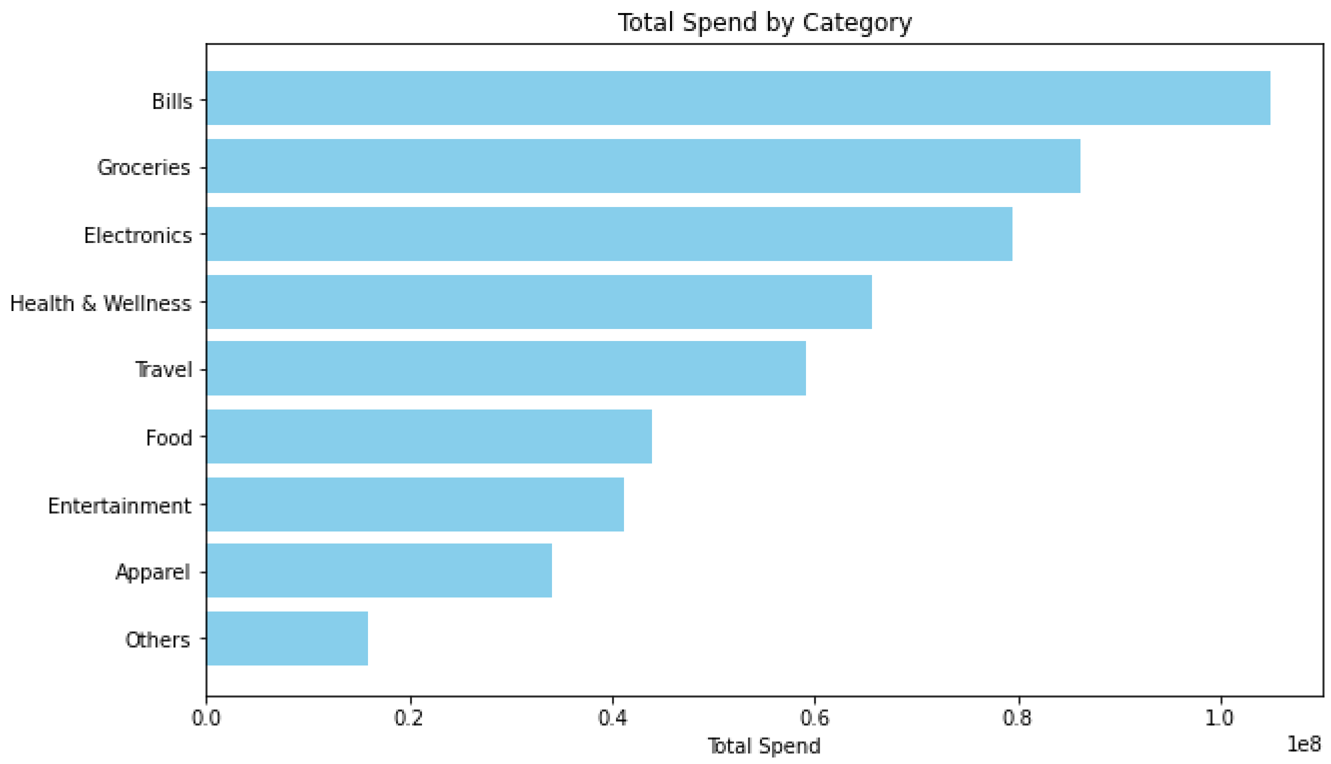
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a histogram of income utilization
plt.figure(figsize=(10,6))
sns.histplot(df['income_utilization'], bins=50, kde=True)
plt.title('Distribution of Average Income Utilization')
plt.xlabel('Average Income Utilization (%)')
plt.ylabel('Count')
plt.show()
```



Spending Insights: Where do people spend money the most? Does it have any impact due to occupation, gender, city, age etc.? This can help you to add relevant credit card features for specific target groups.

```
# Group by category and calculate the total spend for each category
total_spend_by_category = df.groupby('category')['spend'].sum()
```

```
import matplotlib.pyplot as plt

# Data
categories = ['Apparel', 'Bills', 'Electronics', 'Entertainment', 'Food', 'Groceries',
spend_values = [34036001, 104912768, 79562220, 41289162, 44013470, 86303761, 65599867,

# Sort the data in descending order
sorted_data = sorted(zip(spend_values, categories), reverse=True)
spend_values, categories = zip(*sorted_data)

# Create a horizontal bar chart
plt.figure(figsize=(10, 6))
plt.barh(categories, spend_values, color='skyblue')
plt.xlabel('Total Spend')
plt.title('Total Spend by Category')
plt.gca().invert_yaxis()  # Invert y-axis to show categories in descending order
plt.show()
```



```
# Group by occupation, gender, city, age_group, and category, and calculate the total s
total_spend_by_group = df.groupby(['occupation', 'gender', 'city', 'age_group', 'catego

# Print the DataFrame
print(total_spend_by_group)
```

```
occupation                 gender  city       age_group  category
Business Owners            Female  Bengaluru  21-24      Apparel               67515
                                                         Bills                  9021
                                                         Electronics           29876
                                                         Entertainment         38900
                                                         Food                  39123
                                                                                 ...
Salaried Other Employees   Male    Mumbai     45+        Food                 145681
                                                         Groceries            394299
                                                         Health & Wellness    222111
                                                         Others                58136
                                                         Travel               263040
Name: spend, Length: 1800, dtype: int64
```

```
# Create a bar chart for total spend by category
fig1 = px.bar(total_spend_by_category, x=total_spend_by_category.index, y=total_spend_b
fig1.show()
```

```python
# Create a bar chart for total spend by group
fig2 = px.bar(total_spend_by_group.reset_index(), x='category', y='spend', color='occup
fig2.show()
```

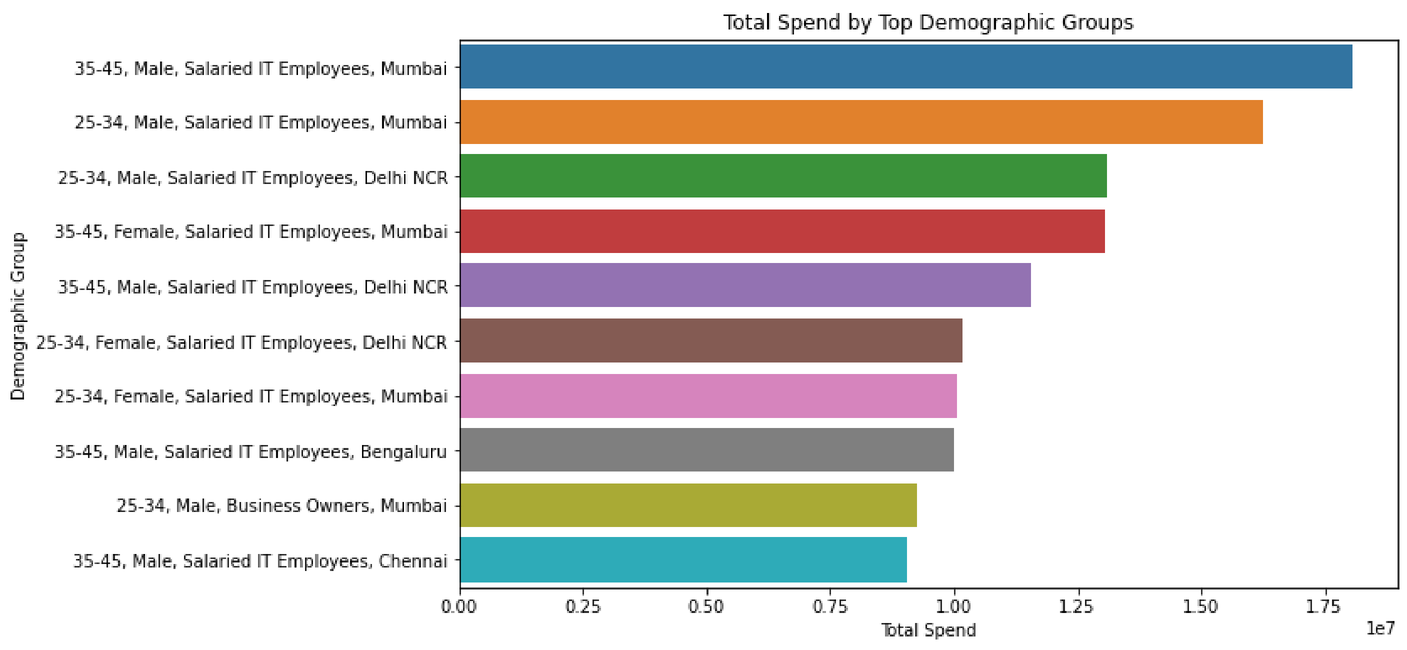
Key Customer Segments: By doing above, you should be able to identify and profile key customer segments that are likely to be the highest-value users of the new credit cards. This includes understanding their demographics, spending behaviours, and financial preferences.

```
grouped = df.groupby(['age_group', 'gender', 'occupation', 'city'])

# Calculate statistics for each group
grouped_stats = grouped.agg({
    'spend': ['mean', 'sum', 'count'],
    'avg_income': ['mean', 'sum'],
    'income_utilization': ['mean']
})

# Sort the groups by total spend
grouped_stats.sort_values(('spend', 'sum'), ascending=False, inplace=True)

# Print the grouped statistics
print(grouped_stats)
```

```
                                                          spend            \
                                                           mean       sum   
age_group gender occupation            city                                  
35-45     Male   Salaried IT Employees Mumbai      1307.337384  18072632   
25-34     Male   Salaried IT Employees Mumbai      1195.502792  16268402   
                                       Delhi NCR   1082.226935  13090617   
35-45     Female Salaried IT Employees Mumbai      1097.948906  13043633   
          Male   Salaried IT Employees Delhi NCR   1188.705864  11554221   
...                                                        ...       ...   
21-24     Female Freelancers           Chennai      198.734568    128780   
45+       Female Freelancers           Hyderabad    279.627315    120799   
21-24     Female Business Owners       Chennai      272.976852    117926   
45+       Female Freelancers           Chennai      255.435185    110348   
                 Government Employees  Chennai      250.444444    108192   

                                                         avg_income  \
                                                   count         mean   
age_group gender occupation            city                              
35-45     Male   Salaried IT Employees Mumbai      13824  65851.343750   
```

```
25-34     Male   Salaried IT Employees Mumbai     13608  62115.777778
                                       Delhi NCR  12096  62329.946429
35-45     Female Salaried IT Employees Mumbai     11880  66004.690909
          Male   Salaried IT Employees Delhi NCR   9720  66357.755556
...                                                 ...           ...
21-24     Female Freelancers           Chennai      648  27117.333333
45+       Female Freelancers           Hyderabad    432  38384.000000
21-24     Female Business Owners       Chennai      432  53405.000000
45+       Female Freelancers           Chennai      432  40656.500000
                 Government Employees  Chennai      432  58801.500000

                                                         income_utilization
                                                     sum               mean
age_group gender occupation            city
35-45     Male   Salaried IT Employees Mumbai     910328976          1.985270
25-34     Male   Salaried IT Employees Mumbai     845271504          1.925157
                                       Delhi NCR  753943032          1.736347
35-45     Female Salaried IT Employees Mumbai     784135728          1.663736
          Male   Salaried IT Employees Delhi NCR  644997384          1.791401
...                                                     ...               ...
21-24     Female Freelancers           Chennai     17572032          0.733993
45+       Female Freelancers           Hyderabad   16581888          0.728602
21-24     Female Business Owners       Chennai     23070960          0.510851
45+       Female Freelancers           Chennai     17563608          0.628042
                 Government Employees  Chennai     25402248          0.424423

[200 rows x 6 columns]
```

```
# Reset the index to make 'age_group', 'gender', 'occupation', and 'city' into columns
grouped_stats.reset_index(inplace=True)

# Create a new column for the demographic group
grouped_stats['demographic_group'] = grouped_stats['age_group'] + ', ' + grouped_stats[

# Select the top 10 groups by total spend
top_groups = grouped_stats.nlargest(10, ('spend', 'sum'))

# Plot a bar graph of total spend for the top groups
plt.figure(figsize=(10, 6))
sns.barplot(x=('spend', 'sum'), y='demographic_group', data=top_groups)
plt.title('Total Spend by Top Demographic Groups')
plt.xlabel('Total Spend')
plt.ylabel('Demographic Group')
plt.show()
```



```
df
```

	customer_id	age_group	city	occupation	gender	marital status	avg_income	avg_spends	income_utilizatio
0	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
1	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
2	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
3	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
4	ATQCUS1825	45+	Bengaluru	Salaried IT Employees	Male	Married	73523	900.018519	1.22413
...	...	...	...	...	...	...	...	...	.
863995	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863996	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863997	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863998	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233
863999	ATQCUS3477	25-34	Mumbai	Business Owners	Male	Single	73541	876.851852	1.19233

864000 rows × 13 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 864000 entries, 0 to 863999
Data columns (total 13 columns):
 #   Column              Non-Null Count   Dtype
```

```
---  ------              --------------  -----
 0   customer_id         864000 non-null  object
 1   age_group           864000 non-null  object
 2   city                864000 non-null  object
 3   occupation          864000 non-null  object
 4   gender              864000 non-null  object
 5   marital status      864000 non-null  object
 6   avg_income          864000 non-null  int64
 7   avg_spends          864000 non-null  float64
 8   income_utilization  864000 non-null  float64
 9   month               864000 non-null  object
 10  category            864000 non-null  object
 11  payment_type        864000 non-null  object
 12  spend               864000 non-null  int64
dtypes: float64(2), int64(2), object(9)
memory usage: 92.3+ MB
```

```python
import pandas as pd

def apply_one_hot_encoding(df, columns_to_encode):
    """
    Function to apply one-hot encoding to specified columns in a DataFrame.

    Parameters:
    - df: DataFrame containing the columns to be encoded.
    - columns_to_encode: List of column names to be one-hot encoded.

    Returns:
    - DataFrame with one-hot encoded columns.
    """
    return pd.get_dummies(df, columns=columns_to_encode)

# Example usage:
# List of columns to be one-hot encoded
columns_for_encoding = ['age_group', 'city', 'occupation', 'gender', 'marital status',

# Apply one-hot encoding to the specified columns in your DataFrame (let's assume your
df_encoded = apply_one_hot_encoding(df, columns_for_encoding)
```

```python
df_encoded
```

	customer_id	avg_income	avg_spends	income_utilization	spend	age_group_21-24	age_group_25-34	age_group_
0	ATQCUS1825	73523	900.018519	1.224132	405	0	0	
1	ATQCUS1825	73523	900.018519	1.224132	1096	0	0	
2	ATQCUS1825	73523	900.018519	1.224132	2765	0	0	
3	ATQCUS1825	73523	900.018519	1.224132	363	0	0	

	customer_id	avg_income	avg_spends	income_utilization	spend	age_group_21-24	age_group_25-34	age_group_
4	ATQCUS1825	73523	900.018519	1.224132	334	0	0	
...	...	...	...	...	...	...	...	
863995	ATQCUS3477	73541	876.851852	1.192331	548	0	1	
863996	ATQCUS3477	73541	876.851852	1.192331	174	0	1	
863997	ATQCUS3477	73541	876.851852	1.192331	346	0	1	
863998	ATQCUS3477	73541	876.851852	1.192331	54	0	1	
863999	ATQCUS3477	73541	876.851852	1.192331	1155	0	1	

864000 rows × 42 columns

```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 864000 entries, 0 to 863999
Data columns (total 42 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   customer_id                          864000 non-null  object
 1   avg_income                           864000 non-null  int64
 2   avg_spends                           864000 non-null  float64
 3   income_utilization                   864000 non-null  float64
 4   spend                                864000 non-null  int64
 5   age_group_21-24                      864000 non-null  uint8
 6   age_group_25-34                      864000 non-null  uint8
 7   age_group_35-45                      864000 non-null  uint8
 8   age_group_45+                        864000 non-null  uint8
 9   city_Bengaluru                       864000 non-null  uint8
 10  city_Chennai                         864000 non-null  uint8
 11  city_Delhi NCR                       864000 non-null  uint8
 12  city_Hyderabad                       864000 non-null  uint8
 13  city_Mumbai                          864000 non-null  uint8
 14  occupation_Business Owners           864000 non-null  uint8
 15  occupation_Freelancers               864000 non-null  uint8
 16  occupation_Government Employees      864000 non-null  uint8
 17  occupation_Salaried IT Employees     864000 non-null  uint8
 18  occupation_Salaried Other Employees  864000 non-null  uint8
 19  gender_Female                        864000 non-null  uint8
 20  gender_Male                          864000 non-null  uint8
 21  marital status_Married               864000 non-null  uint8
 22  marital status_Single                864000 non-null  uint8
 23  month_August                         864000 non-null  uint8
 24  month_July                           864000 non-null  uint8
 25  month_June                      864000 non-null  uint8
 26  month_May                       864000 non-null  uint8
 27  month_October                   864000 non-null  uint8
 28  month_September                 864000 non-null  uint8
 29  category_Apparel                864000 non-null  uint8
 30  category_Bills                  864000 non-null  uint8
 31  category_Electronics            864000 non-null  uint8
 32  category_Entertainment          864000 non-null  uint8
 33  category_Food                   864000 non-null  uint8
 34  category_Groceries              864000 non-null  uint8
 35  category_Health & Wellness      864000 non-null  uint8
 36  category_Others                 864000 non-null  uint8
 37  category_Travel                 864000 non-null  uint8
 38  payment_type_Credit Card        864000 non-null  uint8
 39  payment_type_Debit Card         864000 non-null  uint8
 40  payment_type_Net Banking        864000 non-null  uint8
 41  payment_type_UPI                864000 non-null  uint8
dtypes: float64(2), int64(2), object(1), uint8(37)
memory usage: 70.0+ MB
```

```python
from sklearn.cluster import KMeans

# Assuming df_encoded is your DataFrame with the encoded columns
# Remove non-numeric columns like 'customer_id' before clustering
customer_id = df_encoded['customer_id']  # Save the customer_id column
df_numeric = df_encoded.drop(columns=['customer_id'])  # Remove non-numeric columns

# Apply K-means clustering with 5 clusters
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(df_numeric)

# Get the cluster labels
cluster_labels = kmeans.labels_

# Assign the cluster labels back to the DataFrame
df_encoded['Cluster'] = cluster_labels

# Add customer_id back to the DataFrame
df_encoded['customer_id'] = customer_id

# Display the counts of customers in each cluster
print(df_encoded['Cluster'].value_counts())
```

```python
def get_waitlist_days(customer_id):
    # Assuming 'df_encoded' is your DataFrame containing 'customer_id' and 'Cluster' co
    cluster = df_encoded[df_encoded['customer_id'] == customer_id]['Cluster'].values
    # Dictionary mapping clusters to waitlist days
    waitlist_mapping = {
        0: 1,
        1: 2,
        3: 6,
        4: 8
        # Add more clusters and corresponding waitlist days if needed
    }

    if len(cluster) > 0:
        cluster = int(cluster[0])
        if cluster in waitlist_mapping:
            return waitlist_mapping[cluster]
        else:
            return "Cluster exists but waitlist days not defined."
    else:
        return "Customer ID not found or does not belong to any cluster."

# Example usage:
user_input_customer_id = 'ATQCUS1234'  # Replace this with the user's input

waitlist_days = get_waitlist_days(user_input_customer_id)
if isinstance(waitlist_days, int):
    print(f"Cluster {df_encoded[df_encoded['customer_id'] == user_input_customer_id]['C
    # Customize your message based on the cluster and waitlist days
    if waitlist_days == 1:
        print("You are eligible for the credit card. You'll receive it within 1 day.")
    else:
        print(f"You are in the waitlist. Expected waitlist days: {waitlist_days}")
else:
    print(waitlist_days)  # Print message if customer ID not found or cluster not defir
```