Module 4: Cleaning data with pandas

Weekly Assignment 4A

Brandon Owens, Loan Pham

Q.1 Read in the dataset 'Banking_Marketing.csv'

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

In [2]:
```
df = pd.read_csv("../dataFiles/Banking_Marketing.csv")
df
```

Out[2]:

	age	job	marital	education	default	housing	loan	contact	month	day
0	44.0	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	
1	53.0	technician	married	unknown	no	no	no	cellular	nov	
2	28.0	management	single	university.degree	no	yes	no	cellular	jun	
3	39.0	services	married	high.school	no	no	no	cellular	apr	
4	55.0	retired	married	basic.4y	no	yes	no	cellular	aug	
...	...	...	...	...	...	...	...	...	...	
41194	104.0	retired	married	high.school	unknown	no	yes	telephone	jun	
41195	2.0	housemaid	married	Basic	unknown	no	n	NaN	may	
41196	3.0	admin.	single	university.degree	unknown	yes	y	NaN	may	
41197	NaN	technician	married	professional.course	no	no	n	telephone	oct	
41198	NaN	student	single	high.school	no	no	n	telephone	may	

41199 rows × 21 columns

In [3]:
```
# (a) show the names of the columns of the dataset

for col in df.columns:
    print(col)
```

```
age
job
marital
education
default
housing
loan
contact
month
```

```
day_of_week
duration
campaign
pdays
previous
poutcome
emp_var_rate
cons_price_idx
cons_conf_idx
euribor3m
nr_employed
y
```

In [4]:
```
# (b) print the basic info of each column using .info()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41199 entries, 0 to 41198
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41197 non-null  float64
 1   job             41199 non-null  object 
 2   marital         41199 non-null  object 
 3   education       41199 non-null  object 
 4   default         41199 non-null  object 
 5   housing         41199 non-null  object 
 6   loan            41199 non-null  object 
 7   contact         41193 non-null  object 
 8   month           41199 non-null  object 
 9   day_of_week     41199 non-null  object 
 10  duration        41192 non-null  float64
 11  campaign        41199 non-null  int64  
 12  pdays           41199 non-null  int64  
 13  previous        41199 non-null  int64  
 14  poutcome        41199 non-null  object 
 15  emp_var_rate    41199 non-null  float64
 16  cons_price_idx  41199 non-null  float64
 17  cons_conf_idx   41199 non-null  float64
 18  euribor3m       41199 non-null  float64
 19  nr_employed     41199 non-null  float64
 20  y               41199 non-null  int64  
dtypes: float64(7), int64(4), object(10)
memory usage: 6.6+ MB
```

In [5]:
```
# (c) check any missing values for each variable (column)
df.isnull().any()
```

Out[5]:
```
age              True
job             False
marital         False
education       False
default         False
housing         False
loan            False
contact          True
month           False
day_of_week     False
duration         True
campaign        False
pdays           False
previous        False
poutcome        False
```

```
emp_var_rate      False
cons_price_idx    False
cons_conf_idx     False
euribor3m         False
nr_employed       False
y                 False
dtype: bool
```

In [6]:
```
# (d) drop all the rows with any missing values
df.dropna()
```

Out[6]:

	age	job	marital	education	default	housing	loan	contact	month	day_
0	44.0	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	
1	53.0	technician	married	unknown	no	no	no	cellular	nov	
2	28.0	management	single	university.degree	no	yes	no	cellular	jun	
3	39.0	services	married	high.school	no	no	no	cellular	apr	
4	55.0	retired	married	basic.4y	no	yes	no	cellular	aug	
...	...	...	...	...	...	...	...	...	...	
41182	24.0	admin.	married	high.school	no	yes	no	cellular	may	
41183	59.0	retired	married	high.school	unknown	no	yes	telephone	jun	
41184	31.0	housemaid	married	basic.4y	unknown	no	no	telephone	may	
41187	25.0	student	single	high.school	no	no	no	telephone	may	
41194	104.0	retired	married	high.school	unknown	no	yes	telephone	jun	

41187 rows × 21 columns

In [7]:
```
# (e) do a value count for the variable 'education'
df["education"].value_counts()
```

Out[7]:
```
university.degree      12170
high.school             9521
basic.9y                6045
professional.course     5244
basic.4y                4176
basic.6y                2292
unknown                 1731
illiterate                18
Basic                      2
Name: education, dtype: int64
```

In [8]:
```
# (f)  group the "basic.4y", "basic.9y", and "basic.6y" categories together and call th
df["education"].replace({"basic.9y":"Basic","basic.6y":"Basic","basic.4y":"Basic"},inpl
df["education"].value_counts()
```

Out[8]:
```
Basic                 12515
university.degree     12170
high.school            9521
professional.course    5244
```

```
unknown                 1731
illiterate                18
Name: education, dtype: int64
```

In [9]:

```
# (g) get all the columns with non-numeric data. Then create dummy variables for those
cols = df.select_dtypes([object]).columns
print(cols)
newDF = pd.get_dummies(df.select_dtypes([object]).columns,dtype=str)
newDF.info()
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   contact      10 non-null     object
 1   day_of_week  10 non-null     object
 2   default      10 non-null     object
 3   education    10 non-null     object
 4   housing      10 non-null     object
 5   job          10 non-null     object
 6   loan         10 non-null     object
 7   marital      10 non-null     object
 8   month        10 non-null     object
 9   poutcome     10 non-null     object
dtypes: object(10)
memory usage: 928.0+ bytes
```

Q.2 Read in “flights.csv”

In [10]:

```
fl = pd.read_csv("../dataFiles/flights.csv")
fl
```

Out[10]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
0	2013	1	1	517.0	515	2.0	830.0	819	11.0
1	2013	1	1	533.0	529	4.0	850.0	830	20.0
2	2013	1	1	542.0	540	2.0	923.0	850	33.0
3	2013	1	1	544.0	545	-1.0	1004.0	1022	-18.0
4	2013	1	1	554.0	600	-6.0	812.0	837	-25.0
...	...	...	...	...	...	...	...	...	...
336771	2013	9	30	NaN	1455	NaN	NaN	1634	NaN
336772	2013	9	30	NaN	2200	NaN	NaN	2312	NaN

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
336773	2013	9	30	NaN	1210	NaN	NaN	1330	NaN
336774	2013	9	30	NaN	1159	NaN	NaN	1344	NaN
336775	2013	9	30	NaN	840	NaN	NaN	1020	NaN

336776 rows × 19 columns

In [11]:
```
# (a) Find all flights that
# (i)   Had an arrival delay of two or more hours(“arr_delay” variable is measured in m

fl[fl["arr_delay"]>=120]
```

Out[11]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
119	2013	1	1	811.0	630	101.0	1047.0	830	137.0
151	2013	1	1	848.0	1835	853.0	1001.0	1950	851.0
218	2013	1	1	957.0	733	144.0	1056.0	853	123.0
268	2013	1	1	1114.0	900	134.0	1447.0	1222	145.0
447	2013	1	1	1505.0	1310	115.0	1638.0	1431	127.0
...	...	...	...	...	...	...	...	...	...
336579	2013	9	30	1823.0	1545	158.0	1934.0	1733	121.0
336668	2013	9	30	1951.0	1649	182.0	2157.0	1903	174.0
336724	2013	9	30	2053.0	1815	158.0	2310.0	2054	136.0
336757	2013	9	30	2159.0	1845	194.0	2344.0	2030	194.0
336763	2013	9	30	2235.0	2001	154.0	59.0	2249	130.0

10200 rows × 19 columns

In [12]:
```
# (ii)  Flew to Houston (IAH or HOU)(“dest" is either “IAH” or “HOU”.)

fl[fl["dest"].str.contains("IAH","HOU")|fl["dest"].str.contains("HOU")]
```

Out[12]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
0	2013	1	1	517.0	515	2.0	830.0	819	11.0
1	2013	1	1	533.0	529	4.0	850.0	830	20.0
32	2013	1	1	623.0	627	-4.0	933.0	932	1.0
81	2013	1	1	728.0	732	-4.0	1041.0	1038	3.0
89	2013	1	1	739.0	739	0.0	1104.0	1038	26.0
...	...	...	...	...	...	...	...	...	...
336524	2013	9	30	1729.0	1720	9.0	2001.0	2010	-9.0
336527	2013	9	30	1735.0	1715	20.0	2010.0	2005	5.0
336618	2013	9	30	1859.0	1859	0.0	2134.0	2159	-25.0
336694	2013	9	30	2015.0	2015	0.0	2244.0	2307	-23.0
336737	2013	9	30	2105.0	2106	-1.0	2329.0	2354	-25.0

9313 rows × 19 columns

In [13]:

```
# (iii) Were operated by United or Delta (You also need the dataset “airlines.csv”)
# read file
al = pd.read_csv('../dataFiles/airlines.csv')
al
```

Out[13]:

	carrier	name
0	9E	Endeavor Air Inc.
1	AA	American Airlines Inc.
2	AS	Alaska Airlines Inc.
3	B6	JetBlue Airways
4	DL	Delta Air Lines Inc.
5	EV	ExpressJet Airlines Inc.
6	F9	Frontier Airlines Inc.
7	FL	AirTran Airways Corporation
8	HA	Hawaiian Airlines Inc.

	carrier	name
9	MQ	Envoy Air
10	OO	SkyWest Airlines Inc.
11	UA	United Air Lines Inc.
12	US	US Airways Inc.
13	VX	Virgin America
14	WN	Southwest Airlines Co.
15	YV	Mesa Airlines Inc.

In [14]:
```
# operated by United or Delta
air =fl.merge(al, on="carrier")
air[air["carrier"].str.contains("UA")|air["carrier"].str.contains("DL")]
```

Out[14]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
0	2013	1	1	517.0	515	2.0	830.0	819	11.0
1	2013	1	1	533.0	529	4.0	850.0	830	20.0
2	2013	1	1	554.0	558	-4.0	740.0	728	12.0
3	2013	1	1	558.0	600	-2.0	924.0	917	7.0
4	2013	1	1	558.0	600	-2.0	923.0	937	-14.0
...	...	...	...	...	...	...	...	...	...
194134	2013	9	30	1955.0	2000	-5.0	2219.0	2230	-11.0
194135	2013	9	30	1956.0	1825	91.0	2208.0	2121	47.0

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
194136	2013	9	30	2041.0	2045	-4.0	2147.0	2208	-21.0
194137	2013	9	30	2050.0	2045	5.0	20.0	53	-33.0
194138	2013	9	30	2121.0	2100	21.0	2349.0	14	-25.0

106775 rows × 20 columns

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
age               2
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           6
month             0
day_of_week       0
duration          7
campaign          0
pdays             0
previous          0
poutcome          0
emp_var_rate      0
cons_price_idx    0
cons_conf_idx     0
euribor3m         0
nr_employed       0
y                 0
dtype: int64
```

In [16]:
```
# (c) Sort flights to find
# (i)   the most delayed flights.  (use “dep_delay”)

fl.sort_values(by= 'dep_delay', ascending = False)
fl[["flight", "dep_delay"]]
```

Out[16]:

	flight	dep_delay
0	1545	2.0
1	1714	4.0
2	1141	2.0
3	725	-1.0

	flight	dep_delay
4	461	-6.0
...	...	...
336771	3393	NaN
336772	3525	NaN
336773	3461	NaN
336774	3572	NaN
336775	3531	NaN

336776 rows × 2 columns

In [17]:
```
# (ii)  the fastest flights. (you need to calculate “speed” = “distance”/ “air_time” )

fl["speed"] = fl["distance"]/fl["air_time"]
fl.sort_values(by= 'speed', ascending = False)
fl.head(1)
```

Out[17]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
0	2013	1	1	517.0	515	2.0	830.0	819	11.0	UA

In [18]:
```
# operated by United or Delta
air =fl.merge(al, on="carrier")
air[air["carrier"].str.contains("UA")|air["carrier"].str.contains("DL")]
```

Out[18]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
0	2013	1	1	517.0	515	2.0	830.0	819	11.0
1	2013	1	1	533.0	529	4.0	850.0	830	20.0
2	2013	1	1	554.0	558	-4.0	740.0	728	12.0
3	2013	1	1	558.0	600	-2.0	924.0	917	7.0

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay
4	2013	1	1	558.0	600	-2.0	923.0	937	-14.0
...	...	...	...	...	...	...	...	...	...
194134	2013	9	30	1955.0	2000	-5.0	2219.0	2230	-11.0
194135	2013	9	30	1956.0	1825	91.0	2208.0	2121	47.0
194136	2013	9	30	2041.0	2045	-4.0	2147.0	2208	-21.0
194137	2013	9	30	2050.0	2045	5.0	20.0	53	-33.0
194138	2013	9	30	2121.0	2100	21.0	2349.0	14	-25.0

106775 rows × 21 columns

In [19]:
```
fl.isnull().any()
```

Out[19]:
```
year              False
month             False
day               False
dep_time           True
sched_dep_time    False
dep_delay          True
arr_time           True
sched_arr_time    False
arr_delay          True
carrier           False
flight            False
tailnum            True
origin            False
dest              False
air_time           True
distance          False
hour              False
minute            False
time_hour         False
speed              True
dtype: bool
```

In [20]:
```
# (c) Sort flights to find
# (i)   the most delayed flights.  (use “dep_delay”)

fl.sort_values(by= 'dep_delay', ascending = False)
fl.head(1)
```

Out[20]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
0	2013	1	1	517.0	515	2.0	830.0	819	11.0	UA

In [21]:
```
# (ii)  the fastest flights. (you need to calculate “speed” = “distance”/ “air_time” )

fl["speed"] = fl["distance"]/fl["air_time"]
fl.sort_values(by= 'speed', ascending = False)
fl.head(1)
```

Out[21]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
0	2013	1	1	517.0	515	2.0	830.0	819	11.0	UA

In [22]:
```
#(d) Currently dep_time is convenient to look at, but hard to compute with because they
# Convert them to a more convenient representation of number of minutes since midnight.
fl['midn_time'] = (fl['dep_time'] / 100 * 60 + fl['dep_time'] % 100) % 1440
fl.head()
```

Out[22]:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
0	2013	1	1	517.0	515	2.0	830.0	819	11.0	UA
1	2013	1	1	533.0	529	4.0	850.0	830	20.0	UA
2	2013	1	1	542.0	540	2.0	923.0	850	33.0	AA
3	2013	1	1	544.0	545	-1.0	1004.0	1022	-18.0	B6
4	2013	1	1	554.0	600	-6.0	812.0	837	-25.0	DL

5 rows × 21 columns

In [23]:
```
# (e) Which carrier has the worst delays? (find the average “arr_delay” for each carrie

arr_delay = fl.groupby('carrier', as_index = False)['arr_delay'].mean()
sort_fl = arr_delay.sort_values(by = 'arr_delay', ascending=False)
sort_fl.head(1)
```

Out[23]:

	carrier	arr_delay
6	F9	21.920705

In [24]:

```
# (f)  Find all destinations that are flown by at least two carriers.
df_carrier = fl.groupby("dest")["carrier"].nunique().reset_index(name="carrier_count")
df_carrier[df_carrier["carrier_count"] > 1]
```

Out[24]:

	dest	carrier_count
4	ATL	7
5	AUS	6
6	AVL	2
7	BDL	2
8	BGR	2
...	...	...
99	SYR	3
100	TPA	7
102	TVC	2
103	TYS	2
104	XNA	2

76 rows × 2 columns