The below work is done only by myself and the help of internet for syntax and other minor purposes

Please uncomment these installtion if its not already installed

In [492…

```
#!pip install klib
```

In [493…

```
#!pip install dtale
```

In [494…

```
#pip install Jinja2==3.0.3
```

In [495…

```
# pip install pandas-profiling
```

In [ ]:

```
# !pip install chart_studio
```

In [496…

```
import pandas as pd
import numpy as np
import klib
import dtale
import matplotlib.pyplot as plt
import seaborn as sns
import math
import pandas_profiling
```

In [497…

```
import warnings
warnings.filterwarnings('ignore')
```

Flights Code

In [504…

```
df_Flights=pd.read_csv('/Users/abhishekshastry/Documents/Interview_takehomes/capitalone/
df_Flights.head()
```

Out[504]:

	FL_DATE	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	ORIGIN_CITY_NA
0	2019-03-02	WN	N955WN	4591	14635	RSW	Fort Myers
1	2019-03-02	WN	N8686A	3231	14635	RSW	Fort Myers
2	2019-03-02	WN	N201LV	3383	14635	RSW	Fort Myers
3	2019-03-02	WN	N413WN	5498	14635	RSW	Fort Myers
4	2019-03-02	WN	N7832A	6933	14635	RSW	Fort Myers

In [505…

```
df_Flights.dtypes
```

```
Out[505]: FL_DATE               object
          OP_CARRIER            object
          TAIL_NUM              object
          OP_CARRIER_FL_NUM     object
          ORIGIN_AIRPORT_ID      int64
          ORIGIN                object
          ORIGIN_CITY_NAME      object
          DEST_AIRPORT_ID        int64
          DESTINATION           object
          DEST_CITY_NAME        object
          DEP_DELAY            float64
          ARR_DELAY            float64
          CANCELLED            float64
          AIR_TIME              object
          DISTANCE              object
          OCCUPANCY_RATE       float64
          dtype: object
```

```
In [506…  df_Flights[['Origin_City','Origin_State']]=df_Flights['ORIGIN_CITY_NAME'].str.split(",",
          df_Flights[['Dest_City','Dest_State']]=df_Flights['DEST_CITY_NAME'].str.split(",",expand
          df_Flights.drop('ORIGIN_CITY_NAME',axis=1,inplace=True)
          df_Flights.drop('DEST_CITY_NAME',axis=1,inplace=True)
          df_Flights
```

Out[506]:

	FL_DATE	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	DEST_AIF
0	2019-03-02	WN	N955WN	4591	14635	RSW	
1	2019-03-02	WN	N8686A	3231	14635	RSW	
2	2019-03-02	WN	N201LV	3383	14635	RSW	
3	2019-03-02	WN	N413WN	5498	14635	RSW	
4	2019-03-02	WN	N7832A	6933	14635	RSW	
...	...	...	...	...	...	...	
1915881	3/23/19	AA	N903NN	1433	15370	TUL	
1915882	3/24/19	AA	N965AN	1433	15370	TUL	
1915883	3/25/19	AA	N979NN	1433	15370	TUL	
1915884	3/26/19	AA	N872NN	1433	15370	TUL	
1915885	3/27/19	AA	N945AN	1433	15370	TUL	

1915886 rows × 18 columns

```
In [507…  # klib.describe - functions for visualizing datasets
          # - klib.cat_plot(df) # returns a visualization of the number and frequency of categoric
          # - klib.corr_mat(df) # returns a color-encoded correlation matrix
          # - klib.corr_plot(df) # returns a color-encoded heatmap, ideal for correlations
          # - klib.dist_plot(df) # returns a distribution plot for every numeric feature
          # - klib.missingval_plot(df) # returns a figure containing information about missing val

          # # klib.clean - functions for cleaning datasets
          # - klib.data_cleaning(df) # performs datacleaning (drop duplicates & empty rows/cols, a
          # - klib.clean_column_names(df) # cleans and standardizes column names, also called insi
          # - klib.convert_datatypes(df) # converts existing to more efficient dtypes, also called
          # - klib.drop_missing(df) # drops missing values, also called in data_cleaning()
          # - klib.mv_col_handling(df) # drops features with high ratio of missing vals based on i
          # - klib.pool_duplicate_subsets(df) # pools subset of cols based on duplicates with min.
```

In [508…

```
# df_Flights[df_Flights['TAIL_NUM']=='N955WN'][df_Flights['ORIGIN']=='RSW'][df_Flights['
```

In [509…

```
# df_Flights=klib.data_cleaning(df_Flights)
df_Flights.drop_duplicates()
```

Out[509]:

	FL_DATE	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	DEST_AIF
0	2019-03-02	WN	N955WN	4591	14635	RSW	
1	2019-03-02	WN	N8686A	3231	14635	RSW	
2	2019-03-02	WN	N201LV	3383	14635	RSW	
3	2019-03-02	WN	N413WN	5498	14635	RSW	
4	2019-03-02	WN	N7832A	6933	14635	RSW	
...	...	...	...	...	...	...	
1911336	3/23/19	AA	N903NN	1433	15370	TUL	
1911337	3/24/19	AA	N965AN	1433	15370	TUL	
1911338	3/25/19	AA	N979NN	1433	15370	TUL	
1911339	3/26/19	AA	N872NN	1433	15370	TUL	
1911340	3/27/19	AA	N945AN	1433	15370	TUL	

1911341 rows × 18 columns

In [510…

```
df_Flights.dtypes
```

Out[510]:

```
FL_DATE                  object
OP_CARRIER               object
TAIL_NUM                 object
OP_CARRIER_FL_NUM        object
ORIGIN_AIRPORT_ID         int64
ORIGIN                   object
DEST_AIRPORT_ID           int64
DESTINATION              object
DEP_DELAY               float64
ARR_DELAY               float64
CANCELLED               float64
AIR_TIME                 object
DISTANCE                 object
OCCUPANCY_RATE          float64
Origin_City              object
Origin_State             object
Dest_City                object
Dest_State               object
dtype: object
```

In [511…

```
# df_Flights.distance.dtype.empty
```

In [512…

```
# klib.clean_column_names(df_Flights) # cleans and standardizes column names, also calle
```

In [513…

```
# klib.convert_datatypes(df_Flights)
```

In [514…

```
df_Flights.head()
```

Out[514]:

	FL_DATE	OP_CARRIER	TAIL_NUM	OP_CARRIER_FL_NUM	ORIGIN_AIRPORT_ID	ORIGIN	DEST_AIRPORT_
0	2019-03-02	WN	N955WN	4591	14635	RSW	110
1	2019-03-02	WN	N8686A	3231	14635	RSW	110
2	2019-03-02	WN	N201LV	3383	14635	RSW	110
3	2019-03-02	WN	N413WN	5498	14635	RSW	110
4	2019-03-02	WN	N7832A	6933	14635	RSW	112

In [515…

```
df_Flights['FL_DATE']=pd.to_datetime(df_Flights['FL_DATE'])
```

In [516…

```
# klib.mv_col_handling(df_Flights)
```

In [517…

```
df_Flights.columns
```

Out[517]:

```
Index(['FL_DATE', 'OP_CARRIER', 'TAIL_NUM', 'OP_CARRIER_FL_NUM',
       'ORIGIN_AIRPORT_ID', 'ORIGIN', 'DEST_AIRPORT_ID', 'DESTINATION',
       'DEP_DELAY', 'ARR_DELAY', 'CANCELLED', 'AIR_TIME', 'DISTANCE',
       'OCCUPANCY_RATE', 'Origin_City', 'Origin_State', 'Dest_City',
       'Dest_State'],
      dtype='object')
```

In [518…

```
df_Flights=klib.clean_column_names(df_Flights)
```

In [519…

```
# 1. The 10 busiest round trip routes in terms of number of round trip flights in the qu
#    Exclude canceled flights when performing the calculation.
```

In [520…

```
df_Flights=df_Flights[df_Flights['cancelled']==0]
df_Flights.shape
```

Out[520]:

```
(1864272, 18)
```

In [521…

```
df_Flights.isna().sum()
```

Out[521]:

```
fl_date                 0
op_carrier              0
tail_num                0
op_carrier_fl_num       0
origin_airport_id       0
origin                  0
dest_airport_id         0
destination             0
dep_delay               0
arr_delay            4377
cancelled               0
air_time             5027
distance              610
occupancy_rate        310
origin_city             0
origin_state            0
dest_city               0
dest_state              0
dtype: int64
```

Loading [MathJax]/extensions/Safe.js

```
l_trips_duplicates=df_Flights[['op_carrier','tail_num','origin','destinatio
```

```
df_Flights_all_trips_duplicates.tail()
```

Out[522]:

	op_carrier	tail_num	origin	destination
1915881	AA	N903NN	TUL	CLT
1915882	AA	N965AN	TUL	CLT
1915883	AA	N979NN	TUL	CLT
1915884	AA	N872NN	TUL	CLT
1915885	AA	N945AN	TUL	CLT

In [523…

```
df_Flights_all_trips_duplicates['ID']=list(range(0,len(df_Flights_all_trips_duplicates.i
```

In [524…

```
df_Flights_all_trips_duplicates
```

Out[524]:

	op_carrier	tail_num	origin	destination	ID
0	WN	N955WN	RSW	CLE	0
1	WN	N8686A	RSW	CMH	1
2	WN	N201LV	RSW	CMH	2
3	WN	N413WN	RSW	CMH	3
4	WN	N7832A	RSW	DAL	4
...	...	...	...	...	...
1915881	AA	N903NN	TUL	CLT	1864267
1915882	AA	N965AN	TUL	CLT	1864268
1915883	AA	N979NN	TUL	CLT	1864269
1915884	AA	N872NN	TUL	CLT	1864270
1915885	AA	N945AN	TUL	CLT	1864271

1864272 rows × 5 columns

part 1..

geenrate excel sheet so that postgrasesql querycan remove duplciates it

In [527…

```
df_Flights_all_trips_duplicates.to_csv("/Users/abhishekshastry/Documents/Interview_takeh
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

# part 2

In [529…

```
df_Flights_Unique_round_tripsPart2=df_Flights_all_trips_duplicates.iloc[1048576:,:]
df_Flights_Unique_round_tripsPart2
```

Out[529]:

	op_carrier	tail_num	origin	destination	ID
1074724	YX	N405YX	JAX	MIA	1048576
1074725	YX	N432YX	JAX	MIA	1048577
1074726	YX	N439YX	JAX	MIA	1048578
1074727	YX	N411YX	JAX	MIA	1048579
1074728	YX	N441YX	JAX	MIA	1048580
...	...	...	...	...	...
1915881	AA	N903NN	TUL	CLT	1864267
1915882	AA	N965AN	TUL	CLT	1864268
1915883	AA	N979NN	TUL	CLT	1864269
1915884	AA	N872NN	TUL	CLT	1864270
1915885	AA	N945AN	TUL	CLT	1864271

815696 rows × 5 columns

In [530…

```
df_Flights_Unique_round_tripsPart2.to_csv("/Users/abhishekshastry/Documents/Interview_ta
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

In [531…

```
df_Flights_Unique_round_tripsPart1=pd.read_csv("/Users/abhishekshastry/Documents/Intervi
df_Flights_Unique_round_tripsPart1.head()
```

Out[531]:

	origin	destination	tail_num	roundtrips	rn
0	DUT	ANC	N687PA	39	1
1	HNL	JHM	N805HC	35	1
2	HNL	MKK	N801HC	32	1
3	DUT	ANC	N682PA	29	1
4	DUT	ANC	N681PA	24	1

In [532…

```
df_Flights_Unique_round_tripsPart2=pd.read_csv("/Users/abhishekshastry/Documents/Intervi
df_Flights_Unique_round_tripsPart2.head()
```

Out[532]:

	origin	destination	tail_num	roundtrips	rn
0	HNL	MKK	N801HC	56	1
1	HNL	MKK	N805HC	53	1
2	MKK	HNL	N806HC	34	1
3	DUT	ANC	N687PA	31	1
4	JHM	HNL	N804HC	24	1

In [533…

```
df_Flights_Unique_round_trips=pd.concat([df_Flights_Unique_round_tripsPart1,df_Flights_U
df_Flights_Unique_round_trips.sort_values('roundtrips',ascending=False)
```

Out[533]:

	origin	destination	tail_num	roundtrips	rn
0	HNL	MKK	N801HC	56	1
1	HNL	MKK	N805HC	53	1
0	DUT	ANC	N687PA	39	1
1	HNL	JHM	N805HC	35	1
2	MKK	HNL	N806HC	34	1
...	...	...	...	...	...
4177	ORD	BOS	N910NN	1	1
4178	ORD	BOS	N932AN	1	1
4179	ORD	BTV	N14558	1	1
4180	ORD	BTV	N408AW	1	1
5911	ATL	JFK	N173DZ	1	1

11820 rows × 5 columns

In [534…

```
## visualization
```

In [535…

```
dtale.show(df_Flights_Unique_round_trips)
```

Out[535]:

In [44]:

```
df_Flights_Unique_round_tripsTop=df_Flights_Unique_round_trips.sort_values('roundtrips',
df_Flights_Unique_round_tripsTop.head(10)
```

Out[44]:

	origin	destination	tail_num	roundtrips	rn
0	HNL	MKK	N801HC	56	1
1	HNL	MKK	N805HC	53	1
0	DUT	ANC	N687PA	39	1
1	HNL	JHM	N805HC	35	1
2	MKK	HNL	N806HC	34	1
2	HNL	MKK	N801HC	32	1
3	DUT	ANC	N687PA	31	1
3	DUT	ANC	N682PA	29	1
4	JHM	HNL	N804HC	24	1
4	DUT	ANC	N681PA	24	1

In [536…

```
configure_plotly_browser_state()
```

In [537…

```
def configure_plotly_browser_state():
  import IPython
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
        <script>
          requirejs.config({
            paths: {
              base: '/static/base',
              plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
            },
          });
        </script>
        '''))
```

In [539…

```
df_Flights_Unique_round_trips.profile_report()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Loading [MathJax]/extensions/Safe.js

Overview

Dataset statistics

Number of variables	5
Number of observations	11820
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	67
Duplicate rows (%)	0.6%
Total size in memory	461.8 KiB
Average record size in memory	40.0 B

Variable types

Categorical	4
Numeric	1

Alerts

`rn` has constant value "1"	**Constant**
Dataset has 67 (0.6%) duplicate rows	**Duplicates**
`origin` has a high cardinality: 310 distinct values	**High cardinality**
`destination` has a high cardinality: 308 distinct values	**High cardinality**

Out[539]:

We dont see any high correlation except arr delay and dept delay. That means if a flight departured late

from origin it arrives late to the destination port

visualizing the raw data of df_flights

Distribution plot for every numeric feature

In [541…

```
klib.dist_plot(df_Flights) # returns a distribution plot for every numeric feature
```

```
Large dataset detected, using 10000 random samples for the plots. Summary statistics are
still based on the entire dataset.
```

Out[541]:

```
<AxesSubplot: xlabel='occupancy_rate', ylabel='Density'>
```

In [542…

```
df_Flights.isnull().sum()
```

Out[542]:

```
fl_date                 0
op_carrier              0
tail_num                0
op_carrier_fl_num       0
origin_airport_id       0
origin                  0
dest_airport_id         0
destination             0
dep_delay               0
arr_delay            4377
cancelled               0
air_time             5027
distance              610
occupancy_rate        310
origin_city             0
origin_state            0
dest_city               0
dest_state              0
dtype: int64
```

In [543…

```
df_Flights.shape
```

Out[543]:

```
(1864272, 18)
```

Data Cleaning

Droping duplicates & empty rows and columns

In [546…

```
# df_Flights=klib.data_cleaning(df_Flights)
```

cleans and standardizes column names.

In [547…

```
# df_Flights=klib.clean_column_names(df_Flights)
```

In [548…

```
df_Flights=klib.convert_datatypes(df_Flights)
```

In [549…

```
df_Flights.dtypes
```

Loading [MathJax]/extensions/Safe.js

Out[549]:

```
fl_date                datetime64[ns]
op_carrier                   category
tail_num                     category
op_carrier_fl_num            category
origin_airport_id               int16
origin                       category
dest_airport_id                 int16
destination                  category
dep_delay                     float32
arr_delay                     float32
cancelled                     float32
air_time                     category
distance                     category
occupancy_rate                float32
origin_city                  category
origin_state                 category
dest_city                    category
dest_state                   category
dtype: object
```

In [550…

```
# But keeping date type column as datetime
```

In [551…

```
df_Flights['fl_date']=pd.to_datetime(df_Flights['fl_date'])
```

In [552…

```
df_Flights.head()
```

Out[552]:

	fl_date	op_carrier	tail_num	op_carrier_fl_num	origin_airport_id	origin	dest_airport_id	destination	dep_
0	2019-03-02	WN	N955WN	4591	14635	RSW	11042	CLE	
1	2019-03-02	WN	N8686A	3231	14635	RSW	11066	CMH	
2	2019-03-02	WN	N201LV	3383	14635	RSW	11066	CMH	
3	2019-03-02	WN	N413WN	5498	14635	RSW	11066	CMH	
4	2019-03-02	WN	N7832A	6933	14635	RSW	11259	DAL	

In [553…

```
klib.mv_col_handling(df_Flights)
```

Out[553]:

	fl_date	op_carrier	tail_num	op_carrier_fl_num	origin_airport_id	origin	dest_airport_id	destination
0	2019-03-02	WN	N955WN	4591	14635	RSW	11042	CLE
1	2019-03-02	WN	N8686A	3231	14635	RSW	11066	CMH
2	2019-03-02	WN	N201LV	3383	14635	RSW	11066	CMH
3	2019-03-02	WN	N413WN	5498	14635	RSW	11066	CMH
4	2019-03-02	WN	N7832A	6933	14635	RSW	11259	DAL
...	...	...	...	...	...	...	...	...
1915881	2019-03-23	AA	N903NN	1433	15370	TUL	11057	CLT
1915882	2019-03-24	AA	N965AN	1433	15370	TUL	11057	CLT
1915883	2019-03-25	AA	N979NN	1433	15370	TUL	11057	CLT
1915884	2019-03-26	AA	N872NN	1433	15370	TUL	11057	CLT
1915885	2019-03-27	AA	N945AN	1433	15370	TUL	11057	CLT

1864272 rows × 18 columns

Visualization of historgrams, frequency, value counts after cleaning

In [554…

```
configure_plotly_browser_state()
dtale.show(df_Flights)
```

Loading [MathJax]/extensions/Safe.js

Out[554]:

In [555…

```
def configure_plotly_browser_state():
  import IPython
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
        <script>
          requirejs.config({
            paths: {
              base: '/static/base',
              plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',
            },
          });
        </script>
        '''))
```

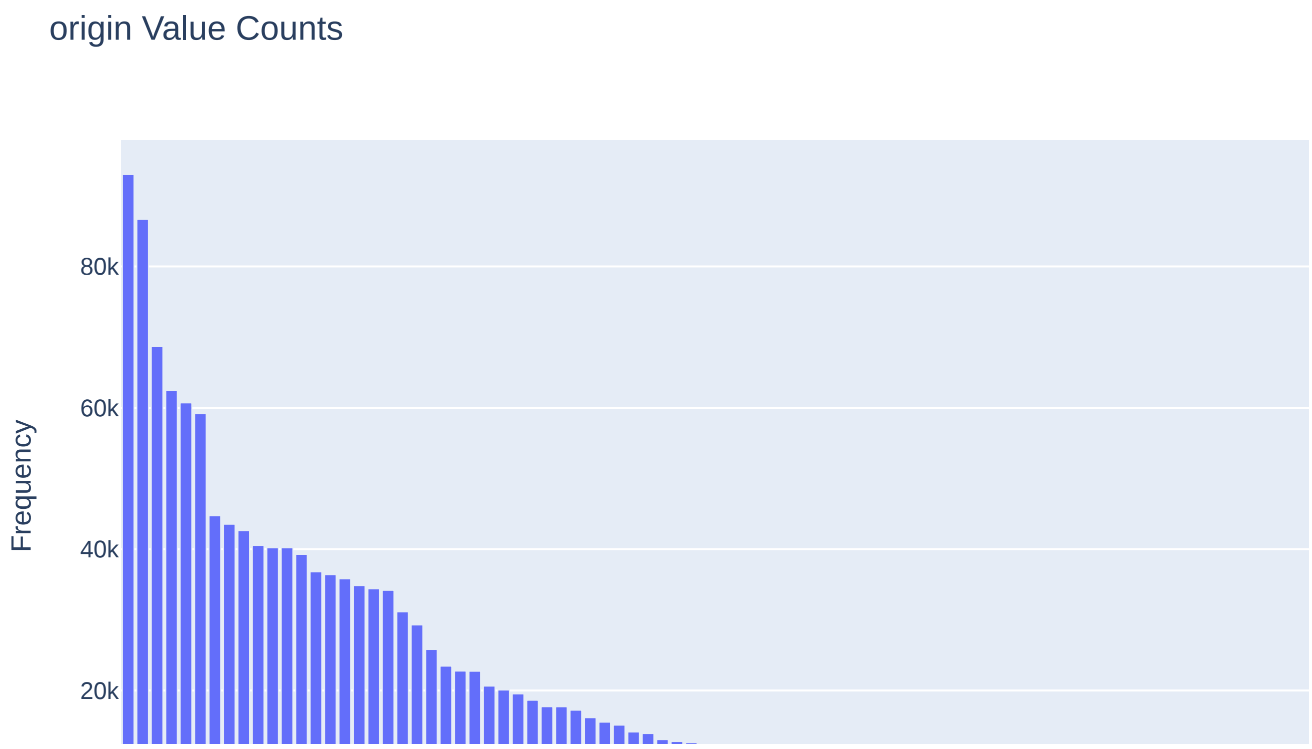
Visualizing the value counts of origin.

In [556…

```
import numpy as np
import pandas as pd
import plotly.graph_objs as go


if isinstance(df_Flights, (pd.DatetimeIndex, pd.MultiIndex)):
    df_Flights = df_Flights.to_frame(index=False)

# remove any pre-existing indices for ease of use in the D-Tale code, but this is not re
df_Flights = df_Flights.reset_index().drop('index', axis=1, errors='ignore')
df_Flights.columns = [str(c) for c in df_Flights.columns]  # update columns to strings i

    df_Flights[~pd.isnull(df_Flights['origin'])]['origin']
```

Loading [MathJax]/extensions/Safe.js

```
chart = pd.value_counts(s).to_frame(name='data')
chart.index.name = 'labels'
chart = chart.reset_index().sort_values(['data', 'labels'], ascending=[False, True])
chart = chart[:100]
charts = [go.Bar(x=chart['labels'].values, y=chart['data'].values, name='Frequency')]
figure = go.Figure(data=charts, layout=go.Layout({
    'barmode': 'group',
    'legend': {'orientation': 'h'},
    'title': {'text': 'origin Value Counts'},
    'xaxis': {'title': {'text': 'origin'}},
    'yaxis': {'title': {'text': 'Frequency'}}
}))


from plotly.offline import iplot, init_notebook_mode

init_notebook_mode(connected=True)
for chart in charts:
    chart.pop('id', None) # for some reason iplot does not like 'id'
configure_plotly_browser_state()
iplot(figure)
```



It is evident from the above ATL, ORD, DFW, DEN, CLT have more number of flights.

In [559…

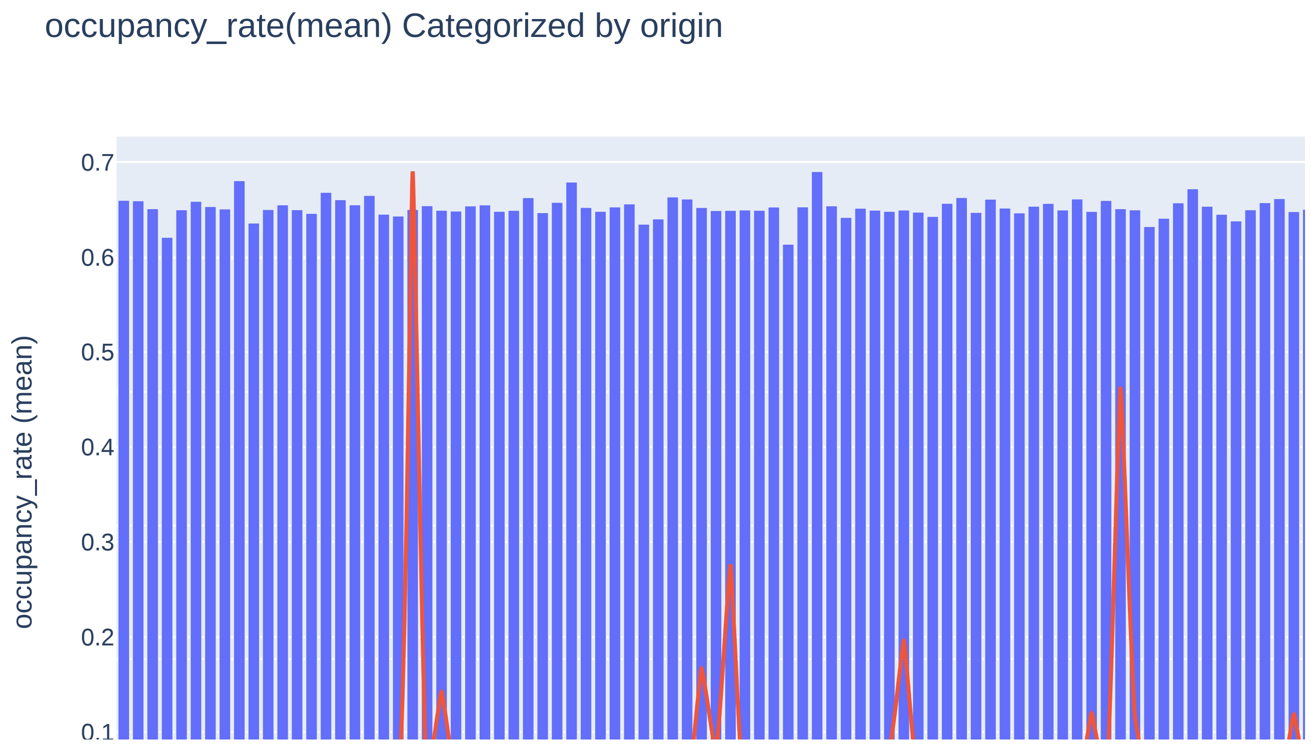
```
# Occupancy rate of origin
```

Loading [MathJax]/extensions/Safe.js

In [560…

```
# DISCLAIMER: 'df_Flights' refers to the data you passed in when calling 'dtale.show'

import numpy as np
import pandas as pd
import plotly.graph_objs as go

if isinstance(df_Flights, (pd.DatetimeIndex, pd.MultiIndex)):
    df_Flights = df_Flights.to_frame(index=False)

# remove any pre-existing indices for ease of use in the D-Tale code, but this is not re
df_Flights = df_Flights.reset_index().drop('index', axis=1, errors='ignore')
df_Flights.columns = [str(c) for c in df_Flights.columns]  # update columns to strings i

chart = df_Flights.groupby('origin')[['occupancy_rate']].agg(['count', 'mean'])
chart.columns = chart.columns.droplevel(0)
chart.columns = ["count", "data"]
chart.index.name = 'labels'
chart = chart.reset_index()
chart = chart[:100]
charts = [
    go.Bar(x=chart['labels'].values, y=chart['data'].values),
    go.Scatter(
        x=chart['labels'].values, y=chart['count'].values, yaxis='y2',
        name='Frequency', line={'shape': 'spline', 'smoothing': 0.3}, mode='lines'
)
]
figure = go.Figure(data=charts, layout=go.Layout({
    'barmode': 'group',
    'legend': {'orientation': 'h'},
    'title': {'text': 'occupancy_rate(mean) Categorized by origin'},
    'xaxis': {'title': {'text': 'origin'}},
    'yaxis': {'side': 'left', 'title': {'text': 'occupancy_rate (mean)'}},
    'yaxis2': {'overlaying': 'y', 'side': 'right', 'title': {'text': 'Frequency'}}
}))


from plotly.offline import iplot, init_notebook_mode

init_notebook_mode(connected=True)
for chart in charts:
    chart.pop('id', None) # for some reason iplot does not like 'id'
configure_plotly_browser_state()
iplot(figure)
```

Loading [MathJax]/extensions/Safe.js



It is evident from the above graph flights from CNY had highest mean occupancy of 68 percent.

ATL is the highest destination travelled and has mean occupancy rate of 64 percent, so increasing occupancy rate will

increase revenue

arrival delay with destinations

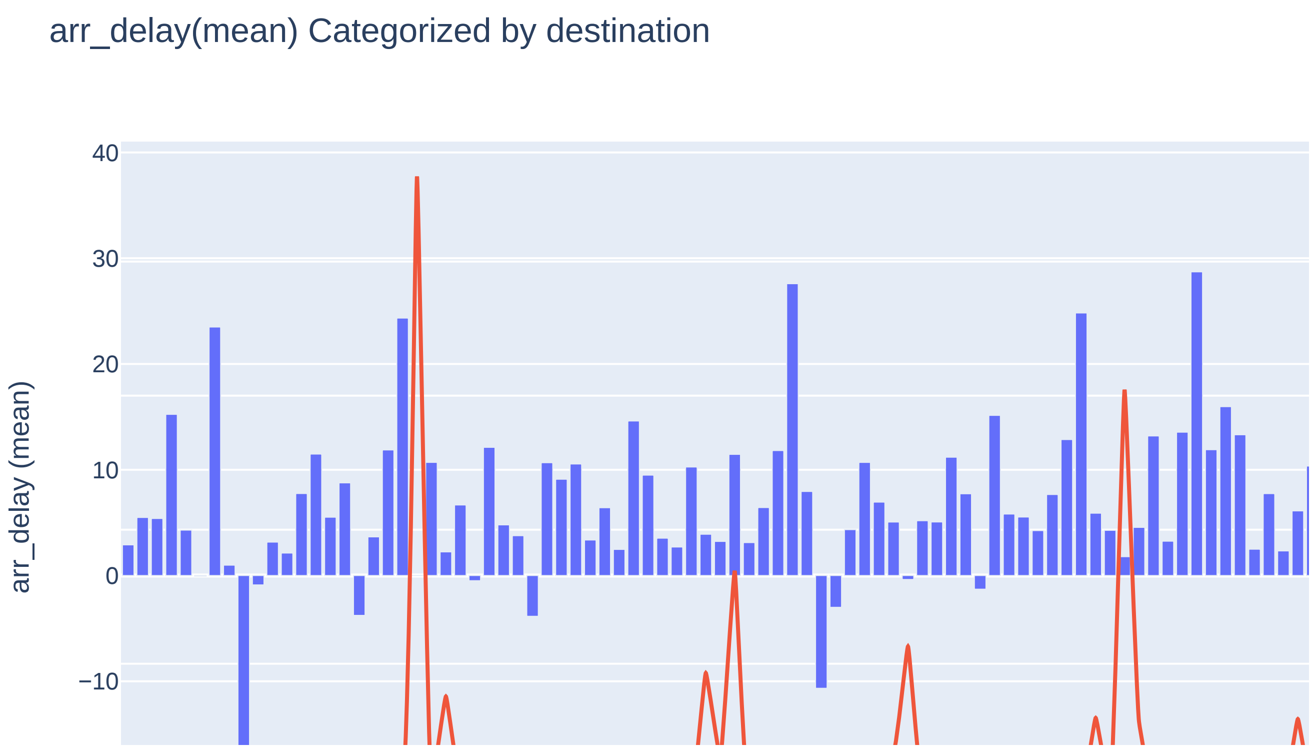
In [561…

```
# DISCLAIMER: 'df_Flights' refers to the data you passed in when calling 'dtale.show'

import numpy as np
import pandas as pd
import plotly.graph_objs as go

if isinstance(df_Flights, (pd.DatetimeIndex, pd.MultiIndex)):
    df_Flights = df_Flights.to_frame(index=False)
```

Loading [MathJax]/extensions/Safe.js

```python
# remove any pre-existing indices for ease of use in the D-Tale code, but this is not re
df_Flights = df_Flights.reset_index().drop('index', axis=1, errors='ignore')
df_Flights.columns = [str(c) for c in df_Flights.columns]  # update columns to strings i

chart = df_Flights.groupby('destination')[['arr_delay']].agg(['count', 'mean'])
chart.columns = chart.columns.droplevel(0)
chart.columns = ["count", "data"]
chart.index.name = 'labels'
chart = chart.reset_index()
chart = chart[:100]
charts = [
    go.Bar(x=chart['labels'].values, y=chart['data'].values),
    go.Scatter(
        x=chart['labels'].values, y=chart['count'].values, yaxis='y2',
        name='Frequency', line={'shape': 'spline', 'smoothing': 0.3}, mode='lines'
    )
]
figure = go.Figure(data=charts, layout=go.Layout({
    'barmode': 'group',
    'legend': {'orientation': 'h'},
    'title': {'text': 'arr_delay(mean) Categorized by destination'},
    'xaxis': {'title': {'text': 'destination'}},
    'yaxis': {'side': 'left', 'title': {'text': 'arr_delay (mean)'}},
    'yaxis2': {'overlaying': 'y', 'side': 'right', 'title': {'text': 'Frequency'}}
}))


from plotly.offline import iplot, init_notebook_mode

init_notebook_mode(connected=True)
for chart in charts:
    chart.pop('id', None) # for some reason iplot does not like 'id'
configure_plotly_browser_state()
iplot(figure)
```

Loading [MathJax]/extensions/Safe.js



from the above graph it is evident that DVL,DUT,DIK, COD,CKB,BRD, ASE, ACV hasan average of more than

15 minutes delay. Each additional minute of delay costs the airline $75. hence these are loss.

In [562…

```
s = df_Flights['arr_delay']
q1 = s.quantile(0.25)
q3 = s.quantile(0.75)
iqr = q3 - q1
iqr_lower = q1 - 1.5 * iqr
iqr_upper = q3 + 1.5 * iqr
outliers = dict(s[(s < iqr_lower) | (s > iqr_upper)])
# print(outliers)
```

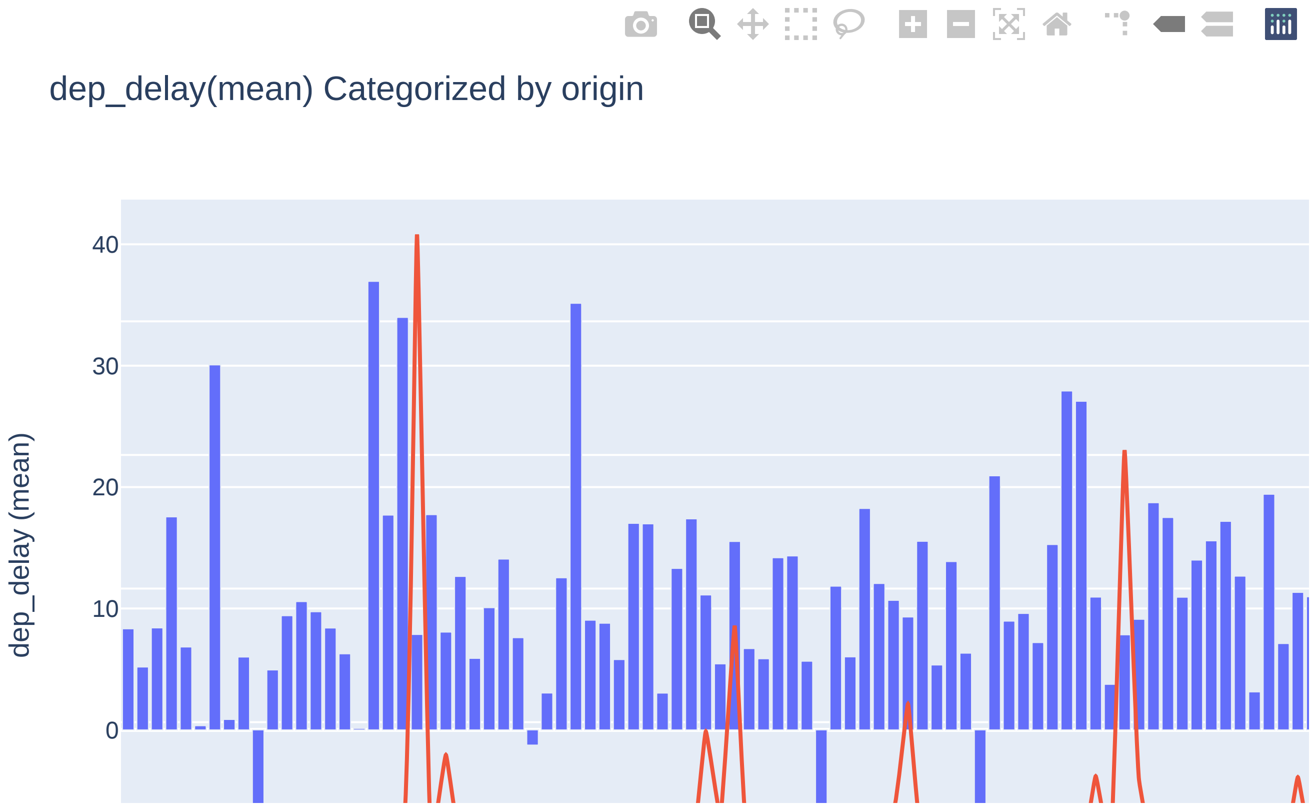
Departure delay with origins

In [564…

```
if isinstance(df_Flights, (pd.DatetimeIndex, pd.MultiIndex)):
    df_Flights = df_Flights.to_frame(index=False)

                         pre-existing indices for ease of use in the D-Tale code, but this is not re
```

```
df_Flights = df_Flights.reset_index().drop('index', axis=1, errors='ignore')
df_Flights.columns = [str(c) for c in df_Flights.columns]  # update columns to strings i

chart = df_Flights.groupby('origin')[['dep_delay']].agg(['count', 'mean'])
chart.columns = chart.columns.droplevel(0)
chart.columns = ["count", "data"]
chart.index.name = 'labels'
chart = chart.reset_index()
chart = chart[:100]
charts = [
    go.Bar(x=chart['labels'].values, y=chart['data'].values),
    go.Scatter(
        x=chart['labels'].values, y=chart['count'].values, yaxis='y2',
        name='Frequency', line={'shape': 'spline', 'smoothing': 0.3}, mode='lines'
    )
]
figure = go.Figure(data=charts, layout=go.Layout({
    'barmode': 'group',
    'legend': {'orientation': 'h'},
    'title': {'text': 'dep_delay(mean) Categorized by origin'},
    'xaxis': {'title': {'text': 'origin'}},
    'yaxis': {'side': 'left', 'title': {'text': 'dep_delay (mean)'}},
    'yaxis2': {'overlaying': 'y', 'side': 'right', 'title': {'text': 'Frequency'}}
}))
from plotly.offline import iplot, init_notebook_mode

init_notebook_mode(connected=True)
for chart in charts:
    chart.pop('id', None) # for some reason iplot does not like 'id'
configure_plotly_browser_state()
iplot(figure)
```

Loading [MathJax]/extensions/Safe.js



DVL on an avergae has a delay of 40 minutes approx.

Origins such as DVl, DIK, DUT, CYS,BGm etc more than 15 min delay in departing the origin

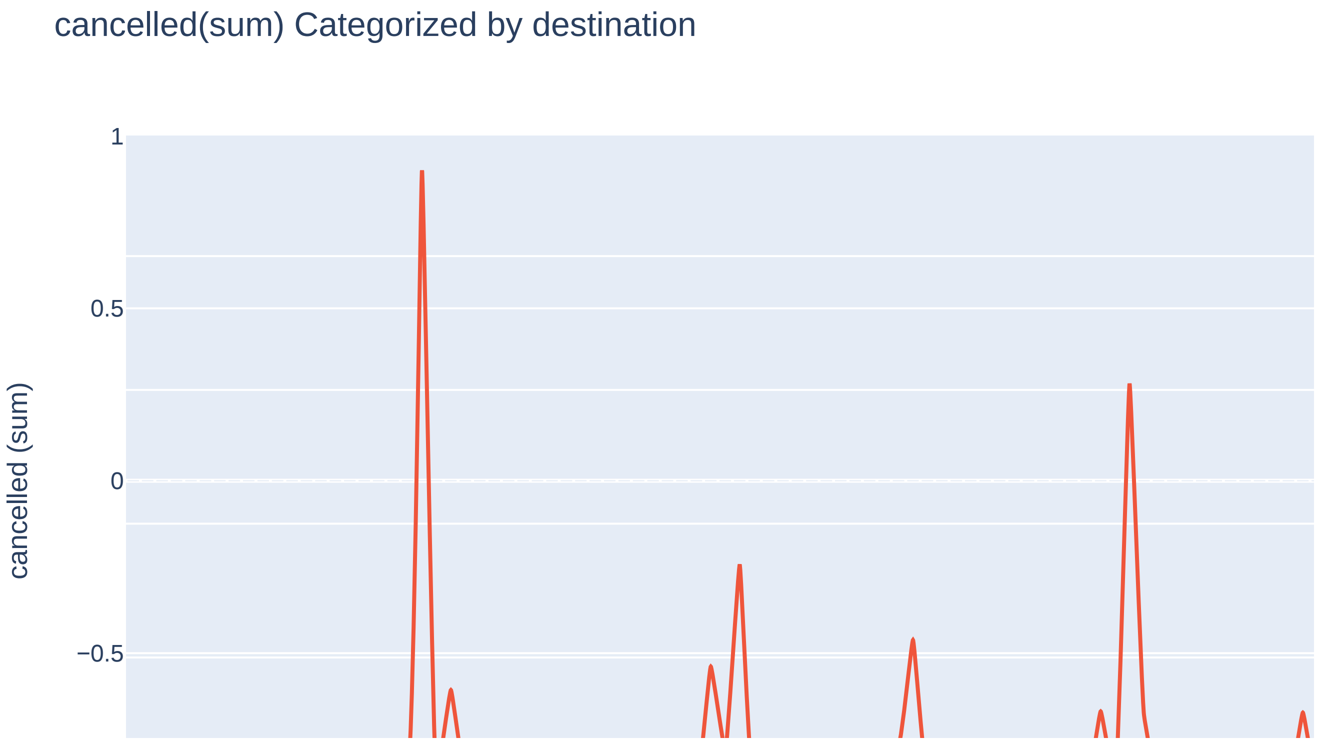
Cancelled flights to the destinations

In [566…

```
import numpy as np
import pandas as pd
import plotly.graph_objs as go

if isinstance(df_Flights, (pd.DatetimeIndex, pd.MultiIndex)):
    df_Flights = df_Flights.to_frame(index=False)

# remove any pre-existing indices for ease of use in the D-Tale code, but this is not re
df_Flights = df_Flights.reset_index().drop('index', axis=1, errors='ignore')
df_Flights.columns = [str(c) for c in df_Flights.columns]  # update columns to strings i

chart = df_Flights.groupby('destination')[['cancelled']].agg(['count', 'sum'])
chart.columns = chart.columns.droplevel(0)
chart.columns = ["count", "data"]
                         ame = 'labels'
```

Loading [MathJax]/extensions/Safe.js

```
chart = chart.reset_index()
chart = chart[:100]
charts = [
    go.Bar(x=chart['labels'].values, y=chart['data'].values),
    go.Scatter(
        x=chart['labels'].values, y=chart['count'].values, yaxis='y2',
        name='Frequency', line={'shape': 'spline', 'smoothing': 0.3}, mode='lines'
    )
]
figure = go.Figure(data=charts, layout=go.Layout({
    'barmode': 'group',
    'legend': {'orientation': 'h'},
    'title': {'text': 'cancelled(sum) Categorized by destination'},
    'xaxis': {'title': {'text': 'destination'}},
    'yaxis': {'side': 'left', 'title': {'text': 'cancelled (sum)'}},
    'yaxis2': {'overlaying': 'y', 'side': 'right', 'title': {'text': 'Frequency'}}
}))
from plotly.offline import iplot, init_notebook_mode

init_notebook_mode(connected=True)
for chart in charts:
    chart.pop('id', None) # for some reason iplot does not like 'id'
configure_plotly_browser_state()
iplot(figure)
```



from the above graph flights going to DEN, DCA, DFW etc destinations have the highest number of

Loading [MathJax]/extensions/Safe.js

cancellations.

Distance travelled the most to least

In [568…

```
df_Flights.shape
```

Out[568]:

```
(1864272, 18)
```

In [569…

```
df_Flights.isnull().sum()
```

Out[569]:

```
fl_date                0
op_carrier             0
tail_num               0
op_carrier_fl_num      0
origin_airport_id      0
origin                 0
dest_airport_id        0
destination            0
dep_delay              0
arr_delay           4377
cancelled              0
air_time            5027
distance             610
occupancy_rate       310
origin_city            0
origin_state           0
dest_city              0
dest_state             0
dtype: int64
```

tail_num has 12111 nan values. This is the field which cannnot be imputated based on median or mean. Hence retaining same

This means every CITY_NAME has a unique AIRPORT_ID

The 10 most profitable round trip routes (without considering the upfront airplane cost) in the quarter. Along with the profit, show total revenue, total cost, summary values of other key components and total round trip flights in the quarter for the top 10 most profitable routes. Exclude canceled flights from these calculations.

Round trip profit calculations

In [570…

```
df_Flights_total_parameters_round_trip=df_Flights[['tail_num','origin','destination','ar
df_Flights_total_parameters_round_trip
```

Out[570]:

	tail_num	origin	destination	arr_delay	dep_delay	distance	occupancy_rate
0	N955WN	RSW	CLE	-6.0	-8.0	1025.0	0.970000
1	N8686A	RSW	CMH	5.0	1.0	930.0	0.550000
2	N201LV	RSW	CMH	4.0	0.0	930.0	0.910000
3	N413WN	RSW	CMH	14.0	11.0	930.0	0.670000
4	N7832A	RSW	DAL	-17.0	0.0	1005.0	0.620000
...	...	...	...	...	...	...	...
1864267	N903NN	TUL	CLT	-6.0	-9.0	****	0.794885
1864268	N965AN	TUL	CLT	-1.0	-2.0	****	0.538399
1864269	N979NN	TUL	CLT	-25.0	-8.0	****	0.955579
1864270	N872NN	TUL	CLT	-6.0	-9.0	****	0.595344
1864271	N945AN	TUL	CLT	5.0	-8.0	****	0.350192

1864272 rows × 7 columns

In [571…

```
df_Flights_total_parameters_round_trip['ID']=range(0,len(df_Flights_total_parameters_rou
```

In [572…

```
df_Flights_total_parameters_round_trip.head()
```

Out[572]:

	tail_num	origin	destination	arr_delay	dep_delay	distance	occupancy_rate	ID
0	N955WN	RSW	CLE	-6.0	-8.0	1025.0	0.97	0
1	N8686A	RSW	CMH	5.0	1.0	930.0	0.55	1
2	N201LV	RSW	CMH	4.0	0.0	930.0	0.91	2
3	N413WN	RSW	CMH	14.0	11.0	930.0	0.67	3
4	N7832A	RSW	DAL	-17.0	0.0	1005.0	0.62	4

In [573…

```
df_Flights_total_parameters_round_trip.dropna(inplace=True)
df_Flights_total_parameters_round_trip = df_Flights_total_parameters_round_trip[df_Fligh
```

In [574…

```
def check(x):
    if x > 15 :
        val=x-15
        return val
    else:
        return 0
df_Flights_total_parameters_round_trip['arr_delay']=df_Flights_total_parameters_round_tr
```

In [575…

```
def check(x):
    if x > 15 :
        val=x-15
        return val
    else:
        return 0
df_Flights_total_parameters_round_trip['dep_delay']=df_Flights_total_parameters_round_tr
```

Loading [MathJax]/extensions/Safe.js

first chunk calculation..

In [576…

```
df_Flights_total_parameters_round_trip.to_csv("/Users/abhishekshastry/Documents/Intervie
```

In [577…

```
df_Flights_total_parameters_round_trip.shape
```

Out[577]:

```
(1844964, 8)
```

second chunk calculation..

In [578…

```
df_Flights_total_parameters_round_tripPart2=df_Flights_total_parameters_round_trip.iloc[
df_Flights_total_parameters_round_tripPart2=df_Flights_total_parameters_round_tripPart2[
```

In [579…

```
df_Flights_total_parameters_round_tripPart2.to_csv("/Users/abhishekshastry/Documents/Int
```

In [580…

```
df_Flights_total_parameters_round_tripPart2.tail()
```

Out[580]:

	tail_num	origin	destination	arr_delay	dep_delay	occupancy_rate	ID	distance
1849295	N254NN	SHV	DFW	1.0	2.0	0.51	1849295	190.0
1849296	N264NN	SHV	DFW	0.0	0.0	0.60	1849296	190.0
1849297	N223NN	SHV	DFW	0.0	0.0	0.32	1849297	190.0
1849298	N239NN	DFW	SHV	0.0	0.0	0.46	1849298	190.0
1849299	N251NN	DFW	SHV	0.0	0.0	0.53	1849299	190.0

reading both files and merging together..

In [581…

```
df_Flights_total_parameters_round_tripPart1UniqueCSV=pd.read_csv("/Users/abhishekshastry
df_Flights_total_parameters_round_tripPart1UniqueCSV.head()
```

Out[581]:

	origin	destination	tail_num	arrdelay	depdelay	occupancyrate	distance	roundtrips
0	DUT	ANC	N687PA	566	740	24.84	30888	39
1	HNL	JHM	N805HC	0	0	23.00	2940	35
2	MKK	HNL	N801HC	44	41	19.87	1728	32
3	DUT	ANC	N682PA	139	310	18.29	22968	29
4	DUT	ANC	N681PA	309	460	16.65	19008	24

In [582…

```
df_Flights_total_parameters_round_tripPart2UniqueCSV=pd.read_csv("/Users/abhishekshastry
df_Flights_total_parameters_round_tripPart2UniqueCSV.head()
```

Out[582]:

	origin	destination	tail_num	arrdelay	depdelay	occupancyrate	distance	roundtrips
0	HNL	MKK	N801HC	22	18	35.34	3024	56
1	HNL	MKK	N805HC	203	211	34.79	2862	53
2	HNL	MKK	N806HC	153	160	21.53	1836	34
3	DUT	ANC	N687PA	576	788	20.72	24552	31
4	JHM	HNL	N804HC	2	3	14.57	2016	24

Loading [MathJax]/extensions/Safe.js

In [583…

```
df_Flights_total_parameters_Unique_round_trip=pd.concat([df_Flights_total_parameters_rou
df_Flights_total_parameters_Unique_round_trip
```

Out[583]:

	origin	destination	tail_num	arrdelay	depdelay	occupancyrate	distance	roundtrips
0	DUT	ANC	N687PA	566	740	24.84	30888	39
1	HNL	JHM	N805HC	0	0	23.00	2940	35
2	MKK	HNL	N801HC	44	41	19.87	1728	32
3	DUT	ANC	N682PA	139	310	18.29	22968	29
4	DUT	ANC	N681PA	309	460	16.65	19008	24
...	...	...	...	...	...	...	...	...
5905	ATL	GNV	N633SK	0	0	0.64	300	1
5906	ATL	GPT	N295PQ	38	33	0.80	352	1
5907	ATL	GRR	N818DA	0	0	0.62	640	1
5908	ATL	GRR	N865DN	0	0	0.32	640	1
5909	ATL	GRR	N902DE	0	0	0.63	640	1

11818 rows × 8 columns

In [584…

```
df_Flights_Unique_round_trips=df_Flights_total_parameters_Unique_round_trip.groupby(['or
df_Flights_Unique_round_trips.sort_values(by='occupancyrate',ascending=False)
```

Out[584]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips
1284	DUT	ANC	117.33	2135	3127	140976	178
1578	HNL	MKK	99.74	453	464	8316	154
2029	LAX	DFW	63.62	162	284	119795	97
2044	LAX	JFK	53.71	375	545	193050	78
1359	EWR	LAX	41.04	904	641	157056	64
...	...	...	...	...	...	...	...
866	DAY	DCA	0.31	0	0	391	1
1085	DFW	CRP	0.31	0	0	354	1
2531	MSP	RDU	0.30	0	0	980	1
481	BUR	HOU	0.30	0	0	1389	1
1614	HOU	OAK	0.30	6	0	1642	1

3798 rows × 7 columns

In [585…

```
df_Flights_Unique_round_trips.sort_values(by='occupancyrate',ascending=False)
```

Out[585]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips
1284	DUT	ANC	117.33	2135	3127	140976	178
1578	HNL	MKK	99.74	453	464	8316	154
2029	LAX	DFW	63.62	162	284	119795	97
2044	LAX	JFK	53.71	375	545	193050	78
1359	EWR	LAX	41.04	904	641	157056	64
...	...	...	...	...	...	...	...
866	DAY	DCA	0.31	0	0	391	1
1085	DFW	CRP	0.31	0	0	354	1
2531	MSP	RDU	0.30	0	0	980	1
481	BUR	HOU	0.30	0	0	1389	1
1614	HOU	OAK	0.30	6	0	1642	1

3798 rows × 7 columns

Calculating the total expenditure assciated with each flights

In [586…

```
df_Flights_Unique_round_trips['Fuel Maintainance']=df_Flights_Unique_round_trips['distan
```

In [587…

```
df_Flights_Unique_round_trips.head()
```

Out[587]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance
0	ABE	ATL	2.41	0	0	2768	4	22144
1	ABE	ORD	0.62	15	11	654	1	5232
2	ABE	SFB	0.61	143	108	882	1	7056
3	ABQ	ATL	2.03	0	0	5076	4	40608
4	ABQ	AUS	0.98	0	0	619	1	4952

In [588…

```
df_Flights_Unique_round_trips['Insurance']=df_Flights_Unique_round_trips['distance'].app
```

In [589…

```
df_Flights_Unique_round_trips.head()
```

Out[589]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24
1	ABE	ORD	0.62	15	11	654	1	5232	771.72
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42

In [590…

```
df_Flights_Unique_round_trips['Arrival_Delay_Charges']=df_Flights_Unique_round_trips['ar
```

```
df_Flights_Unique_round_trips.head()
```

Loading [MathJax]/extensions/Safe.js

Out[591]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [592…

```
df_Flights_Unique_round_trips['Departure_Delay_Charges']=df_Flights_Unique_round_trips['
```

In [593…

```
df_Flights_Unique_round_trips.head()
```

Out[593]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [594…

```
df_Flights_Unique_round_trips['Number_of_passengers']=df_Flights_Unique_round_trips['occ
```

In [595…

```
df_Flights_Unique_round_trips.head()
```

Out[595]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [596…

```
df_Flights_Unique_round_trips['Baggage Fees']=df_Flights_Unique_round_trips['Number_of_p
```

In [597…

```
df_Flights_Unique_round_trips.head()
```

Out[597]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

Airport Codes

In [598…

```
df_Airport_Codes=pd.read_csv('Airport_Codes.csv')
df_Airport_Codes=df_Airport_Codes.dropna(subset=['IATA_CODE'])
df_Airport_Codes.head()
```

Out[598]:

	TYPE	NAME	ELEVATION_FT	CONTINENT	ISO_COUNTRY	MUNICIPALITY	IATA_CODE	CO
223	small_airport	Utirik Airport	4.0	OC	MH	Utirik Island	UTK	169.85
440	small_airport	Ocean Reef Club Airport	8.0	NaN	US	Key Largo	OCA	-80.27 25.32
594	small_airport	Pilot Station Airport	305.0	NaN	US	Pilot Station	PQS	
673	small_airport	Crested Butte Airpark	8980.0	NaN	US	Crested Butte	CSE	
1088	small_airport	LBJ Ranch Airport	1515.0	NaN	US	Johnson City	JCY	-98.6224 30.2518

In [599…

```
df_Airport_Codes.isna().sum()
```

Out[599]:

```
TYPE               0
NAME               0
ELEVATION_FT     352
CONTINENT       2978
ISO_COUNTRY       31
MUNICIPALITY     761
IATA_CODE          0
COORDINATES        0
dtype: int64
```

In [600…

```
df_Airport_Codes['TYPE'].unique()
```

Out[600]:

```
array(['small_airport', 'seaplane_base', 'closed', 'medium_airport',
       'heliport', 'large_airport'], dtype=object)
```

In [601…

```
dictvalues=dict()
for x in range(len(df_Airport_Codes.index)):
    typevalue=df_Airport_Codes.iloc[x:x+1,0].values[0]
    iata_code=df_Airport_Codes.iloc[x:x+1,1].values[0]
    dictvalues.update({typevalue:iata_code})
```

In [602…

```
def getdictvalues(x):
    if x in dictvalues:
        print('inside if')
        return dictvalues.get(x)
    else:
        return 'medium_airport'
```

In [603…

```
df_Flights_Unique_round_trips['origin_airport_size']=df_Flights_Unique_round_trips['orig
```

In [604…

```
df_Flights_Unique_round_trips.head()
```

Loading [MathJax]/extensions/Safe.js

Out[604]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [605…

```
df_Flights_Unique_round_trips['Destination_airport_size']=df_Flights_Unique_round_trips[
```

In [606…

```
df_Flights_Unique_round_trips.head()
```

Out[606]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [607…

```
# Assuming if IATA codes does not match then airport is considered medium sized airport
```

In [608…

```
def airport_size_cost(x):
    if x=='medium_airport':
        return 5000
    elif x=='large_airport':
        return 10000
    else:
        return 0
```

In [609…

```
df_Flights_Unique_round_trips['origin_airport_charges']=df_Flights_Unique_round_trips['o
```

In [610…

```
df_Flights_Unique_round_trips['Destination_airport_charges']=df_Flights_Unique_round_tri
```

In [611…

```
df_Flights_Unique_round_trips.head()
```

Out[611]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

All the above features like occupancyrate,arrdelay ,depdelay, distance are already in the form of total round trips.Hence calculation of Fuel Maintainance,

Insurance,Arrival_Delay_Charges,Departure_Delay_C need not be multiplied by number of round trips.

But origin_airport_size, Destination_airport_size need to be multiplied by number of total round trips

In [612…

```
df_Flights_Unique_round_trips['Total_airport_charges']=(df_Flights_Unique_round_trips['o
```

In [613…

```
df_Flights_Unique_round_trips.head()
```

Out[613]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

Considering Fuel charges, Insurance charges, Airport operational costs , arrival and departure delays, baggage fees for a round trip flight,

In [614…

```
df_Flights_Unique_round_trips['Total_Expenditure']=df_Flights_Unique_round_trips[['Fuel
```

In [615…

```
df_Flights_RoundTrips_Expenditure=df_Flights_Unique_round_trips
df_Flights_RoundTrips_Expenditure.head()
```

Out[615]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [616…

```
# visualization
```

In [617…

```
dtale.show(df_Flights_Unique_round_trips)
```

Out[617]:

In [618…

```
df_Flights_RoundTrips_Expenditure.profile_report()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	19
Number of observations	3798
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	563.9 KiB
Average record size in memory	152.0 B

Variable types

Categorical	6
Numeric	13

Alerts

`origin_airport_size` has constant value "medium_airport"	**Constant**
`Destination_airport_size` has constant value "medium_airport"	**Constant**
`origin_airport_charges` has constant value "5000"	**Constant**
`Destination_airport_charges` has constant value "5000"	**Constant**

Out[618]:

. Total tickets cost

In [619…
```
df_tickets=pd.read_csv('/Users/abhishekshastry/Documents/Interview_takehomes/capitalone/
df_tickets=df_tickets[df_tickets['ROUNDTRIP']==1]
```

In [620…
```
df_tickets.head()
```

Loading [MathJax]/extensions/Safe.js

Out[620]:

	ITIN_ID	YEAR	QUARTER	ORIGIN	ORIGIN_COUNTRY	ORIGIN_STATE_ABR	ORIGIN_STATE_NM
0	201912723049	2019	1	ABI	US	TX	Texas
1	201912723085	2019	1	ABI	US	TX	Texas
2	201912723491	2019	1	ABI	US	TX	Texas
3	201912723428	2019	1	ABI	US	TX	Texas
10	201912723337	2019	1	ABI	US	TX	Texas

The above data tells that in quarter 1 in the year 2019 there are 7 passengers with 168 ticket

price travelling round trip from RSW and CLE and CLE and RSW.It would be any flight

In [621…

```
df_tickets[['ORIGIN','DESTINATION','ITIN_FARE']]
```

Out[621]:

	ORIGIN	DESTINATION	ITIN_FARE
0	ABI	DAB	736.0
1	ABI	COS	570.0
2	ABI	MCO	564.0
3	ABI	LGA	345.0
10	ABI	JAX	1647.0
...	...	...	...
1167275	YAK	ANC	11.0
1167277	YAK	ANC	489.0
1167279	YAK	ANC	493.0
1167281	YAK	JNU	371.0
1167284	YAK	JNU	299.0

708600 rows × 3 columns

In [622…

```
df_tickets_unique=df_tickets[['ORIGIN','DESTINATION','ITIN_FARE']].drop_duplicates()
df_tickets_unique.dtypes
```

Out[622]:

```
ORIGIN         object
DESTINATION    object
ITIN_FARE      object
dtype: object
```

In [623…

```
df_tickets_unique['ITIN_FARE'] = df_tickets_unique['ITIN_FARE'].str.replace(r'[^0-9]+',
```

In [624…

```
df_tickets_unique['ITIN_FARE']=df_tickets_unique['ITIN_FARE'].astype(float)
```

In [625…

```
df_tickets_unique=df_tickets_unique.groupby(['ORIGIN','DESTINATION'],as_index=False)['IT
```

In [626…

```
df_tickets_unique.rename(columns={'ORIGIN':'origin','DESTINATION':'destination'},inplace
```

Loading [MathJax]/extensions/Safe.js

In [627…
```
df_tickets_unique.head()
```

Out[627]:

	origin	destination	ITIN_FARE
0	ABE	ABQ	10680.0
1	ABE	AGS	2990.0
2	ABE	AMA	6540.0
3	ABE	ASE	14840.0
4	ABE	ATL	253580.0

In [628…
```
df_Flights_Unique_round_trips.head()
```

Out[628]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [629…
```
df_merged_Unique_Fair=df_Flights_Unique_round_trips.merge(df_tickets_unique, how='inner'
df_merged_Unique_Fair.head()
```

Out[629]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

In [630…
```
df_merged_Unique_Fair['ITIN_FARE'] = df_merged_Unique_Fair['ITIN_FARE'].apply(np.ceil)
```

In [631…
```
df_merged_Unique_Fair.shape
```

Out[631]:
```
(3753, 20)
```

'ITIN_FARE in the tickets code is given for one person. Hence it has to multiplied for numbe rof passengers.

In [632…
```
df_merged_Unique_Fair['total_ITIN_FARE']=df_merged_Unique_Fair['ITIN_FARE']*df_merged_Un
```

In [633…
```
df_merged_Unique_Fair.shape
```

Out[633]:
```
(3753, 21)
```

```
que_Fair['Profit'] =df_merged_Unique_Fair.apply(lambda x: x['total_ITIN_FAR
df_merged_Unique_Fair.head()
```

Out[634]:

	origin	destination	occupancyrate	arrdelay	depdelay	distance	roundtrips	Fuel Maintainance	Insurance	Arriv
0	ABE	ATL	2.41	0	0	2768	4	22144	3266.24	
1	ABE	ORD	0.62	15	11	654	1	5232	771.72	
2	ABE	SFB	0.61	143	108	882	1	7056	1040.76	
3	ABQ	ATL	2.03	0	0	5076	4	40608	5989.68	
4	ABQ	AUS	0.98	0	0	619	1	4952	730.42	

5 rows × 22 columns

In [635…

```
df_merged_Unique_Fair=df_merged_Unique_Fair[df_merged_Unique_Fair['Profit']>0]
df_merged_Unique_Fair.shape
```

Out[635]:

```
(3750, 22)
```

In [636…

```
df_merged_Unique_Fair['Profit']=df_merged_Unique_Fair['Profit'].astype(int)
```

In [637…

```
df_merged_Total_Unique_Fair=df_merged_Unique_Fair[['origin','destination','Profit']].sor
df_merged_Total_Unique_Fair.head(10)
```

Out[637]:

	origin	destination	Profit
2016	LAX	JFK	60351574351
1864	JFK	LAX	46416598936
1890	JFK	SFO	39406382943
1342	EWR	LAX	32917422750
1984	LAX	ATL	21748713967
1370	EWR	SFO	20069761887
135	ATL	LAX	17957901082
2001	LAX	DFW	16632172891
1703	IAH	EWR	12234331789
1339	EWR	IAH	11858786731

These are the top 10 most profitable round trips.

In [638…

```
dtale.show(df_merged_Unique_Fair)
```

Loading [MathJax]/extensions/Safe.js

```
Out[638]:
```

```
In [639…  df_merged_Unique_Fair.profile_report()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]
```

Overview

Dataset statistics

Number of variables	22
Number of observations	3750
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	673.8 KiB
Average record size in memory	184.0 B

Variable types

Categorical	6
Numeric	16

Alerts

`origin_airport_size` has constant value "medium_airport"	**Constant**
`Destination_airport_size` has constant value "medium_airport"	**Constant**
`origin_airport_charges` has constant value "5000"	**Constant**
`Destination_airport_charges` has constant value "5000"	**Constant**

Out[639]:

Loading [MathJax]/extensions/Safe.js