Milestone 2 - Cleaning/Formatting Flat File Source

Flat File: Excel files from BTS. The Excel data has airline performance factors such as cancelled, diverted, delayed and on-time data. The downloaded raw data has up to 34 columns. https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E (Download Raw Data link for data).

The Flat file is the main data source with scheduled flight information.

In [1]:

```
# Import necessary libraries

import pandas as pd
from datetime import datetime
import numpy as np
```

In [2]:

```
#Read flight data from "https://www.transtats.bts.gov/OT_Delay/OT_DelayCause1.asp?20=E"

flight_data_df = pd.read_csv('T_ONTIME_MARKETING_May.csv')
flight_data_df.head(5)
```

Out[2]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	MKT_UNIQUE_CARRIER	OP_UNIQUE_CAI
0	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
1	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
2	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
3	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
4	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	

5 rows × 39 columns

Data Transformation

i. Drop Columns

Drop unwanted columns to reduce the data size and improve data readability. Columns that I will not be using for this project are as follows:

```
ORIGIN_AIRPORT_ID
ACTUAL_ELAPSED_TIME
AIR_TIME
FLIGHTS
ORIGIN_WAC
```

```
    DEST_AIRPORT_ID
    DEST_WAC
    AIR_TIME
```

In [3]:

```
flight_data_df = flight_data_df.drop(columns=['ORIGIN_AIRPORT_ID','ACTUAL_ELAPSED_TIME',
                          'ORIGIN_WAC','DEST_AIRPORT_ID','DEST_WAC','AIR_TIME'])
flight_data_df.head(5)
```

Out[3]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	FL_DATE	MKT_UNIQUE_CARRIER	OP_UNIQUE_CAI
0	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
1	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
2	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
3	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	
4	2022	2	5	1	7	5/1/2022 12:00:00 AM	AA	

5 rows × 32 columns

ii. Look for Duplicates

Duplicates cause inconsistent results when dealing with statistics. Hence dropping duplicate rows.

In [4]:

```
print('Dataframe before dropping duplicates :', flight_data_df.shape)
flight_data_df = flight_data_df.drop_duplicates() # 1,389 rows dropped
print('Dataframe after dropping duplicates :',flight_data_df.shape)
```

```
Dataframe before dropping duplicates : (602950, 32)
Dataframe after dropping duplicates : (601561, 32)
```

iii. Replace values in a column

Cancellation code is represented as A, B, C and D, which is not very informative. The BTS website provided details on this code as follows:

A Carrier

B Weather

C National Air System

D Security

In [5]:

```
flight_data_df.CANCELLATION_CODE = np.where(flight_data_df.CANCELLATION_CODE=='A', 'Carr
                                   np.where(flight_data_df.CANCELLATION_CODE=='B', 'Weathe
                                            np.where(flight_data_df.CANCELLATION_CODE=='C'
                                                     np.where(flight_data_df.CANCELLATION_
flight_data_df.groupby(['CANCELLATION_CODE'])['CANCELLATION_CODE'].count().sort_index()
```

Out[5]:
```
CANCELLATION_CODE
                          590957
Carrier                     4902
National Air System         1394
Security                       1
Weather                     4307
Name: CANCELLATION_CODE, dtype: int64
```

iv. Rename Column

To make more sense of the information in cancellation_code, replacing the column to cancellation reason.

In [6]:
```
flight_data_df = flight_data_df.rename(columns={"CANCELLATION_CODE": "CANCELLATION_REASO
flight_data_df.columns
```

Out[6]:
```
Index(['YEAR', 'QUARTER', 'MONTH', 'DAY_OF_MONTH', 'DAY_OF_WEEK', 'FL_DATE',
       'MKT_UNIQUE_CARRIER', 'OP_UNIQUE_CARRIER', 'ORIGIN', 'ORIGIN_CITY_NAME',
       'ORIGIN_STATE_ABR', 'ORIGIN_STATE_NM', 'DEST', 'DEST_CITY_NAME',
       'DEST_STATE_ABR', 'DEST_STATE_NM', 'DEP_DELAY', 'DEP_DELAY_NEW',
       'TAXI_OUT', 'TAXI_IN', 'ARR_TIME', 'ARR_DELAY', 'ARR_DELAY_NEW',
       'CANCELLED', 'CANCELLATION_REASON', 'DIVERTED', 'DISTANCE',
       'CARRIER_DELAY', 'WEATHER_DELAY', 'NAS_DELAY', 'SECURITY_DELAY',
       'LATE_AIRCRAFT_DELAY'],
      dtype='object')
```

v. Add new columns

STATUS

In [7]:
```
#Adding a new column 'STATUS' that tells the status of a flight
flight_data_df['STATUS'] = ''

flight_data_df.STATUS = np.where(flight_data_df.CANCELLED==1, 'Cancelled',
                                 np.where(flight_data_df.DIVERTED==1, 'Diverted',
                                          np.where(flight_data_df.ARR_DELAY<=15, 'On-Tim
                                                   np.where(flight_data_df.ARR_DELAY>15,
flight_data_df.groupby(['STATUS'])['STATUS'].count().sort_index()
```

Out[7]:
```
STATUS
Cancelled     10604
Delayed      119624
Diverted       1581
On-Time      469752
Name: STATUS, dtype: int64
```

DELAYED

As a step to data reduction, I will be considering flights arriving 15 minutes or later as delayed

In [8]:
```
#Creating a new column 'DELAYED'. A flag that represents if a flight was delayed. Simila

flight_data_df.loc[(flight_data_df['ARR_DELAY']>15), 'DELAYED'] = True
flight_data_df.loc[(flight_data_df['ARR_DELAY']<=15), 'DELAYED'] = False

flight_data_df.groupby(['DELAYED'])['DELAYED'].count().sort_index()
```

Out[8]:
```
DELAYED
False    469752
True     119624
Name: DELAYED, dtype: int64
```

DELAY REASON

In [9]:

```
#Adding a new column 'DELAY_REASON' that tells the reason for a flight getting delayed
#Using the newly created DELAYED flag and the available columns for each type of delay t

flight_data_df['DELAY_REASON'] = np.where(((flight_data_df.DELAYED==True) & (flight_data
                                           np.where(((flight_data_df.DELAYED==True) & (fl
                                                    np.where(((flight_data_df.DELAYED==Tr
                                                             np.where(((flight_data_df.DE
                                                                      np.where(((flight_d

flight_data_df.groupby(['DELAY_REASON'])['DELAY_REASON'].count().sort_index()
```

Out[9]:

```
DELAY_REASON
                481937
Carrier          72453
LateAircraft     25504
NAS              17384
Security           131
Weather           4152
Name: DELAY_REASON, dtype: int64
```

vi. Implementing arithmetic functions for statistical analysis

In [10]:

```
# Create a new dataframe with total number of flights per operating carrier to calculate

flight_totals = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER']).reset_index()
flight_totals_df = pd.DataFrame(flight_totals) # Convert to dataframe
flight_totals_df.columns = ['OP_UNIQUE_CARRIER','TOTAL'] # Assign Column names
flight_totals_df['PERCENTAGE'] = round(flight_totals_df.TOTAL/flight_totals_df.TOTAL.sum

flight_totals_df = flight_totals_df.sort_values('PERCENTAGE',ascending=False) #Sort by p
flight_totals_df.head(5)
```

Out[10]:

	OP_UNIQUE_CARRIER	TOTAL	PERCENTAGE
0	WN	107950	17.94
1	DL	76021	12.64
2	AA	71471	11.88
3	OO	66615	11.07
4	UA	53535	8.90

In [11]:

```
# Calculate percentage by carrier and flight status
flight_status = flight_data_df.value_counts(subset=['OP_UNIQUE_CARRIER','STATUS']).reset
flight_status_df = pd.DataFrame(flight_status) #create a dataframe
flight_status_df.columns = ['OP_UNIQUE_CARRIER','STATUS', 'COUNT'] #Add column names
flight_status_df = flight_status_df.sort_values('OP_UNIQUE_CARRIER') #Sort by operating

flight_status_df['PERCENTAGE'] = ''

for index, row in flight_status_df.iterrows():
    tot = flight_totals.loc[flight_totals.OP_UNIQUE_CARRIER==row.OP_UNIQUE_CARRIER].TOTA
    val = (row.COUNT/tot * 100)
    flight_status_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2) #Calculate t

flight_status_df.head(10)
```

Out[11]:

	OP_UNIQUE_CARRIER	STATUS	COUNT	PERCENTAGE
33	9E	Delayed	3113	15.33

48	9E	Cancelled	542	2.67
74	9E	Diverted	35	0.17
8	9E	On-Time	16613	81.83
41	AA	Cancelled	973	1.36
56	AA	Diverted	215	0.3
3	AA	On-Time	55403	77.52
11	AA	Delayed	14880	20.82
47	AS	Cancelled	608	3.12
10	AS	On-Time	15502	79.49

In [12]:

```
#Create a new dataframe with the percentage by origin airport and status
flight_origin_totals = flight_data_df.value_counts(subset=['ORIGIN']).reset_index() #get
flight_origin_totals_df = pd.DataFrame(flight_origin_totals) #create a dataframe
flight_origin_totals_df.columns = ['ORIGIN','TOTAL'] #Add column names
flight_origin_totals_df['PERCENTAGE'] = round(flight_origin_totals_df.TOTAL/flight_origi


origin_airport_delays = flight_data_df.value_counts(subset=['ORIGIN','STATUS']).reset_in
origin_airport_df = pd.DataFrame(origin_airport_delays) #create a dataframe
origin_airport_df.columns = ['ORIGIN','STATUS', 'COUNT'] #add column names
origin_airport_df = origin_airport_df.sort_values('ORIGIN') #sort by origin
origin_airport_df['PERCENTAGE'] = ''

for index, row in origin_airport_df.iterrows():
    tot = flight_origin_totals.loc[flight_origin_totals.ORIGIN==row.ORIGIN].TOTAL.values
    val = (row.COUNT/tot * 100)
    origin_airport_df.at[index,'PERCENTAGE'] = round(val[0].astype(float),2) #calulate t

origin_airport_df = origin_airport_df.sort_values('PERCENTAGE',ascending=False) #sort by

origin_airport_df.head(10)
```

Out[12]:

	ORIGIN	STATUS	COUNT	PERCENTAGE
770	GST	On-Time	12	100.0
1208	STC	On-Time	1	100.0
385	LWS	On-Time	95	96.94
623	BGM	On-Time	30	96.77
470	DRT	On-Time	60	96.77
517	PLN	On-Time	51	96.23
488	MCW	On-Time	55	94.83
490	FOD	On-Time	55	94.83
515	TBN	On-Time	51	94.44
529	LAR	On-Time	50	94.34

vii. NULL check

In [13]:

```
#Looking for null values to further reduce the data size.
flight_data_df.isnull().sum()
```

```
Out[13]: YEAR                        0
         QUARTER                     0
         MONTH                       0
         DAY_OF_MONTH                0
         DAY_OF_WEEK                 0
         FL_DATE                     0
         MKT_UNIQUE_CARRIER          0
         OP_UNIQUE_CARRIER           0
         ORIGIN                      0
         ORIGIN_CITY_NAME            0
         ORIGIN_STATE_ABR            0
         ORIGIN_STATE_NM             0
         DEST                        0
         DEST_CITY_NAME              0
         DEST_STATE_ABR              0
         DEST_STATE_NM               0
         DEP_DELAY               10201
         DEP_DELAY_NEW           10201
         TAXI_OUT                10558
         TAXI_IN                 10769
         ARR_TIME                10769
         ARR_DELAY               12185
         ARR_DELAY_NEW           12185
         CANCELLED                   0
         CANCELLATION_REASON         0
         DIVERTED                    0
         DISTANCE                    0
         CARRIER_DELAY          477611
         WEATHER_DELAY          477611
         NAS_DELAY              477611
         SECURITY_DELAY         477611
         LATE_AIRCRAFT_DELAY    477611
         STATUS                      0
         DELAYED                 12185
         DELAY_REASON                0
         dtype: int64
```

Based on the above, it doesn't appear there are any null rows that are irrelevant. Status is a significant column that tells if there are any flights with no relevant status. All flights are now categorized under On-Time, Delayed, Cancelled or Diverted.

Conclusion:

As a part of this milestone, the following Data Transformation steps have been performed.

1. Dropped columns
2. Dropped duplicate rows
3. Replaced values in a dataframe column
4. Renamed a column
5. Added new columns to the dataframe
6. Implemented arithmetic functions for statistical analysis
7. Performed null check to drop rows with null values.

The following cells can be ignored. I will be continuing to work on this file for the upcoming milestones.

#Read Diverted data from "https://www.diverted.eu/" into a dataframe url = 'https://www.diverted.eu/' df = pd.read_html(url) data = df[0] #Format Flight date from string to Date data.Date = pd.to_datetime(data["Date"], format='%d.%m.%Y') data.head(5) #Only select data for May'22 diverted_df = data[(data.Date >= '2022-05-01') & (data.Date < '2022-06-01')] diverted_df.head(5)

#Read Weather Data from API https://visual-crossing-weather.p.rapidapi.com/history?startDateTime={}&aggregateHours=24&location={}&endDateTime={}&unitGroup=us