COGNIZANT NPN_BATCH 4

PROJECT TITLE: POTENTIAL CUSTOMERS FOR UPSELL

DATASET: CDR-Call Details Record Dataset

CTS MENTOR	Mr. NAGASUNDARAM
VEC MENTOR	Mrs. HANNAH ROSE ESTHER
TEAM LEADER	PRABHAKAR V
TEAM MEMBER	DHANUSHIYA S M
TEAM MEMBER	JAGADESSH M
TEAM MEMBER	KANISHKA B
TEAM MEMBER	PRITHI PRASANNA P
TEAM MEMBER	SHREERAM T

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
# pd.set_option('max_columns', 100)
```

EDA

Don't forget to change the dataset path when loading it in (read_csv() function's argument)

```
df = pd.read_csv('CDR-Call-Details.csv')
```

Understanding the data

```
df.shape
```

```
(101174, 17)
```

So there's 101174 rows, 17 columns/features

```
df.columns
```

```
Index(['Phone Number', 'Account Length', 'VMail Message', 'Day Mins',
       'Day Calls', 'Day Charge', 'Eve Mins', 'Eve Calls', 'Eve
Charge',
       'Night Mins', 'Night Calls', 'Night Charge', 'Intl Mins', 'Intl
Calls',
       'Intl Charge', 'CustServ Calls', 'Churn'],
      dtype='object')
```

There's 17 different columns

```
# replacing the spaces with underscores for code purposes i suppose
df.columns = df.columns.str.replace(' ', '_')

df.columns
```

```
Index(['Phone_Number', 'Account_Length', 'VMail_Message', 'Day_Mins',
       'Day_Calls', 'Day_Charge', 'Eve_Mins', 'Eve_Calls',
'Eve_Charge',
       'Night_Mins', 'Night_Calls', 'Night_Charge', 'Intl_Mins',
'Intl_Calls',
       'Intl_Charge', 'CustServ_Calls', 'Churn'],
      dtype='object')
```

```
df

       Phone_Number  Account_Length  VMail_Message  Day_Mins
Day_Calls  \
0          382-4657             128             25     265.1
110
1          371-7191             107             26     161.6
123
2          358-1921             137              0     243.4
114
3          375-9999              84              0     299.4
71
4          330-6626              75              0     166.7
113
...             ...             ...            ...       ...         ..
.
101169     789-9756             222              0     228.2
60
101170     798-5885              88              0     282.2
222
101171     798-5798              22              0     222.2
62
101172     999-9897             228              0     222.0
99
101173     786-7589             228              0     226.2
98

        Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
0            45.07     197.4         99       16.78       244.7
91
1            27.47     195.5        103       16.62       254.4
103
2            41.38     121.2        110       10.30       162.6
104
3            50.90      61.9         88        5.26       196.9
89
4            28.34     148.3        122       12.61       186.9
121
...            ...       ...        ...         ...         ...
...
101169       22.82     229.8        289       28.26       222.8
222
101170       82.88     208.8        220       22.82       282.2
200
101171       88.66     228.0        228       22.08        62.2
209
101172       88.08     220.2         80       22.92       282.9
28
101173       86.28     288.2        208       28.28       800.0
```

```
228

        Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  \
0              11.01       10.0           3         2.70
1
1              11.45       13.7           3         3.70
1
2               7.32       12.2           5         3.29
0
3               8.86        6.6           7         1.78
2
4               8.41       10.1           3         2.73
3
...              ...        ...         ...          ...                .
..
101169          2.28        6.2           2         2.62
2
101170         20.68        9.8           8         2.82
8
101171          2.26        2.8           6         2.22
2
101172         20.22        2.2           8         0.82
0
101173         28.80       20.0           8         2.20
2

        Churn
0       False
1       False
2       False
3       False
4       False
...       ...
101169  False
101170  False
101171  False
101172  False
101173  False

[101174 rows x 17 columns]
```

```
#potential feature engg

# df['day_avg'] = df['Day Mins'] / df['Day Calls']
# df['day_avg']

df.head()
```

```
  Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls  \
0     382-4657             128             25     265.1        110
1     371-7191             107             26     161.6        123
2     358-1921             137              0     243.4        114
3     375-9999              84              0     299.4         71
4     330-6626              75              0     166.7        113

   Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
0       45.07     197.4         99       16.78       244.7
91
1       27.47     195.5        103       16.62       254.4
103
2       41.38     121.2        110       10.30       162.6
104
3       50.90      61.9         88        5.26       196.9
89
4       28.34     148.3        122       12.61       186.9
121

   Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  CustServ_Calls
Churn
0         11.01       10.0           3         2.70               1
False
1         11.45       13.7           3         3.70               1
False
2          7.32       12.2           5         3.29               0
False
3          8.86        6.6           7         1.78               2
False
4          8.41       10.1           3         2.73               3
False

df.tail()

       Phone_Number  Account_Length  VMail_Message  Day_Mins
Day_Calls  \
101169     789-9756             222              0     228.2
60
101170     798-5885              88              0     282.2
222
101171     798-5798              22              0     222.2
62
101172     999-9897             228              0     222.0
99
101173     786-7589             228              0     226.2
98

        Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
```

```
101169       22.82     229.8         289       28.26       222.8
222
101170       82.88     208.8         220       22.82       282.2
200
101171       88.66     228.0         228       22.08        62.2
209
101172       88.08     220.2          80       22.92       282.9
28
101173       86.28     288.2         208       28.28       800.0
228

        Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  \
101169          2.28        6.2           2         2.62
2
101170         20.68        9.8           8         2.82
8
101171          2.26        2.8           6         2.22
2
101172         20.22        2.2           8         0.82
0
101173         28.80       20.0           8         2.20
2

        Churn
101169  False
101170  False
101171  False
101172  False
101173  False
```

Viewed the head and tail to get a look at the data present

Assumptions

assuming Account_Length is measured in days

assuming charges are measured in USD ($)

(because there's no dataset discription provided)

```
print(df.info()) #objects are strings essentially

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101174 entries, 0 to 101173
Data columns (total 17 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Phone_Number    101174 non-null  object
```

```
 1   Account_Length  101174 non-null  int64
 2   VMail_Message   101174 non-null  int64
 3   Day_Mins        101174 non-null  float64
 4   Day_Calls       101174 non-null  int64
 5   Day_Charge      101174 non-null  float64
 6   Eve_Mins        101174 non-null  float64
 7   Eve_Calls       101174 non-null  int64
 8   Eve_Charge      101174 non-null  float64
 9   Night_Mins      101174 non-null  float64
 10  Night_Calls     101174 non-null  int64
 11  Night_Charge    101174 non-null  float64
 12  Intl_Mins       101174 non-null  float64
 13  Intl_Calls      101174 non-null  int64
 14  Intl_Charge     101174 non-null  float64
 15  CustServ_Calls  101174 non-null  int64
 16  Churn           101174 non-null  bool
dtypes: bool(1), float64(8), int64(7), object(1)
memory usage: 12.4+ MB
None
```

There's 15 numbers - 8 float, 7 int columns then 1 bool and 1 object column as depicted above

Note :- Churn column is in 'True' or 'False' not '0' or '1' but python treats them as 0 or 1 so no need to convert them into numeric

```
print(df.describe())
```

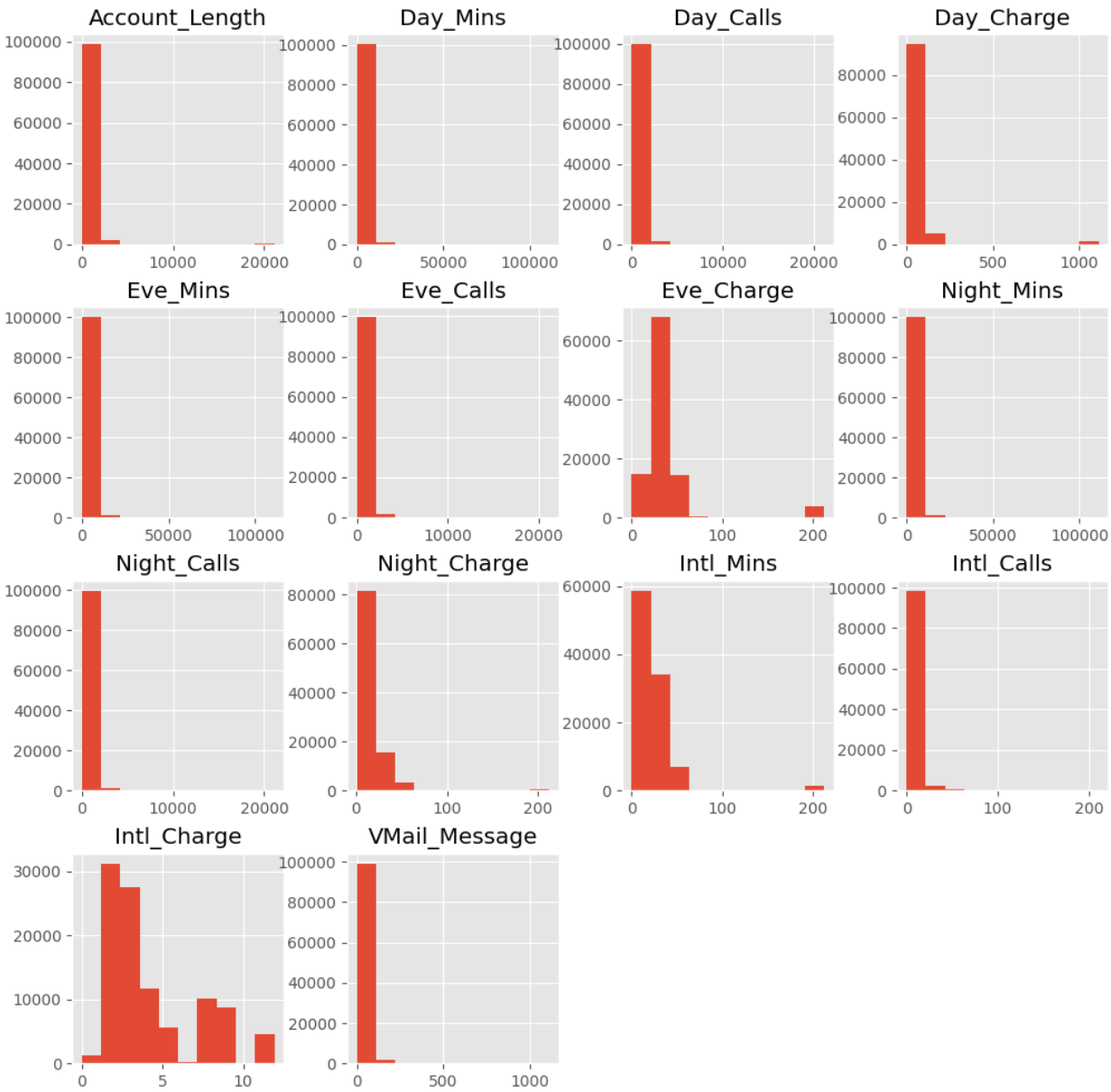
```
       Account_Length  VMail_Message       Day_Mins      Day_Calls  \
count   101174.000000  101174.000000  101174.000000  101174.000000
mean       323.597149      18.142645     587.080798     267.207613
std       1418.073651      75.826932    2193.094319     947.963370
min          1.000000       0.000000       0.000000       0.000000
25%         69.000000       0.000000     222.300000      88.000000
50%         202.000000       0.000000     262.200000     202.000000
75%         240.000000      22.000000     326.275000     224.000000
max       21111.000000    1111.000000  111111.110000   21111.000000

          Day_Charge       Eve_Mins      Eve_Calls     Eve_Charge  \
count  101174.000000  101174.000000  101174.000000  101174.000000
mean       64.958811     655.512526     267.166930      34.358125
std       133.515808    2473.445436     932.914437      36.248577
min         0.000000       0.000000       0.000000       0.000000
25%        23.920000     223.300000      88.000000      22.220000
50%        32.600000     268.200000     202.000000      24.440000
75%        82.200000     329.200000     224.000000      32.020000
max      1111.990000  111111.200000   21111.000000     211.990000

          Night_Mins    Night_Calls   Night_Charge      Intl_Mins  \
count  101174.000000  101174.000000  101174.000000  101174.000000
mean      646.786643     261.673187      13.900142      20.291269
```

```
std      2368.028491     884.963021      16.178685      25.033214
min        20.300000      20.000000       1.040000       0.000000
25%       223.200000      88.000000       6.220000       8.800000
50%       268.200000     202.000000       9.220000      20.200000
75%       329.200000     224.000000      20.440000      22.800000
max    111111.110000   21111.000000     211.920000     211.900000

          Intl_Calls    Intl_Charge  CustServ_Calls
count  101174.000000  101174.000000   101174.000000
mean        5.737798       4.347635        2.559640
std         8.196367       2.761911        2.401904
min         0.000000       0.000000        0.000000
25%         2.000000       2.280000        2.000000
50%         4.000000       2.920000        2.000000
75%         8.000000       4.940000        3.000000
max       211.000000      11.920000       11.000000
```

Inference

- Account_Length has a mix value of 21111 which is highly unlikely (21111 / 365 = 57 ish years which is highly unlikely)
- Similar to account length other fearues also have questionable maximum values indicating clear presence of outliers skewing the dataset essentially
- Every column has mean << 75th percentile so it's very much right skewed
- Higher values of std suggest wider range of values
- Vmail & CustServ features & Intl(mins, call, charge) features aside rest needs to be addressed of outliers because of impossible values

```
#specifying the numeric columns
numeric_cols = ['Account_Length','Day_Mins','Day_Calls','Day_Charge',
'Eve_Mins', 'Eve_Calls','Eve_Charge',
'Night_Mins','Night_Calls','Night_Charge','Intl_Mins','Intl_Calls','In
tl_Charge', 'VMail_Message']

# histograms
df[numeric_cols].hist(figsize=(12, 12))
plt.show()
```



- histogram clearly indicating skewness

dropping the churn column

```
# df = df.drop(['Churn'], axis=1).copy()
```

- From what I've seen account_length seems to indicate the duration the account was open so given the stat summary of it I assume it is measured in days which validates the presence of other features

Null & NaN

```
print(df.isna().sum())
```

```
Phone_Number      0
Account_Length    0
VMail_Message     0
Day_Mins          0
Day_Calls         0
Day_Charge        0
Eve_Mins          0
Eve_Calls         0
Eve_Charge        0
Night_Mins        0
Night_Calls       0
Night_Charge      0
Intl_Mins         0
Intl_Calls        0
Intl_Charge       0
CustServ_Calls    0
Churn             0
dtype: int64
```

Duplicates

```
df.duplicated().sum()
```

```
40729
```

- around 40k rows are duplicates
- df.duplicated() without mentioning any specific features will compare all the features values for exact match essentially looking for 1-1 copy

```
df.loc[df.duplicated()]
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
9999      785-9657             228             28     268.2        220
10000     779-7999             202             26     262.6        228
10001     758-9959             282              0     228.2        222
10002     775-9999              82              0     299.2         22
10003     778-6656              28              0     266.2        228
...            ...             ...            ...       ...        ...
99985     775-9988             292             86     286.2         22
99986     757-7869              68              0     282.2         82
```

```
99987     797-5759              28              0     280.8         209

99988     765-9799             282              0     228.8         208

99989     786-8989              22             28     282.2         228


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
9999        28.02     292.2         99       26.28       222.2
92
10000       22.22     298.8        208       26.62       282.2
208
10001       22.88     222.2        220       20.80       262.6
202
10002       80.90      62.9         88        8.26       296.9
89
10003       28.82     228.8        222       22.62       286.9
222
...           ...       ...        ...         ...         ...
...
99985       26.88     228.8        226       28.82       229.2
88
99986       89.29     288.2         88       28.02       292.8
228
99987       80.22     288.8         88       22.88       292.9
92
99988       86.88     289.6         82       28.82       289.2
282
99989       89.88     268.9         82       22.60       222.2
22

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
9999          22.02       20.0           8         2.20
2  False
10000         22.28       28.2           8         8.20
2  False
10001          2.82       22.2           8         8.29
0  False
10002          8.86        6.6           2         2.28
2  False
10003          8.22       20.2           8         2.28
8  False
...             ...        ...         ...          ...              ..
.    ...
99985         22.86        9.9           6         2.62
2  False
99986          8.62        9.6           2         2.89
8  False
```

```
99987          8.62       22.2           6         8.82
2  False
99988          6.26        8.0          20         2.88
2  False
99989         20.86       28.2           2         8.20
0  False

[40729 rows x 17 columns]
```

```
df.query('Phone_Number == "779-7999"').head()
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
3334      779-7999             407             46     464.6        445

6667      779-7999             202             26     262.6        228

10000     779-7999             202             26     262.6        228

13333     779-7999             202             26     262.6        228

16666     779-7999             202             26     262.6        228


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
3334        47.47     495.5        405       46.64       454.4
405
6667        22.22     298.8        208       26.62       282.2
208
10000       22.22     298.8        208       26.62       282.2
208
13333       22.22     298.8        208       26.62       282.2
208
16666       22.22     298.8        208       26.62       282.2
208

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
3334          44.45       45.7           5          5.7
4  False
6667          22.28       28.2           8          8.2
2  False
10000         22.28       28.2           8          8.2
2  False
13333         22.28       28.2           8          8.2
2  False
16666         22.28       28.2           8          8.2
2  False
```

- Making sure there's indeed outliers by manually quering and yes there's indeed duplicates

```
df = df.loc[~df.duplicated()].reset_index(drop=True).copy()
df
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
0         382-4657             128             25     265.1        110

1         371-7191             107             26     161.6        123

2         358-1921             137              0     243.4        114

3         375-9999              84              0     299.4         71

4         330-6626              75              0     166.7        113

...            ...             ...            ...       ...        ...

60440     789-9756             222              0     228.2         60

60441     798-5885              88              0     282.2        222

60442     798-5798              22              0     222.2         62

60443     999-9897             228              0     222.0         99

60444     786-7589             228              0     226.2         98


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
0           45.07     197.4         99       16.78       244.7
91
1           27.47     195.5        103       16.62       254.4
103
2           41.38     121.2        110       10.30       162.6
104
3           50.90      61.9         88        5.26       196.9
89
4           28.34     148.3        122       12.61       186.9
121
...           ...       ...        ...         ...         ...
...
60440       22.82     229.8        289       28.26       222.8
222
60441       82.88     208.8        220       22.82       282.2
200
60442       88.66     228.0        228       22.08        62.2
209
```

```
60443       88.08     220.2          80       22.92       282.9
28
60444       86.28     288.2         208       28.28       800.0
228

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
0             11.01       10.0           3         2.70
1  False
1             11.45       13.7           3         3.70
1  False
2              7.32       12.2           5         3.29
0  False
3              8.86        6.6           7         1.78
2  False
4              8.41       10.1           3         2.73
3  False
...             ...        ...         ...          ...              ..
.    ...
60440          2.28        6.2           2         2.62
2  False
60441         20.68        9.8           8         2.82
8  False
60442          2.26        2.8           6         2.22
2  False
60443         20.22        2.2           8         0.82
0  False
60444         28.80       20.0           8         2.20
2  False

[60445 rows x 17 columns]
```

- Removed the duplicates which dropped the total rows from 100k to 60k (approx values), so as mentioned earlier 40k rows are duplicates

unique counts in a feature

```
df.nunique()

Phone_Number      7467
Account_Length     322
VMail_Message       72
Day_Mins          2548
Day_Calls          221
Day_Charge        2873
Eve_Mins          2523
Eve_Calls          224
Eve_Charge        2221
Night_Mins        2464
Night_Calls        218
```

```
Night_Charge      1470
Intl_Mins          267
Intl_Calls          39
Intl_Charge        339
CustServ_Calls      11
Churn                2
dtype: int64
```

- Checking for unique counts in hopes of finding anything unusual

```
df.shape

(60445, 17)
```

- Confirming shape once again

Outliers

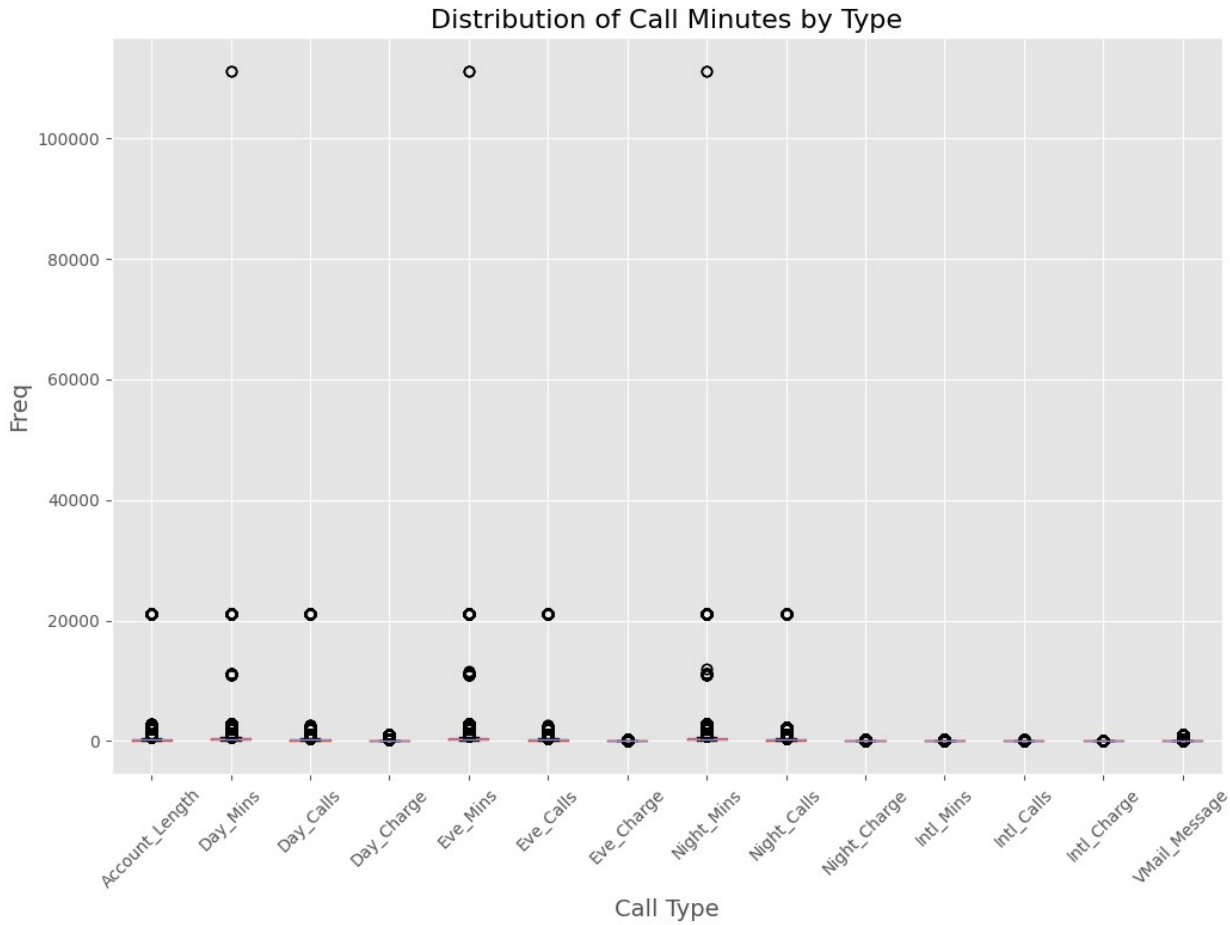
```
df.describe()

       Account_Length  VMail_Message       Day_Mins     Day_Calls  \
count    60445.000000   60445.000000   60445.000000  60445.000000
mean       329.385541      18.101613     596.555649    274.752138
std       1436.763064      76.859936    2231.437206    997.047766
min          1.000000       0.000000       0.000000      0.000000
25%         68.000000       0.000000     222.200000     88.000000
50%        200.000000       0.000000     262.200000    202.000000
75%        243.000000      22.000000     404.400000    226.000000
max      21111.000000    1111.000000  111111.110000  21111.000000

         Day_Charge       Eve_Mins     Eve_Calls    Eve_Charge
Night_Mins  \
count  60445.000000   60445.000000  60445.000000  60445.000000
60445.000000
mean      64.880486     669.288486    272.647911     34.894168
661.395415
std      134.695269    2476.768325    956.454815     36.584588
2422.994289
min        0.000000       0.000000      0.000000      0.000000
20.300000
25%       24.220000     223.400000     88.000000     22.220000
223.200000
50%       32.920000     268.800000    200.000000     24.900000
269.110000
75%       80.480000     440.400000    226.000000     32.620000
440.400000
max     1111.990000  111111.200000  21111.000000    211.990000
111111.110000

        Night_Calls  Night_Charge     Intl_Mins    Intl_Calls
```

```
Intl_Charge  \
count  60445.000000  60445.000000  60445.000000  60445.000000
60445.000000
mean     267.133377     14.066812     20.571371      5.728894
4.315376
std      913.240573     16.532483     25.494288      8.271802
2.711543
min       20.000000      1.040000      0.000000      0.000000
0.000000
25%       88.000000      6.220000      8.800000      2.000000
2.280000
50%      200.000000      9.220000     20.200000      4.000000
2.920000
75%      226.000000     20.400000     22.800000      8.000000
4.940000
max    21111.000000    211.920000    211.900000    211.000000
11.920000

       CustServ_Calls
count    60445.000000
mean         2.563438
std          2.376449
min          0.000000
25%          2.000000
50%          2.000000
75%          4.000000
max         11.000000
```

After removing duplicates and before addressing/removing outliers checking stat summary once more

- mean > 75th percentile so large valued outliers pulling the mean up while majority is on the lower side?
- meadian[50th percentile] 0 for Vmail suggesting majority of customers don't use it at all, so no need to include that for outlier removal

```
fig, ax = plt.subplots(figsize=(12, 8))

# boxplot (all in one)
df.boxplot(column= numeric_cols,
           ax=ax)

# labellingg
ax.set_title('Distribution of Call Minutes by Type', fontsize=16)
ax.set_xlabel('Call Type', fontsize=14)
ax.set_ylabel('Freq', fontsize=14)
ax.set_xticklabels(numeric_cols, rotation=45)

plt.show()
```



By visualizing through box plot we are able to clearly identify outliers

```
df[df['Account_Length'] > 10000]
```

	Phone_Number	Account_Length	VMail_Message	Day_Mins	Day_Calls
18153	777-9685	21111	22	2211.20	1111
18182	987-6577	21111	0	222.11	911
18203	985-5887	21111	0	2011.20	200
18214	755-5578	21111	0	2112.20	112
18278	786-5957	21111	112	2119.11	226
...	...	...	...	...	...
26096	987-9958	21111	0	222.20	2211
26157	989-7887	21111	0	290.00	200
26200	789-7997	21111	0	2116.20	62

```
26204     768-9758           21111           1111  21111.20       2119

26207     785-7595           21111              0   2911.00        222


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
18153     29.2110   2011.90        202     22.1111      222.20
211
18182     22.2200    220.11       1111     22.2900     2911.11
202
18203    112.1111    290.90        202     26.2110      296.00
229
18214    211.2900    229.00       2011     20.2200     2110.00
200
18278    112.2200   2611.11      21111    211.1111      262.11
226
...           ...       ...        ...         ...         ...
...
26096     22.1100    229.11        611     22.2110      222.20
112
26157    112.1100    226.60        211     20.9600     1102.20
202
26200    211.6110   2112.20         60    211.9200      226.20
222
26204     26.1111   2611.11         29     22.1120     2116.20
22
26207    110.2110   2211.60        202     29.0200      229.20
209

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
18153       11.1100      22.60         211       11.920
0   True
18182      211.2900      22.11           2       11.290
11  False
18203       11.1120      11.90           2        2.200
0   True
18214       11.2000      20.11           6        2.112
0  False
18278       22.9200       2.11          11        2.011
2  False
...             ...        ...         ...          ...              ..
.    ...
26096        9.1120     211.60          11       11.620
2  False
26157      211.6900       9.11           2        2.112
2  False
26200        6.1111      22.00           2        2.920
```

```
2   True
26204       11.1190       9.20            2         2.620
11  False
26207       20.1120      22.90            2        11.211
2   True

[270 rows x 17 columns]
```

```
df[df['Account_Length'] > 10000].nunique()
```

```
Phone_Number      267
Account_Length      1
VMail_Message      10
Day_Mins           59
Day_Calls          25
Day_Charge         70
Eve_Mins           61
Eve_Calls          28
Eve_Charge         60
Night_Mins         57
Night_Calls        27
Night_Charge       73
Intl_Mins          29
Intl_Calls          8
Intl_Charge        31
CustServ_Calls      3
Churn               2
dtype: int64
```

- only 200 rows having account length greater then 10k which is all the same value(it has only 1 unique value) from the looks of it [21111]
- could've done a more nuanced outlier detection by doing feature engineering but not sure what features to create so we shall not do that

```
df[(df['Day_Mins'] > 20000)]
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
18121     978-5875             222            112  21111.60        112

18132     767-7969             222              0  21111.20        222

18134     765-7787            2110              0  21111.00        222

18161     757-7858              92             22  21111.20      21111

18186     777-7977            1111              0  21111.11        209

...            ...             ...            ...       ...        ...

26191     988-5689             222              0  21111.20        911
```

```
26204     768-9758           21111           1111  21111.20       2119

26211     999-5977             112              0  21111.00         92

26214     757-8595              99              0  21111.20         96

26215     799-7995             112              0  21111.20        222


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
18121    211.9600    222.00        222     211.112     1126.20
92
18132     26.1120   2119.20        911      20.112     2011.11
21111
18134    112.2200     22.90         99       6.200     2112.11
211
18161     22.6200    222.20       1111     211.260       20.60
29
18186     20.0900   2112.20         92     211.116    21111.20
99
...           ...       ...        ...         ...         ...
...
26191    211.1120    266.90         92      22.690      292.20
2211
26204     26.1111   2611.11         29      22.112     2116.20
22
26211     20.2600    262.11         92     211.911     2112.11
2112
26214     20.1111    226.11       2110      20.911     2911.20
222
26215     26.1111   2211.20        229       9.290     2112.20
92

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
18121        22.690      22.20          11      11.0200
0  False
18132         9.200      20.60           2       2.1160
0  False
18134        11.211       9.11          29       2.1120
0  False
18161        11.211      22.00          11       2.9200
2  False
18186        11.220      22.00          11      11.2200
0  False
...             ...        ...         ...         ...              ..
.    ...
26191        11.900      11.11          11       2.1111
```

```
11   True
26204        11.119       9.20            2       2.6200
11  False
26211        22.220      20.60            6       2.1160
2  False
26214        11.911      22.20            2      11.1111
11  False
26215        11.220      22.11            2      11.2900
0  False

[585 rows x 17 columns]
```

```
df[(df['Day_Mins'] > 20000)]['Day_Mins'].nunique()
```

```
6
```

```
df[(df['Eve_Mins'] > 20000)]
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
18139     797-7558             112              0   2112.11        211

18140     797-6857              20              0    290.00        209

18183     977-6857             262             26    222.90         92

18191     778-9769              20              0    220.20        911

18198     985-7778              92              0   2112.90       2011

...            ...             ...            ...       ...        ...

26227     989-6977            1111             22    222.20        222

26230     799-6777             119              0   2211.20        112

26231     799-9859            2611              0    292.20         90

26237     959-9989             611              0   2112.20        112

26238     999-5667             211              0   2110.11        209


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
18139     22.1111  21111.11        200    211.2200      202.20
611
18140    112.1100  21111.20        112     22.9110     2112.11
202
18183   1111.2110  21111.20        112     26.0000     2112.60
62
```

```
18191    211.9110  21111.20        202    211.2900     2211.60
26
18198     26.1120  21111.00        202    211.2110     2119.60
112
...           ...       ...        ...         ...         ...
...
26227   1111.2000  21111.11       1111     20.1100      222.20
116
26230    110.1111  21111.20         22     29.1160     2112.90
220
26231   1111.1120  21111.11       2211     26.0200      222.20
92
26237    119.2900  21111.20       1111    211.0200      292.11
2211
26238    110.2200  21111.11       1111     22.1111      292.90
92

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
18139        2.1190      22.20           2       11.920
11  False
18140       11.2200       6.11           6        2.200
0  False
18183       22.2600      22.20           2       11.220
0  False
18191       20.2900     211.00           2        2.011
2  False
18198       11.1111      11.00          11        2.260
2   True
...             ...        ...         ...          ...              ..
.    ...
26227        2.1120      22.11           2       11.220
2  False
26230       11.9200       9.20           2        2.260
2  False
26231        9.1100       2.11          11        2.220
2  False
26237       11.6200       9.60           2        2.119
11  False
26238       11.6200      22.20           6       11.112
2  False

[712 rows x 17 columns]
```

```
df[(df['Eve_Mins'] > 20000)]['Eve_Mins'].nunique()
```

```
7
```

```
df[(df['Night_Mins'] > 20000)]
```

```
       Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
18151     796-5567             229              0    220.20         92

18174     779-6759              22           1111   2911.20         92

18180     796-7578             226              0    222.60         20

18186     777-7977            1111              0  21111.11        209

18204     795-7785             110              0    222.11        200

...            ...             ...            ...       ...        ...

26148     798-9775             220              0    292.60       2211

26156     755-5889             116              0    222.90        222

26210     775-7999            2112              0    262.90       2211

26216     799-9898             206            110    220.20       2011

26221     798-9969             911              0    260.00      21111


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
18151     211.112    222.11         92     211.200    21111.11
2011
18174     112.911    226.20         96      20.920    21111.00
92
18180    1111.920    226.90        110     211.220    21111.11
60
18186      20.090   2112.20         92     211.116    21111.20
99
18204      22.211   2211.00        202      22.220    21111.20
62
...           ...       ...        ...         ...         ...
...
26148     112.220    206.20       2011      22.112    21111.20
911
26156     112.220   2211.20        222     211.611    21111.20
1111
26210     211.011    226.11         96      20.211    21111.11
2211
26216     112.220    222.20        209     211.119    21111.20
96
26221      22.200   2211.11        911     211.110    21111.90
112

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
```

```
CustServ_Calls  Churn
18151         11.220      22.20            9        11.000
2  False
18174          6.220      22.60           11        11.920
2  False
18180          6.920       2.11            2         2.220
2  False
18186         11.220      22.00           11        11.220
0  False
18204         22.119       2.90            9         2.211
2  False
...              ...        ...          ...           ...                ..
.    ...
26148         22.220      20.11           11         2.920
2  False
26156         22.960       2.20           11         2.920
0   True
26210         20.211      20.00            9         2.200
2  False
26216          2.211     211.20           11        11.112
0  False
26221         11.110       9.20            2         2.260
0  False

[709 rows x 17 columns]
```

```
df[(df['Night_Mins'] > 20000)]['Night_Mins'].nunique()
```

```
6
```

```
df[df['Day_Charge'] > 1000]
```

```
      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
18128     976-7576            1111             22    296.20       2119

18131     798-6557             211              0    222.20         90

18143     799-7889             211              0    226.20       2011

18158     777-5986             611              0    222.11        220

18169     789-6775             222            110   2911.20        229

...            ...             ...            ...       ...        ...

26167     755-9985              22              0   2211.11         62

26198     787-9996             260              0    206.11         66

26209     997-9758             222             22    222.20        222
```

```
26227     989-6977            1111             22    222.20        222

26231     799-9859            2611              0    292.20         90


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
18128    1111.119   2110.90         90    211.1111      119.11
211
18131    1111.211   2119.11       1111    211.1160      292.11
22
18143    1111.220    202.11        202     22.2110      226.20
911
18158    1111.920    262.60        222    211.1120     2112.20
2211
18169    1111.211    211.11         22      6.2000     2112.20
22
...           ...       ...        ...         ...         ...
...
26167    1111.011    222.11         22     20.1120     2211.11
2116
26198    1111.020    222.20        209     20.2900      222.11
202
26209    1111.211    226.00        206     22.2200      222.20
911
26227    1111.200  21111.11       1111     20.1100      222.20
116
26231    1111.112  21111.11       2211     26.0200      222.20
92

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
18128         2.020     211.11           2      11.2110
2  False
18131        11.611     211.00           2      11.1120
2  False
18143        22.011      20.11          11       2.2110
2  False
18158         6.060     211.20          11      11.1160
11  False
18169        11.211      11.11          11       2.1120
11   True
...             ...        ...         ...         ...              ..
.    ...
26167        20.020      22.11          11      11.1120
11  False
26198        20.211      22.20           6      11.2600
0  False
26209        20.116      11.11           2       2.1111
```

```
2  False
26227         2.112      22.11            2      11.2200
2  False
26231         9.110       2.11           11       2.2200
2  False

[914 rows x 17 columns]
```

```
df[df['Day_Charge'] > 1000]['Day_Charge'].nunique()
```

```
25
```

```
df[df['VMail_Message'] > 1000]['VMail_Message'].nunique()
```

```
1
```

- even though the mins features might have extreme values those have single digit unique values indicating outliers (all those customers only seem to have the same 6 values repeating over so yeah it's best to include them for outlier removal)
- from seeing the above it is best to include day, eve & night mins features for outlier detection because it quite varies unless there's a way to pinpoint them out by means of a intuitive feature engineering?

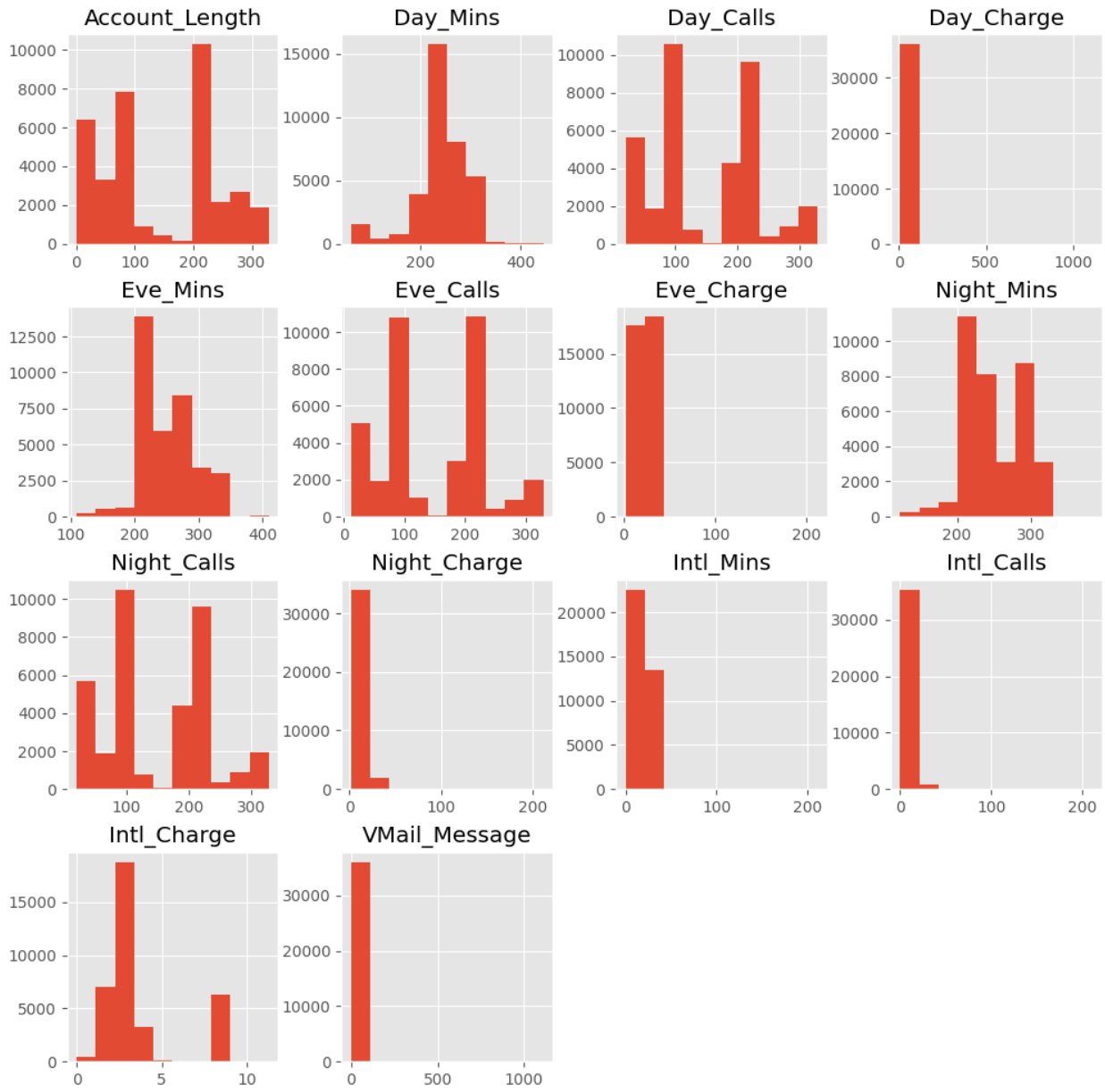
```
def remove_outliers_iqr(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column_name] >= lower_bound) & (df[column_name] <=
upper_bound)]
#     return df[(df[column_name] <= upper_bound)]

outlier_cols = ['Account_Length','Day_Mins','Day_Calls', 'Eve_Mins',
'Eve_Calls', 'Night_Mins','Night_Calls']
# Loop through each selected col/feature
for col in outlier_cols:
    df = remove_outliers_iqr(df, col)

# Resetting index after filtering else it'll be inconsistent!
df.reset_index(drop=True, inplace=True)
```

```
df.shape
```

```
(36077, 17)
```

- dropped 24k ish rows there

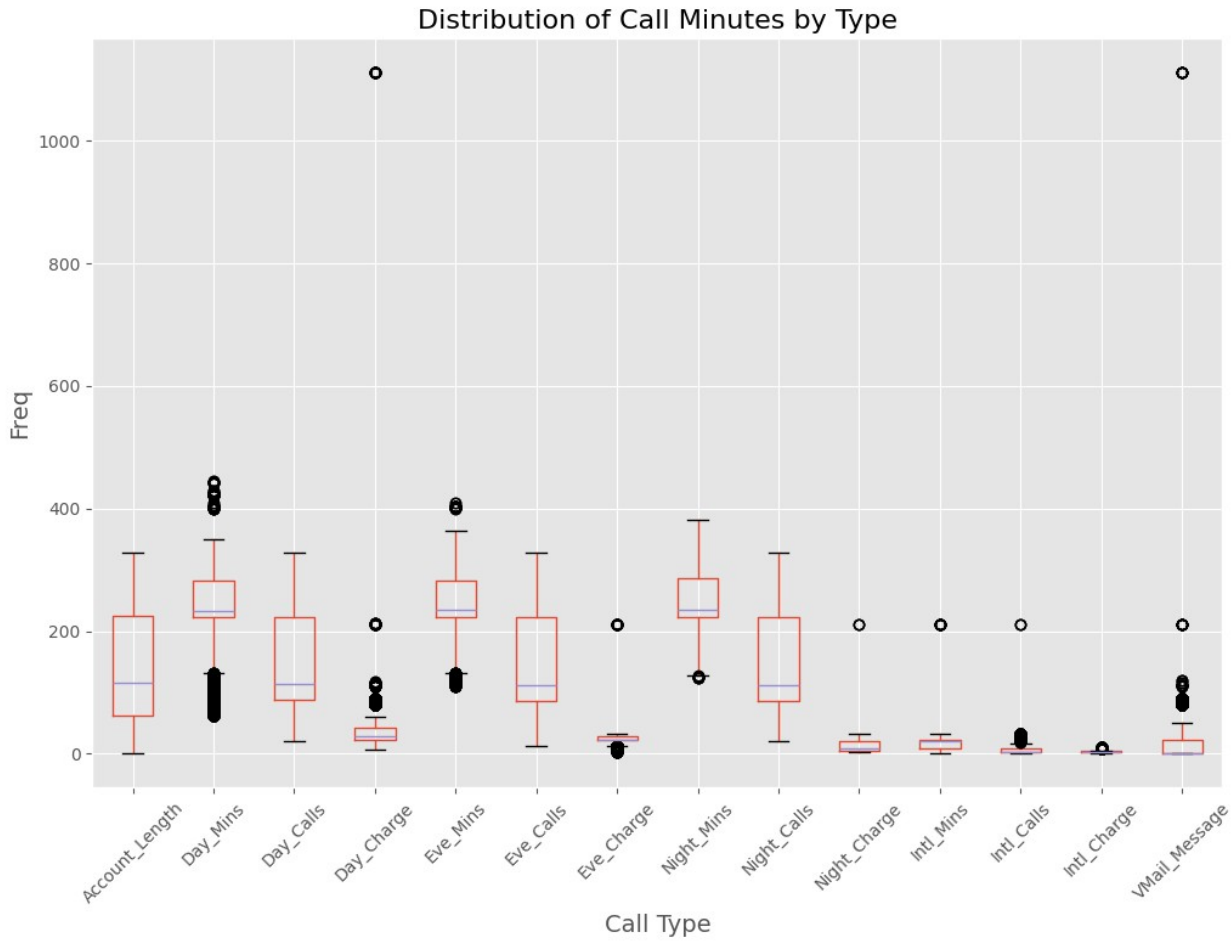
```
# histograms
df[numeric_cols].hist(figsize=(12, 12))
plt.show()
```



- After addressing some outliers the histogram looks not so skewed now

```
fig, ax = plt.subplots(figsize=(12, 8))

# box plot
df.boxplot(column=numeric_cols,
           ax=ax)

# titles and labels
ax.set_title('Distribution of Call Minutes by Type', fontsize=16)
ax.set_xlabel('Call Type', fontsize=14)
ax.set_ylabel('Freq', fontsize=14)
ax.set_xticklabels(numeric_cols, rotation=45)
```

```
plt.show()
```



- By means of another boxplot after removing extreme/impossible values we can get a more better viz

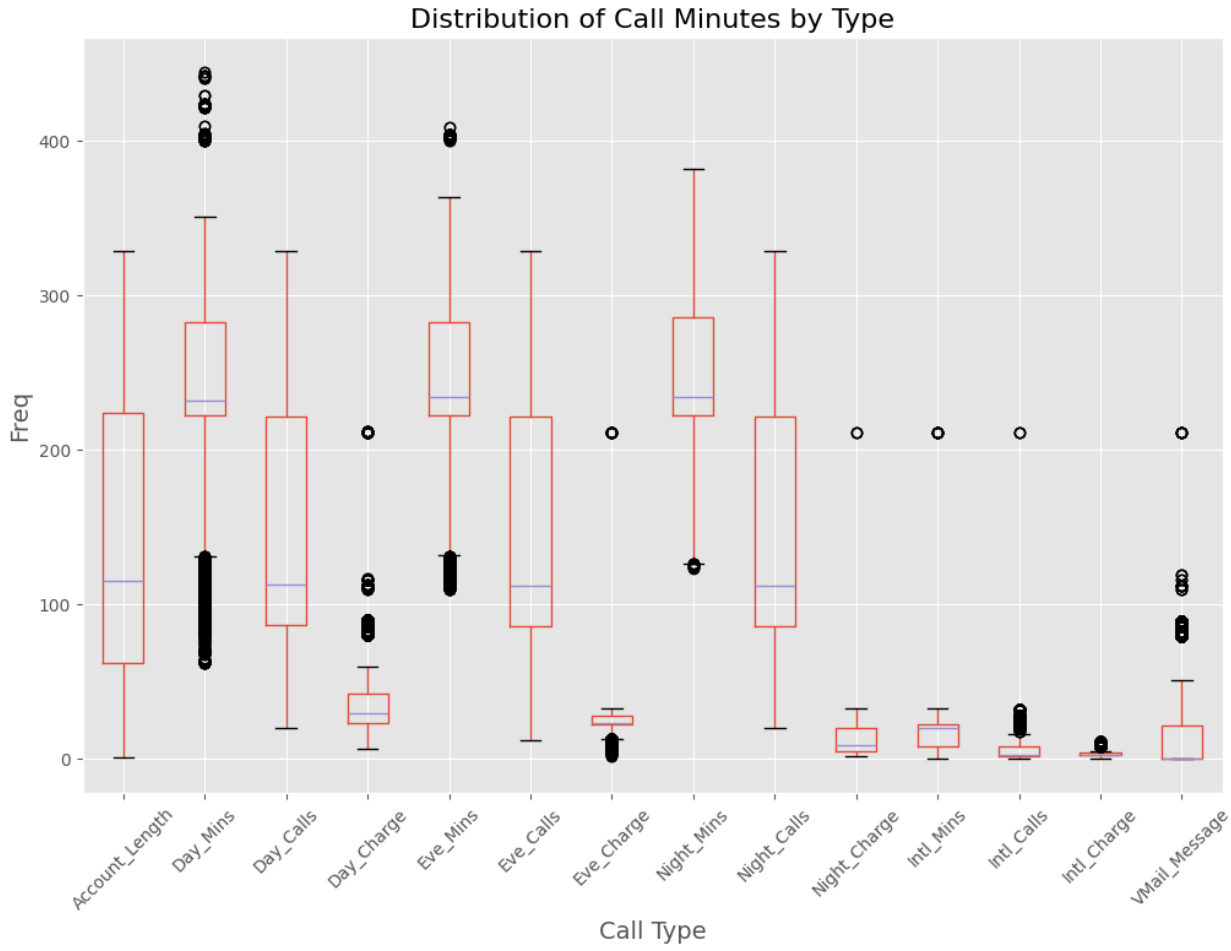
```
df = df[(df['Day_Charge'] < 1000) & (df['VMail_Message'] < 1000)]

df.shape
```

```
(36033, 17)
```

- from 60k rows to 36k rows after removing outliers, that's around 24k rows lost in outliers
- removed another 50ish rows by addressing the outliers in day_charege and vmail_message from previous box plot

```
fig, ax = plt.subplots(figsize=(12, 8))

# box plot
df.boxplot(column=numeric_cols,
           ax=ax)
```

```
# titles and labels
ax.set_title('Distribution of Call Minutes by Type', fontsize=16)
ax.set_xlabel('Call Type', fontsize=14)
ax.set_ylabel('Freq', fontsize=14)
ax.set_xticklabels(numeric_cols, rotation=45)

plt.show()
```



- looks cleaned up now (Note: can't address all outliers in every col as they could very well be potential extreme values given the context of that column)

Checking summary stats after removing outliers

```
df.describe()
```

```
       Account_Length  VMail_Message      Day_Mins     Day_Calls  \
count    36033.000000   36033.000000  36033.000000  36033.000000
mean       147.928205      11.216607    241.267724    149.078456
std         96.566663      22.171680     53.103504     86.367865
min          1.000000       0.000000     62.300000     20.000000
```

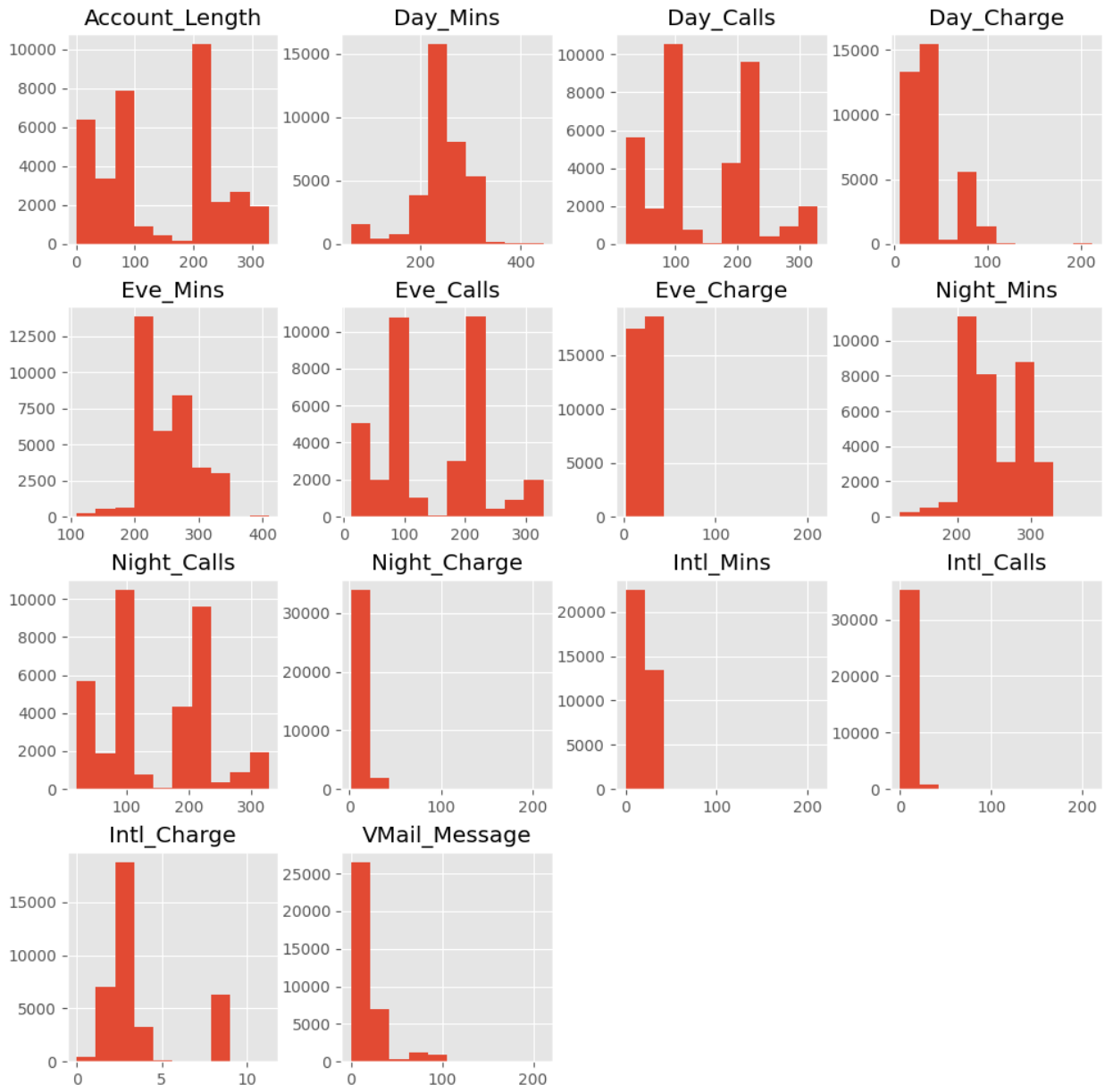
```
25%          62.000000       0.000000    222.200000     87.000000
50%         115.000000       0.000000    232.200000    113.000000
75%         224.000000      22.000000    282.800000    222.000000
max         329.000000     211.000000    444.400000    329.000000

          Day_Charge      Eve_Mins     Eve_Calls    Eve_Charge
Night_Mins  \
count  36033.000000  36033.000000  36033.000000  36033.000000
36033.000000
mean      39.768770    249.978860    148.892904     24.370483
250.316727
std       24.035258     40.496675     86.248911      6.932231
39.543004
min        6.220000    109.600000     12.000000      2.110000
123.500000
25%       23.230000    222.400000     86.000000     22.220000
222.400000
50%       29.240000    234.400000    112.000000     23.220000
234.400000
75%       42.420000    282.900000    222.000000     28.280000
286.200000
max      211.990000    409.200000    329.000000    211.260000
381.900000

         Night_Calls  Night_Charge     Intl_Mins    Intl_Calls
Intl_Charge  \
count  36033.000000  36033.000000  36033.000000  36033.000000
36033.000000
mean     148.205673     11.650730     15.632100      4.838315
3.703050
std       86.024933      8.339419      9.964079      4.686556
2.290302
min       20.000000      2.000000      0.000000      0.000000
0.000000
25%       86.000000      4.920000      8.200000      2.000000
2.280000
50%      112.000000      9.200000     20.000000      3.000000
2.820000
75%      222.000000     20.290000     22.300000      8.000000
4.000000
max      329.000000    211.220000    211.200000    211.000000
11.290000

       CustServ_Calls
count    36033.000000
mean         2.181084
std          2.014803
min          0.000000
25%          2.000000
50%          2.000000
```

```
75%           2.000000
max          11.000000

df

      Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls
\
0         382-4657             128             25     265.1        110

1         371-7191             107             26     161.6        123

2         358-1921             137              0     243.4        114

3         330-6626              75              0     166.7        113

4         391-8027             118              0     223.4         98

...            ...             ...            ...       ...        ...

36072     796-5759             228              0     222.8         92

36073     779-7579              62              0     228.8         92

36074     789-9756             222              0     228.2         60

36075     798-5885              88              0     282.2        222

36076     999-9897             228              0     222.0         99


       Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
0           45.07     197.4         99       16.78       244.7
91
1           27.47     195.5        103       16.62       254.4
103
2           41.38     121.2        110       10.30       162.6
104
3           28.34     148.3        122       12.61       186.9
121
4           37.98     220.6        101       18.75       203.9
118
...           ...       ...        ...         ...         ...
...
36072       29.82     280.0        202       22.28       268.6
229
36073       29.88     292.8         92       26.88       222.9
89
36074       22.82     229.8        289       28.26       222.8
222
36075       82.88     208.8        220       22.82       282.2
```

```
200
36076       88.08     220.2          80       22.92       282.9
28

       Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge
CustServ_Calls  Churn
0             11.01       10.0           3         2.70
1  False
1             11.45       13.7           3         3.70
1  False
2              7.32       12.2           5         3.29
0  False
3              8.41       10.1           3         2.73
3  False
4              9.18        6.3           6         1.70
0  False
...             ...        ...         ...          ...               ..
.    ...
36072         22.09        9.6           2         2.89
2  False
36073          9.62       20.2           2         2.82
8  False
36074          2.28        6.2           2         2.62
2  False
36075         20.68        9.8           8         2.82
8  False
36076         20.22        2.2           8         0.82
0  False

[36033 rows x 17 columns]
```

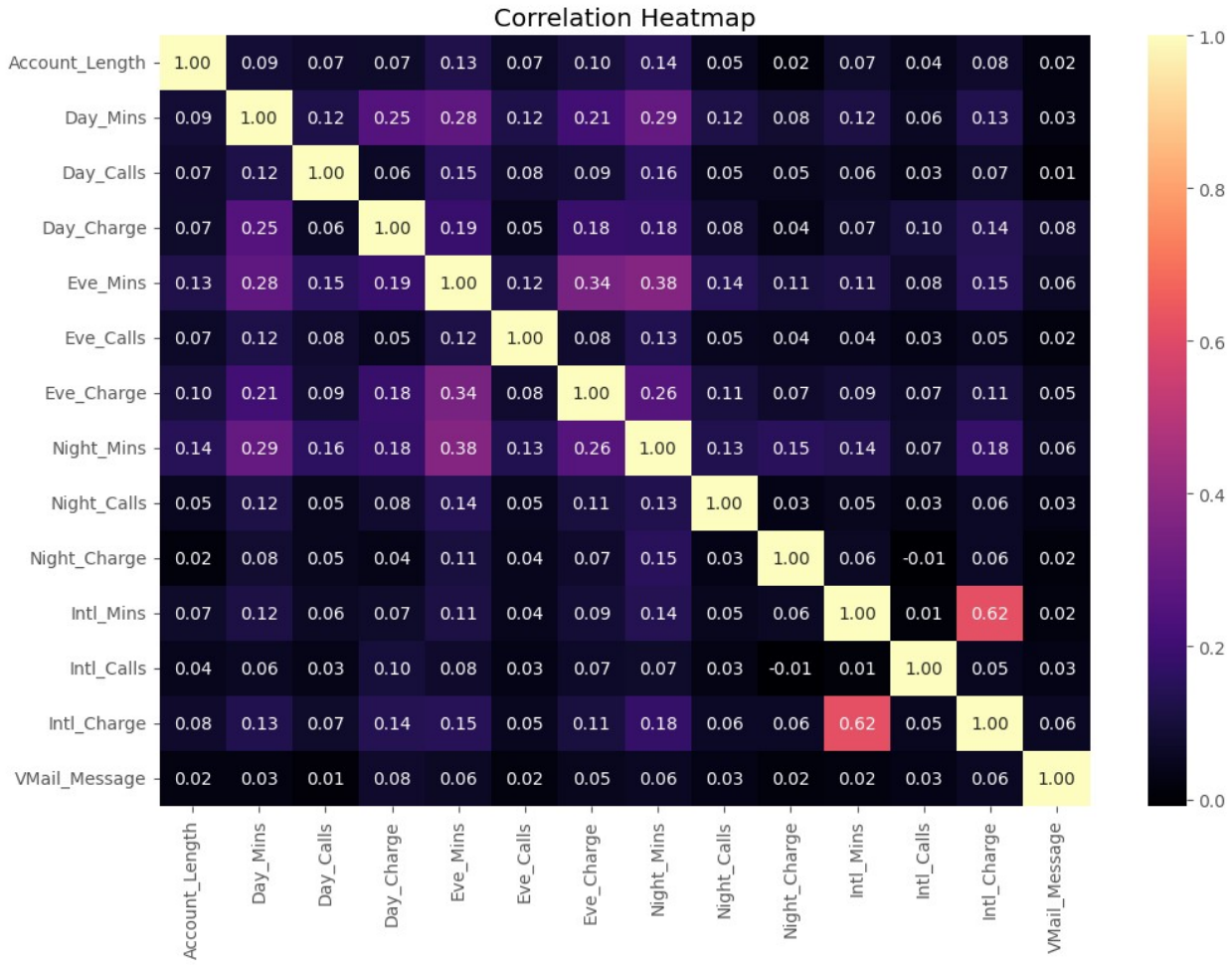
Other visualizations

```
# histograms
df[numeric_cols].hist(figsize=(12, 12))
plt.show()
```



- those columns which have been included for outlier removal has less skewed distribution now

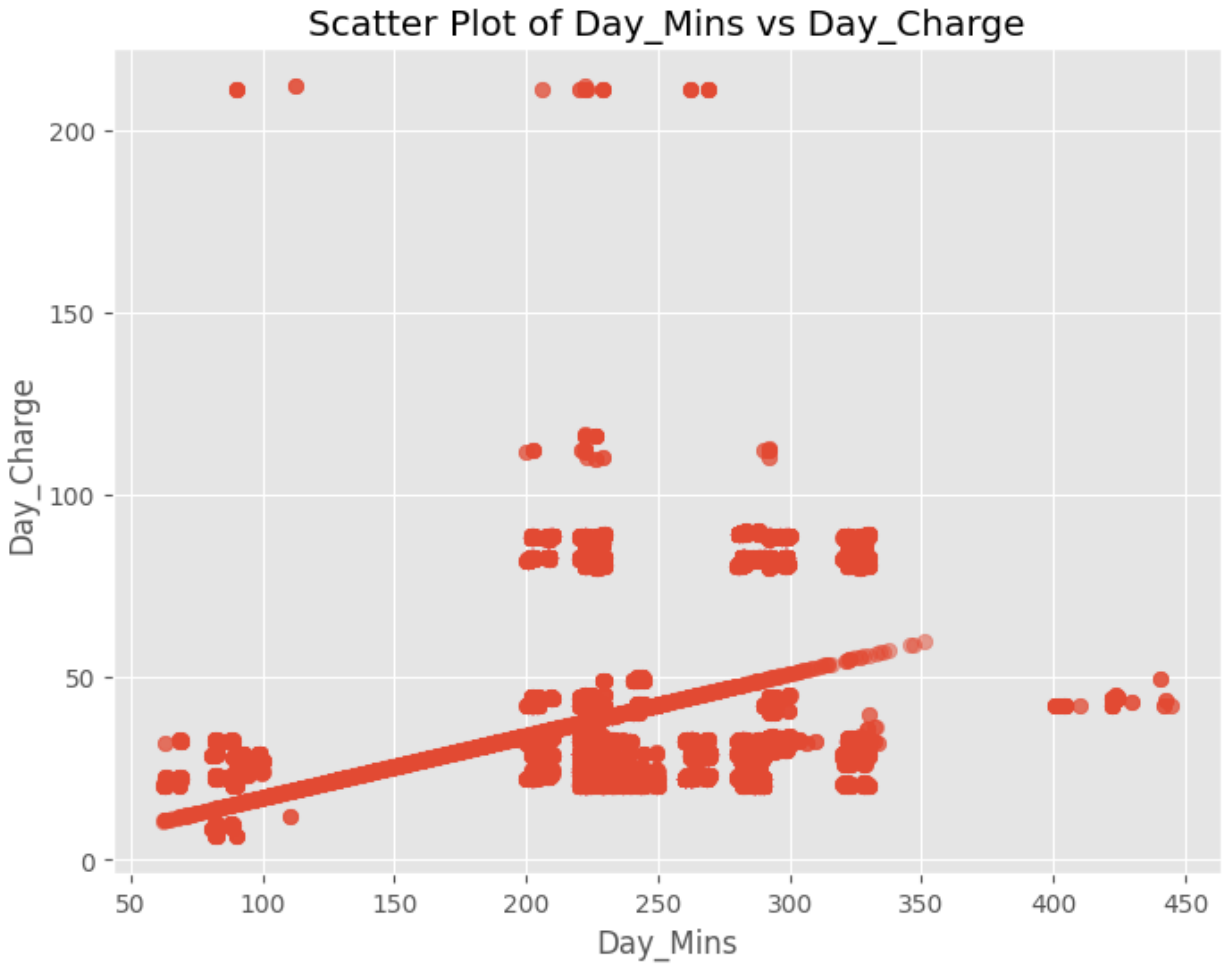
```
# correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df[numeric_cols].corr(), annot=True, fmt='.2f',
cmap='magma')
plt.title('Correlation Heatmap')
plt.show()
```

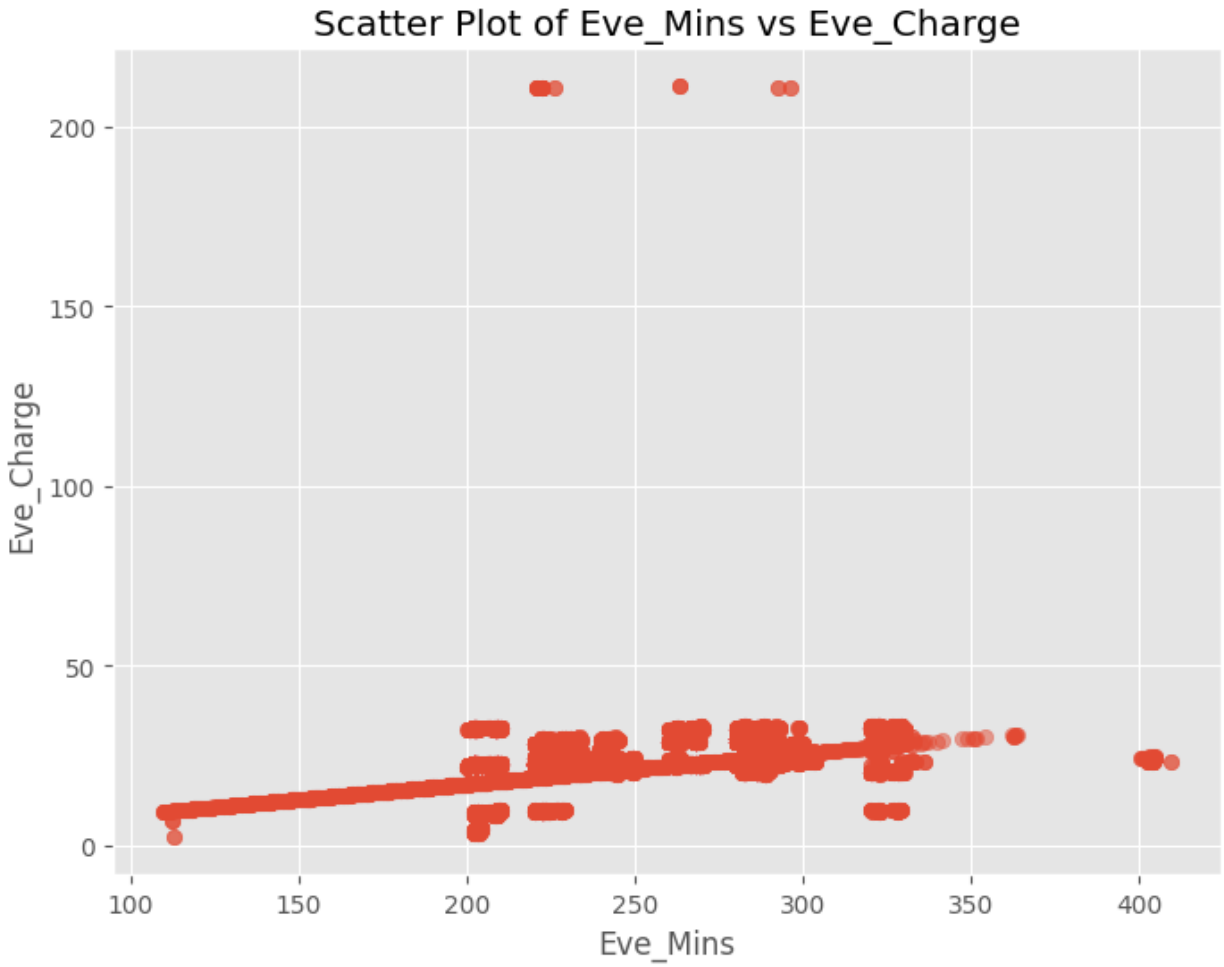


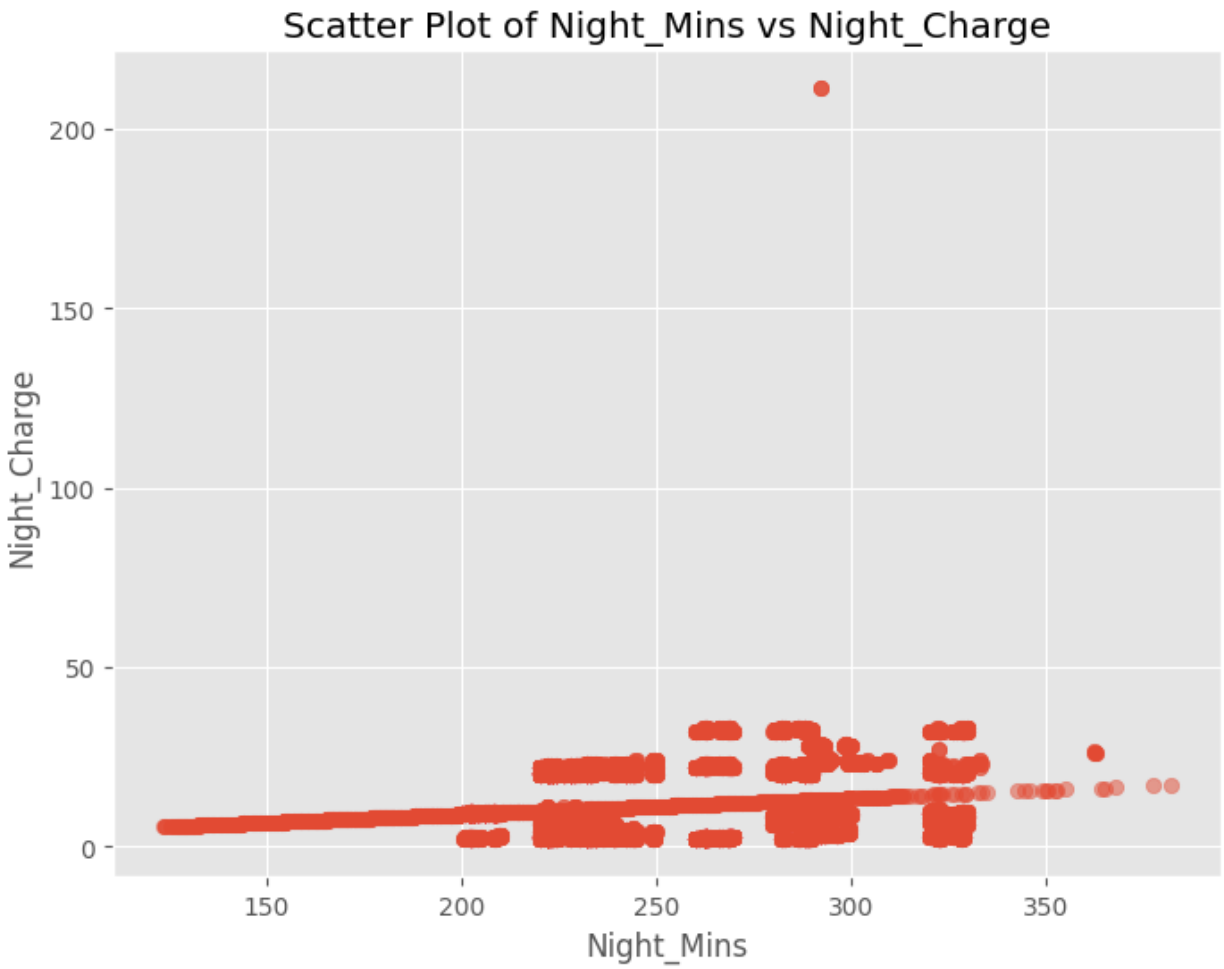
- High correlation between intl_mins and intl_charge which is apparent and expected
- day_mins have 0.25 corr with day_charge
- eve_mins have 0.34 corr with eve_charge
- night_mins have 0.15 corr with night_charge (which is relatively low)
- corr of intl_charge with day_mins, eve_mins & night_mins 0.15 <= corr < 0.20 indicates that these calls are talked around those timeframe?

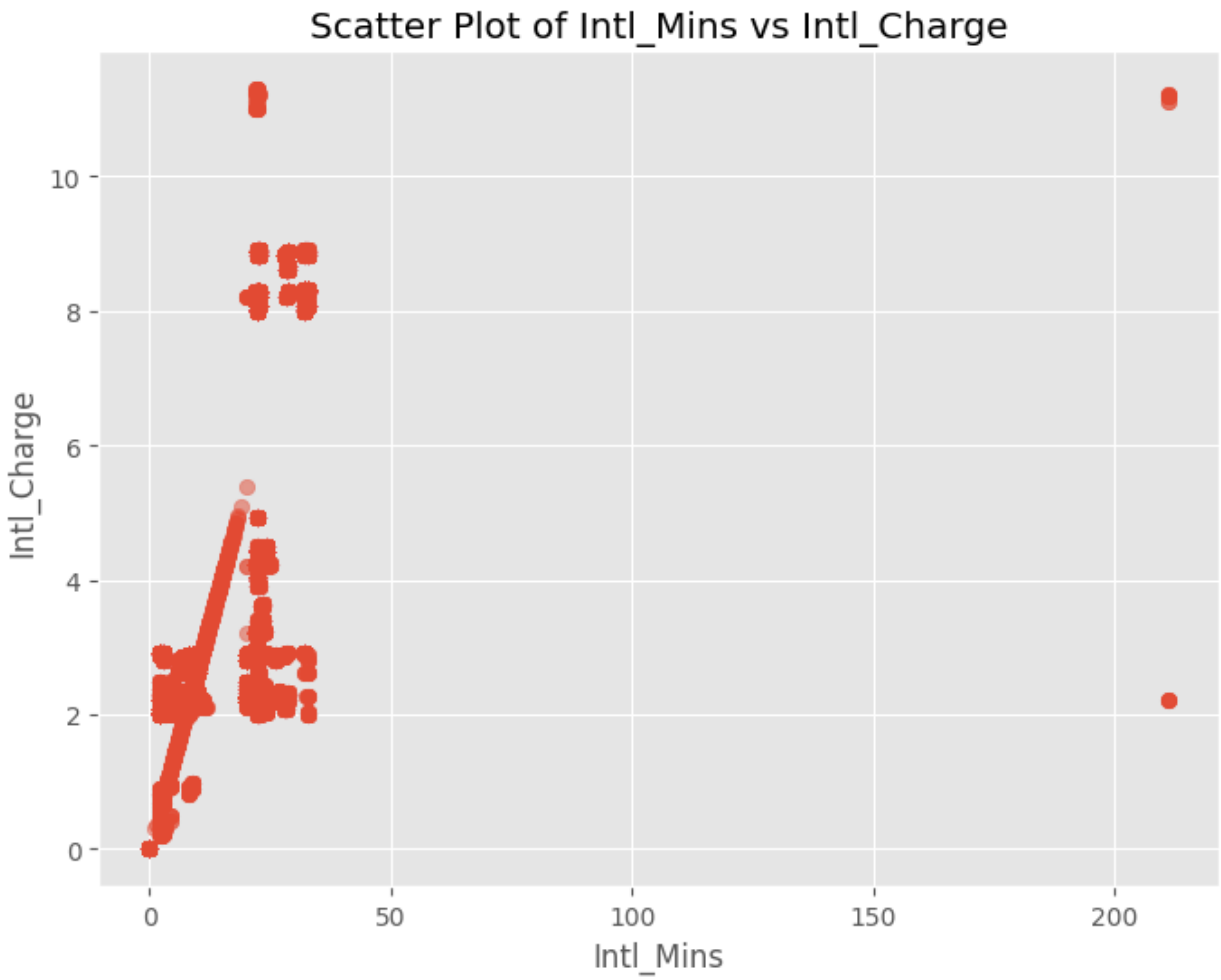
```
# List of feature pairs
feature_pairs = [
    ('Day_Mins', 'Day_Charge'),
    ('Eve_Mins', 'Eve_Charge'),
    ('Night_Mins', 'Night_Charge'),
    ('Intl_Mins', 'Intl_Charge'),
    # ...
]

# scatter plots for each pair
for x, y in feature_pairs:
    plt.figure(figsize=(8, 6))
    plt.scatter(df[x], df[y], alpha=0.5)
```

```
    plt.xlabel(x)
    plt.ylabel(y)
    plt.title(f'Scatter Plot of {x} vs {y}')
    plt.show()
```




Scatter Plot of Eve_Mins vs Eve_Charge
Eve_Charge
Eve_Mins
0
50
100
150
200
100
150
200
250
300
350
400

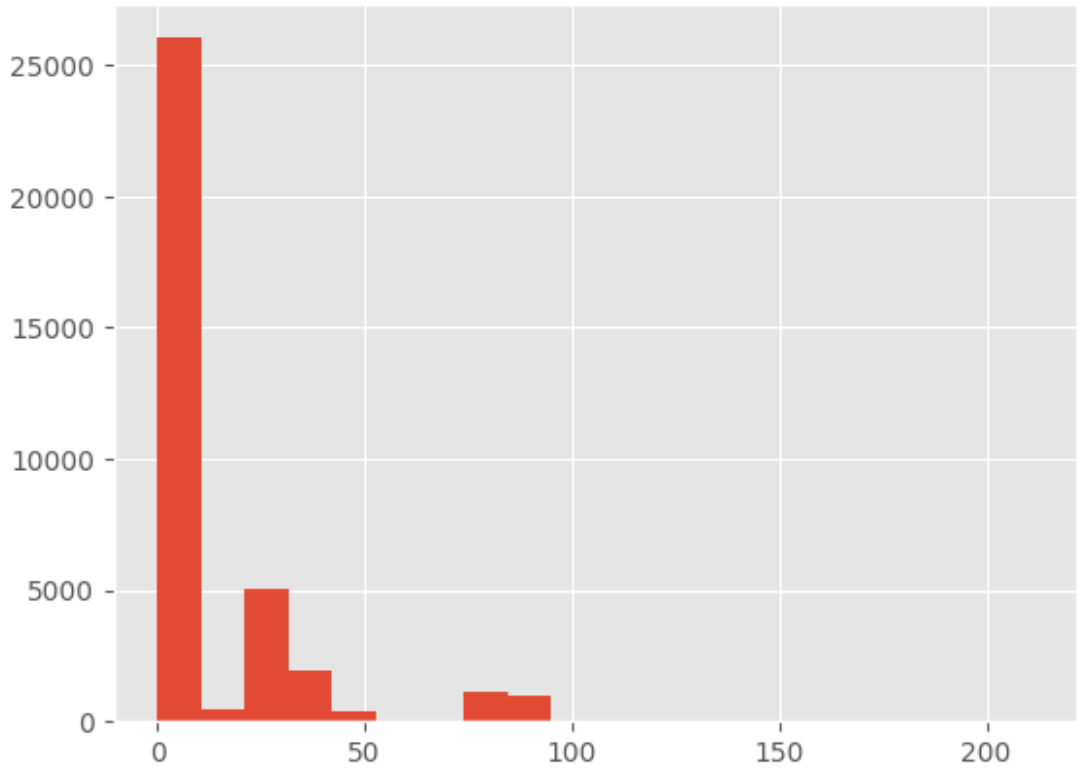

Scatter Plot of Night_Mins vs Night_Charge
Night_Charge
Night_Mins
200
150
100
50
0
150
200
250
300
350



- Shows somewhat linear relationship indicating as mins increases so does price (charge) across day, eve, night & intl

```
plt.hist(df['VMail_Message'], bins=20)
```

```
(array([2.6028e+04, 5.0000e+02, 5.0710e+03, 1.9090e+03, 3.9100e+02,
        0.0000e+00, 0.0000e+00, 1.1630e+03, 9.4700e+02, 0.0000e+00,
        1.3000e+01, 3.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 8.0000e+00]),
 array([  0.  ,  10.55,  21.1 ,  31.65,  42.2 ,  52.75,  63.3 ,
73.85,
         84.4 ,  94.95, 105.5 , 116.05, 126.6 , 137.15, 147.7 ,
158.25,
        168.8 , 179.35, 189.9 , 200.45, 211.  ]),
 <BarContainer object of 20 artists>)
```



- just checking the vmailmessage feature for anything unusual

```
# # pairplot
# sns.set()
# sns.pairplot(df[numeric_cols],size = 2 ,kind
='scatter',diag_kind='kde')
# plt.show()
```

- bunch of pair plots for all numeric features
- some sort of groups are observed in the above viz indicating potential for clustering

```
# df.to_csv("cleaned_churn.csv")

df.describe()
```

```
       Account_Length  VMail_Message      Day_Mins     Day_Calls  \
count    36033.000000   36033.000000  36033.000000  36033.000000
mean       147.928205      11.216607    241.267724    149.078456
std         96.566663      22.171680     53.103504     86.367865
min          1.000000       0.000000     62.300000     20.000000
25%         62.000000       0.000000    222.200000     87.000000
50%        115.000000       0.000000    232.200000    113.000000
75%        224.000000      22.000000    282.800000    222.000000
max        329.000000     211.000000    444.400000    329.000000

         Day_Charge      Eve_Mins     Eve_Calls    Eve_Charge
Night_Mins  \
```

```
count  36033.000000  36033.000000  36033.000000  36033.000000
36033.000000
mean      39.768770    249.978860    148.892904     24.370483
250.316727
std       24.035258     40.496675     86.248911      6.932231
39.543004
min        6.220000    109.600000     12.000000      2.110000
123.500000
25%       23.230000    222.400000     86.000000     22.220000
222.400000
50%       29.240000    234.400000    112.000000     23.220000
234.400000
75%       42.420000    282.900000    222.000000     28.280000
286.200000
max      211.990000    409.200000    329.000000    211.260000
381.900000

        Night_Calls  Night_Charge     Intl_Mins    Intl_Calls
Intl_Charge  \
count  36033.000000  36033.000000  36033.000000  36033.000000
36033.000000
mean     148.205673     11.650730     15.632100      4.838315
3.703050
std       86.024933      8.339419      9.964079      4.686556
2.290302
min       20.000000      2.000000      0.000000      0.000000
0.000000
25%       86.000000      4.920000      8.200000      2.000000
2.280000
50%      112.000000      9.200000     20.000000      3.000000
2.820000
75%      222.000000     20.290000     22.300000      8.000000
4.000000
max      329.000000    211.220000    211.200000    211.000000
11.290000

       CustServ_Calls
count    36033.000000
mean         2.181084
std          2.014803
min          0.000000
25%          2.000000
50%          2.000000
75%          2.000000
max         11.000000
```

Clustering for whole dataset

- clustering be performed on only the numerical features so specifying the numeric_cols below to work with instead of dropping the phone_number and churn feature

Standardizing data

- for performing clustering features need to be standardized so they have the same scale

```
from sklearn.preprocessing import StandardScaler
import pandas as pd

#creating a copy of the df and using that for normalizing
df_standardized = df.copy()

scaler = StandardScaler()

#redifining numeric cols cuz added 2 new aggregate features
numeric_cols = ['Account_Length', 'VMail_Message', 'Day_Mins',
'Day_Calls', 'Day_Charge',
                    'Eve_Mins', 'Eve_Calls', 'Eve_Charge',
'Night_Mins', 'Night_Calls',
                    'Night_Charge', 'Intl_Mins', 'Intl_Calls',
'Intl_Charge', 'CustServ_Calls'
                ]

df_standardized[numeric_cols] =
scaler.fit_transform(df_standardized[numeric_cols])


print(df_standardized.describe())
```

```
       Account_Length  VMail_Message      Day_Mins     Day_Calls  \
count    3.603300e+04   3.603300e+04  3.603300e+04  3.603300e+04
mean     7.020043e-17  -5.205875e-17 -2.520117e-16 -3.470583e-17
std      1.000014e+00   1.000014e+00  1.000014e+00  1.000014e+00
min     -1.521542e+00  -5.059050e-01 -3.370215e+00 -1.494540e+00
25%     -8.898454e-01  -5.059050e-01 -3.590721e-01 -7.187780e-01
50%     -3.409941e-01  -5.059050e-01 -1.707580e-01 -4.177359e-01
75%      7.877755e-01   4.863656e-01  7.821113e-01  8.443251e-01
max      1.875122e+00   9.010872e+00  3.825267e+00  2.083229e+00

         Day_Charge      Eve_Mins     Eve_Calls    Eve_Charge
Night_Mins  \
count  3.603300e+04  3.603300e+04  3.603300e+04  3.603300e+04
3.603300e+04
mean  -1.135827e-16  2.464903e-16 -5.048121e-17  3.013097e-16 -
7.465698e-16
std    1.000014e+00  1.000014e+00  1.000014e+00  1.000014e+00
1.000014e+00
min   -1.395834e+00 -3.466477e+00 -1.587206e+00 -3.211202e+00 -
```

```
3.207103e+00
25%   -6.881141e-01 -6.810249e-01 -7.292124e-01 -3.102194e-01 -
7.059938e-01
50%   -4.380613e-01 -3.847001e-01 -4.277551e-01 -1.659637e-01 -
4.025225e-01
75%    1.103074e-01  8.129457e-01  8.476410e-01  5.639702e-01
9.074619e-01
max    7.165458e+00  3.931764e+00  2.088254e+00  2.695988e+01
3.327645e+00

        Night_Calls  Night_Charge     Intl_Mins    Intl_Calls
Intl_Charge  \
count  3.603300e+04  3.603300e+04  3.603300e+04  3.603300e+04
3.603300e+04
mean  -1.064838e-16 -1.404009e-16  7.138358e-17  5.994644e-17 -
1.577538e-16
std    1.000014e+00  1.000014e+00  1.000014e+00  1.000014e+00
1.000014e+00
min   -1.490352e+00 -1.157258e+00 -1.568867e+00 -1.032396e+00 -
1.616862e+00
25%   -7.231222e-01 -8.071093e-01 -7.458996e-01 -6.056375e-01 -
6.213458e-01
50%   -4.208800e-01 -2.938770e-01  4.383707e-01 -3.922582e-01 -
3.855657e-01
75%    8.578368e-01  1.035970e+00  6.692031e-01  6.746380e-01
1.296574e-01
max    2.101679e+00  2.393117e+01  1.962756e+01  4.399063e+01
3.312688e+00

       CustServ_Calls
count    3.603300e+04
mean     9.149719e-17
std      1.000014e+00
min     -1.082545e+00
25%     -8.987803e-02
50%     -8.987803e-02
75%     -8.987803e-02
max      4.377122e+00

df_standardized.head()

  Phone_Number  Account_Length  VMail_Message  Day_Mins  Day_Calls  \
0     382-4657       -0.206370       0.621675  0.448795  -0.452472
1     371-7191       -0.423840       0.666778 -1.500256  -0.301950
2     358-1921       -0.113169      -0.505905  0.040154  -0.406157
3     330-6626       -0.755221      -0.505905 -1.404215  -0.417736
4     391-8027       -0.309927      -0.505905 -0.336474  -0.591414

   Day_Charge  Eve_Mins  Eve_Calls  Eve_Charge  Night_Mins
Night_Calls  \
```

```
0    0.220564 -1.298368  -0.578484   -1.094970   -0.142043    -
0.664999
1   -0.511704 -1.345286  -0.532106   -1.118051    0.103263    -
0.525502
2    0.067037 -3.180030  -0.450944   -2.029747   -2.218292    -
0.513878
3   -0.475507 -2.510830  -0.311810   -1.696517   -1.603763    -
0.316258
4   -0.074424 -0.725474  -0.555295   -0.810787   -1.173845    -
0.351132

   Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  CustServ_Calls
Churn
0     -0.076833  -0.565248   -0.392258    -0.437961       -0.586211
False
1     -0.024070  -0.193909   -0.392258    -0.001332       -0.586211
False
2     -0.519316  -0.344452    0.034500    -0.180350       -1.082545
False
3     -0.388609  -0.555212   -0.392258    -0.424862        0.406455
False
4     -0.296275  -0.936587    0.247879    -0.874591       -1.082545
False
```

- in case there's need for standardized features

Inference

DR first then clustering gave better cluster results and also was computationally less expensive (time taken to run)

Splitting data on churn

```
# Split the dataset
churn_true = df[df['Churn'] == True]
churn_false = df[df['Churn'] == False]

churn_true_std = churn_true.copy()
churn_false_std = churn_false.copy()

# Standardize the split datasets
scaler = StandardScaler()
churn_true_std[numeric_cols]
=scaler.fit_transform(churn_true[numeric_cols])
```

```
churn_false_std[numeric_cols]
=scaler.fit_transform(churn_false[numeric_cols])

churn_true.shape

(4153, 17)

churn_true.describe()

       Account_Length  VMail_Message     Day_Mins    Day_Calls
Day_Charge  \
count     4153.000000    4153.000000  4153.000000  4153.000000
4153.000000
mean       151.982422       6.745967   251.937334   155.497231
35.101958
std         97.249881      17.699462    52.198994    88.118811
19.722774
min          1.000000       0.000000    62.400000    22.000000
9.890000
25%         66.000000       0.000000   223.300000    88.000000
22.960000
50%        130.000000       0.000000   243.200000   151.000000
28.620000
75%        226.000000       0.000000   288.200000   222.000000
33.220000
max        329.000000      88.000000   444.400000   329.000000
112.011000

          Eve_Mins    Eve_Calls   Eve_Charge   Night_Mins  Night_Calls
\
count  4153.000000  4153.000000  4153.000000  4153.000000  4153.000000

mean    251.668505   149.374187    24.665134   251.186162   148.935228

std      41.799930    87.896389     8.168051    40.990749    88.108880

min     109.600000    20.000000     3.220000   123.500000    20.000000

25%     222.800000    86.000000    22.220000   222.300000    82.000000

50%     236.300000   111.000000    23.230000   236.200000   116.000000

75%     286.000000   222.000000    28.620000   286.600000   222.000000

max     404.200000   329.000000   211.211000   354.900000   329.000000


       Night_Charge    Intl_Mins   Intl_Calls  Intl_Charge
CustServ_Calls
count   4153.000000  4153.000000  4153.000000  4153.000000
4153.000000
```

```
mean      12.134317    16.643142     4.644835     3.853919
2.460149
std        8.327518     9.133902     4.392286     2.354785
2.222761
min        2.020000     2.000000     1.000000     0.320000
0.000000
25%        6.320000     8.900000     2.000000     2.280000
2.000000
50%        9.240000    20.200000     3.000000     2.860000
2.000000
75%       20.360000    22.600000     6.000000     3.920000
2.000000
max       32.980000    32.900000    32.000000     8.920000
9.000000
```

```
churn_false.shape
```

```
(31880, 17)
```

```
churn_false.describe()
```

```
       Account_Length  VMail_Message      Day_Mins     Day_Calls  \
count    31880.000000   31880.000000  31880.000000  31880.000000
mean       147.400063      11.798996    239.877796    148.242284
std         96.466295      22.624759     53.063296     86.103335
min          1.000000       0.000000     62.300000     20.000000
25%         62.000000       0.000000    222.200000     86.000000
50%        112.000000       0.000000    230.400000    110.000000
75%        224.000000      22.000000    282.800000    222.000000
max        329.000000     211.000000    444.400000    329.000000

         Day_Charge      Eve_Mins     Eve_Calls    Eve_Charge
Night_Mins  \
count  31880.000000  31880.000000  31880.000000  31880.000000
31880.000000
mean      40.376714    249.758751    148.830207     24.332099
250.203466
std       24.476160     40.319261     86.033175      6.753818
39.349735
min        6.220000    109.600000     12.000000      2.110000
123.800000
25%       23.240000    222.400000     87.000000     22.220000
222.400000
50%       29.320000    234.400000    112.000000     23.220000
234.400000
75%       42.940000    282.900000    222.000000     28.280000
286.200000
max      211.990000    409.200000    329.000000    211.260000
381.900000
```

```
        Night_Calls  Night_Charge     Intl_Mins    Intl_Calls
Intl_Charge  \
count  31880.000000  31880.000000  31880.000000  31880.000000
31880.000000
mean     148.110634     11.587733     15.500392      4.863519
3.683396
std       85.750684      8.339034     10.059843      4.723023
2.281071
min       20.000000      2.000000      0.000000      0.000000
0.000000
25%       86.000000      4.440000      8.200000      2.000000
2.280000
50%      111.000000      9.090000     20.000000      3.000000
2.820000
75%      222.000000     20.280000     22.300000      8.000000
4.000000
max      329.000000    211.220000    211.200000    211.000000
11.290000

       CustServ_Calls
count    31880.000000
mean         2.144730
std          1.983259
min          0.000000
25%          2.000000
50%          2.000000
75%          2.000000
max         11.000000
```

Final clustering choice

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score

# # PCA
# pca = PCA(n_components=2)
# pca_results = pca.fit_transform(df_standardized[numeric_cols])

# for n_clusters in range(2, 15):
#     kmeans = KMeans(n_clusters=n_clusters,
n_init='auto' ,random_state=42)
#     cluster_labels = kmeans.fit_predict(pca_results)
#     silhouette_avg = silhouette_score(pca_results, cluster_labels)
#     print(f"For n_clusters = {n_clusters}, the average
silhouette_score is : {silhouette_avg}")
```

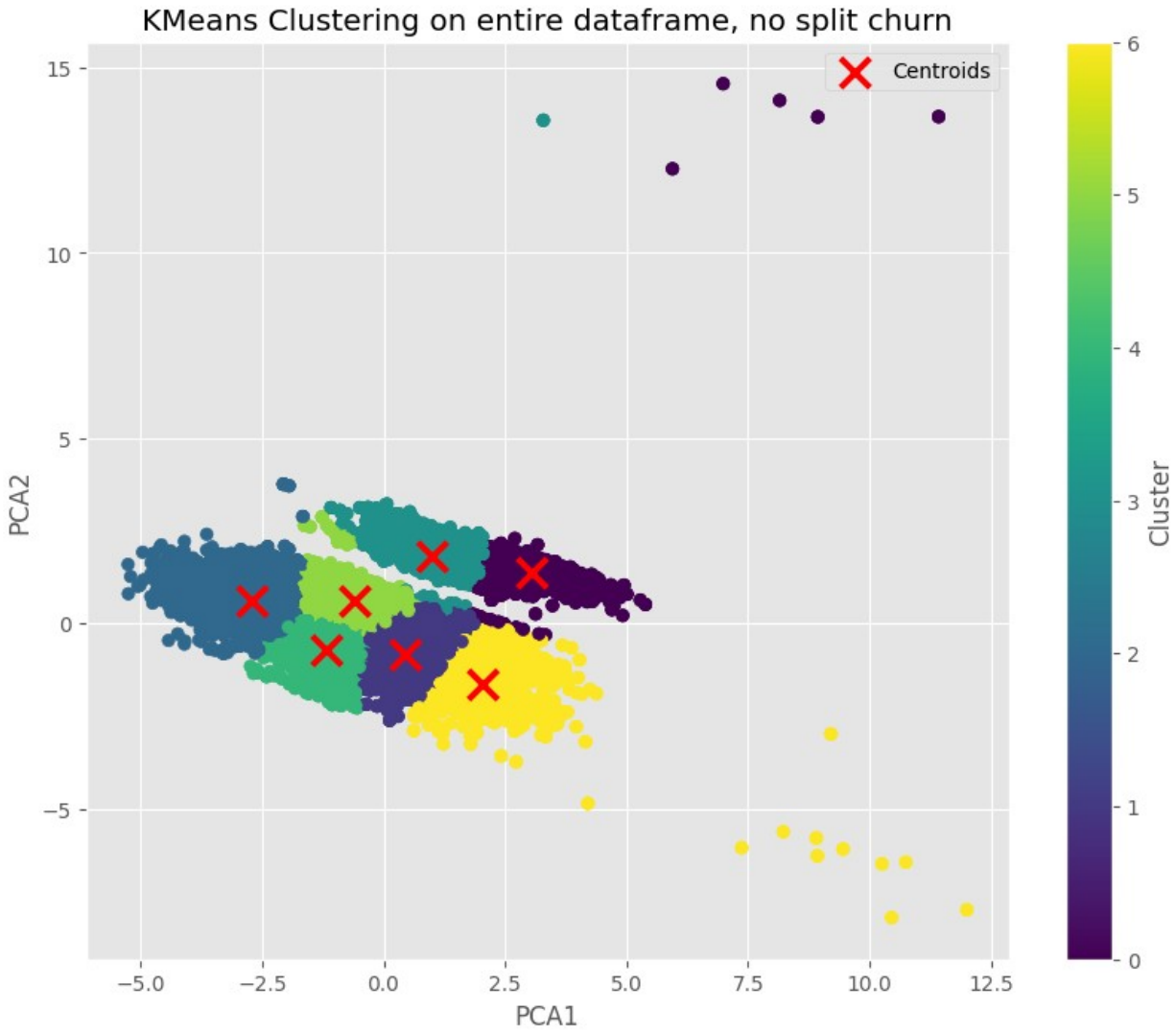
```
# n_clusters = 25
# kmeans = KMeans(n_clusters=n_clusters,
n_init='auto' ,random_state=42)
# cluster_labels = kmeans.fit_predict(pca_results)
# silhouette_avg = silhouette_score(pca_results, cluster_labels)
# print(f"For n_clusters = {n_clusters}, the average silhouette_score
is : {silhouette_avg}")
```

max score for n=3 and second max for n=7

Even though for n=3 we get max silhouette score, n=7 being the second max and not too far behind and 7 clusters we choose n = 7 as the optimal cluster

Clustering on entire df

```
# PCA remains the same
pca = PCA(n_components=2)
pca_results = pca.fit_transform(df_standardized[numeric_cols])

# KMeans Clustering
optimal_clusters = 7
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42,
n_init='auto')
cluster_labels = kmeans.fit_predict(pca_results)

df['Cluster'] = cluster_labels

# Create the scatter plot
plt.figure(figsize=(10, 8))
scatter = plt.scatter(pca_results[:, 0], pca_results[:, 1],
c=cluster_labels, cmap='viridis')

# Add title and labels
plt.title('KMeans Clustering on entire dataframe, no split churn')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

# Add a color bar
plt.colorbar(scatter, label='Cluster')

# Plot the centroids
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x',
s=200, linewidths=3, label='Centroids')
plt.legend()

# Show the plot
plt.show()
```




```
df.head()
```

	Phone_Number	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls
0	382-4657	128	25	265.1	110	45.07	197.4	99	16.78	244.7	91
1	371-7191	107	26	161.6	123	27.47	195.5	103	16.62	254.4	103
2	358-1921	137	0	243.4	114	41.38	121.2	110	10.30	162.6	104
3	330-6626	75	0	166.7	113						
4	391-8027	118	0	223.4	98						

```
3       28.34     148.3         122        12.61        186.9
121
4       37.98     220.6         101        18.75        203.9
118

   Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  CustServ_Calls
Churn  \
0         11.01       10.0           3         2.70                1
False
1         11.45       13.7           3         3.70                1
False
2          7.32       12.2           5         3.29                0
False
3          8.41       10.1           3         2.73                3
False
4          9.18        6.3           6         1.70                0
False

   Cluster
0        4
1        2
2        2
3        2
4        4
```

Grouping for statistical Inference

```
grouped_df = df.groupby('Cluster')

pd.set_option('display.max_columns', None)

grouped_df.describe()

        Account_Length
\
                 count        mean         std  min   25%    50%
75%
Cluster

0               2670.0  190.165543  109.691476  8.0  86.0  206.0
288.0
1               6767.0  161.753953   94.436602  2.0  82.0  202.0
228.0
2               2893.0   99.923263   55.478863  1.0  62.0   95.0
129.0
3               3707.0  151.396547   93.565367  2.0  82.0   99.0
226.0
4               7919.0  128.392979   88.991905  1.0  37.5   93.0
222.0
5               8429.0  137.426860   90.778095  2.0  40.0  112.0
```

```
223.0
6               3648.0  192.583882  112.367301  2.0  82.0  208.0
320.0

               VMail_Message
\
          max         count       mean        std  min  25%  50%
75%
Cluster

0        329.0        2670.0  16.066667  29.479665  0.0  0.0  0.0
28.0
1        329.0        6767.0  13.340624  25.703538  0.0  0.0  0.0
22.0
2        293.0        2893.0   8.258209  15.028283  0.0  0.0  0.0
19.0
3        329.0        3707.0  13.097923  26.819536  0.0  0.0  0.0
22.0
4        329.0        7919.0   8.698952  16.567330  0.0  0.0  0.0
22.0
5        329.0        8429.0   8.613477  15.659019  0.0  0.0  0.0
22.0
6        329.0        3648.0  15.641173  28.937398  0.0  0.0  0.0
28.0

               Day_Mins
\
          max    count        mean         std   min    25%    50%
75%
Cluster

0        211.0   2670.0  287.517528  40.903766  62.6  280.8  289.8
322.6
1         89.0   6767.0  255.406753  47.473965  62.6  222.8  262.3
288.2
2        112.0   2893.0  171.455762  53.608295  62.3  130.5  175.3
220.0
3        211.0   3707.0  241.288608  53.672935  62.6  222.2  229.0
282.8
4         89.0   7919.0  228.106572  39.388587  62.6  222.2  226.2
242.2
5         89.0   8429.0  229.512509  38.287945  62.6  222.2  228.8
242.2
6        211.0   3648.0  292.262971  37.326168  82.2  282.2  292.2
322.8

               Day_Calls
\
          max     count        mean         std   min   25%    50%
75%
```

```
Cluster

0        329.9    2670.0  187.386142  101.152225  20.0  90.0  202.0
288.0
1        444.4    6767.0  161.119994   81.059578  20.0  89.0  202.0
222.0
2        326.3    2893.0  100.823021   41.552893  20.0  82.0   98.0
115.0
3        423.4    3707.0  150.449150   79.791487  20.0  88.0  119.0
222.0
4        402.2    7919.0  133.774088   79.286419  20.0  68.0   98.0
220.0
5        422.2    8429.0  136.679203   81.341637  20.0  49.0   99.0
222.0
6        442.4    3648.0  197.451206  104.143567  20.0  92.0  206.0
322.0

                Day_Charge
\
           max      count       mean         std    min    25%    50%
75%
Cluster

0        329.0     2670.0  56.603306  29.487126  20.02  28.88  32.92
82.88
1        329.0     6767.0  47.221502  28.257681   8.28  26.68  32.08
82.26
2        322.0     2893.0  28.391086   8.287882   8.60  22.78  27.10
33.33
3        328.0     3707.0  42.745465  27.288372   6.28  22.82  28.66
82.06
4        322.0     7919.0  31.645354  14.236718   6.22  22.60  27.59
33.63
5        326.0     8429.0  30.087404  11.240431   6.22  22.62  26.23
33.22
6        329.0     3648.0  59.624505  29.498028   6.32  29.62  80.32
86.06

                  Eve_Mins
\
             max    count        mean        std    min    25%    50%
75%
Cluster

0        211.9900   2670.0  291.408528  33.864482  200.0  282.2  289.2
322.6
1        211.9900   6767.0  265.968135  35.378677  200.2  228.8  280.2
289.2
2         88.9200   2893.0  196.881334  36.936306  109.6  169.7  202.6
222.4
```

```
3        112.1111   3707.0  248.811586  34.122617  112.2  222.2  228.9
282.8
4        116.1120   7919.0  237.139530  26.154503  150.6  222.3  229.3
244.0
5        112.1190   8429.0  234.514716  24.033846  140.9  222.2  228.8
242.2
6        211.2600   3648.0  296.893202  29.587042  200.2  282.6  292.2
322.8

               Eve_Calls
\
           max     count        mean         std   min    25%    50%
75%
Cluster

0        329.9    2670.0  177.798876  104.249346  20.0  89.0  202.0
282.0
1        409.2    6767.0  163.380080   81.295053  20.0  90.0  202.0
222.0
2        313.7    2893.0  100.548220   41.590195  12.0  82.0   96.0
113.0
3        329.2    3707.0  146.998381   79.720088  20.0  88.0   98.0
222.0
4        404.4    7919.0  138.279707   79.696746  20.0  69.0  107.0
222.0
5        404.2    8429.0  136.050540   81.131575  20.0  49.0   99.0
222.0
6        404.0    3648.0  193.839364  104.616135  20.0  90.0  206.0
320.0

               Eve_Charge
\
           max      count       mean        std   min    25%    50%
75%
Cluster

0        329.0     2670.0  28.564801   3.926159  9.29  26.98  28.82
32.26
1        329.0     6767.0  25.811352   3.913686  8.20  22.62  26.29
28.82
2        286.0     2893.0  16.567829   4.689084  2.11  13.57  16.69
20.24
3        329.0     3707.0  24.727983   4.646431  6.99  22.22  22.90
28.29
4        328.0     7919.0  23.141091   3.396943  3.20  22.22  22.62
24.22
5        329.0     8429.0  23.062169   3.281707  3.22  22.22  22.68
24.22
6        329.0     3648.0  30.144047  15.493139  9.28  28.08  28.88
32.29
```

```
                Night_Mins
\
            max      count        mean        std    min    25%    50%
75%
Cluster

0         32.99     2670.0  293.223966  31.349103  200.0  282.2  289.2
322.6
1         32.99     6767.0  261.823696  34.978387  160.6  226.9  268.6
288.9
2         29.62     2893.0  205.232745  37.732954  123.5  178.0  208.2
229.1
3         32.98     3707.0  252.294516  34.055291  200.0  222.6  229.8
286.8
4         32.92     7919.0  234.785041  26.212524  127.1  222.2  228.2
242.4
5         32.88     8429.0  236.404997  25.146768  126.7  222.3  229.4
242.9
6        211.26     3648.0  297.170817  28.672688  200.2  282.6  292.2
322.6

               Night_Calls
\
           max       count        mean         std   min   25%    50%
75%
Cluster

0        329.9      2670.0  184.371161  101.091386  20.0  90.0  202.0
282.0
1        381.9      6767.0  159.462243   82.503956  20.0  88.0  202.0
222.0
2        349.7      2893.0   98.905980   41.615927  20.0  81.0   97.0
113.0
3        329.6      3707.0  147.529539   80.934640  20.0  86.0   98.0
222.0
4        364.9      7919.0  135.356611   79.551046  20.0  69.0   99.0
220.0
5        377.5      8429.0  139.051845   81.070436  20.0  62.0  115.0
222.0
6        329.9      3648.0  189.681743  103.223687  20.0  90.0  202.0
320.0

               Night_Charge
\
           max        count       mean        std   min   25%   50%
75%
Cluster

0        329.0       2670.0  14.969779  11.976593  2.00  8.09  9.28
```

```
20.8975
1        329.0       6767.0  12.391111   8.236189  2.00  6.28  9.08
20.8800
2        229.0       2893.0   8.776308   3.585159  2.03  6.98  8.77
10.0400
3        329.0       3707.0  11.828621   7.441563  2.00  6.96  9.02
20.2800
4        328.0       7919.0  10.675600   7.423966  2.00  3.92  9.22
20.2200
5        328.0       8429.0  10.556600   7.800309  2.00  3.33  9.09
20.2400
6        329.0       3648.0  14.591714  10.019894  2.00  8.20  9.28
20.8900

                Intl_Mins
\
            max     count       mean        std   min   25%   50%
75%
Cluster

0        211.22    2670.0  30.817783  12.663504  20.0  28.2  32.2
32.6
1         32.98    6767.0  12.409137   7.546977   0.0   8.0   9.2
20.2
2         24.42    2893.0  11.090183   4.843993   0.0   8.6  10.3
12.7
3         32.88    3707.0  24.839118   6.360640  22.0  22.2  22.8
28.2
4         32.66    7919.0   5.822240   3.038186   0.0   3.0   4.9
9.2
5         32.08    8429.0  21.924985   2.168091   9.2  20.6  22.2
22.6
6         32.98    3648.0  11.496935   7.683575   0.0   6.8   9.0
20.2

                Intl_Calls
\
            max      count      mean       std  min  25%  50%  75%
max
Cluster

0        211.20     2670.0  5.709363  4.509781  2.0  2.0  6.0  8.0
32.0
1         32.90     6767.0  5.720556  4.935268  0.0  2.0  4.0  8.0
32.0
2         28.20     2893.0  4.032838  2.345936  0.0  2.0  3.0  5.0
20.0
3        211.11     3707.0  5.297275  4.059740  2.0  2.0  6.0  8.0
32.0
4         13.00     7919.0  4.128804  3.897471  0.0  2.0  3.0  4.0
```

```
28.0
5         32.80     8429.0  3.801400  3.325406  1.0  2.0  3.0  4.0
26.0
6         32.90     3648.0  6.672697  8.222489  0.0  2.0  8.0  8.0
211.0

        Intl_Charge
\
               count      mean       std   min   25%   50%   75%
max
Cluster

0            2670.0  8.328269  0.901330  2.22  8.22  8.29  8.82
11.290
1            6767.0  2.546058  0.454120  0.00  2.26  2.44  2.86
4.920
2            2893.0  2.763510  0.823518  0.00  2.30  2.70  3.19
8.860
3            3707.0  8.369879  0.712545  2.00  8.22  8.28  8.82
11.290
4            7919.0  2.305038  0.463517  0.00  2.22  2.29  2.44
3.510
5            8429.0  3.273748  0.921212  2.00  2.42  3.22  4.04
8.290
6            3648.0  2.493514  0.499847  0.00  2.26  2.32  2.82
11.011

        CustServ_Calls
                 count      mean       std  min  25%  50%  75%   max
Cluster
0               2670.0  2.731086  2.638797  0.0  2.0  2.0  2.0  11.0
1               6767.0  2.449387  2.305849  0.0  2.0  2.0  2.0  11.0
2               2893.0  1.547183  1.311299  0.0  1.0  1.0  2.0   9.0
3               3707.0  2.387645  2.377260  0.0  2.0  2.0  2.0  11.0
4               7919.0  1.897967  1.439596  0.0  2.0  2.0  2.0  11.0
5               8429.0  1.861075  1.370771  0.0  2.0  2.0  2.0   9.0
6               3648.0  2.927632  2.741034  0.0  2.0  2.0  2.0  11.0
```

Cluster Analysis for entire df

Cluster 0: Premium High Usage Customers

Characteristics:

- Highest average day minutes: 287.5
- Highest average day charges: 56.6
- Highest average evening minutes: 291.4
- Highest average evening charges: 28.6
- Highest average night minutes: 293.2

- Highest average night charges: 15.0
- High international usage: 30.8 minutes, 5.7 calls
- Above average customer service calls: 2.7

Reason: These customers are heavy users across all time periods, with the highest usage and charges. They likely subscribe to premium plans due to their high usage patterns.

Cluster 1: Moderate Daytime and Evening Users

Characteristics:

- Second-highest day minutes: 255.4
- Second-highest day charges: 47.2
- High evening usage: 266.0 minutes
- Moderate night and international usage
- Above average customer service calls: 2.4

Reason: These customers have significant usage during day and evening hours, suggesting they might be working professionals or businesses.

Cluster 2: Economy Users

Characteristics:

- Lowest usage across all categories (day, evening, night, international)
- Shortest average account length: 99.9 days
- Lowest customer service calls: 1.5

Reason: These customers use their phones minimally, possibly new customers or those on economy plans.

Cluster 3: Night-time International Callers

Characteristics:

- Moderate day and evening usage
- High night minutes: 252.3
- Highest international minutes: 24.8
- Average international calls: 5.3
- Above average customer service calls: 2.4

Reason: These customers have a preference for night-time calls and make longer international calls, possibly due to time zone differences or cheaper night rates.

Cluster 4: Average Balanced Users

Characteristics:

- Usage close to average in most categories
- Slightly below average in day, evening, and night minutes
- Low international usage: 5.8 minutes, 4.1 calls

- Below average customer service calls: 1.9

Reason: These customers represent the typical or average user, with balanced usage across different times of day and services.

Cluster 5: Frequent Short International Callers

Characteristics:

- Slightly below average usage in day, evening, and night minutes
- Highest number of international calls: 21.9
- Low international minutes: 3.3
- Lowest average international charge: 3.3
- Below average customer service calls: 1.9

Reason: These customers make frequent but very short international calls, possibly for quick check-ins or business purposes.

Cluster 6: Ultra-High Usage Customers

Characteristics:

- Highest average account length: 192.6 days
- Highest usage across day minutes: 292.3
- Highest usage across evening minutes: 296.9
- Highest usage across night minutes: 297.2
- Highest charges in all categories
- High international usage: 11.5 minutes, 6.7 calls
- Highest average customer service calls: 2.9

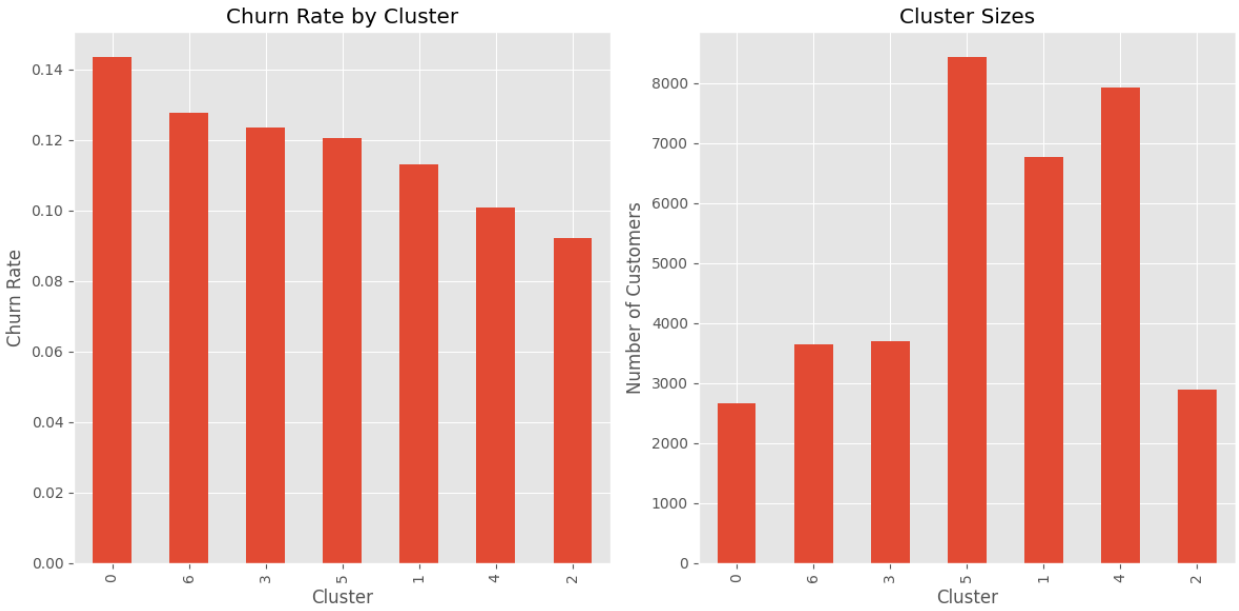
Reason: These are the most intensive users, with the highest usage and charges across all categories. They likely have complex needs and possibly use their phones for both personal and business purposes.

Validating churn rate in each cluster

```
churn_rates = df.groupby('Cluster')
['Churn'].mean().sort_values(ascending=False)

# Step 2: Calculate the number of customers in each cluster
cluster_sizes = df['Cluster'].value_counts().sort_index()

# Step 3: Create a summary dataframe
cluster_summary = pd.DataFrame({
    'Churn Rate': churn_rates,
    'Cluster Size': cluster_sizes
})

# Step 4: Sort by churn rate descending
cluster_summary = cluster_summary.sort_values('Churn Rate',
```

```
ascending=False)

# Step 5: Display the results
print(cluster_summary)

# Step 6: Visualize the results
plt.figure(figsize=(12, 6))

# Bar plot for churn rates
plt.subplot(1, 2, 1)
cluster_summary['Churn Rate'].plot(kind='bar')
plt.title('Churn Rate by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Churn Rate')

# Bar plot for cluster sizes
plt.subplot(1, 2, 2)
cluster_summary['Cluster Size'].plot(kind='bar')
plt.title('Cluster Sizes')
plt.xlabel('Cluster')
plt.ylabel('Number of Customers')

plt.tight_layout()
plt.show()
```

Cluster	Churn Rate	Cluster Size
0	0.143446	2670
6	0.127741	3648
3	0.123550	3707
5	0.120536	8429
1	0.113049	6767
4	0.100770	7919
2	0.092292	2893



Intuition behind churn rate calculation

df.groupby('Cluster'): This groups the dataframe by the 'Cluster' column, creating separate groups for each cluster. ['Churn']: This selects the 'Churn' column within each group. .mean(): This calculates the mean of the 'Churn' column for each cluster group.

The 'Churn' column is typically binary, where:

1 (or True) represents a churned customer 0 (or False) represents a retained customer

When you take the mean of a binary column, you get the proportion of 1s (True values). In this case, it gives you the proportion of churned customers in each cluster. For example:

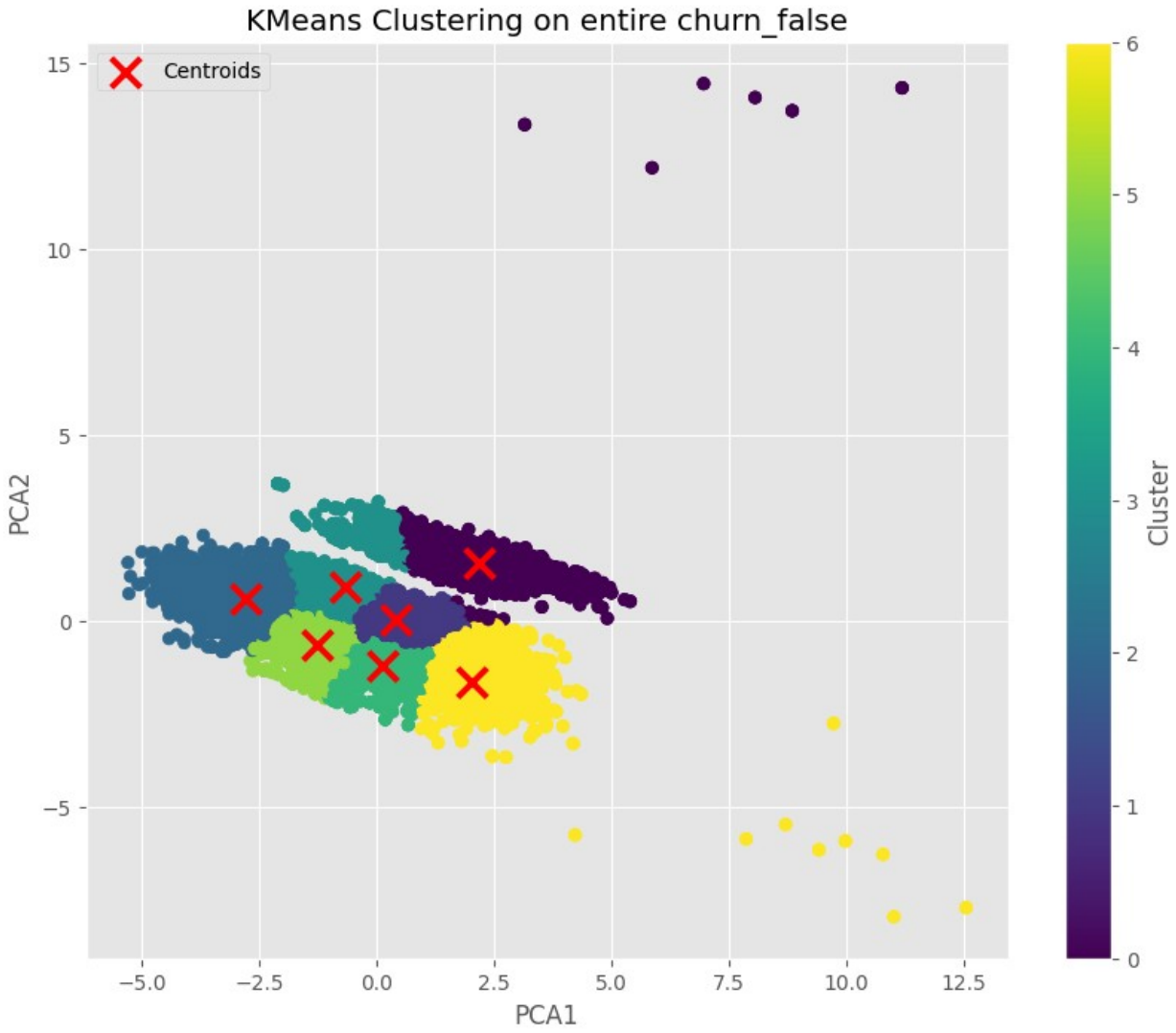
If a cluster has 100 customers and 20 of them churned, the churn rate would be 20/100 = 0.20 or 20%. If another cluster has 50 customers and 15 of them churned, the churn rate would be 15/50 = 0.30 or 30%.

Note :- Churn column is in 'True' or 'False' not '0' or '1' but python treats them as 0 or 1 so no need to convert them into numeric

```
# PCA
pca = PCA(n_components=2)
pca_results = pca.fit_transform(churn_false_std[numeric_cols])

# KMeans Clustering
optimal_clusters = 7
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42,
n_init='auto')
cluster_labels = kmeans.fit_predict(pca_results)

churn_false['Cluster'] = cluster_labels

# Create the scatter plot
```

```
plt.figure(figsize=(10, 8))
scatter = plt.scatter(pca_results[:, 0], pca_results[:, 1],
c=cluster_labels, cmap='viridis')

# Add title and labels
plt.title('KMeans Clustering on entire churn_false')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

# Add a color bar
plt.colorbar(scatter, label='Cluster')

# Plot the centroids
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x',
s=200, linewidths=3, label='Centroids')
plt.legend()

# Show the plot
plt.show()
```

```
C:\Users\tshre\AppData\Local\Temp\ipykernel_14632\2982463046.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  churn_false['Cluster'] = cluster_labels
```




```
churn_false.head()
```

	Phone_Number	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls
0	382-4657	128	25	265.1	110	45.07	197.4	99	16.78	244.7	91
1	371-7191	107	26	161.6	123	27.47	195.5	103	16.62	254.4	103
2	358-1921	137	0	243.4	114	41.38	121.2	110	10.30	162.6	104
3	330-6626	75	0	166.7	113						
4	391-8027	118	0	223.4	98						

```
3       28.34     148.3         122        12.61        186.9
121
4       37.98     220.6         101        18.75        203.9
118

   Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  CustServ_Calls
Churn  \
0         11.01       10.0           3         2.70               1
False
1         11.45       13.7           3         3.70               1
False
2          7.32       12.2           5         3.29               0
False
3          8.41       10.1           3         2.73               3
False
4          9.18        6.3           6         1.70               0
False

   Cluster
0        5
1        2
2        2
3        2
4        5
```

Grouping for statistical Inference

```
grouped_df = churn_false.groupby('Cluster')

grouped_df.describe()

        Account_Length
\
                 count        mean         std  min   25%    50%
75%
Cluster

0               4687.0  170.136121  103.630690  2.0  82.0  202.0
269.0
1               3703.0  163.206589   90.595874  2.0  82.0  206.0
228.0
2               2423.0   99.465126   54.530065  1.0  62.0   95.0
128.0
3               6610.0  132.830560   91.361710  2.0  36.0   94.0
222.0
4               4692.0  156.548167   93.166200  2.0  82.0  202.0
228.0
5               6407.0  125.675199   89.113411  1.0  34.0   92.0
222.0
6               3358.0  190.170637  112.006268  2.0  82.0  208.0
```

```
290.0

                VMail_Message
\
            max          count       mean        std  min  25%  50%
75%
Cluster

0         329.0         4687.0  15.769575  29.488958  0.0  0.0  0.0
22.0
1         329.0         3703.0  12.571429  22.710287  0.0  0.0  0.0
22.0
2         293.0         2423.0   8.421378  15.121347  0.0  0.0  0.0
20.0
3         328.0         6610.0   8.752345  16.698122  0.0  0.0  0.0
22.0
4         329.0         4692.0  13.397698  24.859391  0.0  0.0  0.0
22.0
5         329.0         6407.0   9.088185  16.743138  0.0  0.0  0.0
22.0
6         329.0         3358.0  16.777844  29.569225  0.0  0.0  0.0
28.0

                Day_Mins
\
            max    count        mean        std   min    25%    50%
75%
Cluster

0         211.0   4687.0  267.135790  47.662354  62.6  228.2  282.2
292.60
1          89.0   3703.0  250.543335  44.222523  62.8  222.8  239.3
288.20
2         112.0   2423.0  170.643879  52.252275  62.3  131.7  174.6
215.65
3          89.0   6610.0  222.678720  43.113211  62.3  222.2  224.4
239.30
4          89.0   4692.0  248.924175  45.868605  62.6  222.4  242.2
286.90
5          89.0   6407.0  224.638449  40.049955  62.8  222.0  224.2
239.40
6         211.0   3358.0  290.318490  38.385718  82.2  282.2  290.6
322.60

                Day_Calls
\
            max     count        mean         std   min   25%    50%
75%
Cluster
```

```
0        329.9    4687.0  169.612545   92.218871  20.0  88.0  202.0
222.0
1        444.4    3703.0  159.624359   79.881369  20.0  89.0  202.0
222.0
2        313.8    2423.0  100.880726   40.050374  20.0  83.0   98.0
115.0
3        422.2    6610.0  131.076551   80.812914  20.0  44.0   96.0
220.0
4        423.4    4692.0  156.702046   80.622233  20.0  89.0  202.0
222.0
5        326.0    6407.0  130.955049   79.339172  20.0  62.0   97.0
220.0
6        442.4    3358.0  194.989875  102.713625  20.0  92.0  206.0
320.0

                Day_Charge
\
           max      count       mean        std   min    25%    50%
75%
Cluster

0        329.0     4687.0  52.854962  29.818750  6.28  28.22  32.32
82.88
1        329.0     3703.0  40.684673  22.876622  8.82  24.24  29.99
43.98
2        322.0     2423.0  28.365944   8.093909  8.60  22.82  27.22
33.45
3        322.0     6610.0  29.451832  11.339065  6.22  22.44  24.94
32.33
4        329.0     4692.0  47.152292  28.398045  8.28  26.22  32.20
82.22
5        322.0     6407.0  30.659761  12.480257  6.22  22.44  26.82
33.32
6        329.0     3358.0  61.864250  29.721728  6.32  29.88  80.92
86.28

                 Eve_Mins
\
             max    count        mean        std    min    25%    50%
75%
Cluster

0        211.990   4687.0  272.138937  38.841474  112.2  228.8  282.6
292.8
1        211.112   3703.0  257.890494  34.615644  200.2  226.3  260.8
288.8
2         88.920   2423.0  195.381057  37.217690  109.6  167.6  202.2
222.3
3        112.119   6610.0  232.040372  21.876419  142.3  222.2  226.3
239.2
```

```
4        211.990   4692.0  262.495315  35.464466  200.2  228.2  268.2
288.8
5        116.112   6407.0  234.293638  24.027539  142.3  222.2  228.6
242.3
6        211.260   3358.0  295.379020  30.328555  200.2  282.2  292.0
322.8

                Eve_Calls
\
           max     count        mean         std   min   25%    50%
75%
Cluster

0        329.9    4687.0  164.310646   93.453860  20.0  88.0  200.0
222.0
1        404.4    3703.0  160.506886   78.467169  20.0  90.0  202.0
222.0
2        313.7    2423.0  100.268675   40.201136  12.0  83.0   97.0
113.0
3        402.2    6610.0  132.791982   80.987536  20.0  49.0   96.0
222.0
4        409.2    4692.0  159.649616   82.155802  20.0  89.0  202.0
222.0
5        404.4    6407.0  135.258467   79.538016  20.0  63.0   99.0
220.0
6        404.0    3358.0  191.734068  103.319509  20.0  90.0  206.0
320.0

                Eve_Charge
\
           max      count       mean        std   min      25%    50%
75%
Cluster

0        329.0     4687.0  26.964639   4.274161  6.99  22.8400  28.28
29.08
1        329.0     3703.0  24.991920   3.496793  8.68  22.3200  24.04
28.28
2        286.0     2423.0  16.446971   4.624052  2.11  13.4450  16.52
20.00
3        328.0     6610.0  22.739278   3.670090  3.22  22.2200  22.44
24.02
4        329.0     4692.0  25.671937   3.858029  8.20  22.4375  26.22
28.82
5        328.0     6407.0  22.754951   3.395248  3.20  22.2000  22.42
23.63
6        329.0     3358.0  29.892087  14.533762  9.28  28.0200  28.88
32.28

                 Night_Mins
```

```
\
             max      count        mean        std    min     25%
50%
Cluster

0          32.99     4687.0  274.389917  37.488651  200.0  229.11
282.8
1          32.98     3703.0  255.144267  33.715583  193.9  224.20
244.2
2          28.28     2423.0  204.174094  38.084996  123.8  175.80
206.1
3          32.86     6610.0  234.137658  23.262432  146.2  222.20
228.8
4          32.99     4692.0  258.316240  34.822812  193.9  224.00
262.8
5          32.88     6407.0  234.087770  26.042444  127.1  222.20
228.2
6         211.26     3358.0  295.246534  29.831122  200.2  282.20
292.2

                        Night_Calls
\
             75%    max       count        mean         std   min   25%
50%
Cluster

0         292.80  329.9      4687.0  168.505654   92.612589  20.0  89.0
202.0
1         288.80  367.7      3703.0  159.406967   78.287660  20.0  90.0
202.0
2         228.85  349.7      2423.0   99.533636   40.296020  20.0  82.0
97.0
3         242.20  350.2      6610.0  130.319667   80.813622  20.0  44.0
94.0
4         288.20  381.9      4692.0  161.093990   81.539417  20.0  89.0
202.0
5         242.20  377.5      6407.0  132.328703   78.985803  20.0  66.0
98.0
6         322.20  329.9      3358.0  189.229005  102.085461  20.0  90.0
202.0

                        Night_Charge
\
           75%    max        count       mean        std   min   25%
50%
Cluster

0         222.0  329.0       4687.0  13.661177  10.207685  2.00  8.08
9.22
1         222.0  328.0       3703.0  11.769560   8.263408  2.02  4.22
```

```
8.98
2        113.0  229.0       2423.0   8.733752   3.395016  2.03  7.02
8.75
3        220.0  322.0       6610.0  10.231051   7.575194  2.00  3.32
9.02
4        222.0  329.0       4692.0  12.152755   8.044383  2.00  6.28
9.08
5        220.0  326.0       6407.0  10.595221   7.431409  2.00  3.62
9.22
6        289.0  329.0       3358.0  14.327241   9.856056  2.00  8.20
9.28

                        Intl_Mins
\
            75%     max     count       mean        std   min   25%
50%
Cluster

0        20.820  211.22    4687.0  27.949774  11.303227  20.0  22.6
28.200
1        20.680   32.88    3703.0  21.970132   2.604342   8.8  20.2
20.900
2        10.055   24.42    2423.0  10.844078   4.514926   0.0   8.5
10.200
3        20.220   32.08    6610.0  22.186263   1.998823  11.3  22.0
22.200
4        20.800   32.98    4692.0   6.773719   3.182439   0.0   3.2
8.200
5        20.220   32.66    6407.0   5.953445   3.169192   0.0   3.2
4.900
6        20.880   32.98    3358.0  11.597406   7.579848   0.0   6.9
9.155

                       Intl_Calls
\
         75%    max      count      mean       std  min  25%  50%
75%
Cluster

0        32.2  211.2     4687.0  5.609558  4.228622  2.0  2.0  6.0
8.0
1        22.4   32.9     3703.0  5.092628  4.489595  1.0  2.0  3.0
8.0
2        12.4   26.9     2423.0  4.085844  2.335252  0.0  2.0  4.0
5.0
3        22.8   32.8     6610.0  3.711649  2.997311  1.0  2.0  3.0
4.0
4         9.2   22.4     4692.0  5.694373  5.066678  0.0  2.0  4.0
8.0
5         9.2   20.4     6407.0  4.070080  3.738142  0.0  2.0  3.0
```

```
4.0
6        20.2   32.9     3358.0  6.751042  8.418526  0.0  2.0  8.0
8.0

               Intl_Charge
\
           max       count      mean       std   min   25%   50%   75%

Cluster

0          32.0      4687.0  8.397122  0.676725  2.22  8.22  8.28  8.82

1          28.0      3703.0  2.853568  0.675986  2.00  2.28  2.86  3.02

2          18.0      2423.0  2.713105  0.703471  0.00  2.28  2.70  3.16

3          28.0      6610.0  3.993527  1.894950  2.00  2.92  3.32  4.29

4          32.0      4692.0  2.401813  0.478920  0.00  2.22  2.32  2.82

5          28.0      6407.0  2.314716  0.451630  0.00  2.22  2.30  2.44

6         211.0      3358.0  2.510968  0.485775  0.00  2.28  2.62  2.82


                CustServ_Calls

             max          count      mean       std  min  25%  50%  75%
max
Cluster

0         11.290         4687.0  2.534884  2.504373  0.0  2.0  2.0  2.0
11.0
1          4.920         3703.0  2.139346  1.893012  0.0  2.0  2.0  2.0
11.0
2          5.100         2423.0  1.386711  1.125240  0.0  0.0  1.0  2.0
8.0
3         11.290         6610.0  1.824660  1.381622  0.0  2.0  2.0  2.0
11.0
4          3.050         4692.0  2.391304  2.193359  0.0  2.0  2.0  2.0
11.0
5          3.650         6407.0  1.858124  1.395548  0.0  2.0  2.0  2.0
11.0
6         11.011         3358.0  2.985408  2.785141  0.0  2.0  2.0  2.0
11.0
```

Cluster Analysis for churn_false

Cluster 0: High Usage Premium Customers

Characteristics:

- High average day minutes: 267.1
- High average day charges: 52.9
- High average evening minutes: 272.1
- High average evening charges: 27.0
- Highest average night minutes: 274.4
- Highest average night charges: 13.7
- High international usage: 28.0 minutes, 5.6 calls
- Above average customer service calls: 2.5

Reason: These customers are heavy users across all time periods, with high usage and charges. They likely subscribe to premium plans.

Cluster 1: Moderate Business Users

Characteristics:

- Above average day minutes: 250.5
- Above average day charges: 40.7
- High evening usage: 257.9 minutes
- High international minutes: 22.0
- Average international calls: 5.1
- Slightly above average customer service calls: 2.1

Reason: These customers have significant daytime and evening usage, with long international calls, suggesting business use.

Cluster 2: Economy Users

Characteristics:

- Lowest usage across all categories
- Shortest average account length: 99.5 days
- Lowest customer service calls: 1.4

Reason: These customers use their phones minimally, possibly new customers or those on economy plans.

Cluster 3: Average Users with High International Usage

Characteristics:

- Average usage in day, evening, and night minutes
- Highest international minutes: 22.2
- Highest international calls: 3.7
- Below average customer service calls: 1.8

Reason: These customers have typical domestic usage but make frequent international calls.

Cluster 4: Moderate Daytime Users

Characteristics:

- Above average day minutes: 248.9
- Above average day charges: 47.2
- Moderate evening and night usage
- Low international usage: 6.8 minutes, 5.7 calls
- Average customer service calls: 2.4

Reason: These customers use their phones moderately during the day, possibly for work purposes.

Cluster 5: Basic Plan Users

Characteristics:

- Below average usage in all categories
- Low international usage: 6.0 minutes, 4.1 calls
- Below average customer service calls: 1.9

Reason: These customers have low overall usage, suggesting they might be on basic or entry-level plans.

Cluster 6: Ultra-High Usage Customers

Characteristics:

- Highest average account length: 190.2 days
- Highest usage across day minutes: 290.3
- Highest usage across evening minutes: 295.4
- Highest usage across night minutes: 295.2
- Highest charges in all categories
- High international usage: 11.6 minutes, 6.8 calls
- Highest average customer service calls: 3.0

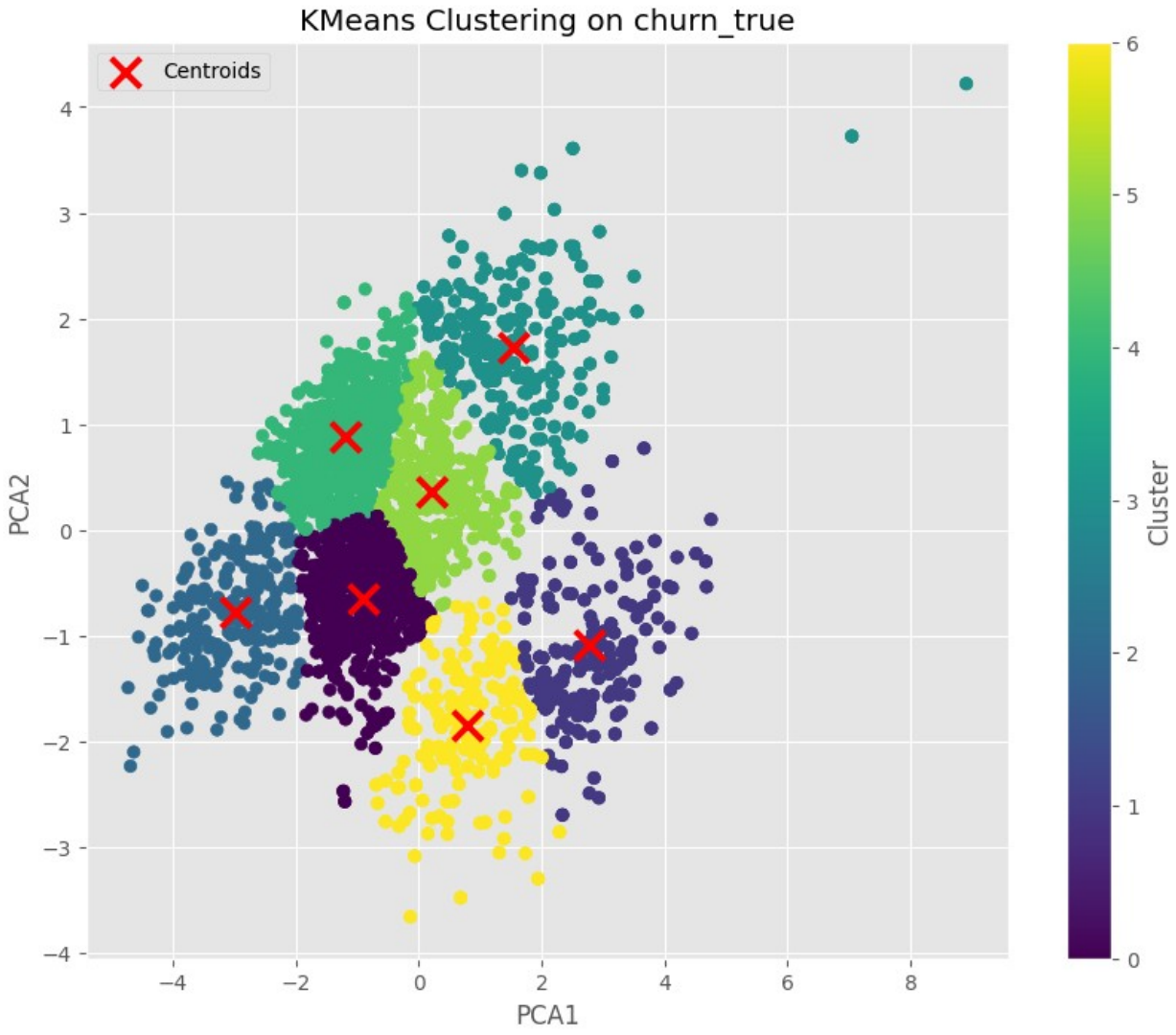
Reason: These are the most intensive users, with the highest usage and charges across all categories.

Clustering on churn_true

```
# PCA
pca = PCA(n_components=2)
pca_results = pca.fit_transform(churn_true_std[numeric_cols])

# KMeans Clustering
optimal_clusters = 7
kmeans = KMeans(n_clusters=optimal_clusters, random_state=42,
n_init='auto')
cluster_labels = kmeans.fit_predict(pca_results)
```

```
churn_true['Cluster'] = cluster_labels

# Create the scatter plot
plt.figure(figsize=(10, 8))
scatter = plt.scatter(pca_results[:, 0], pca_results[:, 1],
c=cluster_labels, cmap='viridis')

# Add title and labels
plt.title('KMeans Clustering on churn_true')
plt.xlabel('PCA1')
plt.ylabel('PCA2')

# Add a color bar
plt.colorbar(scatter, label='Cluster')

# Plot the centroids
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', marker='x',
s=200, linewidths=3, label='Centroids')
plt.legend()

# Show the plot
plt.show()
```

```
C:\Users\tshre\AppData\Local\Temp\ipykernel_14632\1388053657.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  churn_true['Cluster'] = cluster_labels
```




```
churn_true.head()
```

	Phone_Number	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls
8	329-6603	65	0	129.1	137	21.95	228.5	83	19.42	208.8	111
12	351-7269	161	0	332.9	67	56.59	317.8	97	27.01	160.6	128
17	393-7984	77	0	62.4	89	10.61	169.9	121	14.44	209.6	64
26	360-1596	12	0	249.6	118						
39	398-1294	119	0	159.1	114						

```
26       42.43     252.4         119       21.45       280.2
90
39       27.05     231.3         117       19.66       143.2
91

    Night_Charge  Intl_Mins  Intl_Calls  Intl_Charge  CustServ_Calls
Churn  \
8           9.40       12.7           6         3.43               4
True
12          7.23        5.4           9         1.46               4
True
17          9.43        5.7           6         1.54               5
True
26         12.61       11.8           3         3.19               1
True
39          6.44        8.8           3         2.38               5
True

    Cluster
8         2
12        4
17        2
26        4
39        2
```

Grouping for Statistical Inference

```
group_df = churn_true.groupby('Cluster')

group_df.describe()

        Account_Length
\
                count        mean         std   min    25%    50%
75%
Cluster

0               831.0  138.484958   87.786169   2.0  44.00  125.0
223.0
1               422.0  202.279621  107.194863  22.0  89.00  215.5
288.0
2               246.0  103.158537   46.614127   1.0  76.25  100.5
128.0
3               564.0  184.031915  114.209659   2.0  82.00  208.0
320.0
4               909.0  120.360836   83.170210   2.0  34.00   93.0
222.0
5               722.0  161.578947   94.701033   2.0  63.25  208.0
229.0
6               459.0  164.490196   93.205632  20.0  75.00  206.0
```

```
228.0

               VMail_Message
\
           max         count       mean        std  min  25%  50%
75%   max
Cluster

0        298.0         831.0   7.345367  15.804829  0.0  0.0  0.0
0.0  88.0
1        329.0         422.0  12.052133  24.829818  0.0  0.0  0.0
20.0  88.0
2        224.0         246.0   7.146341  13.682811  0.0  0.0  0.0
0.0  48.0
3        329.0         564.0   4.239362  14.941014  0.0  0.0  0.0
0.0  86.0
4        289.0         909.0   3.298130  10.637717  0.0  0.0  0.0
0.0  82.0
5        329.0         722.0   6.340720  19.190055  0.0  0.0  0.0
0.0  88.0
6        326.0         459.0  11.113290  23.760179  0.0  0.0  0.0
0.0  88.0

        Day_Mins
\
           count        mean        std    min      25%     50%
75%
Cluster

0          831.0  232.429242  37.692018   62.6  222.900  230.20
242.400
1          422.0  297.868957  27.065769  202.2  282.800  292.90
322.900
2          246.0  162.603252  53.592667   62.4  125.575  157.55
199.325
3          564.0  295.619592  34.591722  202.0  282.200  289.80
322.800
4          909.0  243.206601  33.713595   82.9  222.900  232.20
262.900
5          722.0  262.263712  47.085666   62.6  228.200  262.80
289.200
6          459.0  240.277342  54.552116   82.2  222.800  229.60
282.800

               Day_Calls
\
           max     count        mean         std   min    25%    50%
75%
Cluster
```

```
0        337.4     831.0  124.787004   80.631777  22.0  42.00   93.0
220.0
1        329.9     422.0  190.296209  101.226798  28.0  90.00  204.0
286.0
2        326.3     246.0   98.479675   34.135110  22.0  83.00   99.0
115.0
3        442.4     564.0  207.618794   98.786087  26.0  93.75  208.0
322.0
4        423.4     909.0  140.970297   76.577695  22.0  82.00  120.0
220.0
5        444.4     722.0  167.444598   78.105927  22.0  92.00  204.0
222.0
6        328.8     459.0  155.592593   82.343789  22.0  88.00  200.0
222.0

                Day_Charge
\
            max      count       mean        std    min      25%
50%
Cluster

0        244.0      831.0  28.050903  10.187186  18.63  22.3300
23.630
1        329.0      422.0  47.483934  26.430795  20.08  28.8200
32.320
2        224.0      246.0  27.179878   8.345559  10.61  21.9525
26.055
3        329.0      564.0  45.295715  25.717065  20.28  28.6200
32.200
4        320.0      909.0  31.808086  13.972160  20.22  22.8600
26.800
5        329.0      722.0  35.240111  19.432743  20.03  23.2200
28.820
6        328.0      459.0  34.509739  21.197404   9.89  22.6800
28.220

                           Eve_Mins
\
             75%      max    count        mean        std    min
25%
Cluster

0        30.2300   89.080    831.0  227.794705  21.867620  140.9
220.600
1        80.8900   89.980    422.0  292.162085  35.596844  202.0
269.900
2        30.7375   55.470    246.0  191.800813  39.153547  109.6
167.275
3        80.2000  112.011    564.0  292.502128  32.678465  202.6
280.900
```

```
4         36.2200   88.920    909.0  242.232013  30.630293  150.6
223.200
5         33.3000   89.280    722.0  259.322022  34.437642  202.2
228.200
6         29.2800   88.820    459.0  246.222004  33.559651  200.6
222.200

                              Eve_Calls
\
            50%    75%    max     count        mean         std   min
25%
Cluster

0         223.3  233.0  404.2     831.0  124.551143   78.793795  20.0
44.00
1         290.1  322.8  329.9     422.0  174.947867  105.430966  20.0
89.00
2         190.8  218.1  339.9     246.0  100.662602   37.145769  20.0
83.25
3         288.8  322.8  404.2     564.0  195.264184  103.870065  22.0
92.00
4         232.3  262.6  404.2     909.0  142.427943   76.538828  22.0
86.00
5         262.8  288.2  404.2     722.0  156.926593   81.582843  22.0
90.00
6         228.8  282.0  328.6     459.0  142.398693   81.572162  20.0
86.00

                               Eve_Charge
\
            50%     75%    max      count       mean         std   min
25%
Cluster

0          93.0  208.00  288.0      831.0  22.432876   3.714940  8.22
22.1950
1         149.5  286.00  329.0      422.0  28.542749   4.233457  9.32
28.0200
2         100.0  113.75  229.0      246.0  15.932846   4.478749  3.22
13.7225
3         208.0  320.00  329.0      564.0  29.960502  17.598791  9.88
26.3300
4         122.0  220.00  289.0      909.0  23.344246   3.271489  8.98
22.2000
5         202.0  222.00  329.0      722.0  25.131773   4.090789  8.32
22.3200
6          98.0  220.00  328.0      459.0  25.196688   4.674594  8.98
22.2200

                                 Night_Mins
```

```
\
             50%     75%      max      count        mean         std
min
Cluster

0        22.330  23.320   32.020      831.0  232.611673  26.251009
126.7
1        28.880  32.260   32.990      422.0  289.873697  34.983084
202.0
2        15.905  18.540   28.890      246.0  199.161789  37.449687
123.5
3        28.680  32.280  211.211      564.0  295.860160  29.958488
202.4
4        22.820  24.240   32.320      909.0  233.097800  27.381780
141.1
5        23.920  28.815   32.980      722.0  255.251524  33.299681
200.2
6        22.990  28.820   32.890      459.0  251.661656  34.362323
202.0

                                          Night_Calls
\
             25%     50%     75%    max       count         mean
std
Cluster

0        222.20  226.20  241.45  332.7       831.0  134.057762
81.032830
1        282.00  288.80  322.60  329.8       422.0  183.680095
102.605462
2        168.90  203.25  224.05  285.0       246.0   98.902439
33.095635
3        282.20  292.20  322.80  329.9       564.0  180.971631
106.261412
4        222.20  226.50  242.20  312.1       909.0  127.668867
77.117147
5        224.25  244.40  287.70  354.9       722.0  167.059557
81.275583
6        222.20  260.90  282.80  329.6       459.0  144.982571
81.187043

                                              Night_Charge
\
          min    25%    50%     75%    max        count       mean
std
Cluster

0        22.0  49.00  103.0  220.00  326.0        831.0   9.103141
6.828475
1        26.0  89.00  202.0  282.00  329.0        422.0  14.750095
```

```
10.255027
2        22.0  81.25   98.0  115.75  222.0        246.0   8.783699
2.906354
3        20.0  88.00  202.0  288.00  329.0        564.0  16.708417
10.414336
4        20.0  68.00   98.0  220.00  288.0        909.0  11.832365
7.195587
5        22.0  91.25  204.0  223.00  329.0        722.0  12.850956
7.958673
6        22.0  82.00   98.0  214.50  328.0        459.0  10.863224
7.065198

                                              Intl_Mins
\
          min    25%     50%      75%    max     count       mean
std
Cluster

0        2.02  3.290   8.220   9.8200  22.96     831.0  21.000842
3.684570
1        2.02  8.065   9.305  22.2200  32.98     422.0  29.870142
3.496841
2        2.22  7.175   8.635   9.8825  22.44     246.0  10.957317
3.707060
3        2.08  8.280  20.020  22.8900  32.96     564.0  11.043387
7.317396
4        2.03  6.320   9.390  20.3000  24.22     909.0   6.328053
3.292441
5        2.06  6.280   9.390  22.0200  32.88     722.0  17.618283
6.708882
6        2.02  6.690   8.980  14.9700  32.88     459.0  25.415033
3.678851

                                         Intl_Calls
\
          min     25%   50%   75%   max      count      mean       std
min
Cluster

0         8.2  20.300  22.2  22.6  28.8      831.0  4.133574  4.115526
1.0
1        20.0  28.800  32.0  32.2  32.9      422.0  5.528436  5.169608
2.0
2         3.3   9.025  10.6  12.9  22.4      246.0  3.674797  2.393856
1.0
3         2.0   6.800   9.0  20.0  32.8      564.0  5.501773  4.738827
2.0
4         2.0   3.200   6.2   9.2  20.3      909.0  3.767877  3.459333
1.0
5         2.2   9.600  20.3  22.2  32.8      722.0  4.912742  4.709229
```

```
2.0
6        20.9  22.200  22.8  28.8  32.9      459.0  5.540305  5.027871
2.0

                            Intl_Charge
\
         25%  50%  75%   max       count      mean       std   min
25%
Cluster

0        2.0  3.0  4.0  28.0       831.0  3.235704  1.029781  2.00
2.36
1        2.0  2.0  8.0  32.0       422.0  7.808318  1.819185  2.26
8.20
2        2.0  3.0  5.0  20.0       246.0  2.852195  0.695202  1.11
2.38
3        2.0  5.0  8.0  32.0       564.0  2.485488  0.348487  0.82
2.26
4        2.0  3.0  4.0  28.0       909.0  2.376986  0.343642  0.32
2.22
5        2.0  3.0  8.0  28.0       722.0  2.795956  0.543195  2.00
2.28
6        2.0  2.0  8.0  32.0       459.0  8.144946  1.305600  2.22
8.22

                          CustServ_Calls
\
          50%   75%   max          count      mean       std  min  25%
50%
Cluster

0        3.22  3.62  8.92          831.0  2.208183  1.680141  0.0  2.0
2.0
1        8.28  8.68  8.92          422.0  2.848341  2.711940  0.0  2.0
2.0
2        2.81  3.32  4.83          246.0  2.926829  2.014927  0.0  1.0
3.0
3        2.32  2.82  4.24          564.0  2.590426  2.494181  0.0  2.0
2.0
4        2.32  2.62  3.70          909.0  1.951595  1.582486  0.0  1.0
2.0
5        2.82  3.00  4.92          722.0  2.595568  2.426137  0.0  2.0
2.0
6        8.28  8.82  8.92          459.0  2.943355  2.774072  0.0  2.0
2.0


         75%  max
Cluster
0        3.0  9.0
```

```
1         2.0  9.0
2         4.0  9.0
3         2.0  9.0
4         2.0  8.0
5         2.0  9.0
6         2.0  9.0
```

Cluster Analysis for churn_true

Cluster 0: Average Users with Short Calls

Characteristics:

- Average usage across categories
- Low day calls: 124.8 despite average minutes
- High international calls: 4.1
- Low international minutes: 21.0
- Average customer service calls: 2.2

Reason: These customers make frequent but shorter calls, especially internationally.

Cluster 1: High Usage Premium Customers

Characteristics:

- Highest day minutes: 297.9
- Highest day charges: 47.5
- High evening minutes: 292.2
- High night minutes: 289.9
- High international usage: 29.9 minutes, 5.5 calls
- Above average customer service calls: 2.8

Reason: These customers are heavy users across all time periods, likely on premium plans.

Cluster 2: Low Usage Economy Customers

Characteristics:

- Lowest usage across all categories
- Shortest average account length: 103.2 days
- Highest customer service calls: 2.9

Reason: These customers use their phones minimally but require more support, possibly new customers on economy plans.

Cluster 3: Night-time Heavy Users

Characteristics:

- Highest night minutes: 295.9

- Highest night charges: 16.7
- High day minutes: 295.6
- High evening minutes: 292.5
- Low voicemail messages: 4.2
- Average customer service calls: 2.6

Reason: These customers are heavy users, especially during night time.

Cluster 4: Moderate Balanced Users

Characteristics:

- Moderate usage across all categories
- Balanced day minutes: 243.2
- Balanced evening minutes: 242.2
- Balanced night minutes: 233.1
- Low international usage: 6.3 minutes, 3.8 calls
- Below average customer service calls: 2.0

Reason: These customers have balanced usage across different times of day.

Cluster 5: Daytime Business Users

Characteristics:

- High day minutes: 262.3
- High day charges: 35.2
- Moderate evening and night usage
- High international calls: 4.9
- Average international minutes: 17.6
- Above average customer service calls: 2.6

Reason: These customers have high daytime usage and frequent short international calls, suggesting business use.

Cluster 6: Evening and International Callers

Characteristics:

- High evening minutes: 246.2 relative to day and night
- Highest international minutes: 25.4
- Highest international charges: 8.1
- Above average customer service calls: 2.9

Reason: These customers prefer evening calls and have significant international usage.

EDA PART VISUALIZED USING STREAMLIT

Navigation

Select a section:

- (•) Exploratory Data Analysis (EDA)
- ( ) Clustering

Customer Analysis

Exploratory Data Analysis (EDA)

Dataset Shape: `(101174, 17)`

There are `101174` rows and `17` columns/features.

Columns in the Dataset:

	0
0	Phone Number
1	Account Length
2	VMail Message
3	Day Mins
4	Day Calls
5	Day Charge
6	Eve Mins
7	Eve Calls
8	Eve Charge
9	Night Mins

Columns After Replacing Spaces with Underscores:

	0
0	Phone_Number
1	Account_Length
2	VMail_Message
3	Day_Mins
4	Day_Calls
5	Day_Charge
6	Eve_Mins
7	Eve_Calls
8	Eve_Charge
9	Night_Mins

First Few Rows of the Dataset:

	Phone_Number	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls	Nigl
0	382-4657	128	25	265.1	110	45.07	197.4	99	16.78	244.7	91	
1	371-7191	107	26	161.6	123	27.47	195.5	103	16.62	254.4	103	
2	358-1921	137	0	243.4	114	41.38	121.2	110	10.3	162.6	104	
3	375-9999	84	0	299.4	71	50.9	61.9	88	5.26	196.9	89	
4	330-6626	75	0	166.7	113	28.34	148.3	122	12.61	186.9	121	

Assumptions:

- Assuming Account_Length is measured in days.
- Assuming charges are measured in USD ($) because there is no dataset description provided.

Dataset Information:

```
None
```

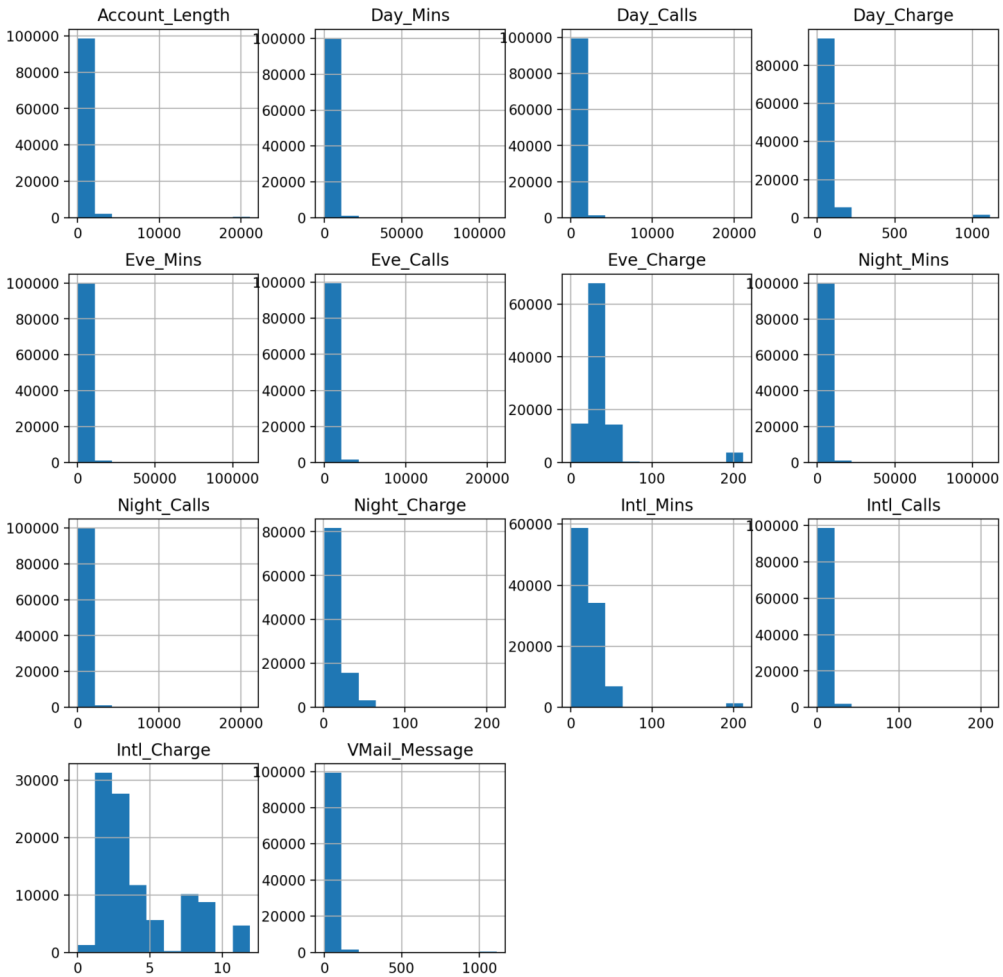
There are 15 numeric columns - 8 float and 7 int, 1 boolean, and 1 object column.

Descriptive Statistics:

	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls	Night_Charge	Intl
count	101,174	101,174	101,174	101,174	101,174	101,174	101,174	101,174	101,174	101,174	101,174	10
mean	323.5971	18.1426	587.0808	267.2076	64.9588	655.5125	267.1669	34.3581	646.7866	261.6732	13.9001	20
std	1,418.0737	75.8269	2,193.0943	947.9634	133.5158	2,473.4454	932.9144	36.2486	2,368.0285	884.963	16.1787	25
min	1	0	0	0	0	0	0	0	20.3	20	1.04	
25%	69	0	222.3	88	23.92	223.3	88	22.22	223.2	88	6.22	
50%	202	0	262.2	202	32.6	268.2	202	24.44	268.2	202	9.22	
75%	240	22	326.275	224	82.2	329.2	224	32.02	329.2	224	20.44	
max	21,111	1,111	111,111.11	21,111	1,111.99	111,111.2	21,111	211.99	111,111.11	21,111	211.92	

- Account_Length has a maximum value of 21111, which is highly unlikely (57 years).
- Similar to Account_Length, other features also have questionable maximum values indicating the presence of outliers skewing the dataset.
- Every column has a mean << 75th percentile, indicating a right-skewed distribution.
- High standard deviation values suggest a wide range of values.
- VMail_Message, CustServ, Intl_Mins, Intl_Calls, and Intl_Charge aside, the rest of the columns need outlier detection and removal due to impossible values.

Histograms:



The histogram clearly indicates skewness.

Missing Values:

	0
Phone_Number	0
Account_Length	0
VMail_Message	0
Day_Mins	0
Day_Calls	0
Day_Charge	0
Eve_Mins	0
Eve_Calls	0
Eve_Charge	0
Night_Mins	0

Duplicate Rows:

Total Duplicate Rows: `40729`

- Around 40k rows are duplicates.
- The df.duplicated() function compares all the feature values for an exact match, looking for a 1-1 copy.

Dataset After Removing Duplicates:

```
(60445, 17)
```

Unique Counts in Each Feature:

	0
Phone_Number	7,467
Account_Length	322
VMail_Message	72
Day_Mins	2,548
Day_Calls	221
Day_Charge	2,873
Eve_Mins	2,523
Eve_Calls	224
Eve_Charge	2,221
Night_Mins	2,464

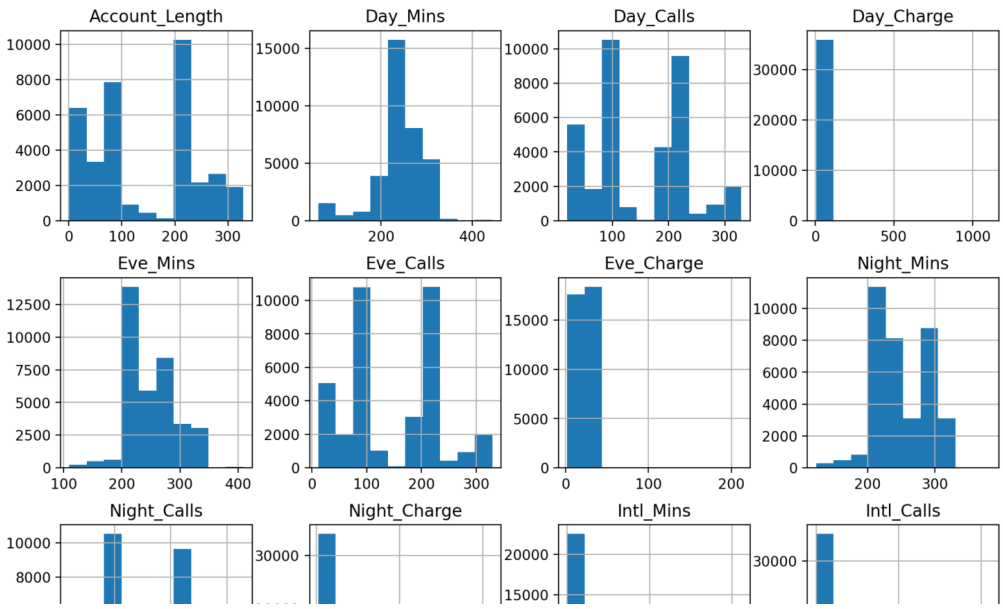
Checking for unusual unique counts in hopes of finding anything unusual.

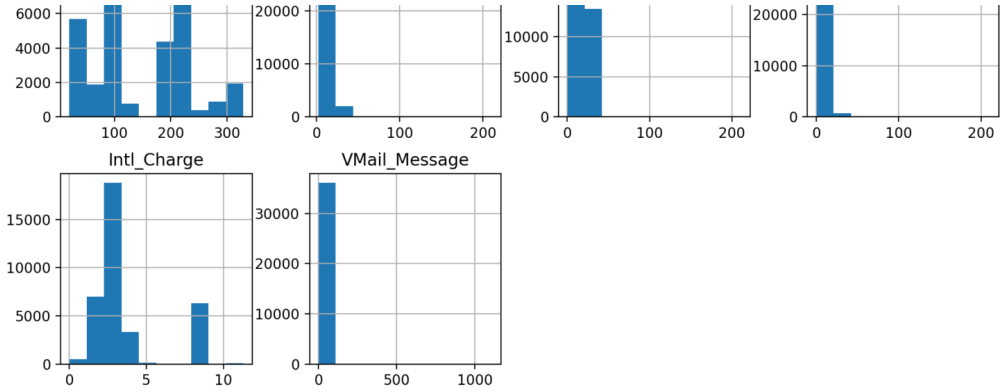
Outlier Detection and Removal:

Dataset After Removing Outliers:

```
(36077, 17)
```

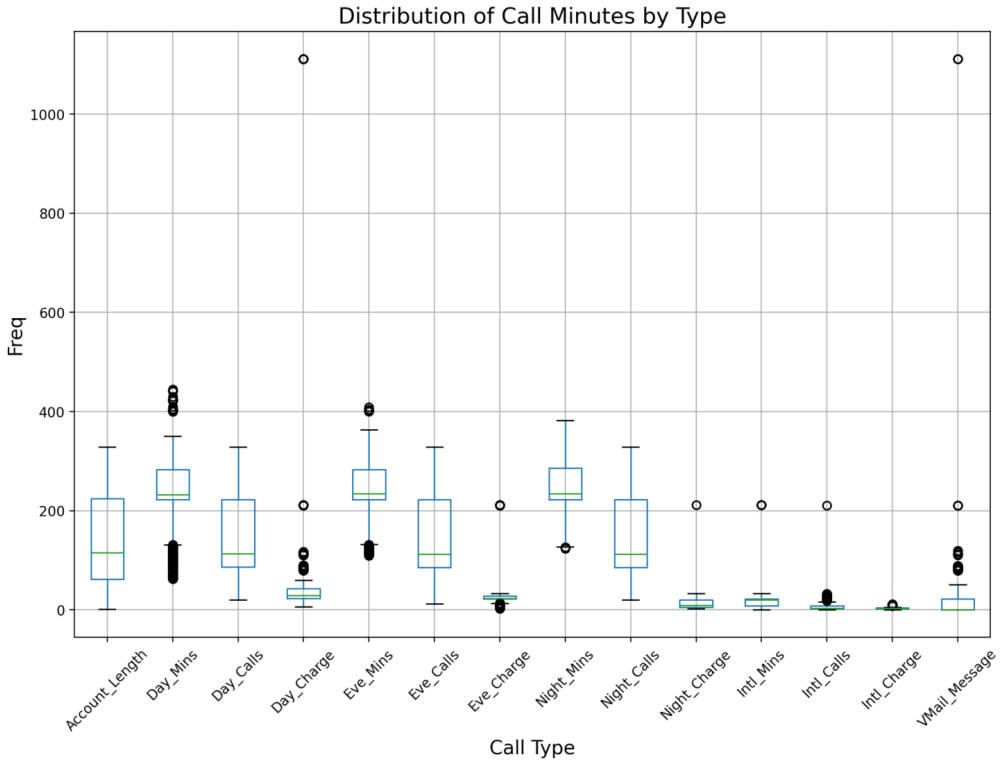
Histograms After Outlier Removal:





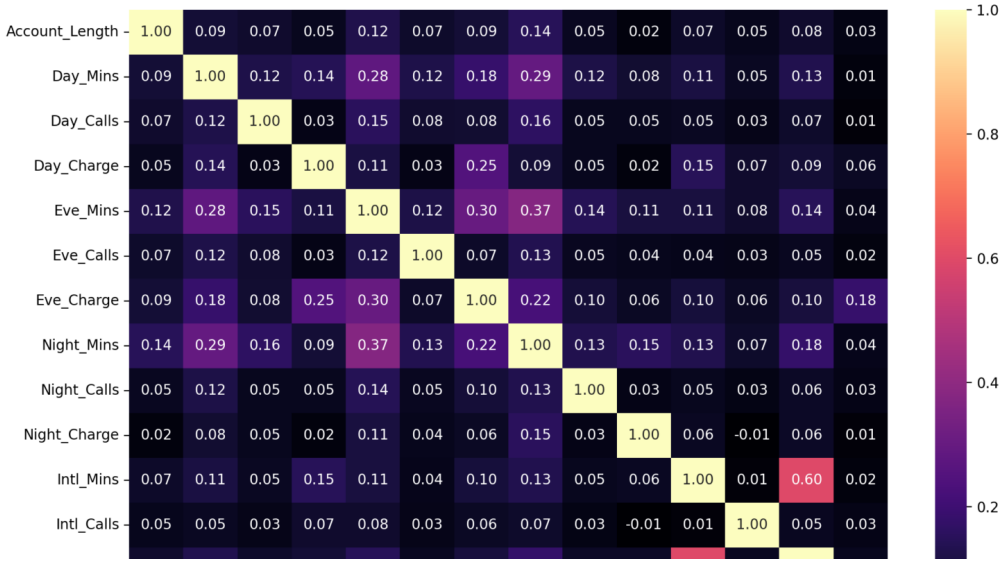
After addressing outliers, the histogram looks less skewed.

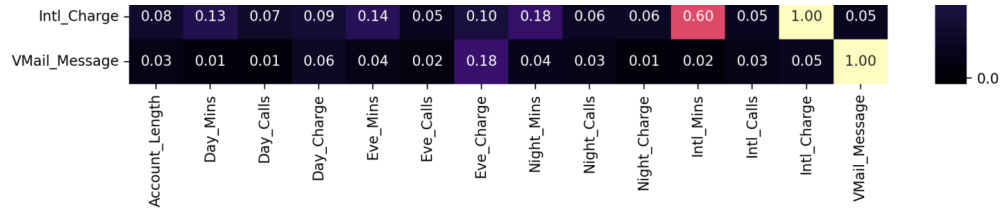
Box Plot After Outlier Removal:



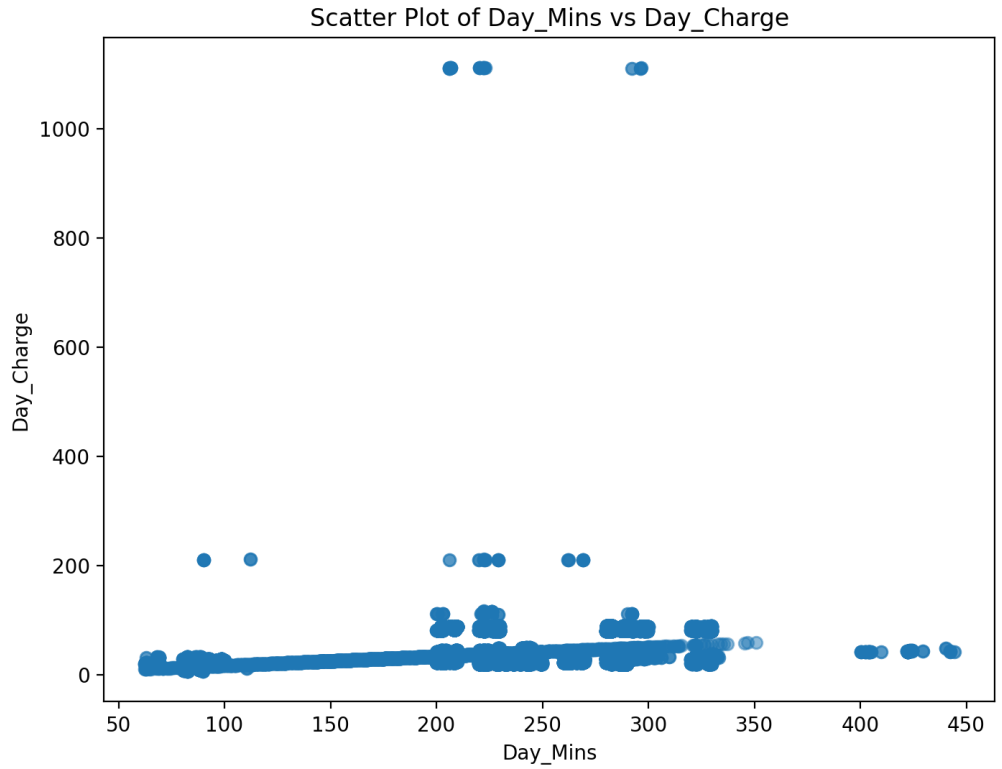
By visualizing the box plot, we can clearly identify outliers.

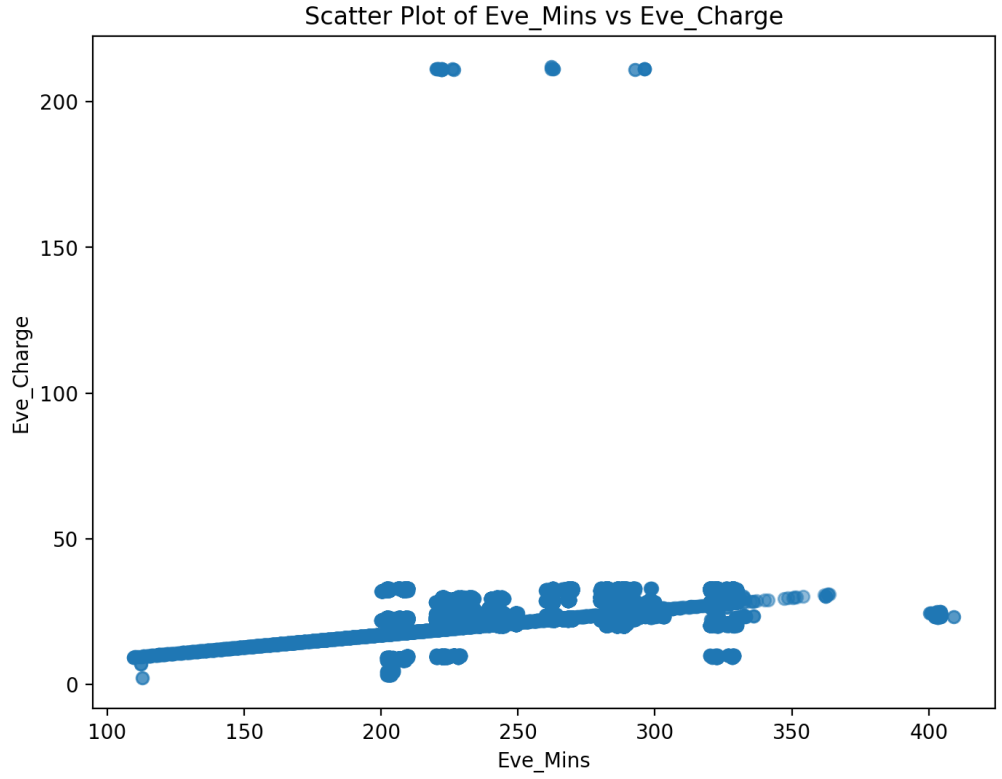
Correlation Heatmap:



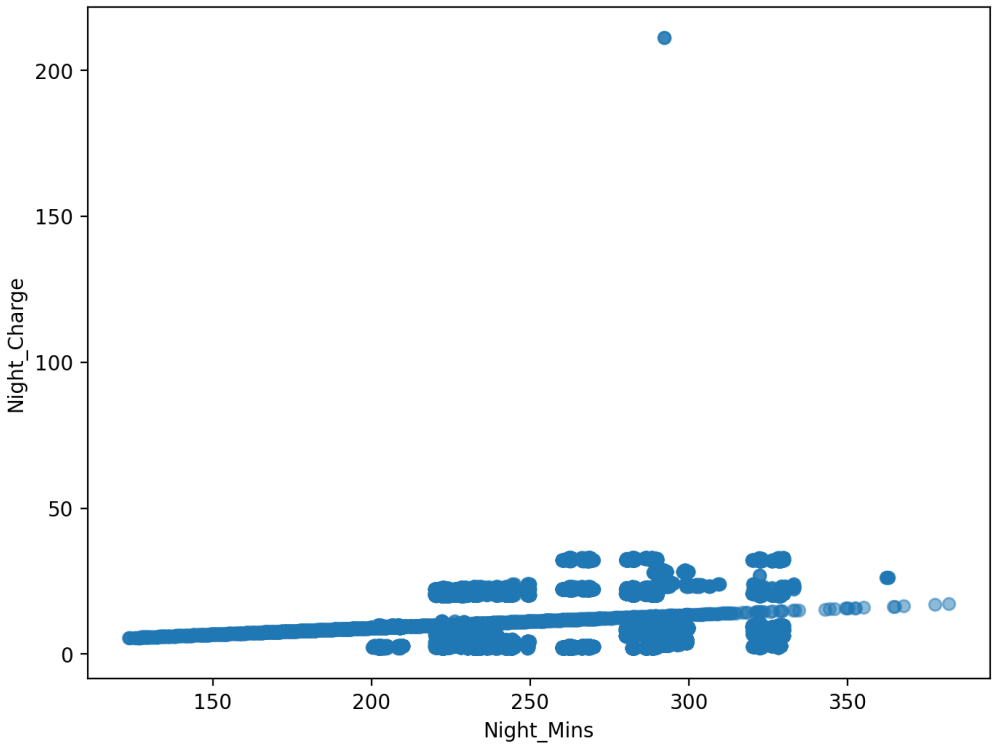


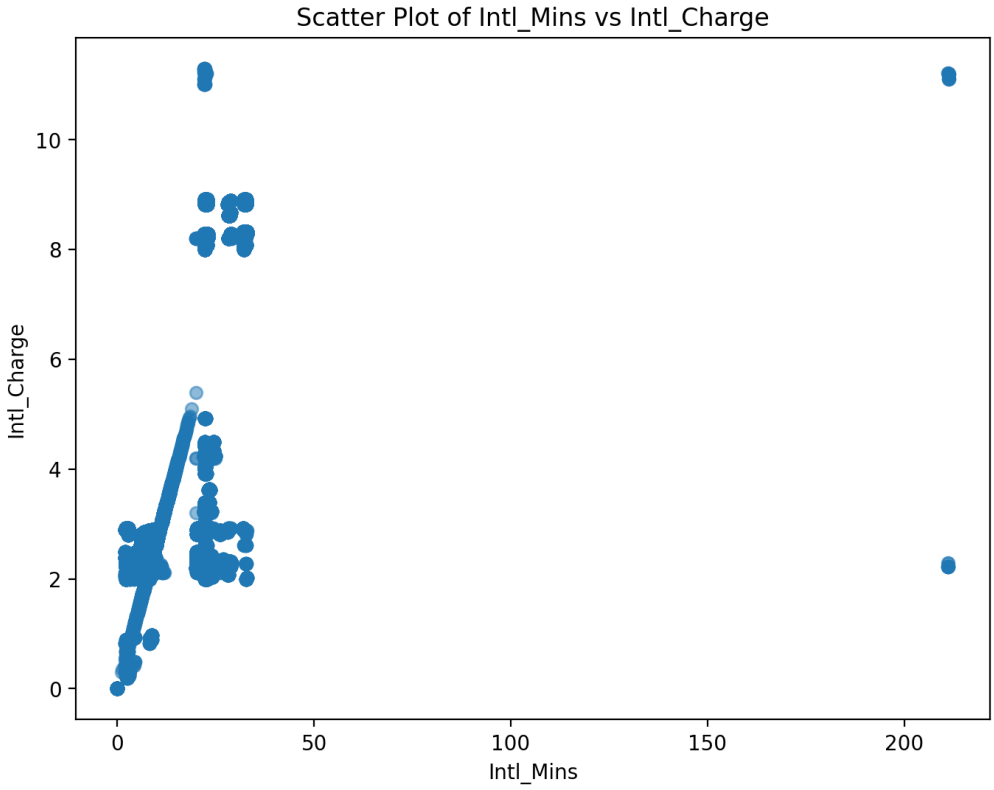
Shows somewhat linear relationships indicating that as minutes increase, so does the price (charge) across day, eve, night & intl.



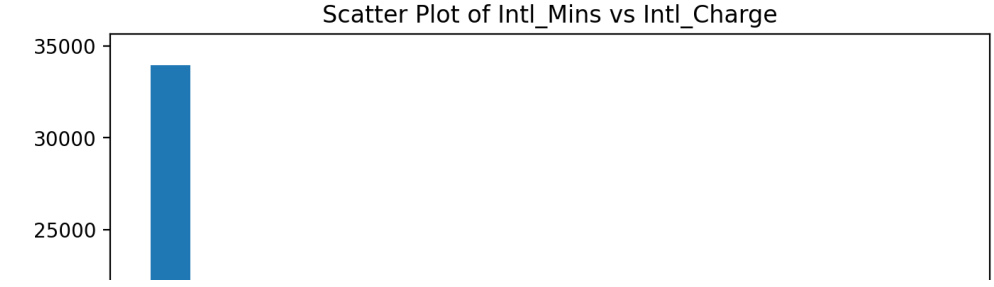


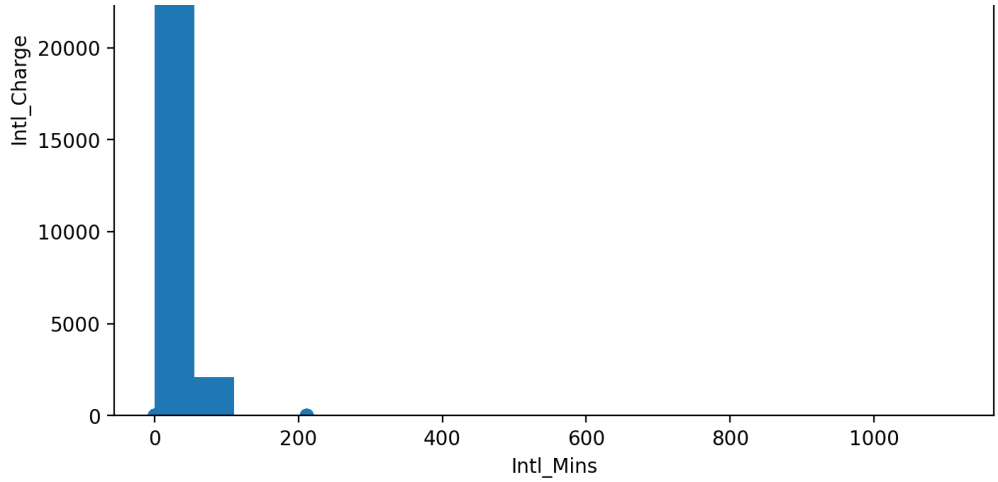
Scatter Plot of Night_Mins vs Night_Charge





Histogram for VMail_Message Feature:





Inference:

Just checking the VMail_Message feature for anything unusual.

	Account_Length	VMail_Message	Day_Mins	Day_Calls	Day_Charge	Eve_Mins	Eve_Calls	Eve_Charge	Night_Mins	Night_Calls	Night_Charge	Intl_M
count	36,077	36,077	36,077	36,077	36,077	36,077	36,077	36,077	36,077	36,077	36,077	36,(
mean	147.9666	11.5152	241.2501	149.0671	40.8114	249.971	148.9111	24.4473	250.2916	148.2313	11.648	15.6
std	96.56	28.2733	53.0843	86.3538	41.1102	40.485	86.2326	7.9073	39.533	86.0034	8.3371	10.3
min	1	0	62.3	20	6.22	109.6	12	2.11	123.5	20	2	
25%	62	0	222.2	87	23.23	222.4	86	22.22	222.4	86	4.92	
50%	115	0	232.2	113	29.24	234.4	112	23.22	234.4	112	9.2	
75%	224	22	282.8	222	42.43	282.9	222	28.28	286.2	222	20.29	2
max	329	1,111	444.4	329	1,111.22	409.2	329	211.911	381.9	329	211.22	21

Developed by COGNIZANT BATCH 4

CLUSTERING PART VISUALIZED USING STREAMLIT

Navigation

Select a section:

- Exploratory Data Analysis (EDA)
- Clustering

Clustering Configuration

Select the number of clusters:

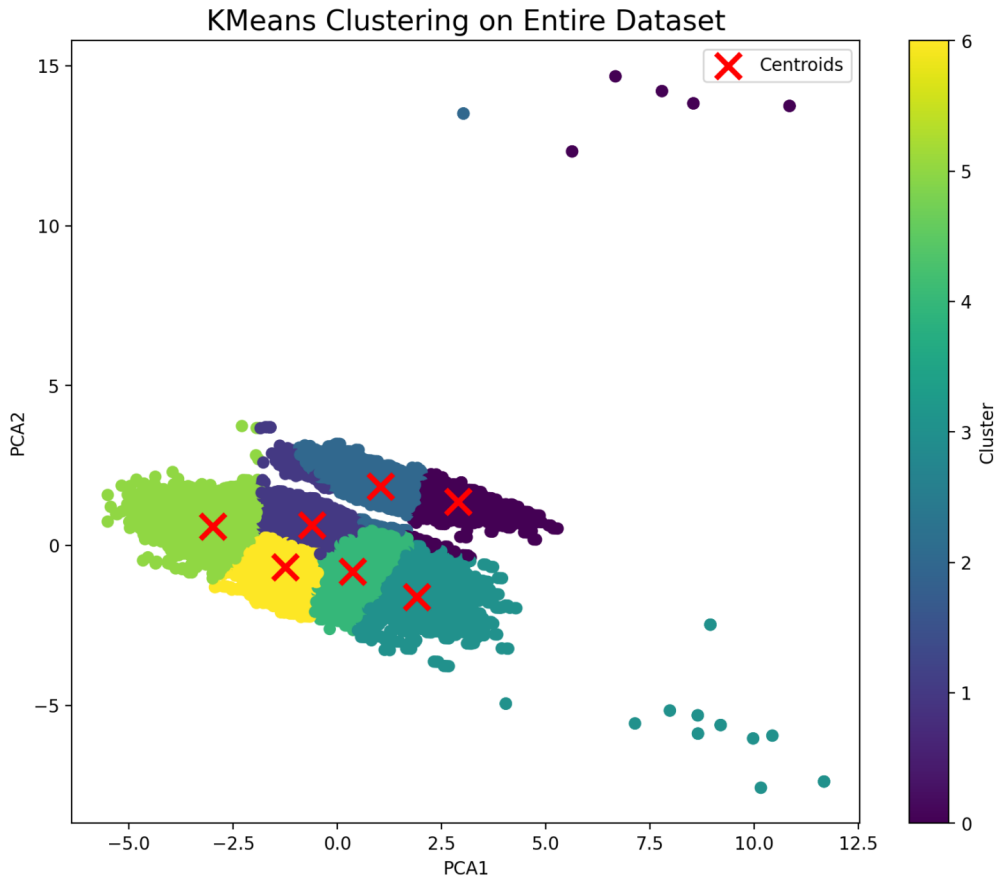
7

2 10

Customer Analysis

Customer Clustering Analysis

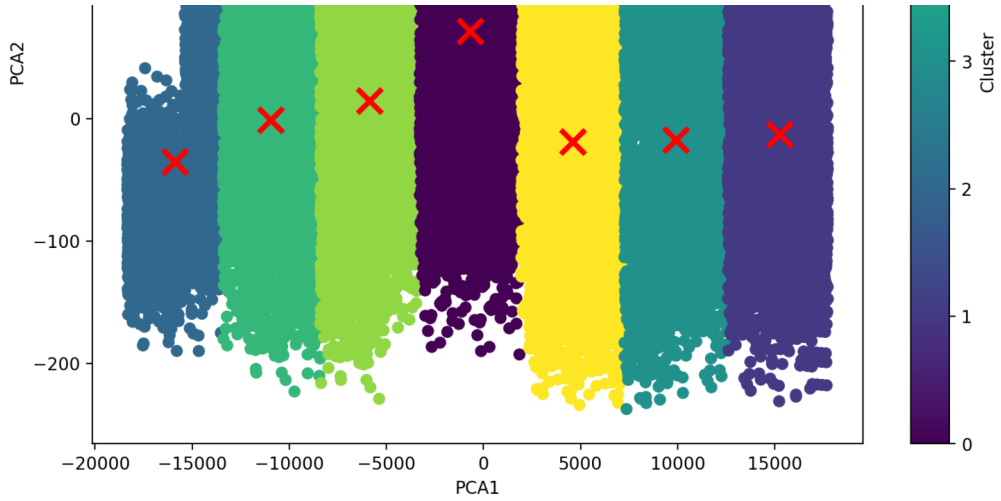
KMeans Clustering on Entire Dataset



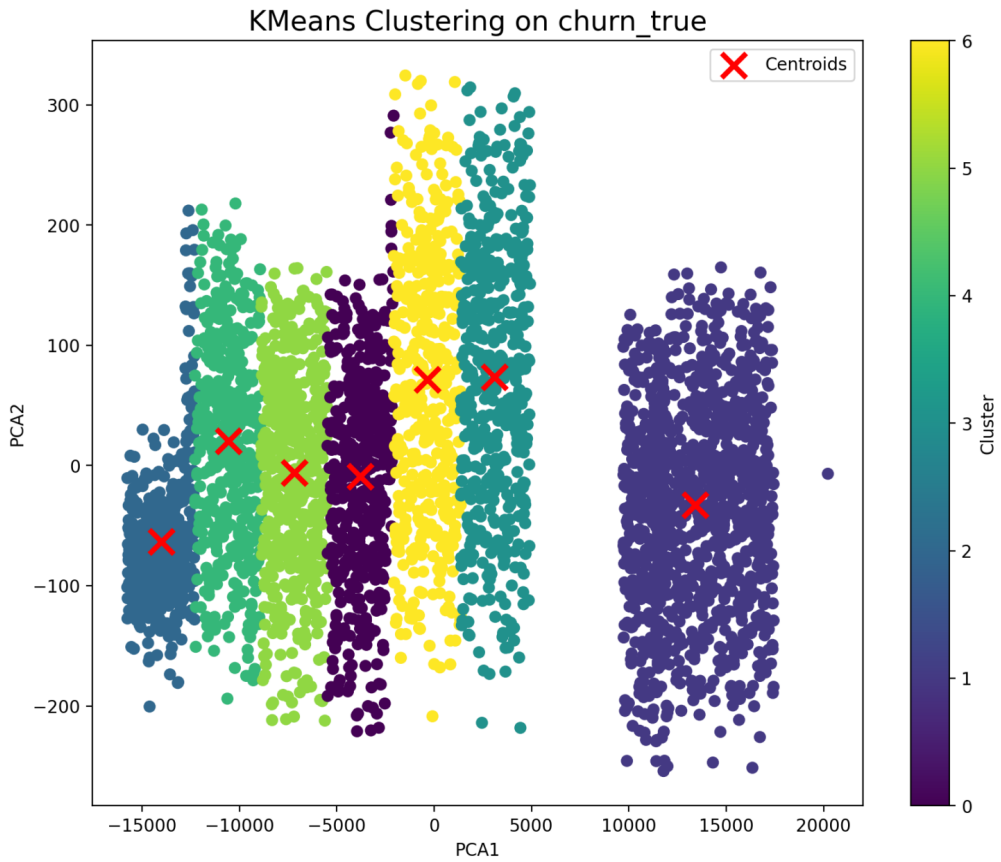
Churn Rate Analysis per Cluster

	Cluster	Churn Rate
0	0	0.1352
1	1	0.1206
2	2	0.1286
3	3	0.1219
4	4	0.105
5	5	0.1002
6	6	0.107

KMeans Clustering on churn_false



KMeans Clustering on churn_true



Descriptive Statistics by Cluster

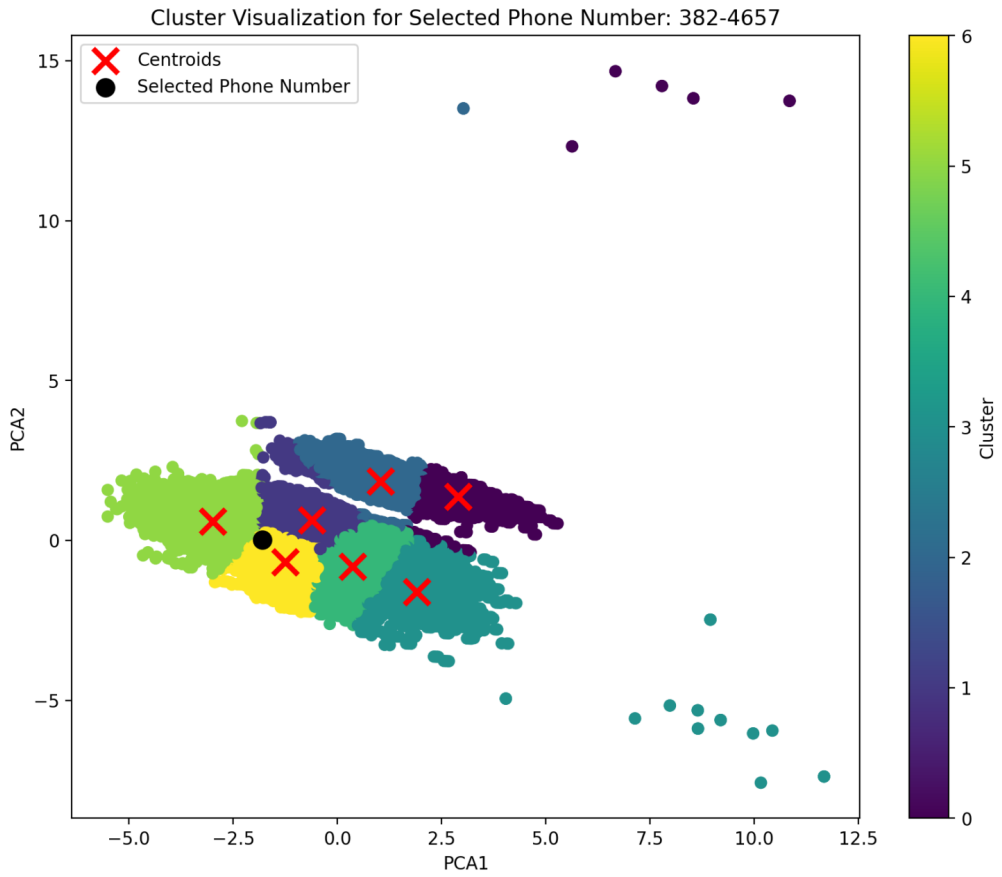
Cluster	Unnamed: 0	Unnamed: 0	Unnamed: 0	Unnamed: 0	Unnamed: 0	Unnamed: 0	Unnamed: 0	Unnamed: 0	Account_Length	Account_Length	Accoun
None	count	mean	std	min	0.25	0.5	0.75	max	count	mean	std
0	2,921	20,474.0332	7,328.5941	2,963	15,697	18,208	20,676	36,060	2,921	190.0743	
1	8,209	17,130.244	8,730.4621	20	9,157	15,774	24,848	36,066	8,209	137.2307	
2	3,484	22,784.1613	11,946.9595	2,964	13,326.75	29,187	32,469.75	36,067	3,484	148.9027	
3	4,103	19,821.2808	6,781.1959	2,979	15,622.5	17,969	20,279.5	36,075	4,103	190.3651	
4	7,235	22,553.573	10,355.8114	5	14,476.5	25,703	31,517.5	36,076	7,235	158.8435	
5	2,794	3,568.0361	5,753.7695	1	890.25	1,769.5	2,693.75	36,014	2,794	99.412	
6	7,287	15,903.6416	9,761.2979	0	7,588.5	12,420	24,620.5	36,074	7,287	126.4894	

Phone Number Cluster Identification

Select a Phone Number

382-4657

The selected phone number belongs to Cluster 6.



Developed by COGNIZANT BATCH 4