MSPA PREDICT 420

Graded Exercise 5: Still Flying But How Many

Introduction

This document presents the results of the forth graded exercise for the Masters of Science in Predictive Analytics course: PREDICT 420.

Assessment

1. Loading the Data

Load datasets into pandas dataframes.

In [1]:
```
import pandas as pd

df_passenger = pd.read_csv("data/2014+CY-YTD+Passenger+Raw+Data_2-1.csv", skiprows = [0], thousa
nds = ",")
df_a2010 = pd.read_table("data/A2010_14.txt", encoding = "latin-1")
df_causefactors = pd.read_table("data/causefactors.txt", header = None)
```

2. Pre-process the Data

Set field names for 'passenger' dataframe.

In [2]:
```
df_passenger.columns = ["month",
                        "orgApt",
                        "destApt",
                        "orgWAC",
                        "destWAC",
                        "carrier",
                        "group",
                        "type",
                        "total",
                        "schedule",
                        "charter"]
```

Confirm dtypes for 'passenger' dataframe fields.

In [3]:
```
print(df_passenger.dtypes)
```

```
month        int64
orgApt      object
destApt     object
orgWAC       int64
destWAC      int64
carrier     object
group        int64
type        object
total        int64
schedule     int64
charter      int64
dtype: object
```

Print first five records for dataframes.

In [4]:
```
df_passenger.head(5)
```

Out[4]:

	month	orgApt	destApt	orgWAC	destWAC	carrier	group	type	total	schedule	charter
0	201401	AEX	GUA	72	127	FCQ	1	Passengers	398	0	398
1	201401	AEX	GUA	72	127	XP	1	Passengers	68	0	68
2	201401	AEX	GYE	72	337	FCQ	1	Passengers	202	0	202
3	201401	AEX	MGA	72	153	FCQ	1	Passengers	17	0	17
4	201401	AEX	PAP	72	238	K8	1	Passengers	73	0	73

In [6]:
```
df_a2010.head(5)
```

Out[6]:

	c5	c1	c2	c3	c4	c6	c7	c8	c9	c10	c75	c132	c134	c136	c138	c139	c140	c141	c144	c145	c147
0	20100101025609A	A	091			2010	01	01	20100101	1940	9										
1	20100101025799A	A	137			2010	01	01	20100101	1142	9										
2	20100101025829I	I	091			2010	01	01	20100101	910	9								1H72	1	H
3	20100101030559I	I	091			2010	01	01	20100101	900	9								1L72	1	L
4	20100102026249I	I	091			2010	01	02	20100102	1258	9										

In [8]:
```
df_causefactors.head(5)
```

Out[8]:

	0	1	2	3
0	0	NaN	UNKNOWN	NaN
1	1	AA	FAIL ADVISE UNSAFE APT COND	APT/COND
2	2	AF	IMPROPER MAINTENANCE APT FAC	APT/FAC
3	3	AI	INADEQUATELY MAINTAIN AWY FAC	AWY/FAC
4	4	AL	DIDN'T FLY ASG ALT IFR CLRNS	ASG/ALT

3. Departure and Arrival Statistics

For airports LAX, SFO, ATL, MIA, and JFK, determine how many passenger departures and arrivals there were during 2014.

In [9]:
```
apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Departures",
              "Arrivals"]

results_df = pd.DataFrame([])

for a in apt:
    departures = df_passenger.loc[df_passenger["orgApt"] == a, "total"].sum()
    arrivals = df_passenger.loc[df_passenger["destApt"] == a, "total"].sum()
    temp_df = pd.DataFrame([[departures, arrivals]], index = [a], columns = resultcols)
    results_df = results_df.append(temp_df)

results_df
```

Out[9]:

	Departures	Arrivals
LAX	18681107	0
SFO	10066556	0
ATL	10583444	0
MIA	20020381	0
JFK	27515961	0

For airports LAX, SFO, ATL, MIA, and JFK, determine which airline was the largest departure carrier.

In [10]:
```
apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Carrier",
              "Departures"]

results_df = pd.DataFrame([])

for a in apt:
    df_temp = df_passenger
    df_temp = df_temp[df_temp["orgApt"] == a] # Return matches for desired airport.
    df_temp = df_temp.groupby(["carrier"])["total"].max() # Group based on max occurrences.
    df_temp = df_temp.sort_values(ascending = False) # Sort by descending occurrences.
    temp_df = pd.DataFrame([[df_temp.index[0], df_temp[0]]], index = [a], columns = resultcols)
    results_df = results_df.append(temp_df)

results_df = results_df.sort_values(by = "Departures", ascending = False)
results_df
```

Out[10]:

	Carrier	Departures
JFK	BA	134263
LAX	AC	62504
ATL	DL	55480
MIA	AA	51349
SFO	AC	48917

For airports LAX, SFO, ATL, MIA, and JFK, determine which airline was the the largest arrival carrier.

In [11]:
```
#Note: No arrivals for each
apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Carrier",
              "Arrivals"]

results_df = pd.DataFrame([])

for a in apt:
    df_temp = df_passenger
    df_temp = df_temp[df_temp["destApt"] == a] # Return matches for desired airport.
    df_temp = df_temp.groupby(["carrier"])["total"].max() # Group based on max occurrences.
    df_temp = df_temp.sort_values(ascending = False) # Sort by descending occurrences.
    #temp_df = pd.DataFrame([[df_temp.index[0], df_temp[0]]], index = [a], columns = resultcols
)
    #results_df = results_df.append(temp_df)

#results_df = results_df.sort_values(by = "Arrivals", ascending = False)
results_df
```

Out[11]: □

For airports LAX, SFO, ATL, MIA, and JFK, determine w hat airports the largest number of departures w ent to.

In [12]:
```
apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Airport",
              "Departures"]

results_df = pd.DataFrame([])

for a in apt:
    df_temp = df_passenger
    df_temp = df_temp[df_temp["orgApt"] == a] # Return matches for desired airport.
    df_temp = df_temp.groupby(["destApt"])["total"].sum() # Group based on number of occurrence
s.
    df_temp = df_temp.sort_values(ascending = False) # Sort by descending count of occurrences.
    temp_df = pd.DataFrame([[df_temp.index[0], df_temp[0]]], index = [a], columns = resultcols)
    results_df = results_df.append(temp_df)

results_df = results_df.sort_values(by = "Departures", ascending = False)
results_df
```

Out[12]:

	Airport	Departures
JFK	LHR	2892396
LAX	LHR	1428718
MIA	GRU	1039120
SFO	LHR	911760
ATL	CUN	704666

For airports LAX, SFO, ATL, MIA, and JFK, determine w hat airports the largest number of arrivals w ere from.

In [13]:
```
#Note: No arrivals for each
apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Airport",
              "Arrivals"]

results_df = pd.DataFrame([])

for a in apt:
    df_temp = df_passenger
    df_temp = df_temp[df_temp["destApt"] == a] # Return matches for desired airport.
    df_temp = df_temp.groupby(["orgApt"])["total"].sum() # Group based on number of occurrences
.
    df_temp = df_temp.sort_values(ascending = False) # Sort by descending count of occurrences.
    #temp_df = pd.DataFrame([[df_temp.index[0], df_temp[0]]], index = [a], columns = resultcols
)
    #results_df = results_df.append(temp_df)

#results_df = results_df.sort_values(by = "Arrivals", ascending = False)
results_df
```

Out[13]: □

4. Accident and Fatality Statistics

For airports LAX, SFO, ATL, MIA, and JFK, determine the number of accidents or incidents that occurred at them betw een 2010 and 2014 inclusive, according to the FAA.

In [14]:
```
# c143                              Char                4 Airport identification code of the accide
nt/incident location, if on airport.

apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

resultcols = ["Incidents"]

results_df = pd.DataFrame([])

for a in apt:
    incidents = len(df_a2010[df_a2010["c143"].str.contains(a, na = False)]) # Return count of m
atches for desired airport.
    temp_df = pd.DataFrame([[incidents]], index = [a], columns = resultcols) # Create dataframe
 of match count.
    results_df = results_df.append(temp_df) # Append match count to summary dataframe.

results_df = results_df.sort_values(by = "Incidents", ascending = False)
results_df
```

Out[14]:

	Incidents
ATL	28
LAX	16
JFK	13
MIA	10
SFO	7

For airports LAX, SFO, ATL, MIA, and JFK, determine the number of deaths that occurred in each event.

In [15]:
```
# c76                              VarChar            3 Total Fatalities
# c78                              Char               2 Primary cause factor code
# c94                              Char               2 Type of accident code
# c143                             Char               4 Airport identification code of the accide
nt/incident location, if on airport.

import numpy as np

apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

for a in apt:
    df_temp = df_a2010
    df_temp = df_temp[df_temp["c143"].str.contains(a, na = False)] # Return matches for desired
 airport.
    df_temp = df_temp[["c78", "c94", "c76"]] # Return relevant dataframe columns (see above).
    df_temp = df_temp.replace({"c78" : {np.NaN : "Unknown", "  " : "Unknown"}}) # Replace NaN a
nd blank values with "Unknown".
    df_temp = df_temp.replace({"c94" : {np.NaN : "Unknown", "  " : "Unknown"}})
    df_temp.index.name = None
    df_temp.columns = ["causefactorCode", "accidentCode", "fatalities"]
    print(a, df_temp)
    print("")
```

```
LAX       causefactorCode accidentCode  fatalities
312           Unknown      Unknown           0
706           Unknown      Unknown           0
1475          Unknown      Unknown           0
1630               55      Unknown           0
2151          Unknown      Unknown           0
2498          Unknown      Unknown           0
2857          Unknown      Unknown           0
3847          Unknown      Unknown           0
5184          Unknown      Unknown           0
5264               32      Unknown           0
5615          Unknown      Unknown           0
5968               32      Unknown           0
6450          Unknown      Unknown           0
7764          Unknown      Unknown           0
9358          Unknown      Unknown           0
10495         Unknown      Unknown           0

SFO       causefactorCode accidentCode  fatalities
1721          Unknown      Unknown           0
1892          Unknown      Unknown           0
1954               71      Unknown           0
3150               57      Unknown           0
4208          Unknown      Unknown           0
5324               32      Unknown           0
10842         Unknown      Unknown           3

ATL       causefactorCode accidentCode  fatalities
7             Unknown      Unknown           0
25            Unknown      Unknown           0
42            Unknown      Unknown           0
95            Unknown      Unknown           0
152           Unknown      Unknown           0
159           Unknown      Unknown           0
173           Unknown      Unknown           0
210           Unknown      Unknown           0
220           Unknown      Unknown           0
223                78      Unknown           0
242           Unknown      Unknown           0
245           Unknown      Unknown           0
283           Unknown      Unknown           0
458           Unknown      Unknown           0
501           Unknown      Unknown           0
847                32      Unknown           0
848           Unknown      Unknown           0
2280          Unknown      Unknown           0
2631          Unknown      Unknown           0
3294          Unknown      Unknown           0
3948          Unknown      Unknown           0
4073          Unknown      Unknown           0
4257          Unknown      Unknown           0
5401               75      Unknown           0
6177          Unknown      Unknown           0
9593               14      Unknown           0
9636          Unknown      Unknown           0
10459         Unknown      Unknown           0

MIA       causefactorCode accidentCode  fatalities
43            Unknown      Unknown           0
875           Unknown      Unknown           0
2367          Unknown      Unknown           0
2466          Unknown      Unknown           0
```

For airports LAX, SFO, ATL, MIA, and JFK, determine w hat the top ten (primary) causes of 2010-2014 incidents and accidents are for all events resulting in deaths regardless of w here they occurred. Provide descriptions (not codes) for the causes.

In [16]:
```
import numpy as np

df_a2010 = df_a2010.replace({"c78" : {np.NaN : 0,
                                      "  " : 0}}) # Replace NaN's and blanks within cause factor code column with zero digit.
df_a2010["c78"] = df_a2010["c78"].astype(int) # Convert cause factor code column to integer type.
df_causefactors = df_causefactors[[0, 2]] # Define relevant columns of cause factor code description dataframe.
causedict = df_causefactors.set_index(0).to_dict() # Convert cause factor code description data frame to dictonary.
```

In [17]:
```
# c76                           VarChar            3 Total Fatalities
# c78                           Char               2 Primary cause factor code
# c143                          Char               4 Airport identification code of the accident/incident location, if on airport.

import numpy as np

apt = ["LAX", "SFO", "ATL", "MIA", "JFK"]

for a in apt:
    df_temp = df_a2010
    df_temp = df_temp[df_temp["c143"].str.contains(a, na = False)] # Return matches for desired airport.
    df_temp = df_temp[["c78", "c76"]] # Return relevant dataframe columns (see above).
    df_temp = df_temp.replace({"c78" : causedict[2]}) # Replace cause factor code values based on dictonary of descriptions.
    df_temp = df_temp.groupby(["c78"]).sum() # Group based on number of occurrences.
    df_temp = df_temp.sort_values(by = "c76", ascending = False) # Sort by descending count of occurrences.
    df_temp.columns = ["fatalities"]
    df_temp.index.name = None
    print(a, df_temp[0:10])
    print("")
```

```
LAX                                fatalities
IMPROPER MGT/FUEL TANK SELECTO              0
INADEQ SPACE AC/WKE TURBULENCE              0
UNKNOWN                                     0

SFO                                fatalities
UNKNOWN                                     3
AEROBATICS BELOW SAFE ALTITUDE              0
IMPROPER MGT/FUEL TANK SELECTO              0
IMPROPER OPERATION OF FAC                   0

ATL                                fatalities
FAIL TO ATTAIN PROPER OP TEMP               0
IMPROPER MGT/FUEL TANK SELECTO              0
ISSUED IMPR CONFLICTING INSTS               0
PILOT INCAP EXCLUDES ALCOHOL                0
UNKNOWN                                     0

MIA                                fatalities
IMPROPER INST PROC T/O LDG                  0
STARTED ENG W/OUT ASSIST/EQUIP              0
UNKNOWN                                     0

JFK          fatalities
UNKNOWN               0
```