Winter 2017 PRED 420 Exercise 1 Student: Sunny Wu

1.Read each data file into a Pandas DataFrame. Add meaningful names (i.e., names that would make sense to other people, given the data) to the columns of each DataFrame.

Provide your syntactically correct, commented code. Print the first three rows of each DataFrame. Provide your code, and the results it produced.

In [1]:
```
#The first row is added as the name of each column, following the naming co
import pandas as pd

airports = pd.read_csv('/Users/sunnywu/Desktop/Python_files/Exercise1/airpo
                  names = ['AirportID','Name','City','Country','IATA/FAA','
                           'Longitude','Altitude','Timezone','DST','Tz'])
airports.head(3)
```

Out[1]:

	AirportID	Name	City	Country	IATA/FAA	ICAO	Latitude	Longitude	Altitude	T
0	1	Goroka	Goroka	Papua New Guinea	GKA	AYGA	-6.081689	145.391881	5282	1
1	2	Madang	Madang	Papua New Guinea	MAG	AYMD	-5.207083	145.788700	20	1
2	3	Mount Hagen	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826789	144.295861	5388	1

In [2]:
```
airlines = pd.read_csv('/Users/sunnywu/Desktop/Python_files/Exercise1/airli
                  names = ['AirlineID','Name','Alias','IATA','ICAO','Callsi
airlines.head(3)
```

Out[2]:

	AirlineID	Name	Alias	IATA	ICAO	Callsign	Country	Active
0	1	Private flight	\N	-	NaN	NaN	NaN	Y
1	2	135 Airways	\N	NaN	GNL	GENERAL	United States	N
2	3	1Time Airline	\N	1T	RNX	NEXTIME	South Africa	Y

In [3]:
```
routes = pd.read_csv('/Users/sunnywu/Desktop/Python_files/Exercise1/routes.c
                  names = ['Airline','AirlineID','Source airport','Source ai
                           'Destination airport ID', 'Codeshare','Stops','Eq
routes.head(3)
```

Out[3]:

	Airline	AirlineID	Source airport	Source airport ID	Destination airport	Destination airport ID	Codeshare	Stops	Equipmen
0	2B	410	AER	2965	KZN	2990	NaN	0	CR2
1	2B	410	ASF	2966	KZN	2990	NaN	0	CR2
2	2B	410	ASF	2966	MRV	2962	NaN	0	CR2

2.Check each DataFrame for duplicate records. For each, report the number of duplicates you found.

Provide your commented, syntactically correct code and the results it produced.

In [4]:
```
#Find duplicated airport record.
#Step 1: find out the total number of rows of the dataframe "airports"
airport_total_number = len(airports)
#Step 2: find out the row number with unique names, column name is "Name"
airport_number_unique_names = len(airports.Name.unique())
# Step 3: calculate the difference
airport_duplicates= airport_total_number - airport_number_unique_names
print ("duplicated airport record is:", airport_duplicates)
```

```
duplicated airport record is: 199
```

In [5]:
```
#Find duplicated airline record.
#Step 1: find out the total number of rows of the dataframe "airlines"
airline_total_number = len(airlines)
#Step 2: find out the row number with unique names in the column named as "1
airline_number_unique_names = len(airlines.Name.unique())
# Step 3: calculate the difference
airline_duplicates= airline_total_number - airline_number_unique_names
print ("duplicated airline record is:", airline_duplicates)
```

```
duplicated airline record is: 89
```

In [6]:
```
#Find duplicated route record.
#Step 1: Find total number of route records from dataframe "routes".
total_routes=len(routes)
#Step2: Find out unique routes, the routes are considered unique when two ro
unique_routes = len(routes.drop_duplicates())
#Step 3: calculate the difference to find duplicated records.
route_duplicates= total_routes - unique_routes
print ("duplicated route record is: ", route_duplicates)
```

```
duplicated route record is:  0
```

3.Describe the data types of the columns in each of the DataFrames.¶

In [7]:
```
#Data types of each column in the DataFrame "airports" is listed below:
airports.dtypes
```

Out[7]:
```
AirportID      int64
Name          object
City          object
Country       object
IATA/FAA      object
ICAO          object
Latitude     float64
Longitude    float64
Altitude       int64
Timezone     float64
DST           object
Tz            object
dtype: object
```

In [8]:
```
#Data types of each column in the DataFrame "airlines" is listed below:
airlines.dtypes
```

Out[8]:
```
AirlineID     int64
Name         object
Alias        object
IATA         object
ICAO         object
Callsign     object
Country      object
Active       object
dtype: object
```

In [9]:
```
#Data types of each column in the DataFrame "routes" is listed below:
routes.dtypes
```

Out[9]:
```
Airline                  object
AirlineID                object
Source airport           object
Source airport ID        object
Destination airport      object
Destination airport ID   object
Codeshare                object
Stops                     int64
Equipment                object
dtype: object
```

4.Determine how many of the airlines are "defunct."

Provide your definition of what a defunct airline is. Provide your commented, syntactically correct code and the results it produced.

In [10]:
```
#Definition of "defunct airline" is airlines with active status of  "N".
#The active status of an airline is displayed in the last column of the data
Filter = airlines['Active']=='N'

print ("The number of defunct airlines are: ",
       len(airlines[Filter]))
```

```
The number of defunct airlines are:  4886
```

5.Determine how many "routes from nowhere" there are in the data. These are flights that don't originate from an airport.

Provide your commented, syntactically correct code and the results it produced.

In [11]:
```
#Use DataFrame "routes" as input, and the column 'Source airport' as searchi
#When column "Source airport" is empty string in a row, the route is conside
print ("routes from nowhere: ",
       len (routes [routes['Source airport']==''']))
```

```
routes from nowhere:  0
```

6.Save your DataFrames for future use. You may pickle them, put them in a shelve db, on in a the tables of a SQL db. Check to make sure that they are saved correctly.

Provide your commented, syntactically correct code and the results it produced.

In [12]:
```
#Step 1: pickle DataFrame airports, airlines, routes.
# create airports, airline, and routes files.
import pickle
airports.to_pickle('airports')
airlines.to_pickle('airlines')
routes.to_pickle('routes')
```

In [13]:
```
#Step 2: shelve DataFrame airports, airlines, routes.
import shelve

#create myairports_db, myairlines_db, and myroutes_db files.
myairports_db=shelve.open('myairports')
myairports_db['airports']=airports
myairports_db.close()
```

In [14]:
```
myairlines_db=shelve.open('myairlines')
myairlines_db['airlines']=airlines
myairlines_db.close()
```

In [15]:
```
myroutes_db=shelve.open('myroutes')
myroutes_db['routes']=routes
myroutes_db.close()
```

End of Exercise 1