

PYTHON INTERNSHIP MANUAL 5(B)

Batch A - 2025

Week#9 (Pandas Library in Python)

Important Instructions:

- Read the manual carefully and understand every topic before starting the assignment.
- Follow the deadlines properly; marks will be deducted for late submissions.
- Avoid to use AI Tools and make your own logic.
- Completing all tasks in the assignment is compulsory to receive full marks.

• Manual Posted Date:

30th April, 2025 (Wednesday)

• Assignment-5 Deadline:

5th April, 2025 (Monday 11:59pm)

More on Pandas

Obtaining Basic Information About DataFrame

```
df.columns
```

```
Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size',  
      'price_per_person', 'Payer Name', 'CC Number', 'Payment ID'],  
      dtype='object')
```

```
df.index
```

```
RangeIndex(start=0, stop=244, step=1)
```

```
df.head(3)
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458

```
df.tail(3)
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 11 columns):
total_bill      244 non-null float64
tip             244 non-null float64
sex             244 non-null object
smoker         244 non-null object
day            244 non-null object
time           244 non-null object
size           244 non-null int64
price_per_person 244 non-null float64
Payer Name     244 non-null object
CC Number      244 non-null int64
Payment ID     244 non-null object
dtypes: float64(3), int64(2), object(6)
memory usage: 21.0+ KB
```

```
len(df)
```

244

Operations on Columns

1) Create a new column:

For example: `df['tip_percentage'] = 100* df['tip'] / df['total_bill']`

2) Adjust existing columns:

For Example: `df['price_per_person'] = np.round(df['price_per_person'],2)`

3) Set Index:

```
df.set_index('Payment ID')
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Payment ID										
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

4) Reset Index:

```
df = df.reset_index()
```

...

```
df.head()
```

	Payment ID	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
0	Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
1	Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
2	Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
3	Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994
4	Sun2251	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221

Operations on Rows

1) Grab Single Row:

```
# Name Based
```

```
df.loc['Sun2959']
```

```
total_bill    16.99
tip           1.01
sex           Female
smoker        No
day           Sun
time          Dinner
size          2
price_per_person  8.49
Payer Name    Christy Cunningham
CC Number     3560325168603410
Name: Sun2959, dtype: object
```

2) Grab Multiple Rows:

```
df.iloc[0:4]
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun4608	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230
Sun4458	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

```
df.loc[['Sun2959', 'Sun5260']]
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number
Payment ID										
Sun2959	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410
Sun5260	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994

3) Loc/iLoc Filtering:

Syntax:

`df.loc[Start_row: End_row , Start_column:End_column]`

`df.loc[Start_row: End_row , Start_column:End_column]`

```
df.iloc[0:2,0:2]
```

	total_bill	tip
0	16.99	1.01
1	10.34	1.66

Merging DataFrames

```
registrations = pd.DataFrame({'reg_id':[1,2,3,4], 'name':['Andrew', 'Bobo', 'Claire', 'David']})
logins = pd.DataFrame({'log_id':[1,2,3,4], 'name':['Xavier', 'Andrew', 'Yolanda', 'Bobo']})
```

registrations

	reg_id	name
0	1	Andrew
1	2	Bobo
2	3	Claire
3	4	David

logins

	log_id	name
0	1	Xavier
1	2	Andrew
2	3	Yolanda
3	4	Bobo

1) Inner Join

Match up where the key is present in both tables. There should be no NaN due to the join, since by definition to be part of the inner join they need information in both tables.

```
# Notice pd.merge doesn't take in a list like concat
pd.merge(registrations,logins,how='inner',on='name')
```

	reg_id	name	log_id
0	1	Andrew	2
1	2	Bobo	4

2) Group By

A groupby() operation allows us to examine data on a per category basis.

Data

```
df = pd.read_csv('mpg.csv')
```

```
df
```

```
3]:
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

398 rows × 9 columns

```
# model_year becomes the index! It is NOT a column name, it is now the name of the index
df.groupby('model_year').mean()
```

	mpg	cylinders	displacement	weight	acceleration	origin
model_year						
70	17.689655	6.758621	281.413793	3372.793103	12.948276	1.310345
71	21.250000	5.571429	209.750000	2995.428571	15.142857	1.428571
72	18.714286	5.821429	218.375000	3237.714286	15.125000	1.535714
73	17.100000	6.375000	256.875000	3419.025000	14.312500	1.375000
74	22.703704	5.259259	171.740741	2877.925926	16.203704	1.666667
75	20.266667	5.600000	205.533333	3176.800000	16.050000	1.466667
76	21.573529	5.647059	197.794118	3078.735294	15.941176	1.470588
77	23.375000	5.464286	191.392857	2997.357143	15.435714	1.571429
78	24.061111	5.361111	177.805556	2861.805556	15.805556	1.611111
79	25.093103	5.827586	206.689655	3055.344828	15.813793	1.275862
80	33.696552	4.137931	115.827586	2436.655172	16.934483	2.206897
81	30.334483	4.620690	135.310345	2522.931034	16.306897	1.965517
82	31.709677	4.193548	128.870968	2453.548387	16.638710	1.645161

3) Group By Multiple Columns

```
df.groupby(['model_year', 'cylinders']).mean()
```

```
[3]:
```

		mpg	displacement	weight	acceleration	origin
model_year	cylinders					
70	4	25.285714	107.000000	2292.571429	16.000000	2.285714
	6	20.500000	199.000000	2710.500000	15.500000	1.000000
	8	14.111111	367.555556	3940.055556	11.194444	1.000000
71	4	27.461538	101.846154	2056.384615	16.961538	1.923077
	6	18.000000	243.375000	3171.875000	14.750000	1.000000
	8	13.428571	371.714286	4537.714286	12.214286	1.000000
72	3	19.000000	70.000000	2330.000000	13.500000	3.000000
	4	23.428571	111.535714	2382.642857	17.214286	1.928571
	8	13.615385	344.846154	4228.384615	13.000000	1.000000
73	3	18.000000	70.000000	2124.000000	13.500000	3.000000
	4	22.727273	109.272727	2338.090909	17.136364	2.000000
	6	19.000000	212.250000	2917.125000	15.687500	1.250000
74	8	13.200000	365.250000	4279.050000	12.250000	1.000000
	4	27.800000	96.533333	2151.466667	16.400000	2.200000
	6	17.857143	230.428571	3320.000000	16.857143	1.000000
	8	14.200000	315.200000	4438.400000	14.700000	1.000000
	4	25.250000	114.833333	2489.250000	15.833333	2.166667

4) The .apply() Method

This allows to apply and broadcast custom functions on a DataFrame column.

```
import pandas as pd
import numpy as np
```

```
df = pd.read_csv('tips.csv')
```

```
df.head()
```

```
[3]:
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251

```
def last_four(num):
    return str(num)[-4:]
```

```
df['CC Number'][0]
```

```
3560325168603410
```

```
last_four(3560325168603410)
```

```
'3410'
```

```
df['last_four'] = df['CC Number'].apply(last_four)
```

```
df.head()
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	last_four
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	3410
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	9230
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	1322
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	5994
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carler	4832732618637221	Sun2251	7221

Activate 'Win

Using .apply() with more complex functions

```
df['total_bill'].mean()
```

```
2]: 19.78594262295082
```

```
def yelp(price):  
    if price < 10:  
        return '$'  
    elif price >= 10 and price < 30:  
        return '$$'  
    else:  
        return '$$$'
```

```
df['Expensive'] = df['total_bill'].apply(yelp)
```

```
df
```

```
5]:
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	last_four	Expensive
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	3410	\$\$
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	9230	\$\$
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458	1322	\$\$
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260	5994	\$\$
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carler	4832732618637221	Sun2251	7221	\$\$

Activate Win
Go to Settings to

5) Between

```
df['total_bill'].between(10,20,inclusive=True)
```

```
54]: 0      True  
      1      True  
      2     False  
      3     False  
      4     False  
      5     False  
      6     False  
      7     False  
      8      True
```



```
df[df['total_bill'].between(10,20,inclusive=True)]
```

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	last_four	Expensive	Tip Quality
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959	3410	\$\$	Ok
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608	9230	\$\$	Ok
8	15.04	1.96	Male	No	Sun	Dinner	2	7.52	Joseph McDonald	3522866365840377	Sun6820	0377	\$\$	Ok
9	14.78	3.23	Male	No	Sun	Dinner	2	7.39	Jerome Abbott	3532124519049786	Sun3775	9786	\$\$	Ok
10	10.27	1.71	Male	No	Sun	Dinner	2	5.14	William Riley	566287581219	Sun2546	1219	\$\$	Ok
12	15.42	1.57	Male	No	Sun	Dinner	2	7.71	Chad Harrington	577040572932	Sun1300	2932	\$\$	Ok
13	18.43	3.00	Male	No	Sun	Dinner	4	4.61	Joshua Jones	6011163105616890	Sun2971	6890	\$\$	Ok
14	14.83	3.02	Female	No	Sun	Dinner	2	7.42	Vanessa Jones	30016702287574	Sun3848	7574	\$\$	Ok

6) nlargest and nsmallest

```
df.nlargest(10,'tip')
```

1]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID	last_four	Expensive	Tip Quality
170	50.81	10.00	Male	Yes	Sat	Dinner	3	16.94	Gregory Clark	5473850968388236	Sat1954	8236	\$\$\$	Ok
212	48.33	9.00	Male	No	Sat	Dinner	4	12.08	Alex Williamson	676218815212	Sat4590	5212	\$\$\$	Ok
23	39.42	7.58	Male	No	Sat	Dinner	4	9.86	Lance Peterson	3542584061609808	Sat239	9808	\$\$\$	Ok
59	48.27	6.73	Male	No	Sat	Dinner	4	12.07	Brian Ortiz	6596453823950595	Sat8139	0595	\$\$\$	Ok
141	34.30	6.70	Male	No	Thur	Lunch	6	5.72	Steven Carlson	3526515703718508	Thur1025	8508	\$\$\$	Ok
183	23.17	6.50	Male	Yes	Sun	Dinner	4	5.79	Dr. Michael James	4718501859162	Sun6059	9162	\$\$	Generous
214	28.17	6.50	Female	Yes	Sat	Dinner	3	9.39	Marissa Jackson	4922302538691962	Sat3374	1962	\$\$	Ok
47	32.40	6.00	Male	No	Sun	Dinner	4	8.10	James Barnes	3552002592874186	Sun9677	4186	\$\$\$	Ok
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657	2842	\$\$	Ok
88	24.71	5.85	Male	No	Thur	Lunch	2	12.36	Roger Taylor	4410248629955	Thur9003	9955	\$\$	Ok

Activate Windows
Go to Settings to activate Windows.