Data Analytics Project

Credit Card Data Analytics

In this project we have the data of customers and their orders we need to find some useful insights from the data like

1. We need to segment the customers in different-different categories according to their age.
    - Young age Females
    - Mid-age Females
    - Old age Females
    - Young age Males
    - Mid-age Males
    - Old age Males
2. We need to calculate the amount in terms of Product, State and Payment Method.
3. We need to calculate the highest five spending in all above categories.
4. We need to give opinion on return category like customers returning the products belongs to which state, age group, condition, category of the product or is it related to discount.
5. We need to create a profile of customers in terms of timing of their orders.
6. We want to know which payment method provides more discount for customers?
7. We must create a profile for high value items vs low value items and relate that with respect to their number of orders.
8. We need to find the answer if the merchant increases the discount price will it leads to increase in orders?

Explanation:

Firstly we need to download the Jupyter notebook and we also need to install python and in python we need to install some libraries like Pandas ,NumPy and Matplotlib.

Importing the libraries:

We need to import all the above libraries and read the dataset.

```
[175]: #importing the liberaries
       import pandas as pd
       import seaborn as sns
       import numpy as np
       import matplotlib as plt
```

```
[176]: #Reading the csv file
       data=pd.read_csv('Credit Banking - 3.csv')
```

Data Cleaning:

Orders: -

- Firstly, we need to do data cleaning. We need to find the **nan** values and replace them.
- We need to change some object data type like object to integer.
- We need to remove the **$** sign from the integer values.
- We need to check whether the return date is after the delivery date.
- We need to change the datatype of date from object to Date.
- We also need to check where the selling price is equal to Price.

Data Cleaning

```
[181]: data['Credit_card']=data['Credit_card'].replace([np.nan],0)
[182]: data['Credit_card']=data['Credit_card'].astype(int)
[183]: data['Price '] = data['Price '].replace('[$]', '', regex=True)
[184]: data['Price '] = data['Price '].replace('[,]', '', regex=True).astype(float)
[185]: data['Price ']=data['Price '].astype(int)
[186]: data['Selling_price'] = data['Selling_price'].replace('[$]', '', regex=True)
[187]: data['Selling_price'] = data['Selling_price'].replace('[,]', '', regex=True)
[188]: data['Selling_price'] = data['Selling_price'].replace('[()]', '', regex=True).astype(float)
[189]: data['Selling_price']=data['Selling_price'].astype(int)
[190]: data['Date']=pd.to_datetime(data['Date'],dayfirst=True)
[191]: data['Return_date']=data['Return_date'].fillna(0)
[192]: data['Return_date']=pd.to_datetime(data['Return_date'],dayfirst=True)
[193]: equal_selling_price_and_price= np.where(data['Price ']==data['Selling_price'])
[194]: print(equal_selling_price_and_price)
(array([  16,   27,   77,  127,  177,  178,  179,  180,  189,  190,  204,
         205,  206,  220,  221,  222,  411,  412,  413,  453,  471,  485,
         495,  496,  511,  512,  524,  525,  526,  559,  560,  561,  806,
         807,  808,  845,  846,  847,  882,  883,  884,  885,  886,  887,
         888,  931,  932,  933,  934,  935, 5372, 5373, 5374, 5375, 5376,
        5377, 5378, 7210, 7211, 7212, 7213, 7214, 8215, 8216, 8217, 8218,
        8219, 8220, 8221, 8771, 8772, 8773, 8774, 8775, 8776, 8777, 9075,
        9076, 9077, 9078, 9079, 9080, 9081, 9310, 9311, 9312, 9313, 9314,
        9584, 9585, 9586, 9587, 9588, 9993, 9994, 9995], dtype=int64),)
```

```
[195]: selected_rows = data.iloc[equal_selling_price_and_price]
[196]: selected_values = selected_rows['Selling_price']
[197]: discounted_values= (selected_values-(selected_values*5/100))
[198]: data['return_date_is_after_purchase date']=data['Return_date']>data['Date']
[199]: coupon_code_is_null= data.index[data['Coupon_ID'].isna()]
[200]: print(coupon_code_is_null)
Index([  5,  23,  37,  61,  85, 105, 134, 166, 201, 237, 272, 298, 323, 353,
       377, 399, 417, 418, 445, 505],
      dtype='int64')
[201]: data.loc[coupon_code_is_null, 'Selling_price'] = data.loc[coupon_code_is_null, 'Price ']
```

Customer Data:-

- We need to check whether the age of the customer is more than 18 or not .
- We need to check the customer id is unique for every customer.
- We need to change the selling price is equal to price where the coupon id is null.

[202]:
```
data2=pd.read_csv('Credit Banking - 3 second page.csv')
```

[203]:
```
print(data2)
```

```
     Credit_card                Email       Name  Mobile_number Gender  Age  \
0           3768    eddie@yahoomail.com      EDDIE     9045258449      M   83
1           4852       rose@hotmail.com       ROSE     8834789103      F   87
2           1174        amy@yahoomail.com        AMY     9557690013      F   31
3           4807     clarence@gmail.com   CLARENCE     9394398429      M   37
4           9131     johnny@hotmail.com     JOHNNY     9976623538      F   80
..           ...                    ...        ...           ...    ...  ...
193         4575     tina@yahoomail.com       TINA     8819353220      F   56
194         3958     gordon@hotmail.com     GORDON     8853153218      M   87
195         8609   jennifer@hotmail.com   JENNIFER     9935042642      F   93
196         4542   michelle@hotmail.com   MICHELLE     9730382145      F   60
197         9532  katherine@hotmail.com  KATHERINE     9033150890      M   18

           City       State Address
0     Louisville    Kentucky  Masked
1    Kansas City    Missouri  Masked
2        Seattle  Washington  Masked
3        Seattle  Washington  Masked
4       Columbus        Ohio  Masked
..           ...         ...     ...
193       Austin       Texas  Masked
194      Seattle  Washington  Masked
195      Seattle  Washington  Masked
196   Louisville    Kentucky  Masked
197     San Jose  California  Masked

[198 rows x 9 columns]
```

[204]:
```
data2.dtypes
```

[204]:
```
Credit_card       int64
Email            object
Name             object
Mobile_number     int64
Gender           object
Age               int64
City             object
State            object
Address          object
dtype: object
```

[205]:
```
age_is_less=data2.index[data2['Age']<18]
```

[206]:
```
data2 = data2.drop(index=age_is_less)
```

[207]:
```
data2=data2.reset_index(drop=True)
```

[208]:
```
data2.loc[38]
```

[208]:
```
Credit_card                  3500
Email            cheryl@hotmail.com
Name                         CHERYL
Mobile_number            8881689041
Gender                           F
Age                             73
City                       Seattle
State                   Washington
Address                     Masked
Name: 38, dtype: object
```

[209]:
```
data2['Credit_card'].unique
```

[209]:
```
<bound method Series.unique of 0      3768
1      4852
2      1174
3      4807
4      9131
       ...
182    4575
183    3958
184    8609
185    4542
186    9532
Name: Credit_card, Length: 187, dtype: int64>
```

From the above code we can say that the credit card id is unique for every customer because this code only give the unique value of the column and the length of column and the data set are same.

[210]:
```
data.columns
```

[210]:
```
Index(['Credit_card', 'Product_ID', 'P_CATEGORY', 'CONDTION', 'Brand',
       'Price ', 'Selling_price', 'Coupon_ID', 'Date', 'Time', 'GTIN', 'MPN',
       'Merchant_name', 'M_ID', 'Payment Method', 'Transaction ID',
       'Return_ind', 'Return_date', 'return_date_is_after_purchase date'],
      dtype='object')
```

Task 1: We need to segment the customers in different-different categories according to their age.

- Young age Females
- Mid-age Females
- Old age Females
- Young age Males
- Mid-age Males
- Old age Males

Task 1

```
[211]: data2.columns
```

```
[211]: Index(['Credit_card', 'Email', 'Name', 'Mobile_number', 'Gender', 'Age',
              'City ', 'State', 'Address'],
             dtype='object')
```

```
[212]: data2['Age_Category']=''
```

```
[213]: df_young_female_index=data2.index[(data2['Age']<=40) & (data2['Gender']=='F')]
```

```
[214]: data2.loc[df_young_female_index, 'Age_Category'] = 'Young_Females'
```

```
[215]: df_mid_age_female_index=data2.index[(data2['Age']>40) & (data2['Age']<=60)&(data2['Gender']=='F')]
```

```
[216]: data2.loc[df_mid_age_female_index, 'Age_Category'] = 'Mid_age_Females'
```

```
[217]: df_old_age_female_index=data2.index[(data2['Age']>60) & (data2['Gender']=='F')]
```

```
[218]: data2.loc[df_old_age_female_index, 'Age_Category'] = 'Old_Females'
```

```
[219]: Young_Males=data2.index[(data2['Age']<=40) & (data2['Gender']=='M')]
```

```
[220]: data2.loc[Young_Males,'Age_Category']='Young_Males'
```

```
[221]: Mid_age_Males=data2.index[(data2['Age']>40) & (data2['Age']<=60) & (data2['Gender']=='M')]
```

```
[222]: data2.loc[Mid_age_Males,'Age_Category']='Mid_age_Males'
```

```
[223]: Old_age_Males=data2.index[(data2['Age']>60) & (data2['Gender']=='M')]
```

```
[224]: data2.loc[Old_age_Males,'Age_Category']='Old_age_Males'
```

```
[225]: data2
```

[225]:

	Credit_card	Email	Name	Mobile_number	Gender	Age	City	State	Address	Age_Category
0	3768	eddie@yahoomail.com	EDDIE	9045258449	M	83	Louisville	Kentucky	Masked	Old_age_Males
1	4852	rose@hotmail.com	ROSE	8834789103	F	87	Kansas City	Missouri	Masked	Old_Females
2	1174	amy@yahoomail.com	AMY	9557690013	F	31	Seattle	Washington	Masked	Young_Females
3	4807	clarence@gmail.com	CLARENCE	9394398429	M	37	Seattle	Washington	Masked	Young_Males
4	9131	johnny@hotmail.com	JOHNNY	9976623538	F	80	Columbus	Ohio	Masked	Old_Females
...	...	...	...	...	...	...	...	...	...	...
182	4575	tina@yahoomail.com	TINA	8819353220	F	56	Austin	Texas	Masked	Mid_age_Females
183	3958	gordon@hotmail.com	GORDON	8853153218	M	87	Seattle	Washington	Masked	Old_age_Males
184	8609	jennifer@hotmail.com	JENNIFER	9935042642	F	93	Seattle	Washington	Masked	Old_Females
185	4542	michelle@hotmail.com	MICHELLE	9730382145	F	60	Louisville	Kentucky	Masked	Mid_age_Females
186	9532	katherine@hotmail.com	KATHERINE	9033150890	M	18	San Jose	California	Masked	Young_Males

187 rows × 10 columns

Task 2 : We need to calculate the amount in terms of Product, State and Payment Method.

Task 2

```
[226]: df_combined=data.merge(data2,on='Credit_card')

[227]: df_spend_on_product_category=data.groupby('P_CATEGORY')['Selling_price'].sum().sort_values(ascending=False)

[228]: df_spend_in_city=df_combined.groupby('City ')['Selling_price'].sum().sort_values(ascending=False)

[229]: df_spend_from_different_payment_method=df_combined.groupby('Payment Method')['Selling_price'].sum().sort_values(ascending=False)
```

Task 3 : We need to calculate the highest five spending in all above categories.

Task 3

```
[230]: df_spend_on_product_category.head()

[230]: P_CATEGORY
       DECOR              3200423
       COMPUTERS          3060948
       ELECTRONICS        2684788
       OFFICE SUPPLIES    2465747
       SHOES              2297932
       Name: Selling_price, dtype: int32

[231]: df_spend_in_city.head()

[231]: City
       Seattle          2122500
       Chicago          2006585
       Los Angeles      1962464
       Louisville       1950493
       San Francisco    1892234
       Name: Selling_price, dtype: int32

[232]: df_spend_from_different_payment_method.head()

[232]: Payment Method
       Mobile carrier Billing    10104350
       Credit card                8825501
       Prepaid card               4225844
       Debit card                  831263
       Paypal wallet               223621
       Name: Selling_price, dtype: int32
```

Task 4: We need to give opinion on return category like customers returning the products belongs to which state, age group, condition, category of the product or is it related to discount.

Task 4

```
[233]: df_return_customer_data=df_combined.index[df_combined['Return_date']!='1970-01-01']

[234]: df_combined[['Return_date','State','Age_Category','CONDTION','P_CATEGORY','Coupon_ID']].loc[df_return_customer_data]
```

[234]:

	Return_date	State	Age_Category	CONDTION	P_CATEGORY	Coupon_ID
5	2014-05-22	Massachusetts	Old_age_Males	Refurbished	DECOR	GK692
6	2014-10-28	Massachusetts	Old_age_Males	New	BABY CLOTHING	EJ951
7	2014-01-23	Massachusetts	Old_age_Males	Used	OFFICE SUPPLIES	BW988
8	2014-09-02	Massachusetts	Old_age_Males	New	SHOES	BN710
9	2014-04-11	Massachusetts	Old_age_Males	Used	LUGGAGE	GQ188
...	...	...	...	...	...	...
9334	2014-02-21	Arizona	Old_age_Males	New	KITCHEN & DINING	CI397
9359	2014-03-24	Illinois	Old_age_Males	New	KITCHEN & DINING	QB548
9360	2014-04-08	Illinois	Old_age_Males	Used	GAMES	HA460
9361	2014-03-29	Illinois	Old_age_Males	New	KITCHEN & DINING	XY385
9362	2014-01-29	Illinois	Old_age_Males	Refurbished	COMPUTERS	OP777

1441 rows × 6 columns

Task 5 : We need to create a profile of customers in terms of timing of their orders.

Task 5

[235]:
```
df_combined.columns
```

[235]:
```
Index(['Credit_card', 'Product_ID', 'P_CATEGORY', 'CONDTION', 'Brand',
       'Price ', 'Selling_price', 'Coupon_ID', 'Date', 'Time', 'GTIN', 'MPN',
       'Merchant_name', 'M_ID', 'Payment Method', 'Transaction ID',
       'Return_ind', 'Return_date', 'return_date_is_after_purchase date',
       'Email', 'Name', 'Mobile_number', 'Gender', 'Age', 'City ', 'State',
       'Address', 'Age_Category'],
      dtype='object')
```

[236]:
```
df_combined['Time']
```

[236]:
```
0       17:16:17
1       17:00:44
2       15:55:47
3       21:25:54
4       14:28:23
          ...
9382    10:48:06
9383    09:38:28
9384    22:53:20
9385    10:15:58
9386    10:48:06
Name: Time, Length: 9387, dtype: object
```

[237]:
```
df_index_of_morning=df_combined.index[(df_combined['Time']>='06:00:00') & (df_combined['Time']<'12:00:00')]
```

[238]:
```
df_index_of_afternoon=df_combined.index[(df_combined['Time']>='12:00:00') & (df_combined['Time']<'17:00:00')]
```

[239]:
```
df_index_of_evening=df_combined.index[(df_combined['Time']>='17:00:00') & (df_combined['Time']<'21:00:00')]
```

[240]:
```
df_index_of_night=df_combined.index[(df_combined['Time']>='21:00:00') & (df_combined['Time']<='24:00:00')]
```

[241]:
```
df_index_of_late_night=df_combined.index[(df_combined['Time']>='00:00:00') & (df_combined['Time']<'05:59:59')]
```

[242]:
```
df_combined['Timing']=''
```

[243]:
```
df_combined.loc[df_index_of_morning,'Timing']='Morning'
```

[244]:
```
df_combined.loc[df_index_of_afternoon,'Timing']='Afternoon'
```

[245]:
```
df_combined.loc[df_index_of_evening,'Timing']='Evening'
```

[246]:
```
df_combined.loc[df_index_of_night,'Timing']='Night'
```

[247]:
```
df_combined.loc[df_index_of_late_night,'Timing']='late Night'
```

[268]:
```
Morning=df_combined.loc[df_index_of_morning]
```

[269]:
```
Afternoon=df_combined.loc[df_index_of_afternoon]
```

[270]:
```
Evening=df_combined.loc[df_index_of_evening]
```

[271]:
```
Night=df_combined.loc[df_index_of_night]
```

[272]:
```
Late_Night=df_combined.loc[df_index_of_late_night]
```

Task 6 : We want to know which payment method provides more discount for customers ?

Task 6

```
[254]: data.columns
```

```
[254]: Index(['Credit_card', 'Product_ID', 'P_CATEGORY', 'CONDTION', 'Brand',
       'Price ', 'Selling_price', 'Coupon_ID', 'Date', 'Time', 'GTIN', 'MPN',
       'Merchant_name', 'M_ID', 'Payment Method', 'Transaction ID',
       'Return_ind', 'Return_date', 'return_date_is_after_purchase date'],
      dtype='object')
```

```
[255]: data['Payemnt_Method_discount']=data['Price ']-data['Selling_price']
```

```
[256]: Payment_method_discount=data.groupby('Payment Method')['Payemnt_Method_discount'].sum().sort_values(ascending=False)
```

```
[257]: Payment_method_discount
```

```
[257]: Payment Method
       Mobile carrier Billing    122786
       Credit card               111606
       Prepaid card               46739
       Debit card                  9178
       Paypal wallet               2381
       Deirect debits              2339
       Gift card                   1777
       Name: Payemnt_Method_discount, dtype: int32
```

Task 7: We must create a profile for high value items vs low value items and relate that with respect to their number of orders.

Task 7

```
[258]: data.columns
```

```
[258]: Index(['Credit_card', 'Product_ID', 'P_CATEGORY', 'CONDTION', 'Brand',
       'Price ', 'Selling_price', 'Coupon_ID', 'Date', 'Time', 'GTIN', 'MPN',
       'Merchant_name', 'M_ID', 'Payment Method', 'Transaction ID',
       'Return_ind', 'Return_date', 'return_date_is_after_purchase date',
       'Payemnt_Method_discount'],
      dtype='object')
```

```
[274]: data.groupby('P_CATEGORY')['P_CATEGORY'].count().sort_values(ascending=False)
```

```
[274]: P_CATEGORY
       COMPUTERS           1228
       DECOR               1132
       ELECTRONICS         1008
       SHOES                931
       CLOTHING             809
       OFFICE SUPPLIES      782
       KITCHEN & DINING     776
       GAMES                693
       LUGGAGE              588
       APPLIANCES           545
       BEDDING              544
       BABY CLOTHING        502
       BABY TOYS            461
       Name: P_CATEGORY, dtype: int64
```

Task 8: We need to find the answer if the merchant increases the discount price will it leads to increase in orders?

Task 8

```
[260]: coupon_code_is_null
```

```
[260]: Index([  5,  23,  37,  61,  85, 105, 134, 166, 201, 237, 272, 298, 323, 353,
               377, 399, 417, 418, 445, 505],
              dtype='int64')
```

```
[261]: null_coupon_id=data.iloc[coupon_code_is_null]
```

```
[262]: null_coupon_id.columns
```

```
[262]: Index(['Credit_card', 'Product_ID', 'P_CATEGORY', 'CONDTION', 'Brand',
               'Price ', 'Selling_price', 'Coupon_ID', 'Date', 'Time', 'GTIN', 'MPN',
               'Merchant_name', 'M_ID', 'Payment Method', 'Transaction ID',
               'Return_ind', 'Return_date', 'return_date_is_after_purchase date',
               'Payemnt_Method_discount'],
              dtype='object')
```

```
[263]: discount_not_given=null_coupon_id.groupby('P_CATEGORY')['P_CATEGORY'].count()
```

```
[264]: coupon_code_is_not_null=data[data['Coupon_ID'].notna()]
```

```
[265]: discount_given=coupon_code_is_not_null.groupby('P_CATEGORY')['P_CATEGORY'].count()
```

```
[266]: Increase_in_orders_or_not_if_discount_given=pd.concat([discount_not_given ,discount_given] ,axis=1 ,keys=['Discount not Given','Discount given'])
```

```
[267]: Increase_in_orders_or_not_if_discount_given
```

[267]:

P_CATEGORY	Discount not Given	Discount given
BABY TOYS	1.0	460
CLOTHING	2.0	807
COMPUTERS	5.0	1223
DECOR	1.0	1131
ELECTRONICS	2.0	1006
GAMES	1.0	692
KITCHEN & DINING	1.0	775
LUGGAGE	3.0	585
OFFICE SUPPLIES	1.0	781
SHOES	3.0	928
APPLIANCES	NaN	545
BABY CLOTHING	NaN	502
BEDDING	NaN	544

In the above code we can see that when discount is given the number of orders in high and when discount is not given the number of order is less then we can say that if more discount is given then the number of order will increase