Working_File

November 28, 2022

[1]:
```
import pandas as pd
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession
appName = "PySparkFlight_tbl"
master = "local"


# Create Spark session
spark = SparkSession.builder \
    .appName(appName) \
    .master(master) \
    .enableHiveSupport() \
    .getOrCreate()

#spark.sparkContext.setLogLevel("WARN")


# Create DF by reading from Hive
df = spark.sql("select * from f_db.flight_analysis2;")
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
22/11/28 22:28:00 INFO org.apache.spark.SparkEnv: Registering MapOutputTracker
22/11/28 22:28:00 INFO org.apache.spark.SparkEnv: Registering BlockManagerMaster
22/11/28 22:28:00 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMasterHeartbeat
22/11/28 22:28:00 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
ivysettings.xml file not found in HIVE_HOME or
HIVE_CONF_DIR,/etc/hive/conf.dist/ivysettings.xml will be used
```

[2]:
```
df=df.toPandas()
print("PySpark Dataframe converted to Pandas Dataframe")
```

```
PySpark Dataframe converted to Pandas Dataframe
```

[3]:
```
df_lad= spark.sql("select late_aircraft_delay from f_db.ontimerep")
```

[4]:
```
df_lad=df_lad.toPandas()
```

```
22/11/28 22:29:10 WARN org.apache.spark.sql.catalyst.util.package: Truncated the
string representation of a plan since it was too large. This behavior can be
adjusted by setting 'spark.sql.debug.maxToStringFields'.
```

[5]:
```
df_dest= spark.sql("select dest_city_name from f_db.ontimerep;")

df_dest=df_dest.toPandas()
```

[6]:
```
df['late_aircraft_delay']=df_lad['late_aircraft_delay']
df['dest_city_name']=df_dest['dest_city_name']
```

[7]:
```
df.reset_index(inplace = True, drop = True)
df['dte'] = pd.to_datetime(df['dte']) # date to datetime datatype
df[['op_carrier_fl_num',␣
 ↪'origin_airport_id','dep_time','dep_delay_new','dep_del15','distance','carrier_delay','weat
 ↪= df[['op_carrier_fl_num',␣
 ↪'origin_airport_id','dep_time','dep_delay_new','dep_del15','distance','carrier_delay','weat
 ↪apply(pd.to_numeric)
```

[14]:
```
#pd.DataFrame.hist(df.op_carrier_fl_num)
```

[8]:
```
#Exploratory Questions
#Q1: Which carriers are most and least reliable for on-time departure?
#Q2: Which airports are best and worst for on-time departures?

df_dep_delay=df[df.dep_delay_new>0.0]
df_dep_delay=df_dep_delay[["dep_delay_new","carrier_name",'display_airport_name']]
```

[112]:
```
#df_dep_delay.info()
```

[9]:
```
#Answer of Q1

len(pd.unique(df_dep_delay.carrier_name))
len(pd.unique(df.carrier_name))
#if the above two results match then execute next line of code

carriers=round(((df_dep_delay.groupby(by='carrier_name').size()/df.
 ↪groupby(by='carrier_name').size())*100),2).sort_values()

carriers=carriers.reset_index()
#carriers.info()
```

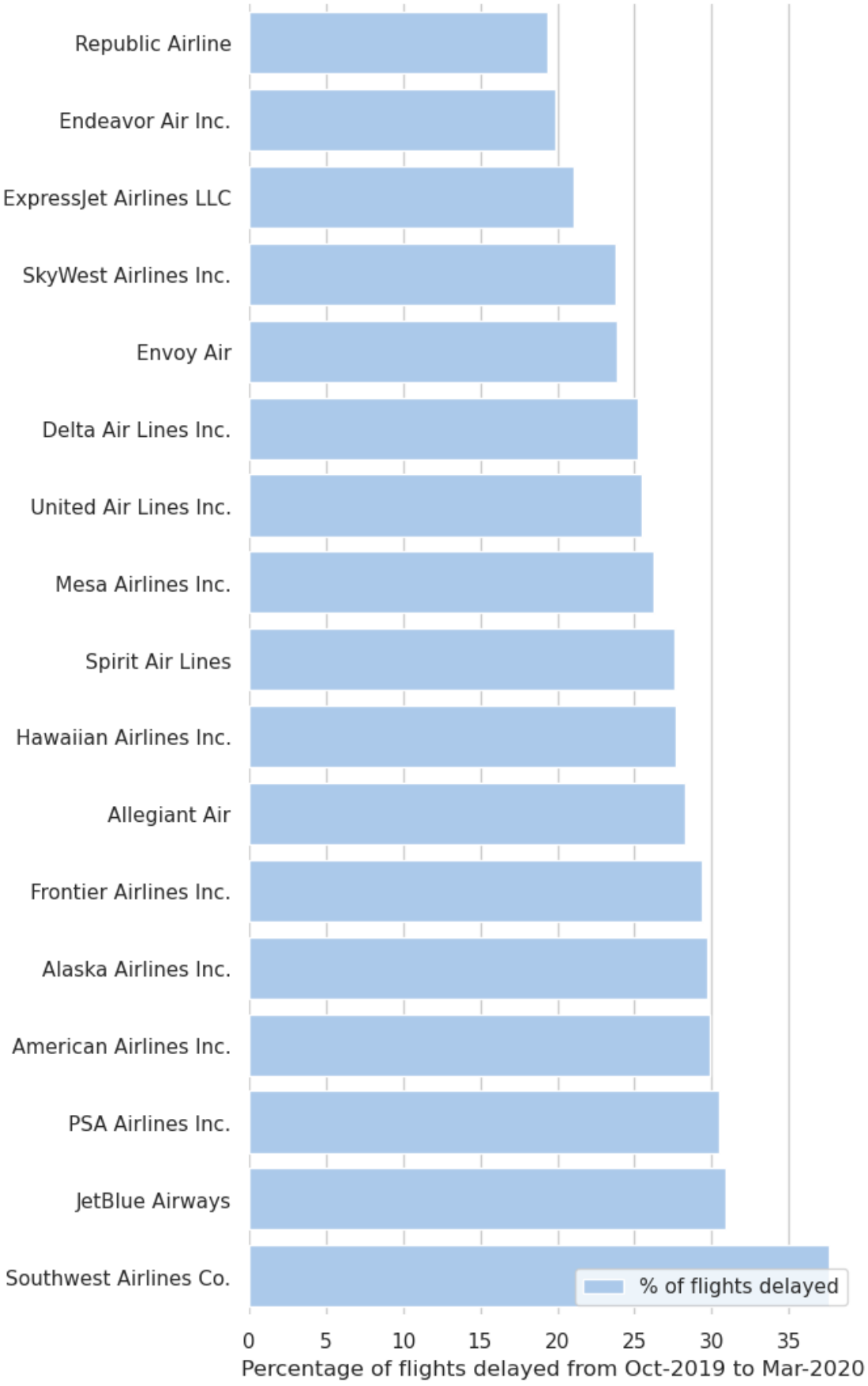
```
carriers=carriers.rename(columns={0:"% of flights delayed"})
```

[61]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import voila
#!voila --show_tracebacks=True
sns.set_theme(style="whitegrid")
f,ax = plt.subplots(figsize=(6,13))

sns.set_color_codes("pastel")

sns.barplot(x="% of flights delayed", y="carrier_name", data=carriers,
            label="% of flights delayed", color="b")

ax.legend(ncol=2, loc="lower right", frameon=True)
ax.set(ylabel="",
       xlabel="Percentage of flights delayed from Oct-2019 to Mar-2020")
sns.despine(left=True, bottom=True)
```


Republic Airline
Endeavor Air Inc.
ExpressJet Airlines LLC
SkyWest Airlines Inc.
Envoy Air
Delta Air Lines Inc.
United Air Lines Inc.
Mesa Airlines Inc.
Spirit Air Lines
Hawaiian Airlines Inc.
Allegiant Air
Frontier Airlines Inc.
Alaska Airlines Inc.
American Airlines Inc.
PSA Airlines Inc.
JetBlue Airways
Southwest Airlines Co.
% of flights delayed
0
5
10
15
20
25
30
35
Percentage of flights delayed from Oct-2019 to Mar-2020

[10]:
```
#Answer to Q2 :Begin
len(pd.unique(df_dep_delay.display_airport_name)) #result 350
len(pd.unique(df.display_airport_name)) #result 351

#find missing row
for i in pd.unique(df.display_airport_name) :
  if i not in pd.unique(df_dep_delay.display_airport_name):
       print(i)
```

[13]:
```
#Need to remove 'Yellowstone Regional' from df dataframe

df=df.drop(df[df.display_airport_name=='Yellowstone Regional'].index)

#len(pd.unique(df.display_airport_name))

#((df_dep_delay.groupby(by='display_airport_name').size()/df.
 ↪groupby(by='display_airport_name').size())*100).sort_values()
```

[14]:
```
#Create new dataframe from above results. We'll plot this data in our map
df_q2=df[['display_airport_name', 'latitude', 'longitude']]
df_q2.drop_duplicates(subset=['display_airport_name'],inplace=True)

temp=df.groupby(by='display_airport_name').size()
df_q2['Total Flights']=temp[df_q2.display_airport_name].values

temp=df_dep_delay.groupby(by='display_airport_name').size()
df_q2['No of flights delayed']=temp[df_q2.display_airport_name].values

temp=(df_dep_delay.groupby(by='display_airport_name').size()/df.
 ↪groupby(by='display_airport_name').size())*100
df_q2['% of flights delayed']=temp[df_q2.display_airport_name].values

#df_q2.iloc[349]
```

```
/tmp/ipykernel_12959/2011631351.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_q2.drop_duplicates(subset=['display_airport_name'],inplace=True)
/tmp/ipykernel_12959/2011631351.py:6: 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
  df_q2['Total Flights']=temp[df_q2.display_airport_name].values
/tmp/ipykernel_12959/2011631351.py:9: 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
  df_q2['No of flights delayed']=temp[df_q2.display_airport_name].values
/tmp/ipykernel_12959/2011631351.py:12: 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
  df_q2['% of flights delayed']=temp[df_q2.display_airport_name].values
```

[15]:
```
df_q2['display_airport_name']=df_q2.display_airport_name+" Airport"
```

```
/tmp/ipykernel_12959/2672709948.py:1: 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
  df_q2['display_airport_name']=df_q2.display_airport_name+" Airport"
```

[16]:
```
import folium

m = folium.Map(location=[47.116386,-101.299591],zoom_start=4)
df_q2_len=df_q2.head(50)
for index in range(len(df_q2_len)):

    iframe = folium.IFrame('<b>Total number of flights in  2019: <b>'+str(df_q2.
 ↪iloc[index][3])+"<br><br><b>No of fligths delayed in 2019:<b> "+str(df_q2.
 ↪iloc[index][4])+"<br><br><b>Percentage of flights delayed in 2019:<b>␣
 ↪"+str(round(df_q2.iloc[index][5],2)))

    popup=folium.Popup(iframe,min_width=300,max_width=300)

    folium.Marker([df_q2.iloc[index][1], df_q2.iloc[index][2]], popup=popup,␣
 ↪tooltip=df_q2.iloc[index][0]
                ).add_to(m)
```

```
m
```

[16]:
```
<folium.folium.Map at 0x7f1c6c0807c0>
```

[17]:
```
correlations=df.corr().round(2)
#correlations.dep_delay_new
```

[18]:
```
dep_corr=correlations.dep_delay_new

dep_corr=dep_corr.reset_index()
```

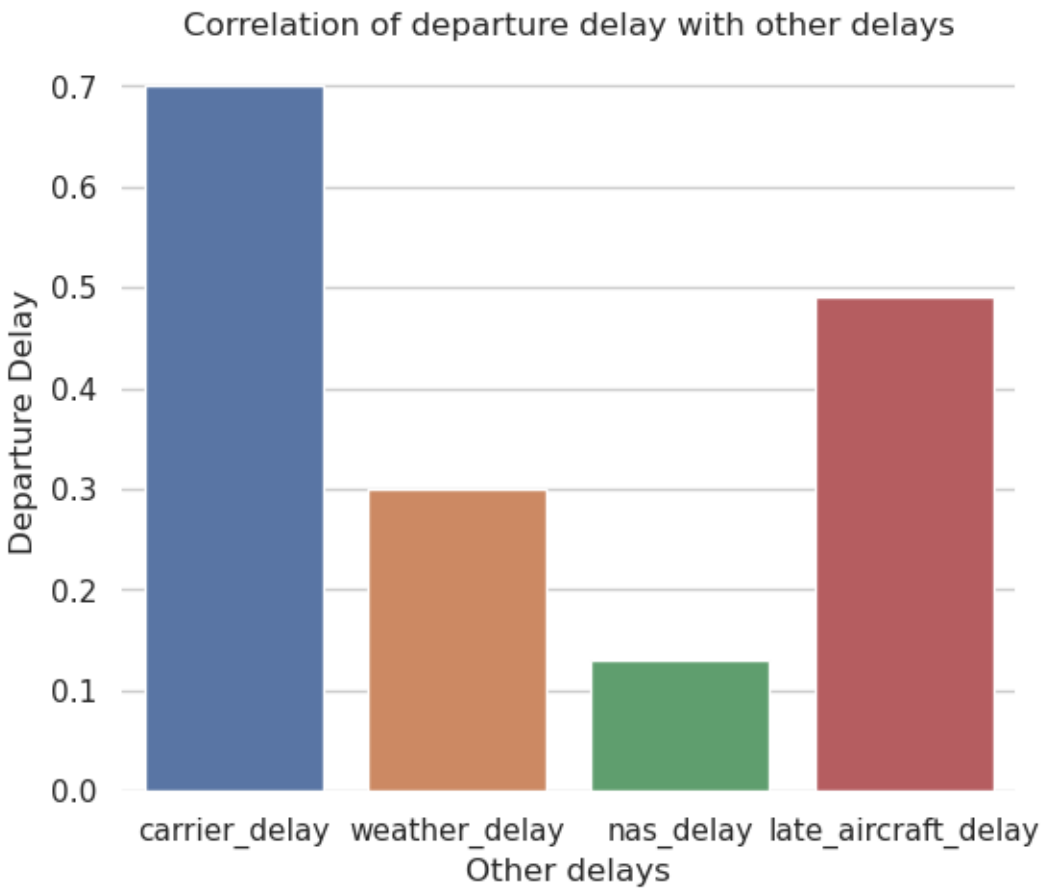
[20]:
```
#dep_corr
```

[21]:
```
#dep_corr.iloc[[6,7,8,16]]
```

[23]:
```
import seaborn as sns
sns.set_theme(style="whitegrid")
f,ax = plt.subplots(figsize=(6,5))

#bar_p=sns.barplot(data=dep_corr.iloc[[6,7,8,16]], x="index", y="dep_delay_new")

sns.set_color_codes("pastel")
sns.barplot(data=dep_corr.iloc[[6,7,8,16]], x="index", y="dep_delay_new")
ax.set(ylabel='Departure Delay', xlabel="Other delays",title='Correlation of␣
 ↪departure delay with other delays')
sns.despine(left=True, bottom=True)
```



[30]:
```
df_carrier=df.groupby(by=['carrier_name','origin_city_name','dest_city_name']).
  ↪size()
```

[33]:
```
df_carrier=df_carrier.reset_index()
df_carrier.rename(columns={0:"No of Flights"},inplace=True)
```

[34]:
```
df_dep_delay_carrier=df[df.dep_delay_new>0.0].
  ↪groupby(by=['carrier_name','origin_city_name','dest_city_name']).size()
df_dep_delay_carrier=df_dep_delay_carrier.reset_index()
df_dep_delay_carrier.rename(columns={0:"No of Flights"},inplace=True)
```

[35]:
```
#output=pd.
  ↪DataFrame(columns=['carrier_name','origin_city_name','dest_city_name','flight_delay'])
#output=output.append(df_carrier[df_carrier.carrier_name=="United Air Lines Inc.
  ↪"].sort_values(by='No of Flights',ascending=False).
  ↪head(5)[['carrier_name','origin_city_name','dest_city_name','No of␣
  ↪Flights']])
```

```
#output[(output.carrier_name.str.contains('United Air Lines Inc.') & output.
↪origin_city_name.str.contains('Newark, NJ'))]['No of Flights'].iloc[0]
```

[36]:
```
def Carrier_Delay_Freq_Cal(df_carrier,df_dep_delay_carrier):
    output=pd.
↪DataFrame(columns=['carrier_name','origin_city_name','dest_city_name','flight_delay'])
    output_delay=pd.
↪DataFrame(columns=['carrier_name','origin_city_name','dest_city_name'])
    carriers=pd.unique(df_carrier['carrier_name'])
    carriers_delay=pd.unique(df_dep_delay_carrier['carrier_name'])

    for carrier in carriers:
        output=output.append(df_carrier[df_carrier.carrier_name==carrier].
↪sort_values(by='No of Flights',ascending=False).
↪head(5)[['carrier_name','origin_city_name','dest_city_name','No of␣
↪Flights']])
        #df_carrier[df_carrier.carrier_name==carrier].sort_values(by='No of␣
↪Flights',ascending=False).head(5))
        output_delay=output_delay.
↪append(df_dep_delay_carrier[df_dep_delay_carrier.carrier_name==carrier].
↪sort_values(by='No of Flights',ascending=False).head(5))

    for index in range(len(output)):
        output.flight_delay.iloc[index]=output_delay[(output_delay.carrier_name.
↪str.contains(output.carrier_name.iloc[index]) & output_delay.
↪origin_city_name.str.contains(output.origin_city_name.iloc[index]) &␣
↪output_delay.dest_city_name.str.contains(output.dest_city_name.
↪iloc[index]))]['No of Flights']


    return output
save=Carrier_Delay_Freq_Cal(df_carrier,df_dep_delay_carrier)
```

```
/opt/conda/miniconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1637:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

[60]:
```
save.flight_delay
```

[60]:
```
268      266    1078.0
Name: No of Flights, dtype: float64
20         Series([], Name: No of Flights, dtype: float64)
128       126    740.0
```

```
Name: No of Flights, dtype: float64
293       291    920.0
Name: No of Flights, dtype: float64
250       248    588.0
Name: No of Flights, dtype: float64
                                  …
11549    11092    825.0
Name: No of Flights, dtype: flo…
11689    11222    789.0
Name: No of Flights, dtype: flo…
11767      Series([], Name: No of Flights, dtype: float64)
11448      Series([], Name: No of Flights, dtype: float64)
11683      Series([], Name: No of Flights, dtype: float64)
Name: flight_delay, Length: 85, dtype: object
```

[51]:
```
for index in range(len(save)):
    save.flight_delay.iloc[index]
```

[ ]:
```

```