Credit Card case study

BUSINESS PROBLEM: In order to effectively produce quality decisions in the modern credit card industry, knowledge must be gained through effective data analysis and modeling. Through the use of dynamic datadriven decision-making tools and procedures, information can be gathered to successfully evaluate all aspects of credit card operations. PSPD Bank has banking operations in more than 50 countries across the globe. Mr. Jim Watson, CEO, wants to evaluate areas of bankruptcy, fraud, and collections, respond to customer requests for help with proactive offers and service.

DATA AVAILABLE -

Customer Acquisition: At the time of card issuing, company maintains the details of customers. Spend (Transaction data): Credit card spend for each customer Repayment: Credit card Payment done by customer

In [298]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [48]:
```
cust_acqu=pd.read_csv('Customer Acqusition.csv')
spend=pd.read_csv('spend.csv')
repayment=pd.read_csv('Repayment.csv')
```

In [ ]:
```
cust_acqu.head()
#cust_acqu.info()
```

In [172]:
```
spend.rename(columns={'Sl No:':'Sl'}, inplace=True)
repayment.rename(columns={'SL No:':'Sl'}, inplace=True)
```

In [ ]:
```
spend.head()
spend.info()
```

In [ ]:
```
repayment.head()
repayment.info()
```

1. In the above dataset, a. In case age is less than 18, replace it with mean of age values.

In [13]:
```
cust_acqu[cust_acqu.Age<18]=np.mean(cust_acqu.Age)
```

In [12]:
```
np.mean(cust_acqu.Age)
```

Out[12]:
```
46.49
```

b. In case spend amount is more than the limit, replace it with 50% of that customer's limit. (customer's limit provided in acquisition table is the per transaction limit on his card)

In [174]:
```
cust1=pd.merge(left=spend, right=cust_acqu, left_on= spend.Customer, right_on=cust_acqu.Customer, how='left')
```

In [ ]:
```
cust1.Amount[cust1.Amount>cust1.Limit]=cust1.Limit/2
```

In [178]:
```
spend.Amount=np.where(cust1['Sl']==spend['Sl'],cust1.Amount,0)
```

c. Incase the repayment amount is more than the limit, replace the repayment with the limit.

In [200]:
```
cust2=pd.merge(left=repayment, right=cust_acqu, left_on= repayment.Customer, right_on=cust_acqu.Customer, how='left')
```

In [204]:
```
repayment.Amount=np.where(cust2['Sl']==repayment['Sl'],cust2.Amount,0)
```

2. From the above dataset create the following summaries:

a. How many distinct customers exist?

In [83]:
```
cust_acqu.Customer.unique().size
```

Out[83]:
```
100
```

b. How many distinct categories exist?

In [90]:
```
cust_acqu.Product.unique()
```

Out[90]:
```
array(['Gold', 'Silver', 'Platimum'], dtype=object)
```

In [92]:
```
spend.Type.unique().size
```

Out[92]:
```
15
```

c. What is the average monthly spend by customers?

In [205]:
```
spend.Month=pd.to_datetime(spend.Month)
```

In [114]:
```
spend.groupby(spend.Month.dt.month)['Amount'].mean().reset_index()
```

Out[114]:

	Month	Amount
0	1	256221.133553
1	2	233037.216471
2	3	246255.783723
3	4	236671.847533
4	5	241277.023476
5	6	241329.181250
6	7	268042.287143
7	8	236056.328571
8	9	219369.892143
9	10	215352.487857
10	11	249099.824405
11	12	208987.020357

d. What is the average monthly repayment by customers?

In [117]:
```
repayment.Month=pd.to_datetime(repayment.Month)
```

In [119]:
```
repayment.groupby(spend.Month.dt.month)['Amount'].mean().reset_index()
```

Out[119]:

	Month	Amount
0	1.0	244402.285043
1	2.0	258172.397269
2	3.0	254892.557143
3	4.0	247516.319667
4	5.0	243372.891143
5	6.0	244673.197500
6	7.0	224553.805714
7	8.0	243773.223333
8	9.0	223081.544048
9	10.0	264389.870714
10	11.0	242931.127857
11	12.0	265275.984286

In [ ]:
```
cust_acqu.Customer.unique()
```

e. If the monthly rate of interest is 2.9%, what is the profit for the bank for each month? (Profit is defined as interest earned on Monthly Profit. Monthly Profit = Monthly repayment – Monthly spend. Interest is earned only on positive profits and not on negative amounts)

In [131]:
```
profit_tab=pd.DataFrame(spend.groupby(spend.Month.dt.month)['Amount'].sum().reset_index()).sort_values(by='Month')
```

In [139]:
```
profit_tab['repayment']=repayment.groupby(spend.Month.dt.month)['Amount'].sum().reset_index().sort_values(by='Month').il
```

In [140]:
```
profit_tab['Profit']=profit_tab.repayment-profit_tab.Amount
```

In [ ]:
```
profit_tab.Profit[profit_tab.Profit<0]=0
```

In [148]:
```
profit_tab['interest']=profit_tab.Profit*.029
```

In [149]:
```
profit_tab
```

Out[149]:

	Month	Amount	repayment	Profit	interest
0	1	89421175.61	85296397.48	0.00	0.00000
1	2	55462857.52	61445030.55	5982173.03	173483.01787
2	3	56885086.04	58880180.70	1995094.66	57857.74514
3	4	35500777.13	37127447.95	1626670.82	47173.45378
4	5	50668174.93	51108307.14	440132.21	12763.83409
5	6	13514434.15	13701699.06	187264.91	5430.68239
6	7	11257776.06	9431259.84	0.00	0.00000
7	8	9914365.80	10238475.38	324109.58	9399.17782
8	9	9213535.47	9369424.85	155889.38	4520.79202
9	10	6029869.66	7402916.38	1373046.72	39818.35488
10	11	20924385.25	20406214.74	0.00	0.00000
11	12	5851636.57	7427727.56	1576090.99	45706.63871

f. What are the top 5 product types?

In [183]:
```
spend.groupby(spend.Type)['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head()
```

Out[183]:

	Type	Amount
10	PETRO	28597384.98
4	CAMERA	27690738.44
7	FOOD	20519243.60
0	AIR TICKET	20155847.12
14	TRAIN TICKET	19995825.72

g. Which city is having maximum spend?

In [191]:
```
cust1.groupby(cust1.City)['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(1)
```

Out[191]:

	City	Amount
4	COCHIN	45963513.5

h. Which age group is spending more money?

The age group above 25 spends the most money.

In [276]:
```
cust1.groupby(cust1.Age)['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(10)
```

Out[276]:

	Age	Amount
10	28	17365270.23
53	78	12329508.54
18	37	9992645.71
7	25	9749700.72
29	51	9670808.79
26	47	9157809.12
24	44	9002326.39
11	29	8231607.26
28	50	8082803.39
3	16	7935969.47

i. Who are the top 10 customers in terms of repayment?

In [214]:
```
cust2.groupby(cust2.Customer_x)['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(10)
```

Out[214]:

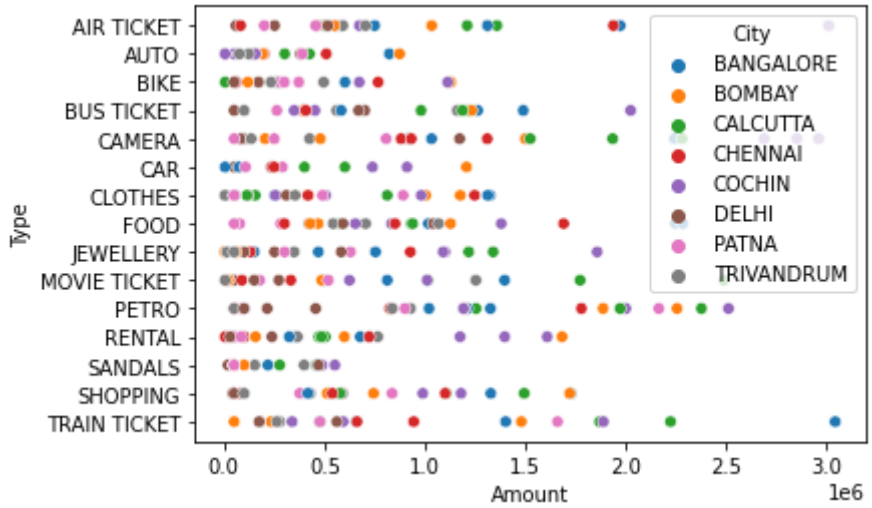
	Customer_x	Amount
58	A61	10539142.91
57	A60	9876290.74
5	A13	9572000.66
38	A43	8489871.46
40	A45	8448334.87
4	A12	8334760.16
6	A14	7943268.63
39	A44	7744730.12
33	A39	7622483.30
37	A42	7615460.86

3. Calculate the city wise spend on each product on yearly basis. Also include a graphical representation for the same.

In [335]:
```
plot4=cust1.groupby([cust1.Month.dt.year, cust1.Type, cust1.City])['Amount'].sum().reset_index()
```

In [337]:
```
sns.scatterplot(x=plot4.Amount, y=plot4.Type, hue=plot4.City)
```

Out[337]:
```
<AxesSubplot:xlabel='Amount', ylabel='Type'>
```



4. Create graphs for a. Monthly comparison of total spends, city wise

In [313]:
```
plot2|K)cust1.groupby([cust1.Month.dt.month,cust1.City])['Amount'].sum().reset_index()
```

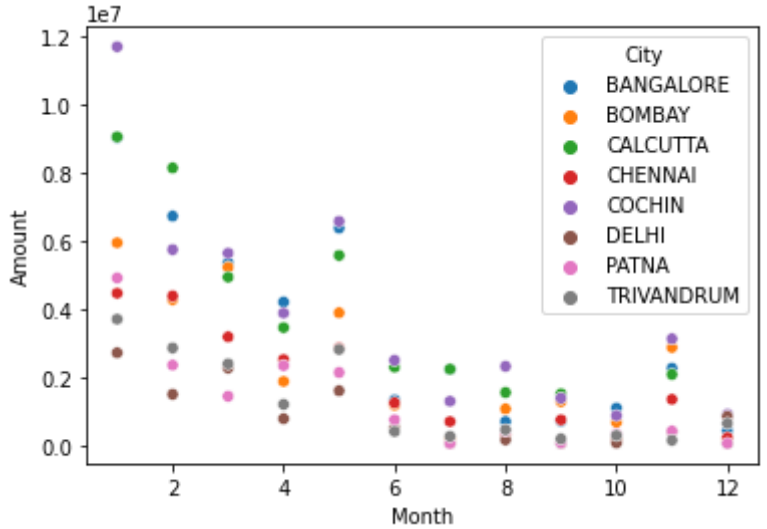
Out[313]:

	Month	City	Amount
0	1	BANGALORE	9041796.84
1	1	BOMBAY	5948993.33
2	1	CALCUTTA	9064864.90
3	1	CHENNAI	4466961.36
4	1	COCHIN	11714373.54
...	...	...	...
91	12	CHENNAI	223628.27
92	12	COCHIN	909533.05
93	12	DELHI	842342.02
94	12	PATNA	50000.00
95	12	TRIVANDRUM	640922.26

96 rows × 3 columns

In [314]:
```
sns.scatterplot(x=plot2.Month, y=plot2.Amount, hue=plot2.City)
```

Out[314]:
```
<AxesSubplot:xlabel='Month', ylabel='Amount'>
```



b. Comparison of yearly spend on air tickets

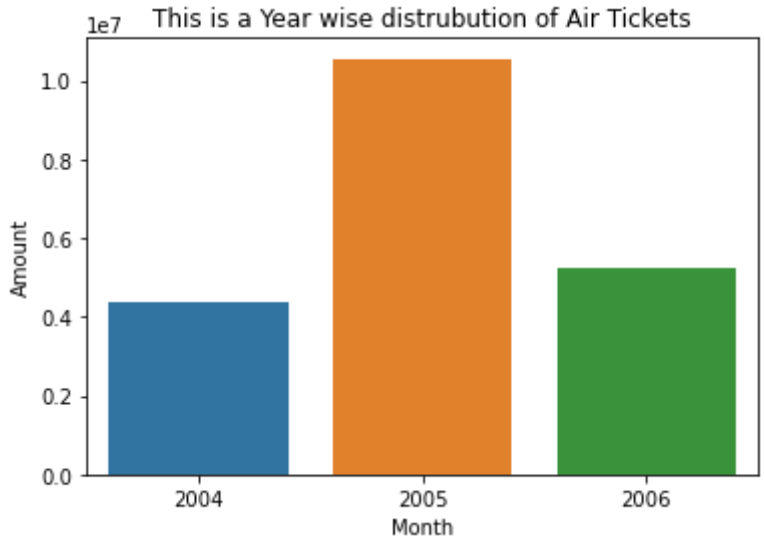
In [309]:
```
plot1=spend[spend.Type=='AIR TICKET'].groupby(spend.Month.dt.year)['Amount'].sum().reset_index()
plot1
```

Out[309]:

	Month	Amount
0	2004	4357225.35
1	2005	10550152.21
2	2006	5248469.56

In [299]:
```
plt.title('This is a Year wise distrubution of Air Tickets')
sns.barplot(x=plot1.Month, y=plot1.Amount)
```

Out[299]:
```
<AxesSubplot:title={'center':'This is a Year wise distrubution of Air Tickets'}, xlabel='Month', ylabel='Amount'>
```



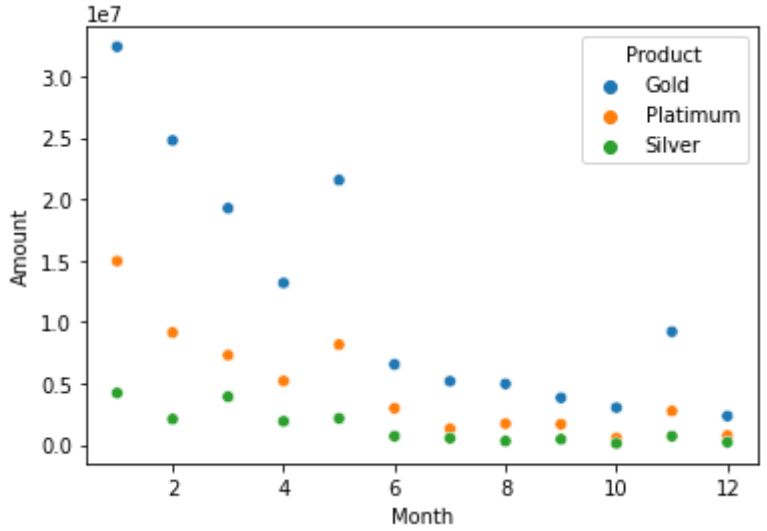
c. Comparison of monthly spend for each product (look for any seasonality that exists in terms of spend)

```
All customers, espcially the gold customers, spend more in the begning of the year and less and less towards the end
of the year.
```

In [331]:
```
plot3=cust1.groupby([cust1.Product,cust1.Month.dt.month])['Amount'].sum().reset_index()
```

In [330]:
```
sns.scatterplot(x=plot3.Month, y=plot3.Amount, hue=plot3.Product)
```

Out[330]: <AxesSubplot:xlabel='Month', ylabel='Amount'>



5. Write user defined PYTHON function to perform the following analysis: You need to find top 10 customers for each city in terms of their repayment amount by different products and by different time periods i.e. year or month. The user should be able to specify the product (Gold/Silver/Platinum) and time period (yearly or monthly) and the function should automatically take these inputs while identifying the top 10 customers.

In [261]:
```
def func1(prod,timep):
    if timep=='year':
        cust=cust2[cust2.Product==prod].groupby([cust2.Month.dt.year, cust2.City, cust2.Customer_x])['Amount'].sum().\
        reset_index().sort_values(by=['Month','City','Amount'], ascending=[True, True, False]).head(20)
    else:
        cust=cust2[cust2.Product==prod].groupby([cust2.Month.dt.month, cust2.City, cust2.Customer_x])['Amount'].sum().\
        reset_index().sort_values(by=['Month','City','Amount'], ascending=[True, True, False]).head(20)
    print('Top customers for product type '+prod+' and time period grouped by '+timep+ ' is as follows:')
    print(cust)
```

In [262]:
```
func1('Platimum','year')
```

```
Top customers for product type Platimum and time period grouped by year is as follows:
     Month       City Customer_x      Amount
1   2004.0  BANGALORE        A37   770552.62
3   2004.0  BANGALORE        A52   263853.18
2   2004.0  BANGALORE         A5    68809.07
0   2004.0  BANGALORE        A19    50000.00
5   2004.0     BOMBAY        A36   772335.72
7   2004.0     BOMBAY        A51   423029.79
6   2004.0     BOMBAY         A4    30003.00
4   2004.0     BOMBAY        A33    12275.98
8   2004.0     BOMBAY        A71    10002.00
11  2004.0   CALCUTTA        A40  2576916.73
12  2004.0   CALCUTTA        A49   100020.00
10  2004.0   CALCUTTA        A34    81344.52
9   2004.0   CALCUTTA        A20    70007.00
14  2004.0   CALCUTTA        A99    10002.00
13  2004.0   CALCUTTA        A98    10001.00
15  2004.0    CHENNAI        A38  1951311.01
16  2004.0    CHENNAI        A47    60000.00
17  2004.0    CHENNAI        A56    36537.90
21  2004.0     COCHIN        A41  1752953.17
23  2004.0     COCHIN        A54   490081.00
```