Introduction

In this assignment we will import data from the San Francisco airport and create a dataset that allows us to perform some summary statistics on the customers' experience with the airport and generate a list of customers that we will target for a 2017 follow up focus group.

The code below assumes that you are running this Notebook from the folder you unzipped the .zip package containing the datafiles and other assets as needed to run thes scripts.

All code, sourcefiles, outputs and pickles are available on Github:
https://github.com/jarrardenator/jarrard-predictiveanalytics/tree/master/P420
(https://github.com/jarrardenator/jarrard-predictiveanalytics/tree/master/P420)

Part 1

Part 1: Import the data and create a single data frame containing key fields on survey responses for further analysis. (a) List the the variables from each year that you used to create this data set using the original names they had in the data set they appeared in.

(b) Document any variable name changes so that it's clear what the original variables are whose names you changed. (Otherwise, a user of your data wouldn't be able to know what these variables are, or how to use them.)

(c) Describe your DataFrame in terms of its size, the variables in it, and how the data types of the variables. How many missing values do you have on the ratings variables?

(d) Write your new DataFrame to a csv file with an initial header record that includes the variable names. Verify that you wrote this file without errors.

In [2]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas_profiling
import pickle

sfo_2016 = pd.read_csv("SFO_2016.csv", parse_dates=True)
sfo_2015 = pd.read_csv("SFO_2015.csv", parse_dates=True)
sfo_2014 = pd.read_csv("SFO_2014.csv", parse_dates=True)
```

```
C:\Users\Jeff\Anaconda3\lib\site-packages\matplotlib\__init__.py:1405: UserWarn
ing:
This call to matplotlib.use() has no effect because the backend has already
been chosen; matplotlib.use() must be called *before* pylab, matplotlib.pyplot,
or matplotlib.backends is imported for the first time.

  warnings.warn(_use_error_msg)
```

We'll create some profiling reports using the pandas_profiling just to get a sense for what's in each of these data sets. Some manual analysis and comparison between them is required, then we will select the columns that we need for all the parts of the assignment below and combine them all into a single dataset. Some columns may need to be renamed, some values may need to be rekeyed for year to year consistency, and we'll need to handle missing or otherwise incorrect values.

In [6]:
```
pandas_profiling.ProfileReport(df=sfo_2014)
```

Out[6]:

Overview

Dataset info

Number of variables	95
Number of observations	2820
Total Missing (%)	24.1%
Total size in memory	2.0 MiB
Average record size in memory	760.0 B

Variables types

Numeric	72
Categorical	11
Date	0
Text (Unique)	0
Rejected	12

In [8]:
```
pandas_profiling.ProfileReport(df=sfo_2015)
```

Out[8]:

Overview

Dataset info

Number of variables	90
Number of observations	2958
Total Missing (%)	17.9%
Total size in memory	2.0 MiB
Average record size in memory	720.0 B

Variables types

Numeric	75
Categorical	7
Date	0
Text (Unique)	0
Rejected	8

In [9]:
```
pandas_profiling.ProfileReport(df=sfo_2016)
```

Out[9]:

Overview

Dataset info

Number of variables	97
Number of observations	3087
Total Missing (%)	12.8%
Total size in memory	2.3 MiB
Average record size in memory	776.0 B

Variables types

Numeric	66
Categorical	14
Date	0
Text (Unique)	0
Rejected	17

This image shows the columns that we have selected that should satisfy the requirements for Part 1 and the subesequent parts below. We have a single dataframe that we can then manipulate to create subsets that will be needed for analyzing comments, demographic data, and creating a targeting list for 2017 follow up surveys.

2014	2015	2016	Legend
RESPNUM	RESPNUM	*RESPNUM	Part 1
CCGID	CCGID	CCGID	Part 2
RUN	RUNID	RUNID	Part 3
INTDATE	INTDATE	INTDATE	Part4
	DESTGEO	DESTGEO	
	DESTMARK	DESTMARK	
	Q2PURP1	Q2PURP1	
	Q3PARK	Q3PARK	
	Q4BAGS	Q4BAGS	
	Q4STORE	Q4STORE	
	Q4FOOD	Q4FOOD	
	Q4WIFI	Q4WIFI	
	Q5TIMESFLOWN	Q5TIMESFLOWN	
	Q5FIRSTTIME	Q5FIRSTTIME	
	Q6LONGUSE	Q6LONGUSE	
Q7ART	Q7ART	Q7ART	
Q7FOOD	Q7FOOD	Q7FOOD	
Q7STORE	Q7STORE	Q7STORE	
Q7SIGN	Q7SIGN	Q7SIGN	
Q7WALKWAY	Q7WALKWAYS	Q7WALKWAYS	
Q7SCREENS	Q7SCREENS	Q7SCREENS	
Q7INFODOW	Q7INFODOWN	Q7INFODOWN	
Q7INFOUP	Q7INFOUP	Q7INFOUP	
Q7WIFI	Q7WIFI	Q7WIFI	
Q7ROADS	Q7ROADS	Q7ROADS	
Q7PARK	Q7PARK	Q7PARK	
Q7AIRTRAIN	Q7AIRTRAIN	Q7AIRTRAIN	
Q7LTPARKING	Q7LTPARKING	Q7LTPARKING	
Q7RENTAL	Q7RENTAL	Q7RENTAL	
Q7ALL	Q7ALL	Q7ALL	
	Q8COM1	Q8COM	
	Q8COM2	Q8COM2	
	Q8COM3	Q8COM3	
		Q8COM4	
		Q8COM5	
Q9BOARDING	Q9BOARDING	Q9BOARDING	
Q9AIRTRAIN	Q9AIRTRAIN	Q9AIRTRAIN	
Q9RENTAL	Q9RENTAL	Q9RENTAL	
Q9FOOD	Q9FOOD	Q9FOOD	
Q9RESTROOM	Q9RESTROOM	Q9RESTROOM	
Q9ALL	Q9ALL	Q9ALL	
Q10SAFE	Q10SAFE	Q10SAFE	
Q12PRECHEK	Q12PRECHEKCRAT	Q12PRECHEKCRATE	
Q13GETRATE	Q13GETRATE	Q13GETRATE	
Q14PASSTHR	Q14PASSTHRU	Q14PASSTHRU	
Q16LIVE	Q16LIVE	Q16LIVE	
	Q18AGE	Q18AGE	
	Q19GENDER	Q19GENDER	
	Q20INCOME	Q20INCOME	
	Q21FLY	Q21FLY	
	Q22SJC	Q22SJC	
	Q22OAK	Q22OAK	
	LANG	LANG	

Now, we will do some cleanup that will be needed to create a union of all three years' with the columns we need for ech part of the assignment. We'll make it easy on ourselves now by doing the minimal amount of cleanup needed. As we go, we may find things we need to clean up, as noted int he pandas_profiling reports. Several fields have missing values, and there are a number of key columns that reference data supplied by the data dictionary. Ideally, we'd have tables that supplied the key-value pairs for the coded columns. We will add those as we go if needed.

Once this initial cleansing step is complete, the columns needed for Part 1 are extracted and a new dataframe is created containing the ratings data for key airport amenities and the airport overall. Finally, the new dataframe is output to a .csv and a pickle file for reuse later.

In [6]:
```
#Fix column names that need adjusting
sfo_2016 = sfo_2016.rename(index=str,columns={"*RESPNUM": "RESPNUM"})

#Add a Year column to each
sfo_2016['Year'] = 2016
sfo_2015['Year'] = 2015
sfo_2014['Year'] = 2014


#Append the files together with the common fields needed for Parts 2-4 into a Pan
sfo_all = pd.concat([sfo_2014,sfo_2015,sfo_2016],ignore_index=True)

cols_part1 = ('RESPNUM','Year','CCGID','RUNID','INTDATE',
              'Q7ART','Q7FOOD','Q7STORE','Q7SIGN',
              'Q7WALKWAYS','Q7SCREENS','Q7INFODOWN',
              'Q7INFOUP','Q7WIFI','Q7ROADS','Q7PARK',
              'Q7AIRTRAIN','Q7LTPARKING','Q7RENTAL','Q7ALL',
              'Q9BOARDING','Q9AIRTRAIN','Q9RENTAL','Q9FOOD',
              'Q9RESTROOM','Q9ALL','Q10SAFE','Q12PRECHEKCRATE',
              'Q13GETRATE','Q14PASSTHRU','Q16LIVE')

sfo_part1 = pd.DataFrame(data=sfo_all,columns=cols_part1)

#c. profile new dataframe
#run the profiling and discuss

#d write to a new .csv file with initial header; reimport and check values
sfo_part1.to_csv("SFO_PART1.csv",header=True,index=True)

#test
sfo_part1_test = pd.read_csv("SFO_PART1.csv",parse_dates=True)

#pickleize
sfo_part1.to_pickle('sfo_part1_pickle')
```

Part 2: In this section, we analyze the comments data to determine the top 3 comments for 2015 and 2016. For each year, there are multiple columns for comments, presumably so the customer could add more than one piece of feedback. We will 'union' all of the columns for each year, then summarize and count the number of instances for each comment.

The comments in these fields are actually IDs, that map to text in a large dictionary in the provided documentation. Once we have aggregated the total instances of each comment ID, we will look up the values and provide a lookup table to provide the text of the comments ranked by the percent of total comments for each year.

In [7]:
```
# Part 2: Identify the top three comments made in 2015 and 2016
# Create a Comments dataset based on the information from the Data Dictionary for
#
sfo_part2 = sfo_all['Year'] >= 2015
sfo_part2 = sfo_all[sfo_part2]
cols_part2 = ('Q8COM1','Q8COM2','Q8COM3','Q8COM4','Q8COM5')
sfo_part2 = pd.DataFrame(data=sfo_part2,columns=cols_part2)

comments = []
for col in cols_part2:
    s = pd.Series(sfo_part2[col])
    s = s.dropna()
    comments.append(s)

comments = pd.concat(comments)
comments = pd.DataFrame(comments)
comments = comments.rename(columns={0:'CommentCode'})
comments['Count'] = 1
comments['CommentCode'] = comments.loc[:,('CommentCode')].astype('category')
comments = comments.pivot_table(values='Count',index='CommentCode',aggfunc=sum)
comments['TotalComments']=comments.Count.sum()
comments['Prop']=comments.Count/comments.TotalComments

top5comments = comments.sort_values(by='Count', ascending=False).head(5)
sfo_part2.to_pickle('sfo_part2_pickle')
print(top5comments)
```

```
             Count  TotalComments      Prop
CommentCode
999.0          192           2389  0.080368
202.0          186           2389  0.077857
505.0          102           2389  0.042696
203.0           72           2389  0.030138
501.0           70           2389  0.029301
```

Part 3: Summarize the Q7ALL data by Home residence location. We will create a "lookup table" to provide a meaningful desription for the location codes in the HOME field. Looking at the data dictionary for home residence location (field named "HOME"), the codes are all the same, so we can analyze the Q7ALL field by location.

We can take the frequencies of each response to the question in Q7ALL, "Rating SFO Airport as a whole", analyze the mean and variance for those providing a response (we will ignore responses other than a 1-5 on this question). Fortunatley, only 3.7% of respondents didn't answer this question in 2015, and in 2016 the percent blank was only2.4%.

In [8]:
```
#Part 3:  Analyze the Q7ALL "SFO Rating as a whole" question by residence Home lo

#set the columns
cols_part3 = ('RESPNUM','Year','RUNID','HOME','Q7ALL')

sfo_part3 = pd.DataFrame(data=sfo_all,columns=cols_part3)
sfo_part3 = sfo_part3.dropna()

#for some reason, probably the javascript in the browser, need to convert Q7ALL t
sfo_part3['Q7ALL'] = sfo_part3['Q7ALL'].astype(float)
#only use the ratings between 1 and 5; 0 = no answer, 6= no experience, we only w

sfo_part3 = sfo_part3[(sfo_part3.Q7ALL > int(0)) & (sfo_part3.Q7ALL < int(6))]
print(sfo_part3.dtypes)

sfo_part3_summary = sfo_part3.groupby(['Year','HOME'])

part3_stats = sfo_part3_summary['Q7ALL'].describe()
sfo_part3.to_pickle('sfo_part3_pickle')
print(part3_stats)
```

```
RESPNUM    float64
Year         int64
RUNID      float64
HOME       float64
Q7ALL      float64
dtype: object
           count      mean       std  min   25%  50%   75%  max
Year HOME
2015 1.0   264.0  4.041667  0.677003  2.0  4.00  4.0  4.00  5.0
     2.0   159.0  4.113208  0.746139  2.0  4.00  4.0  5.00  5.0
     3.0   147.0  4.129252  0.733467  1.0  4.00  4.0  5.00  5.0
     4.0   134.0  4.126866  0.630332  2.0  4.00  4.0  5.00  5.0
     5.0   114.0  4.087719  0.759150  2.0  4.00  4.0  5.00  5.0
     6.0    55.0  4.236364  0.637229  3.0  4.00  4.0  5.00  5.0
     7.0    39.0  4.025641  0.668351  2.0  4.00  4.0  4.00  5.0
     8.0    12.0  4.333333  0.492366  4.0  4.00  4.0  5.00  5.0
     9.0    13.0  4.153846  0.554700  3.0  4.00  4.0  4.00  5.0
     10.0  658.0  4.051672  0.744583  1.0  4.00  4.0  5.00  5.0
     11.0  299.0  4.066890  0.701538  2.0  4.00  4.0  5.00  5.0
     12.0  367.0  4.019074  0.722145  1.0  4.00  4.0  5.00  5.0
     13.0   79.0  3.924051  0.729808  1.0  4.00  4.0  4.00  5.0
     14.0   16.0  4.250000  0.577350  3.0  4.00  4.0  5.00  5.0
     15.0  107.0  3.953271  0.678277  2.0  4.00  4.0  4.00  5.0
     16.0  132.0  3.909091  0.692981  2.0  3.00  4.0  4.00  5.0
     17.0    5.0  4.600000  0.547723  4.0  4.00  5.0  5.00  5.0
     18.0    2.0  4.500000  0.707107  4.0  4.25  4.5  4.75  5.0
     19.0   36.0  4.027778  0.696362  2.0  4.00  4.0  4.00  5.0
     90.0   41.0  4.073171  0.519146  3.0  4.00  4.0  4.00  5.0
     91.0   11.0  4.272727  0.646670  3.0  4.00  4.0  5.00  5.0
     99.0  120.0  3.891667  0.807484  1.0  3.00  4.0  4.00  5.0
2016 1.0   277.0  4.068592  0.701181  2.0  4.00  4.0  5.00  5.0
     2.0   136.0  4.117647  0.760690  1.0  4.00  4.0  5.00  5.0
     3.0   168.0  4.035714  0.699811  1.0  4.00  4.0  4.00  5.0
     4.0   145.0  3.931034  0.751435  2.0  4.00  4.0  4.00  5.0
```

```
5.0    84.0  4.214286  0.602633  3.0  4.00  4.0  5.00  5.0
6.0    41.0  3.975610  0.790184  2.0  3.00  4.0  5.00  5.0
7.0    35.0  4.000000  0.685994  3.0  4.00  4.0  4.00  5.0
8.0    27.0  4.111111  0.697982  3.0  4.00  4.0  5.00  5.0
9.0    10.0  4.100000  0.567646  3.0  4.00  4.0  4.00  5.0
10.0  727.0  4.038514  0.743124  1.0  4.00  4.0  5.00  5.0
11.0  224.0  4.013393  0.748571  2.0  4.00  4.0  5.00  5.0
12.0  334.0  4.146707  0.657165  2.0  4.00  4.0  5.00  5.0
13.0  213.0  3.962441  0.628338  2.0  4.00  4.0  4.00  5.0
14.0   22.0  4.318182  0.476731  4.0  4.00  4.0  5.00  5.0
15.0   96.0  3.958333  0.631067  2.0  4.00  4.0  4.00  5.0
16.0  119.0  3.932773  0.620703  3.0  4.00  4.0  4.00  5.0
17.0    7.0  3.857143  0.690066  3.0  3.50  4.0  4.00  5.0
18.0    2.0  4.000000  0.000000  4.0  4.00  4.0  4.00  4.0
19.0   45.0  3.977778  0.722649  2.0  4.00  4.0  4.00  5.0
90.0   57.0  3.894737  0.771923  1.0  3.00  4.0  4.00  5.0
91.0    9.0  4.000000  0.707107  3.0  4.00  4.0  4.00  5.0
99.0  141.0  4.000000  0.870140  1.0  4.00  4.0  5.00  5.0
```

Part 4: a) We will import the targeting dataset and join with a copy of our original dataset with just the years 2015 and 2016 to create a dataframe that consists of the following: •Respondent ID •Year surveyed •Destination geographic area •Size of destination market •Purpose(s) of travel (Make sure that the meanings of the codes are consistent, e.g. that a code of "5" means the same thing year to year.) •Used parking? •Checked baggage? •Purchased from a store? •Purchased in a restaurant? •Used free WiFi? •Times Flown in last 12 mo. •First time flying out of SFO? •How Long Using SFO? •Residence Location? Bay Area, or ...? (Q16LIVE) •Age •Gender •Income •Fly 100K miles or more per year? •Language version of questionnaire •Have used the San Jose airport •Have used the Oakland airport b) SAve the file as a headered .csv file and profile the results ensuring good data quality (c) Tabulate the frequencies of the codes for Parking, Times Flown, Gender, How Long Using SFO. Account for missing values.

Part 5: Pickleize all the data frames for use in a later session.

In [10]:
```
#Part 4:  Create a targeting dataset
#Import the targeting data


select_resps_2017 = pd.read_csv("select_resps_2017.csv",parse_dates=True)
cols_part4_resps = ('RESPNUM','year')
select_resps_2017 = pd.DataFrame(data=select_resps_2017,columns=cols_part4_resps)
select_resps_2017 = select_resps_2017.rename(columns={"year" : "Year"})

cols_part4_demos = ('RESPNUM','Year','DESTGEO','DESTMARK','Q3PARK','Q4BAGS',
                    'Q4STORE','Q4FOOD','Q4WIFI','Q5TIMESFLOWN','Q5FIRSTTIME',
                    'Q6LONGUSE','Q18AGE','Q19GENDER','Q20INCOME','Q21FLY',
                    'Q22SJC','Q22OAK','LANG')
sfo_all['Year'] = sfo_all['Year'].astype(int)
part4_demos = sfo_all[(sfo_all.Year >=2015)]
part4_demos=pd.DataFrame(data=part4_demos,columns=cols_part4_demos)

#we'll fill the na values with -1 so our targeting system can account for missing
part4_demos = part4_demos.fillna(-1)


part4_targeting = pd.merge(select_resps_2017, part4_demos, how='inner', on=['RESP
print(part4_targeting.head(5))

part4_targeting.to_pickle('part4_targeting_pickle')
```

```
   RESPNUM  Year  DESTGEO  DESTMARK Q3PARK  Q4BAGS  Q4STORE  Q4FOOD  Q4WIFI  \
0     1054  2016      2.0       3.0      4     1.0      2.0     1.0     1.0
1     1088  2016      2.0       2.0     -1     2.0      2.0     2.0     1.0
2     1952  2016      8.0       4.0     -1     1.0      1.0     2.0     2.0
3      794  2016      4.0       1.0     -1     1.0      2.0     2.0     2.0
4      406  2016      2.0       4.0     -1     1.0      0.0     1.0     1.0

   Q5TIMESFLOWN  Q5FIRSTTIME  Q6LONGUSE  Q18AGE  Q19GENDER  Q20INCOME  Q21FLY
 \
0           2.0          1.0        4.0    -1.0       -1.0       -1.0    -1.0

1           5.0          1.0        4.0    -1.0       -1.0       -1.0    -1.0

2           2.0          1.0        4.0    -1.0       -1.0       -1.0    -1.0

3           2.0          1.0        2.0    -1.0       -1.0       -1.0    -1.0

4           2.0          1.0        4.0    -1.0       -1.0       -1.0    -1.0


   Q22SJC  Q22OAK  LANG
0    -1.0    -1.0   1.0
1    -1.0    -1.0   1.0
2    -1.0    -1.0   1.0
3    -1.0    -1.0   1.0
4    -1.0    -1.0   1.0
```

In [ ]: