

Applied Data Science with Python

Course-End Project: Sales Analysis

Project Statement:

AAL is a household name in Australia, known for its clothing business since 2000. It caters to all groups— kids, women, men, and seniors. It has opened branches in many states in Australia, in metropolises and tier-1 and tier-2 cities.

The business is booming, and the company is in an expansion mode. It wants to get an in-depth understanding of its sales so that it can make investment decisions. The CEO has tasked the Head of Sales and Marketing (S&M) of AAL to:

- 1) Determine the states that are generating the highest revenues and
- 2) Formulate sales programs for states with lower revenues. The Head of S&M has approached you for the same.

Analyze the company's sales data for the fourth quarter across Australia, and state by state and help the company make data-driven decisions for the coming year.

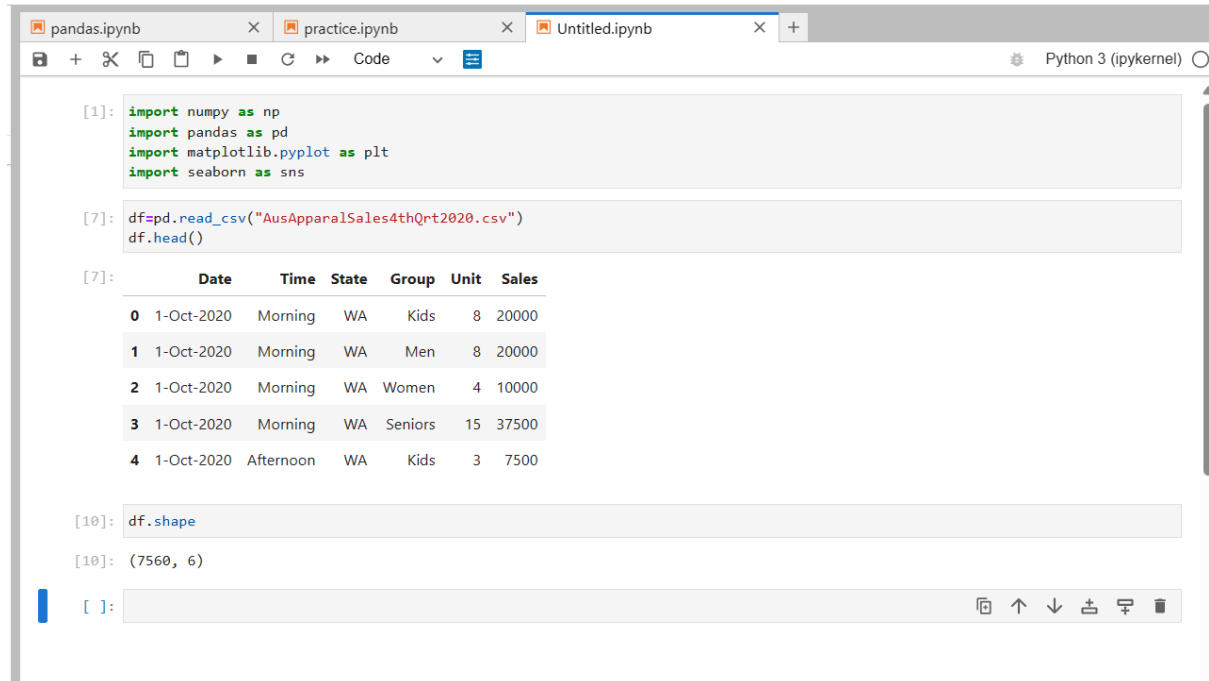
Perform the following steps:

As a data scientist, you must perform the following main steps on the enclosed data.

1. Data Wrangling
2. Data Analysis
3. Data Visualization and
4. Report Generation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
df=pd.read_csv("AusApparalSales4thQrt2020.csv")
df.head()
df.shape
```



The screenshot shows a Jupyter Notebook interface with three tabs: 'pandas.ipynb', 'practice.ipynb', and 'Untitled.ipynb'. The 'Untitled.ipynb' tab is active. The code cell [1] contains import statements for numpy, pandas, matplotlib.pyplot, and seaborn. The code cell [7] reads a CSV file and displays the first five rows of the DataFrame. The output shows a table with columns: Date, Time, State, Group, Unit, and Sales. The code cell [10] displays the shape of the DataFrame, which is (7560, 6).

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

[7]: df=pd.read_csv("AusApparalSales4thQrt2020.csv")
df.head()

[7]:
```

	Date	Time	State	Group	Unit	Sales
0	1-Oct-2020	Morning	WA	Kids	8	20000
1	1-Oct-2020	Morning	WA	Men	8	20000
2	1-Oct-2020	Morning	WA	Women	4	10000
3	1-Oct-2020	Morning	WA	Seniors	15	37500
4	1-Oct-2020	Afternoon	WA	Kids	3	7500

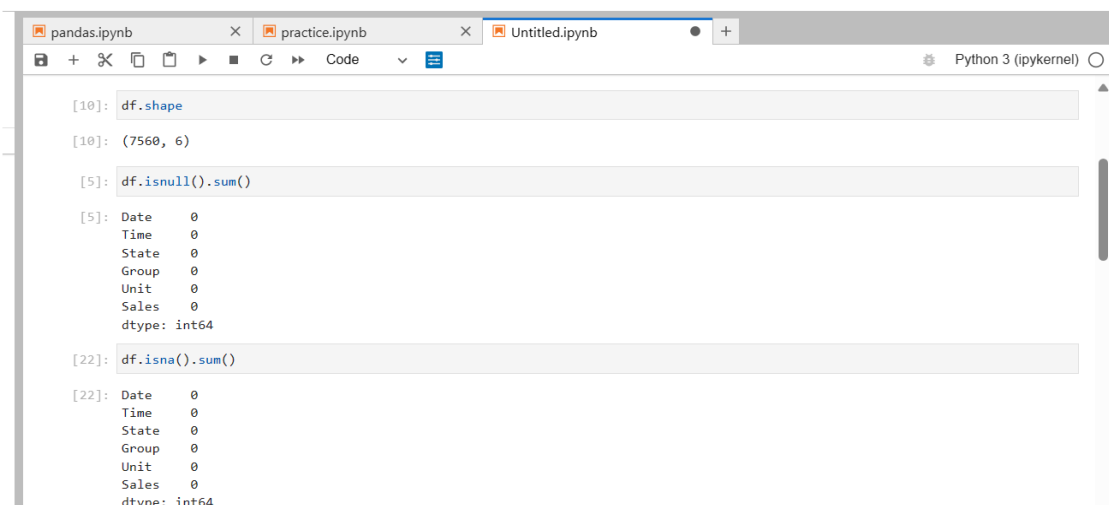
```
[10]: df.shape

[10]: (7560, 6)

[ ]:
```

1. Data Wrangling

- Ensure that the data is clean and that there is no missing or incorrect data.
 - Inspect the data manually for missing/incorrect data using the functions `isna()`, and `notna()`.
- Based on your knowledge of Data Analytics, include your recommendations for treating missing data and incorrect data. (*dropping the null values or filling them*).



The screenshot shows a Jupyter Notebook interface with three tabs: 'pandas.ipynb', 'practice.ipynb', and 'Untitled.ipynb'. The 'Untitled.ipynb' tab is active. The code cell [10] displays the shape of the DataFrame, which is (7560, 6). The code cell [5] checks for null values using `df.isnull().sum()`, resulting in 0 for all columns. The code cell [22] checks for missing values using `df.isna().sum()`, also resulting in 0 for all columns.

```
[10]: df.shape

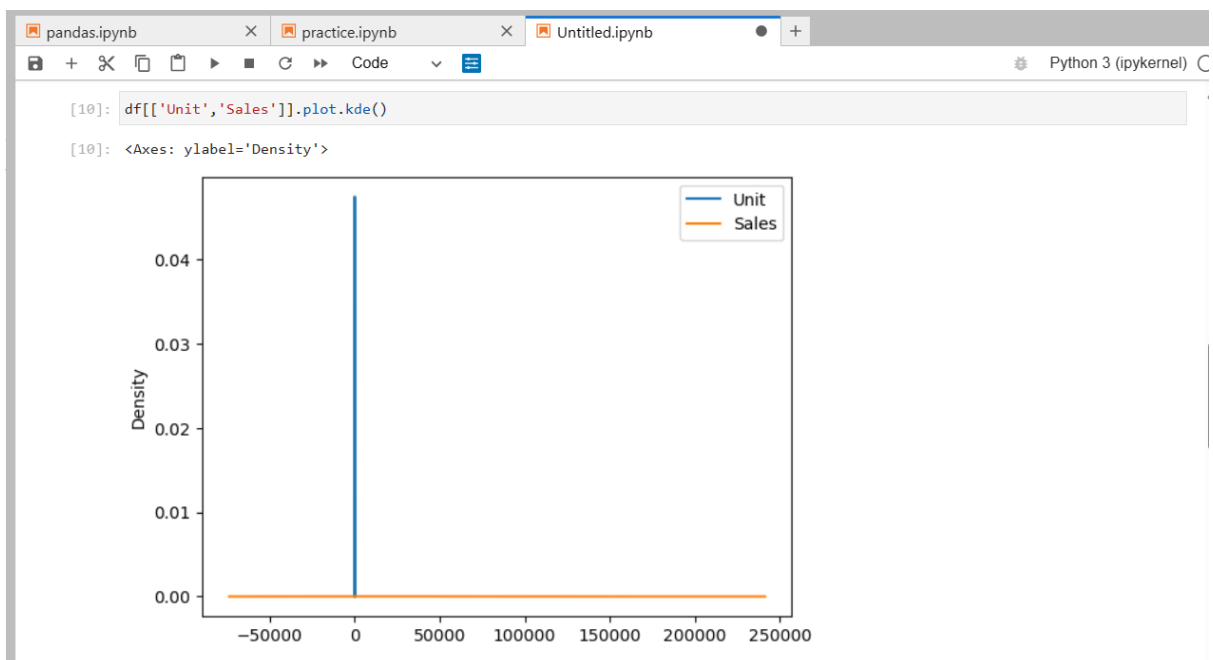
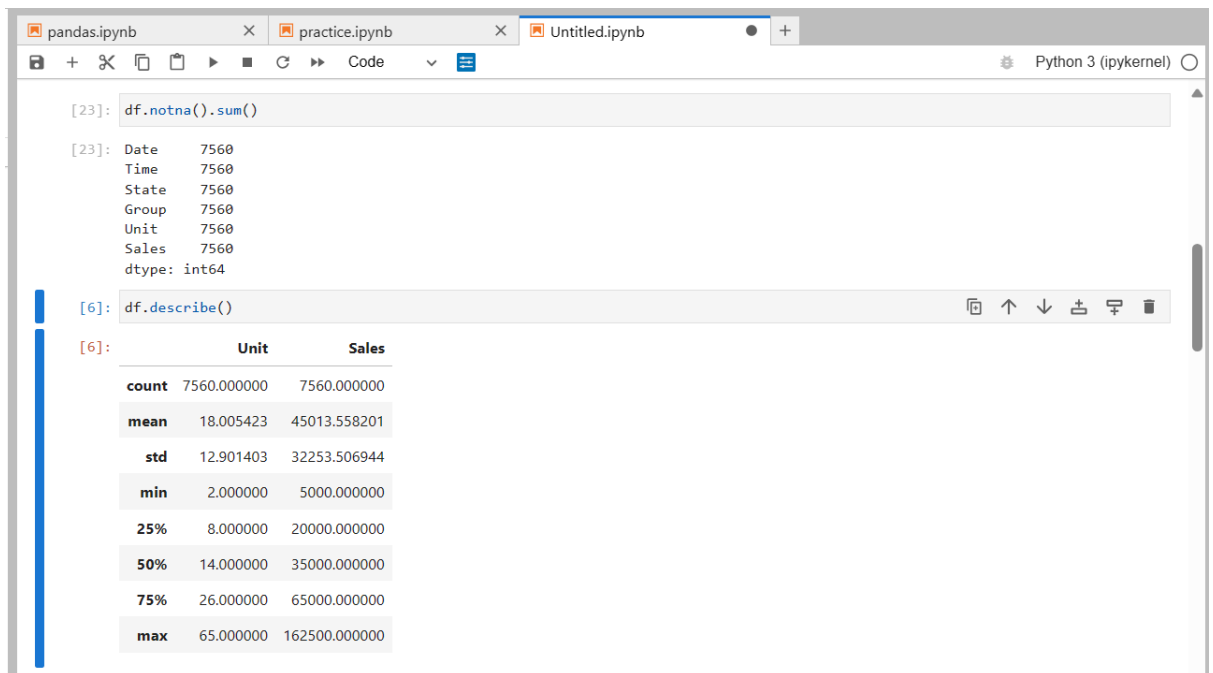
[10]: (7560, 6)

[5]: df.isnull().sum()

[5]: Date      0
Time      0
State      0
Group      0
Unit      0
Sales      0
dtype: int64

[22]: df.isna().sum()

[22]: Date      0
Time      0
State      0
Group      0
Unit      0
Sales      0
dtype: int64
```



- Select an appropriate Data Wrangling approach — data standardization or data normalization. Perform the standardization or normalization and present the data. (*Normalization is the preferred approach for this problem.*)

Data standardization

```
pandas.ipynb x practice.ipynb x Untitled.ipynb + Python 3 (ipykernel)

[14]: from sklearn.preprocessing import StandardScaler
      ss= StandardScaler()
      df1 =df[['Unit','Sales']]
      data_transformed=ss.fit_transform(df1)

[15]: type(data_transformed)

[15]: numpy.ndarray

[17]: df1 = pd.DataFrame(data_transformed,columns =['Unit','Sales'])
```

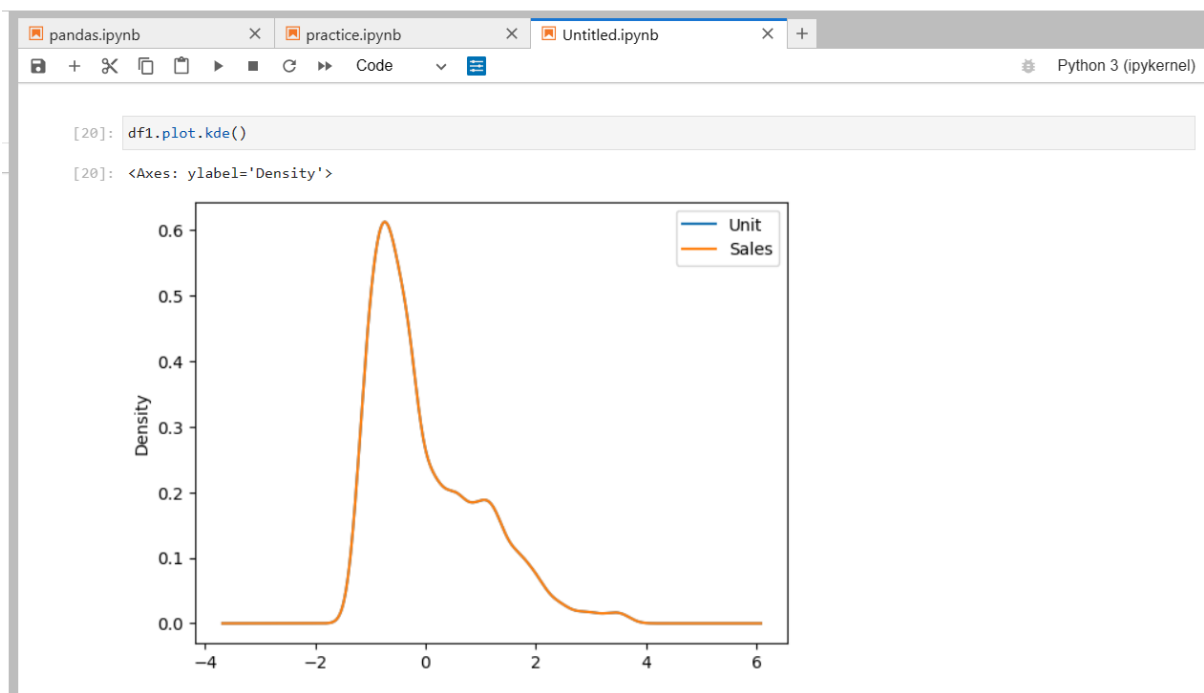
```
pandas.ipynb x practice.ipynb x Untitled.ipynb + Python 3 (ipykernel)

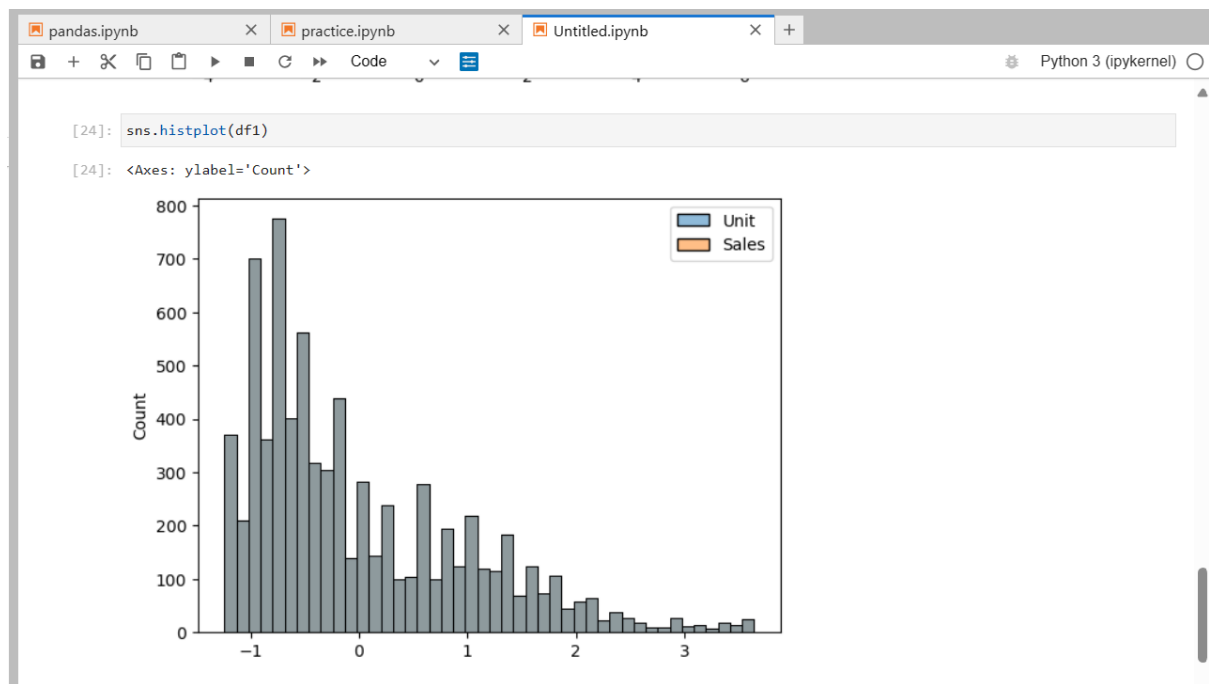
[18]: df1

[18]:
```

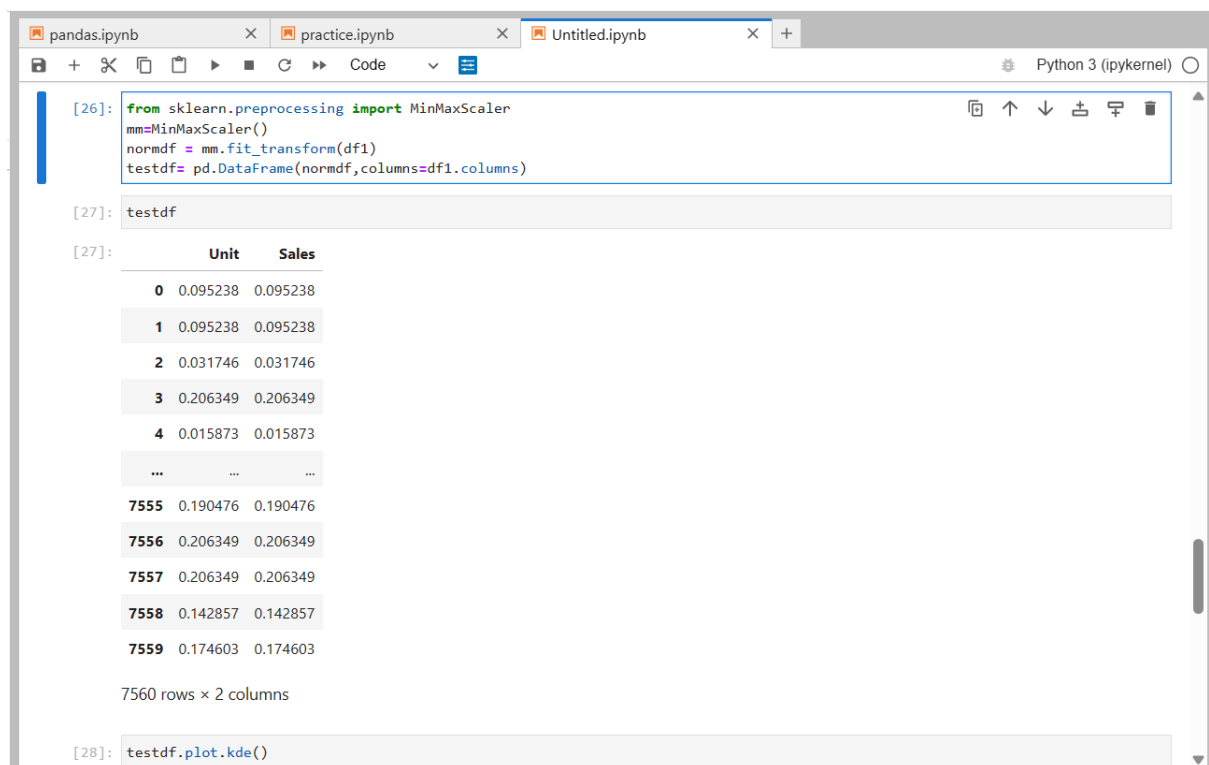
	Unit	Sales
0	-0.775581	-0.775581
1	-0.775581	-0.775581
2	-1.085645	-1.085645
3	-0.232969	-0.232969
4	-1.163162	-1.163162
...
7555	-0.310485	-0.310485
7556	-0.232969	-0.232969
7557	-0.232969	-0.232969
7558	-0.543033	-0.543033
7559	-0.388001	-0.388001

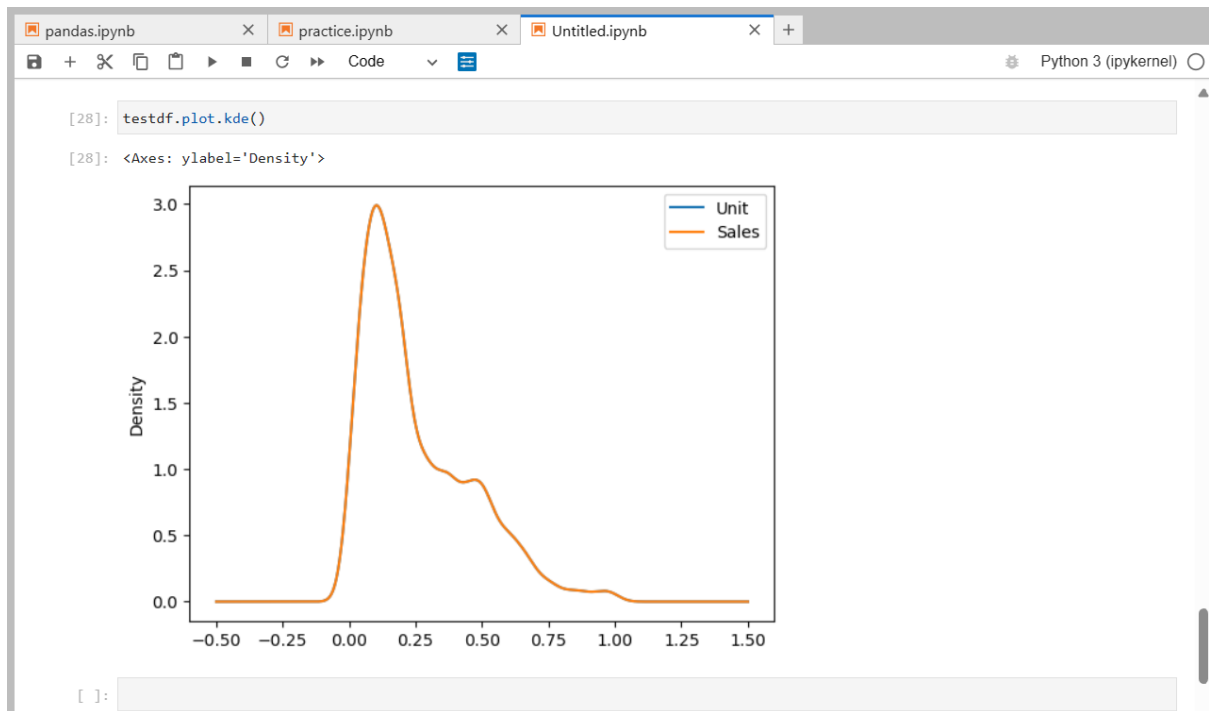
7560 rows x 2 columns





Data normalization





- Share your recommendation on the usage of the *groupby()* function for data chunking or merging.

pandas.ipynb practice.ipynb Untitled.ipynb Python 3 (ipykernel)

```
[29]: df
```

[29]:

	Date	Time	State	Group	Unit	Sales
0	1-Oct-2020	Morning	WA	Kids	8	20000
1	1-Oct-2020	Morning	WA	Men	8	20000
2	1-Oct-2020	Morning	WA	Women	4	10000
3	1-Oct-2020	Morning	WA	Seniors	15	37500
4	1-Oct-2020	Afternoon	WA	Kids	3	7500
...
7555	30-Dec-2020	Afternoon	TAS	Seniors	14	35000
7556	30-Dec-2020	Evening	TAS	Kids	15	37500
7557	30-Dec-2020	Evening	TAS	Men	15	37500
7558	30-Dec-2020	Evening	TAS	Women	11	27500
7559	30-Dec-2020	Evening	TAS	Seniors	13	32500

7560 rows × 6 columns

The screenshot shows a Jupyter Notebook with three tabs: 'pandas.ipynb', 'practice.ipynb', and 'Untitled.ipynb'. The 'Untitled.ipynb' tab is active. The code cell [31] contains the following Python code: `state_vise_totalsales=df.groupby('State')['Sales'].sum().sort_values()`. The output cell [32] displays the result as a Series with the index 'State' and values for each state. The values are: WA (22152500), NT (22580000), TAS (22760000), QLD (33417500), SA (58857500), NSW (74970000), and VIC (105565000). The output is labeled 'Name: Sales, dtype: int64'.

```
[31]: state_vise_totalsales=df.groupby('State')['Sales'].sum().sort_values()

[32]: state_vise_totalsales

[32]: State
WA      22152500
NT      22580000
TAS      22760000
QLD      33417500
SA       58857500
NSW      74970000
VIC     105565000
Name: Sales, dtype: int64
```

2. Data Analysis

- Perform descriptive statistical analysis on the data (Sales and Unit columns) *(Techniques such as mean, median, mode and standard deviation can be used.).*

The screenshot shows a Jupyter Notebook with three tabs: 'pandas.ipynb', 'practice.ipynb', and 'Untitled.ipynb'. The 'Untitled.ipynb' tab is active. The code cell [33] contains the following Python code: `df1=df[['Unit', 'Sales']]`. The output cell [34] displays the resulting DataFrame 'df1' with two columns: 'Unit' and 'Sales'. The DataFrame contains 7560 rows. The first few rows are: (0, 8, 20000), (1, 8, 20000), (2, 4, 10000), (3, 15, 37500), (4, 3, 7500). The output is labeled '7560 rows x 2 columns'.

```
[33]: df1=df[['Unit', 'Sales']]

[34]: df1

[34]:   Unit  Sales
0     8  20000
1     8  20000
2     4  10000
3    15  37500
4     3   7500
...   ...   ...
7555  14  35000
7556  15  37500
7557  15  37500
7558  11  27500
7559  13  32500

7560 rows x 2 columns
```

```
pandas.ipynb  X  practice.ipynb  X  Untitled.ipynb  +
Python 3 (ipykernel)

[35]: df1.mean()

[35]: Unit      18.005423
      Sales    45013.558201
      dtype: float64

[36]: df1.median()

[36]: Unit      14.0
      Sales    35000.0
      dtype: float64

[37]: df1.mode()

[37]:   Unit  Sales
      0    9  22500

[38]: df1.std()

[38]: Unit      12.901403
      Sales   32253.506944
      dtype: float64

[ ]:
```

- Determine which group is generating the highest sales, and which group is generating the lowest sales.
- Determine which state is generating the highest sales, and which state is generating the lowest sales.

```
pandas.ipynb  X  practice.ipynb  X  Untitled.ipynb  X  +
Python 3 (ipykernel)

[40]: group_df1=df.groupby('Group')
      group_df1.apply(lambda x:x.sort_values(by='Sales',ascending=False))

[40]:   Date      Time  State  Group  Unit  Sales
      Group
      Kids  7432  29-Dec-2020  Afternoon  VIC  Kids  65  162500
           6340  16-Dec-2020  Afternoon  VIC  Kids  65  162500
           6928  23-Dec-2020  Afternoon  VIC  Kids  63  157500
           7008  24-Dec-2020  Morning    VIC  Kids  63  157500
           7180  26-Dec-2020  Afternoon  VIC  Kids  63  157500
           ...   ...      ...      ...   ...   ...   ...
      Women  3366  11-Nov-2020  Afternoon  WA   Women  2   5000
           3358  10-Nov-2020  Evening    TAS  Women  2   5000
           3286  10-Nov-2020  Evening    WA   Women  2   5000
           3686  14-Nov-2020  Morning    TAS  Women  2   5000
           3130  8-Nov-2020   Evening    NT   Women  2   5000

7560 rows x 6 columns
```



```
pandas.ipynb x practice.ipynb x Untitled.ipynb + Python 3 (ipykernel) O

[43]: group_df1.apply(lambda x:x.sort_values(by='Sales',ascending=False)).max()

[43]: Date      9-Oct-2020
      Time      Morning
      State     WA
      Group     Women
      Unit       65
      Sales    162500
      dtype: object

[45]: group_df1.apply(lambda x:x.sort_values(by='Sales',ascending=False)).min()

[45]: Date      1-Dec-2020
      Time      Afternoon
      State     NSW
      Group     Kids
      Unit       2
      Sales     5000
      dtype: object

[ ]:
```

- Generate weekly, monthly and quarterly reports for the analysis made.

(Use suitable libraries such as NumPy, Pandas, SciPy etc. for performing the analysis.)

```
pandas.ipynb x practice.ipynb x Untitled.ipynb + Python 3 (ipykernel) O

[20]: df.groupby(['Unit'])['Sales'].describe()

[20]:
```

	count	mean	std	min	25%	50%	75%	max
Unit								
2	130.0	5000.0	0.0	5000.0	5000.0	5000.0	5000.0	5000.0
3	240.0	7500.0	0.0	7500.0	7500.0	7500.0	7500.0	7500.0
4	210.0	10000.0	0.0	10000.0	10000.0	10000.0	10000.0	10000.0
5	331.0	12500.0	0.0	12500.0	12500.0	12500.0	12500.0	12500.0
6	370.0	15000.0	0.0	15000.0	15000.0	15000.0	15000.0	15000.0
...
61	9.0	152500.0	0.0	152500.0	152500.0	152500.0	152500.0	152500.0
62	10.0	155000.0	0.0	155000.0	155000.0	155000.0	155000.0	155000.0
63	13.0	157500.0	0.0	157500.0	157500.0	157500.0	157500.0	157500.0
64	13.0	160000.0	0.0	160000.0	160000.0	160000.0	160000.0	160000.0
65	11.0	162500.0	0.0	162500.0	162500.0	162500.0	162500.0	162500.0

64 rows x 8 columns

```
pandas.ipynb X practice.ipynb X Untitled.ipynb X + Python 3 (ipykernel)

[21]: df.groupby(['State'])['Sales'].describe()

[21]:
```

	count	mean	std	min	25%	50%	75%	max
State								
NSW	1080.0	69416.666667	20626.651646	30000.0	52500.0	70000.0	85000.0	112500.0
NT	1080.0	20907.407407	8961.907893	5000.0	15000.0	20000.0	27500.0	37500.0
QLD	1080.0	30942.129630	13344.638002	7500.0	20000.0	30000.0	40000.0	62500.0
SA	1080.0	54497.685185	17460.965183	25000.0	40000.0	52500.0	67500.0	87500.0
TAS	1080.0	21074.074074	9024.684205	5000.0	15000.0	20000.0	27500.0	37500.0
VIC	1080.0	97745.370370	26621.597092	50000.0	77500.0	95000.0	112500.0	162500.0
WA	1080.0	20511.574074	9231.905897	5000.0	12500.0	20000.0	27500.0	37500.0

```
[22]: df.groupby(['Group'])['Sales'].describe()

[22]:
```

	count	mean	std	min	25%	50%	75%	max
Group								
Kids	1890.0	45011.904762	31871.491085	5000.0	20000.0	35000.0	65000.0	162500.0
Men	1890.0	45370.370370	32177.180712	5000.0	20000.0	35000.0	65000.0	160000.0
Seniors	1890.0	44464.285714	32195.360017	5000.0	20000.0	35000.0	62500.0	162500.0
Women	1890.0	45207.671958	32781.639869	5000.0	20000.0	35000.0	67500.0	162500.0

```
pandas.ipynb X practice.ipynb X Untitled.ipynb Python 3 (ipykernel)

[36]: from datetime import datetime
df['Date'] = pd.to_datetime(df['Date'])

[32]: # Monthly sales
monthly_sales = df.groupby(df['Date'].dt.to_period('M'))['Sales'].sum()

# Weekly sales
weekly_sales = df.groupby(df['Date'].dt.to_period('W'))['Sales'].sum()

# Quarterly sales
quarterly_sales = df.groupby(df['Date'].dt.to_period('Q'))['Sales'].sum()

[33]: monthly_sales

[33]: Date
2020-10      114290000
2020-11      90682500
2020-12     135330000
Freq: M, Name: Sales, dtype: int64
```

```
pandas.ipynb X practice.ipynb X Untitled.ipynb Python 3 (ipykernel)

[34]: weekly_sales

[34]: Date
2020-09-28/2020-10-04      15045000
2020-10-05/2020-10-11     27002500
2020-10-12/2020-10-18     26640000
2020-10-19/2020-10-25     26815000
2020-10-26/2020-11-01     21807500
2020-11-02/2020-11-08     20865000
2020-11-09/2020-11-15     21172500
2020-11-16/2020-11-22     21112500
2020-11-23/2020-11-29     21477500
2020-11-30/2020-12-06     29622500
2020-12-07/2020-12-13     31525000
2020-12-14/2020-12-20     31655000
2020-12-21/2020-12-27     31770000
2020-12-28/2021-01-03     13792500
Freq: W-SUN, Name: Sales, dtype: int64

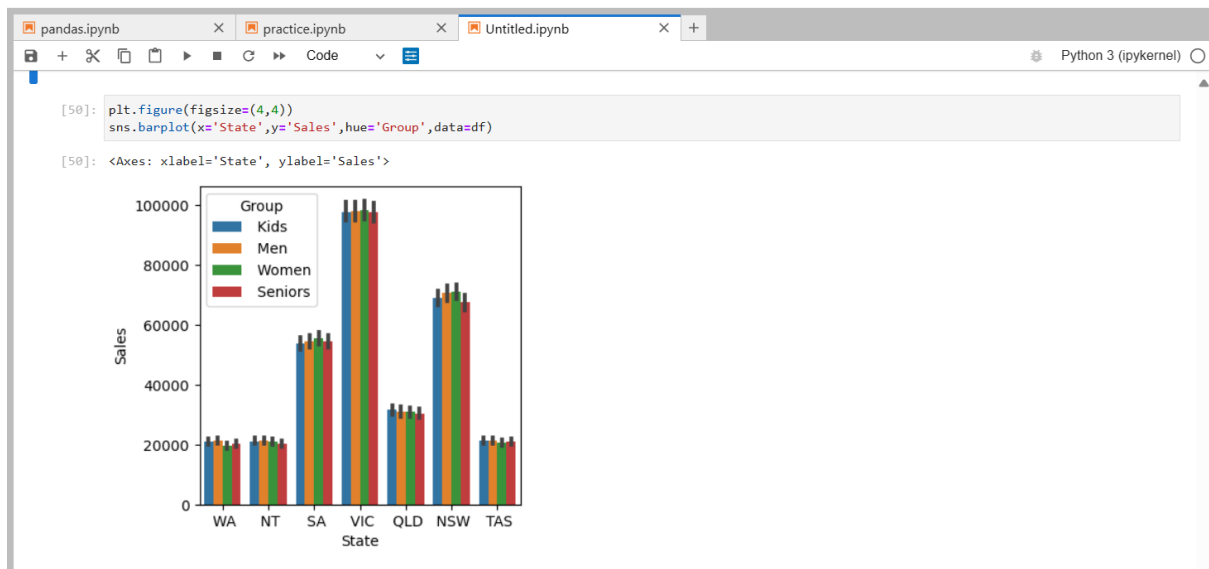
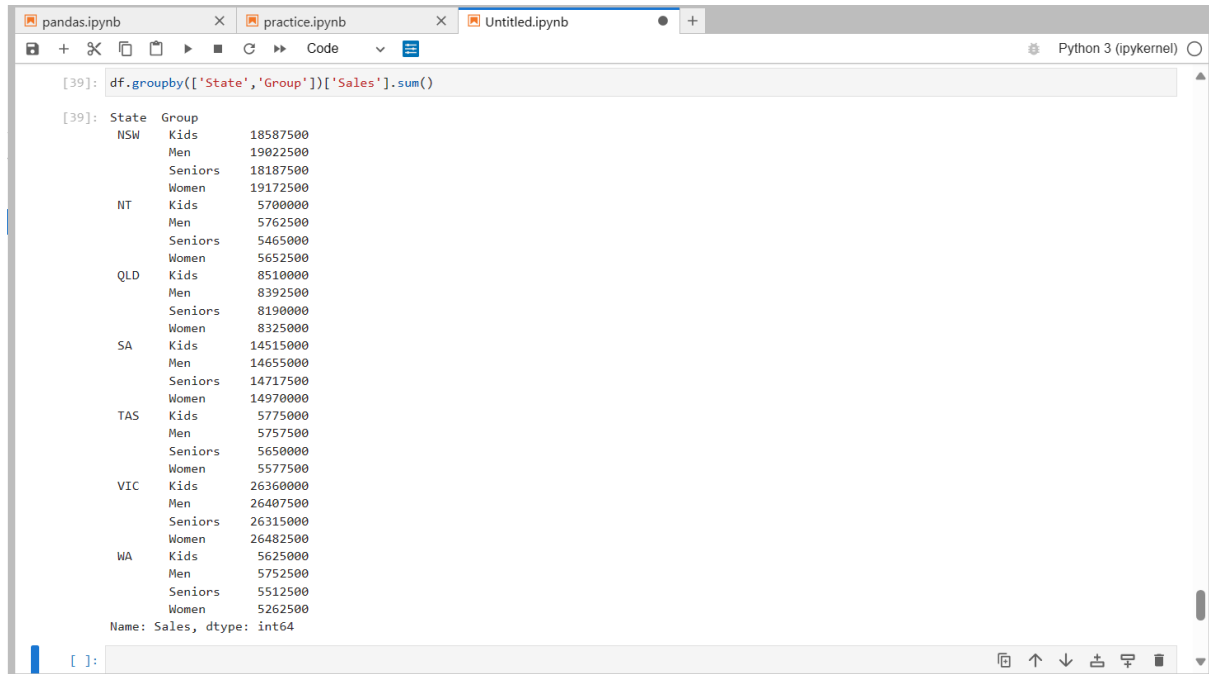
[35]: quarterly_sales

[35]: Date
2020Q4      340302500
Freq: Q-DEC, Name: Sales, dtype: int64

[ ]:
```

3. Data Visualization

- Use appropriate data visualization libraries to build a dashboard for the Head of S&M that includes for the key parameters like
 - State-wise sales analysis for different groups (kids, women, men, and seniors)

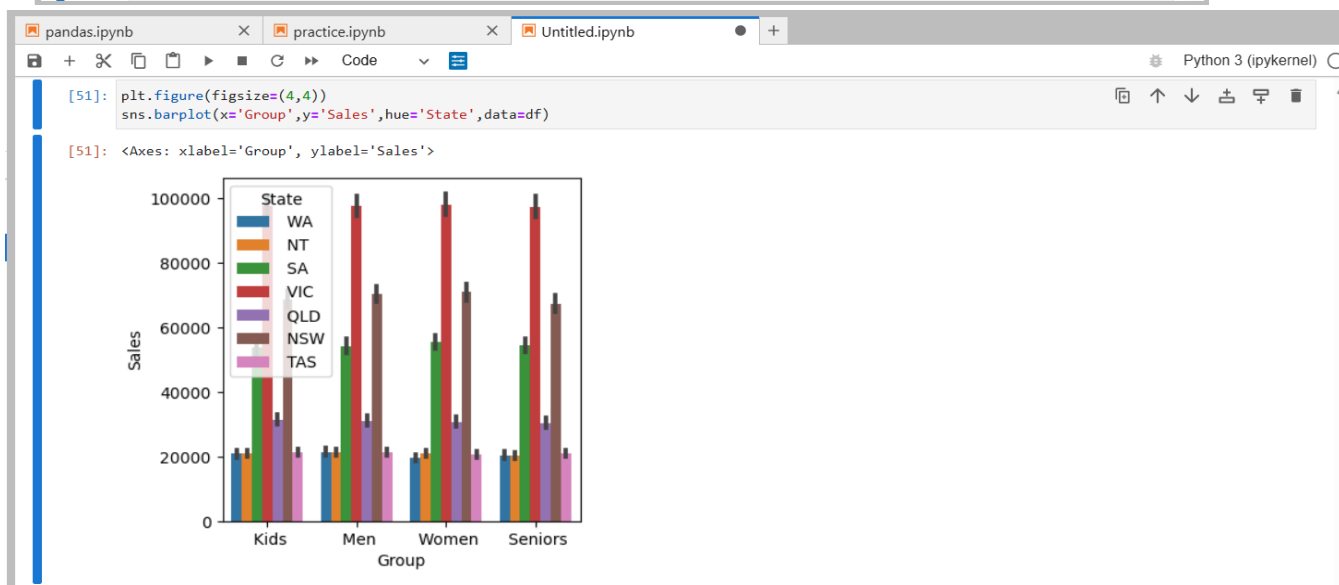


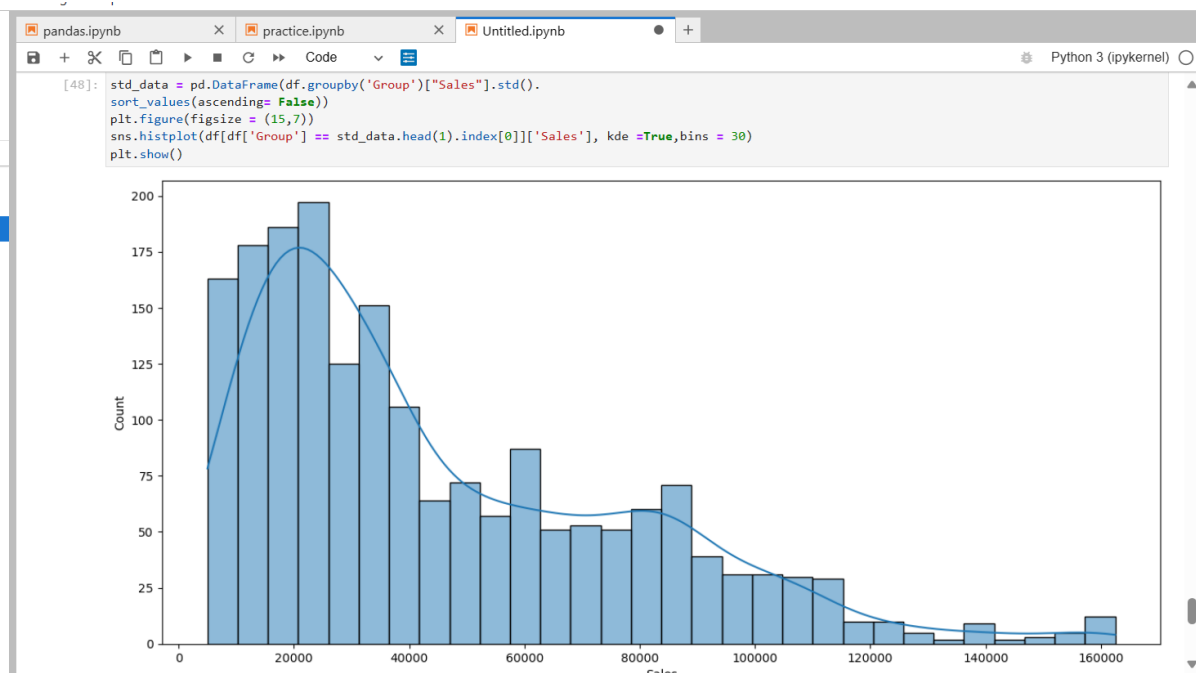
- Group-wise sales analysis (kids, women, men, and seniors) across different states.

```
pandas.ipynb x practice.ipynb x Untitled.ipynb
Python 3 (ipykernel)

[38]: df.groupby(['Group', 'State'])['Sales'].sum()

[38]: Group State Sales
Kids NSW 18587500
Kids NT 5700000
Kids QLD 8510000
Kids SA 14515000
Kids TAS 5775000
Kids VIC 26360000
Kids WA 5625000
Men NSW 19022500
Men NT 5762500
Men QLD 8392500
Men SA 14655000
Men TAS 5757500
Men VIC 26407500
Men WA 5752500
Seniors NSW 18187500
Seniors NT 5465000
Seniors QLD 8190000
Seniors SA 14717500
Seniors TAS 5650000
Seniors VIC 26315000
Seniors WA 5512500
Women NSW 19172500
Women NT 5652500
Women QLD 8325000
Women SA 14970000
Women TAS 5577500
Women VIC 26482500
Women WA 5262500
Name: Sales, dtype: int64
```





- Time-of-the-day analysis: during which time of the day are sales the highest, and during which time are sales the lowest? [This helps S&M teams design programs for increasing sales such as hyper-personalization and Next Best Offers].

The dashboard must contain daily, weekly, monthly and quarterly charts.

(Any visualization library can be used for this purpose. However, since statistical analysis is being done, Seaborn is preferred.)

```

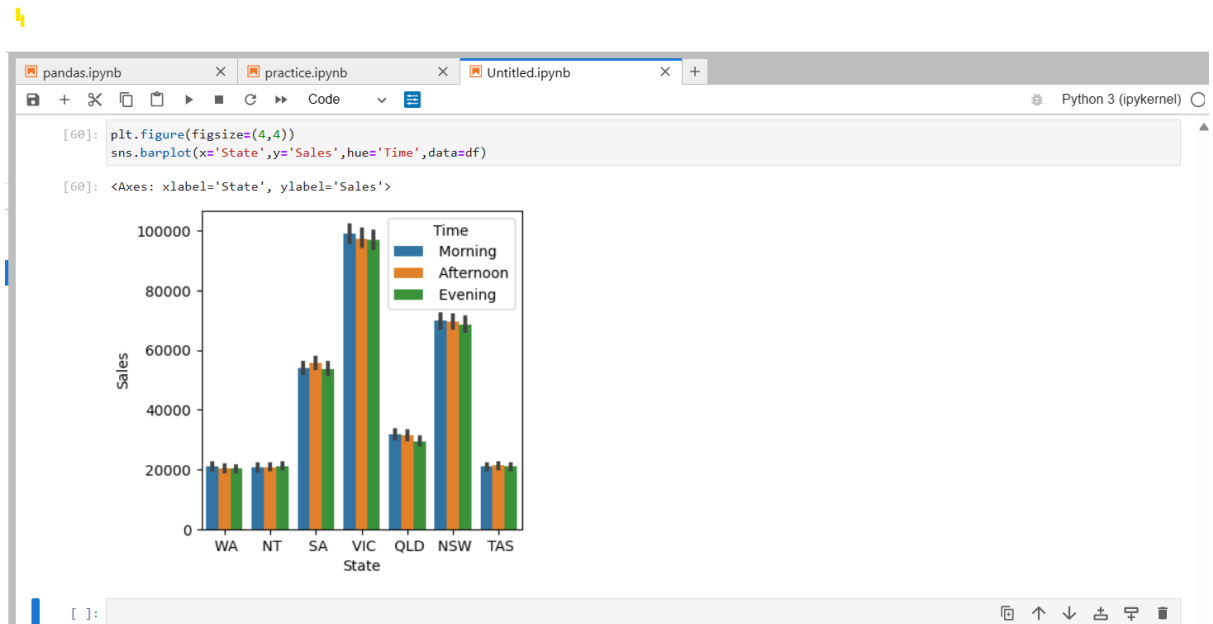
[59]: group_df1.apply(lambda x:x.sort_values(by= 'Sales',ascending=False)).max()

[59]: Date      2020-12-30 00:00:00
      Time      Morning
      State      WA
      Group      Women
      Unit      65
      Sales      162500
      dtype: object

[61]: group_df1.apply(lambda x:x.sort_values(by= 'Sales',ascending=False)).min()

[61]: Date      2020-10-01 00:00:00
      Time      Afternoon
      State      NSW
      Group      Kids
      Unit      2
      Sales      5000
      dtype: object

```



4. Report Generation

- Use JupyterLab Notebook for report generation (wrangling, analysis and visualization) *Please note that JupyterLab allows you to mix code with graphs and plots etc.*
- Use Markdown in suitable places, while presenting your report.
- The report should contain suitable graphs, plots and analysis reports, and recommendations. *Please note that different aspects of analysis demand different graphs/plots.*
 - Use box plot for descriptive statistics
 - Use Seaborn distribution plot for any other statistical plotting.