```
# This is formatted as code
```

DHV LAB Sheet 7

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Enable inline plotting
%matplotlib inline
```

Pandas is a python package that deals mostly with :

- **Series** (1d homogeneous array)
- **DataFrame** (2d labeled heterogeneous array)
- **Panel** (general 3d array)

Pandas Series

Pandas *Series* is one-dimentional labeled array containing data of the same type (integers, strings, floating point numbers, Python objects, etc.). The axis labels are often referred to as *index*.

```
Start coding or generate with AI.
```

```
# Read a dataset with missing values
flights = pd.read_csv("flights.csv")
flights.head()
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT
0	2015	1	1	4	AS	98	N407AS	ANC
1	2015	1	1	4	AA	2336	N3KUAA	LAX
2	2015	1	1	4	US	840	N171US	SFO
3	2015	1	1	4	AA	258	N3HYAA	LAX
4	2015	1	1	4	AS	135	N527AS	SEA

5 rows × 31 columns

```
flights.info()
```

Missing Values

```
# Select the rows that have at least one missing value
flights[flights.isnull().any(axis=1)].head()
```

LINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPAR
AS	98	N407AS	ANC	SEA	
AA	2336	N3KUAA	LAX	PBI	
US	840	N171US	SFO	CLT	
AA	258	N3HYAA	LAX	MIA	
AS	135	N527AS	SEA	ANC	

```
# Filter all the rows where arr_delay value is missing:
flights1 = flights[ flights['ARRIVAL_DELAY'].notnull( )]
flights1.head()
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT
0	2015	1	1	4	AS	98	N407AS	ANC
1	2015	1	1	4	AA	2336	N3KUAA	LAX
2	2015	1	1	4	US	840	N171US	SFC
3	2015	1	1	4	AA	258	N3HYAA	LAX
4	2015	1	1	4	AS	135	N527AS	SEA

5 rows × 31 columns

```
# Remove all the observations with missing values
flights2 = flights.dropna()
```

```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135433 entries, 0 to 135432
Data columns (total 31 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   YEAR                 135433 non-null  int64
 1   MONTH                135433 non-null  int64
 2   DAY                  135433 non-null  int64
 3   DAY_OF_WEEK          135433 non-null  int64
 4   AIRLINE              135433 non-null  object
 5   FLIGHT_NUMBER        135433 non-null  int64
 6   TAIL_NUMBER          135144 non-null  object
 7   ORIGIN_AIRPORT       135433 non-null  object
 8   DESTINATION_AIRPORT  135433 non-null  object
 9   SCHEDULED_DEPARTURE  135433 non-null  int64
 10  DEPARTURE_TIME       131836 non-null  float64
 11  DEPARTURE_DELAY      131836 non-null  float64
 12  TAXI_OUT             131719 non-null  float64
 13  WHEELS_OFF           131718 non-null  float64
 14  SCHEDULED_TIME       135432 non-null  float64
 15  ELAPSED_TIME         131328 non-null  float64
 16  AIR_TIME             131328 non-null  float64
 17  DISTANCE             135432 non-null  float64
 18  WHEELS_ON            131573 non-null  float64
 19  TAXI_IN              131573 non-null  float64
 20  SCHEDULED_ARRIVAL    135432 non-null  float64
 21  ARRIVAL_TIME         131573 non-null  float64
 22  ARRIVAL_DELAY        131328 non-null  float64
 23  DIVERTED             135432 non-null  float64
 24  CANCELLED            135432 non-null  float64
 25  CANCELLATION_REASON  3748 non-null    object
 26  AIR_SYSTEM_DELAY     43797 non-null   float64
 27  SECURITY_DELAY       43797 non-null   float64
 28  AIRLINE_DELAY        43797 non-null   float64
 29  LATE_AIRCRAFT_DELAY  43797 non-null   float64
 30  WEATHER_DELAY        43797 non-null   float64
dtypes: float64(20), int64(6), object(5)
memory usage: 32.0+ MB
```

```
# Fill missing values with zeros
nomiss =flights['DEPARTURE_DELAY'].fillna(0)
nomiss.isnull().any()
```

```
False
```

Exercise

```
# Count how many missing data are in dep_delay and arr_delay columns
```

Common Aggregation Functions:

Function	Description
min	minimum
max	maximum
count	number of non-null observations

Function	Description
sum	sum of values
mean	arithmetic mean of values
median	median
mad	mean absolute deviation
mode	mode
prod	product of values
std	standard deviation
var	unbiased variance

```
# Find the number of non-missing values in each column
flights.describe()
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULED_DEPA
count	135433.0	135433.0	135433.000000	135433.000000	135433.000000	135433.0
mean	2015.0	1.0	4.847105	4.076296	2242.200416	1309.0
std	0.0	0.0	2.464646	1.843236	1817.438683	473.3
min	2015.0	1.0	1.000000	1.000000	1.000000	5.0
25%	2015.0	1.0	3.000000	3.000000	753.000000	915.0
50%	2015.0	1.0	5.000000	4.000000	1696.000000	1305.0
75%	2015.0	1.0	7.000000	5.000000	3419.000000	1710.0
max	2015.0	1.0	9.000000	7.000000	7438.000000	2359.0

8 rows × 26 columns

```
flights.info()
```

```
# Let's compute summary statistic per a group':
flights.groupby('ORIGIN_AIRPORT')['DEPARTURE_DELAY'].mean()
```

```
ORIGIN_AIRPORT
ABE    10.500000
ABI    18.200000
ABQ    19.648148
ABR     6.312500
ABY     1.761905
          ...
VPS    12.008772
WRG     6.312500
XNA    20.557214
YAK     6.928571
YUM     9.100000
Name: DEPARTURE_DELAY, Length: 312, dtype: float64
```

```
# We can use agg() methods for aggregation:
flights[['DEPARTURE_DELAY','ARRIVAL_DELAY']].agg(['min','mean','max'])
```

	DEPARTURE_DELAY	ARRIVAL_DELAY
min	-39.0000	-74.000000
mean	18.8949	16.957305
max	1380.0000	1384.000000

```
# An example of computing different statistics for different columns
flights.agg({'DEPARTURE_DELAY':['min','mean',max], 'ORIGIN_AIRPORT':['nunique']})
```

	DEPARTURE_DELAY	ORIGIN_AIRPORT
min	-39.0000	NaN
mean	18.8949	NaN
max	1380.0000	NaN
nunique	NaN	312.0

Basic descriptive statistics

Function	Description
min	minimum
max	maximum
mean	arithmetic mean of values
median	median
mad	mean absolute deviation
mode	mode
std	standard deviation
var	unbiased variance
sem	standard error of the mean
skew	sample skewness
kurt	kurtosis
quantile	value at %

```
# Convinient describe() function computes a veriety of statistics
flights.DEPARTURE_DELAY.describe()
```

```
count    131836.00000
mean         18.89490
std          45.46225
min         -39.00000
25%          -3.00000
50%           2.00000
75%          23.00000
max        1380.00000
Name: DEPARTURE_DELAY, dtype: float64
```

```
# find the index of the maximum or minimum value
# if there are multiple values matching idxmin() and idxmax() will return the first match
flights['DEPARTURE_DELAY'].idxmin()  #minimum value
```

```
20989
```

```
# Count the number of records for each different value in a vector
flights['ORIGIN_AIRPORT'].value_counts()
```

```
ORIGIN_AIRPORT
ATL    8219
ORD    6676
DFW    6641
DEN    5160
LAX    5051
       ...
ADQ       7
GFK       6
OTH       5
ADK       3
PPG       3
Name: count, Length: 312, dtype: int64
```

```
#factorplot
sns.catplot(x='ORIGIN_AIRPORT',y='DEPARTURE_DELAY', data=flights, kind='bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7e4848fcfa30>
```

Exercise

```
#Using seaborn package explore the dependency of arr_delay on dep_delay (scatterplot or regplot) using flights dataset
```

```
df=pd.read_csv('Salaries.csv')
#Use matplotlib to draw a histogram of a salary data
plt.hist(df['salary'],bins=20, density=True)
```

```
# Use regular matplotlib function to display a barplot
df.groupby(['rank'])['salary'].count().plot(kind='bar')
```

```
# Use seaborn package to display a barplot
sns.set_style("whitegrid")

ax = sns.barplot(x='rank',y ='salary', data=df, estimator=len)
```

```
# Split into 2 groups:
ax = sns.barplot(x='rank',y ='salary', hue='sex', data=df, estimator=len)
```

```
#Violinplot
sns.violinplot(x = "salary", data=df)


#Scatterplot in seaborn
sns.jointplot(x='service', y='salary', data=df)


sns.scatterplot(x='service', y='salary', data=df)


#If we are interested in linear regression plot for 2 numeric variables we can use regplot
sns.regplot(x='service', y='salary', data=df)


# box plot
sns.boxplot(x='rank',y='salary', data=df)


# side-by-side box plot
sns.boxplot(x='rank',y='salary', data=df, hue='sex')


# swarm plot
sns.swarmplot(x='rank',y='salary', data=df)


# Pairplot
sns.pairplot(df)
```

Start coding or generate with AI.