

DATA SCIENCE LAB 3

Roll no: 20K-0409

Screen Shots

Task # 1

The screenshot shows a Jupyter Notebook titled "DS LAB # 3 EDA_20K-0409_Mukand". The first code cell imports pandas and numpy. The second code cell reads a CSV file and displays the first five rows of the data frame.

```
[1] 1 import pandas as pd
     2 import numpy as np

[2] 1 df = pd.read_csv('/content/telecom_churn.csv')
     2 df.head(5)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total eve minutes	total eve calls	total night minutes	total night calls	churn
0	KS	128	415	382-4657	no	yes	25	285.1	110	4	2	10	1	0
1	OH	107	415	371-7191	no	yes	26	161.6	123	2	1	10	1	0
2	NJ	137	415	358-1921	no	no	0	243.4	114	4	2	10	1	0
3	OH	84	408	375-9999	yes	no	0	299.4	71	5	2	10	1	0
4	OK	75	415	330-6626	yes	no	0	166.7	113	2	1	10	1	0

5 rows x 21 columns

```
[8] 1 df['churn'].value_counts()

0    2850
1     483
Name: churn, dtype: int64

[9] 1 df['churn'].value_counts(normalize=True)

0    0.855086
1    0.144914
Name: churn, dtype: float64

[10] 1 df.sort_values(by='total night charge', ascending=
```

The screenshot shows a Jupyter Notebook with a section titled "groupby methods". It contains two code cells. The first code cell uses groupby to describe statistics for different churn levels. The second code cell uses crosstab to show the relationship between voice mail plan and churn.

```
[22] 1 columns = ['total day minutes', 'total eve minutes', 'total night minutes']
     2 df.groupby(['churn'])[columns].describe(percentiles=[])

total day minutes      total eve minutes
count mean  std  min 50% max count mean  std
churn
0  2850.0 175.175754 50.181655 0.0 177.2 315.6 2850.0 199.043298 50.292174
1   483.0 206.914079 68.997792 0.0 217.6 350.8  483.0 212.410145 51.728910

[23] 1 pd.crosstab(df['churn'], df['voice mail plan'])

voice mail plan  no  yes
churn
0              2008  842
1               403   80
```

```

[14] 1 ## use any one column : e.g.,
      2 df[df['churn']==1]['total day calls'].mean()

101.33540372670808

[15] 1 df[(df['churn']==0) & (df['international plan']=="No")]['total intl minutes'].#
      nan

[16] 1 ## loc : indexing by name
      2 ## iloc : indexing by number

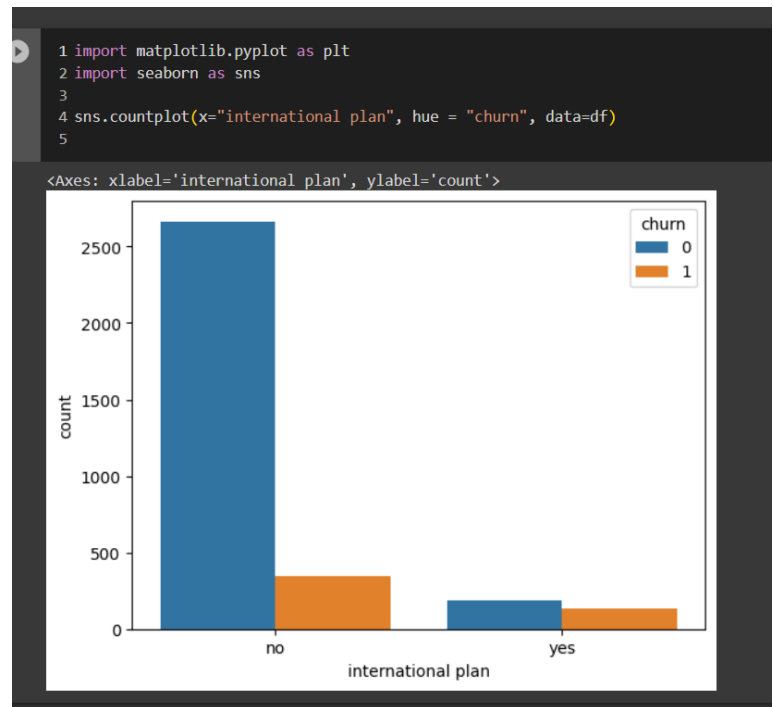
+ Code + Text

[17] 1 df.loc[0:5, "state":"area code"]

```

	state	account	length	area code
0	KS		128	415
1	OH		107	415
2	NJ		137	415

TASK # 2



TASK # 2

```

[36] 1
      2 data1 = {'cities': ['lahore', 'karachi'], 'provinces': ['punjab', 'sindh']}
      3 frame1 = pd.DataFrame(data1)
      4
      5 data2 = {'cities': ['islamabad', 'karachi', 'peshawar', 'quetta'], 'provinces': ['capital', 'sindh', 'KPK', 'Balochistan']}
      6 frame2 = pd.DataFrame(data2)
      7
      8 frame3 = pd.concat([frame1, frame2], ignore_index=True)
      9
     10 frame3 = frame3.drop_duplicates()
     11
     12 frame3 = frame3.sort_values(by='provinces')
     13
     14 frame3 = frame3.reset_index(drop=True)
     15
     16 print(frame3)
     17

```

	cities	provinces
0	quetta	Balochistan
1	peshawar	KPK
2	islamabad	capital
3	lahore	punjab
4	karachi	sindh

TASK#3 and 4

Task 3

```
[37] 1
2 data = {'Name': ['C', 'Ali', 'Ahmed', 'Nida', np.nan],
3         'Field': ['C', 'E', 'E', 'C', 'C'],
4         'Age': [np.nan, np.nan, np.nan, np.nan, np.nan],
5         'Marks': [-90, 60, -10, 70, 75]}
6 df = pd.DataFrame(data)
7
8 # Drop the 'Age' column
9 df = df.drop(columns=['Age'])
10
11 # Replace empty strings in the 'Name' column with '---'
12 df['Name'].replace('', '---', inplace=True)
13
14 # Replace 'C' with 0 and 'E' with 1 in the 'Field' column
15 df['Field'].replace({'C': 0, 'E': 1}, inplace=True)
16
17 # Replace negative values in the 'Marks' column
18 Avgmarks = df.loc[df['Marks'] >= 0, 'Marks'].mean()
19 df['Marks'] = df['Marks'].apply(lambda x: Avgmarks if x < 0 else x)
20
21 print(df)
22
```

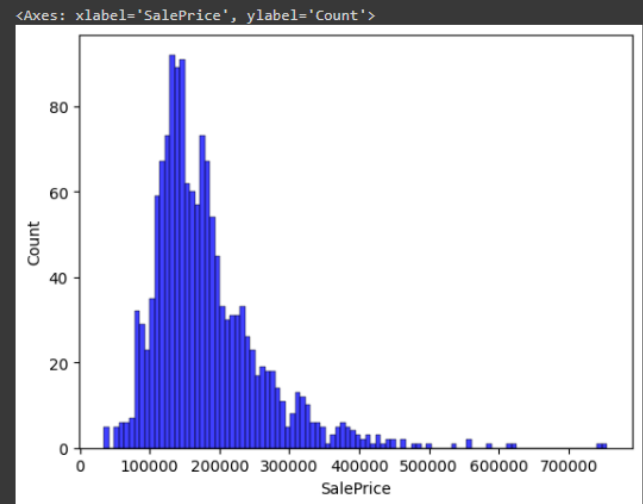
	Name	Field	Marks
0	C	0	68.333333
1	Ali	1	60.000000
2	Ahmed	1	68.333333
3	Nida	0	70.000000
4	NaN	0	75.000000

sale price

```
[40] 1 print(train['SalePrice'].describe())
```

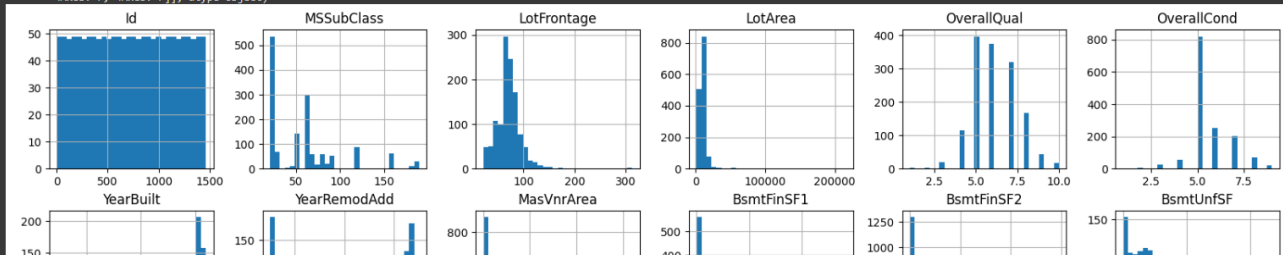
```
count    1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max       755000.000000
Name: SalePrice, dtype: float64
```

```
[42] 1 sns.histplot(train['SalePrice'], color = 'blue', bins = 100)
```



```
[44] 1 train_num = train.select_dtypes(include = ['float64', 'int64'])
2 train_num.head()
```

```
<Axes: title=[center: 'SalePrice'], <Axes: >, <Axes: >,
<Axes: >, <Axes: >], dtype=object>
```



```
[50] 1 corr = train_num.drop('SalePrice', axis=1).corr()
2
3 plt.figure(figsize=(16,14))
4
5 sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.4)],
6             cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
7             annot=True, annot_kws={"size": 8}, square=True);
```

