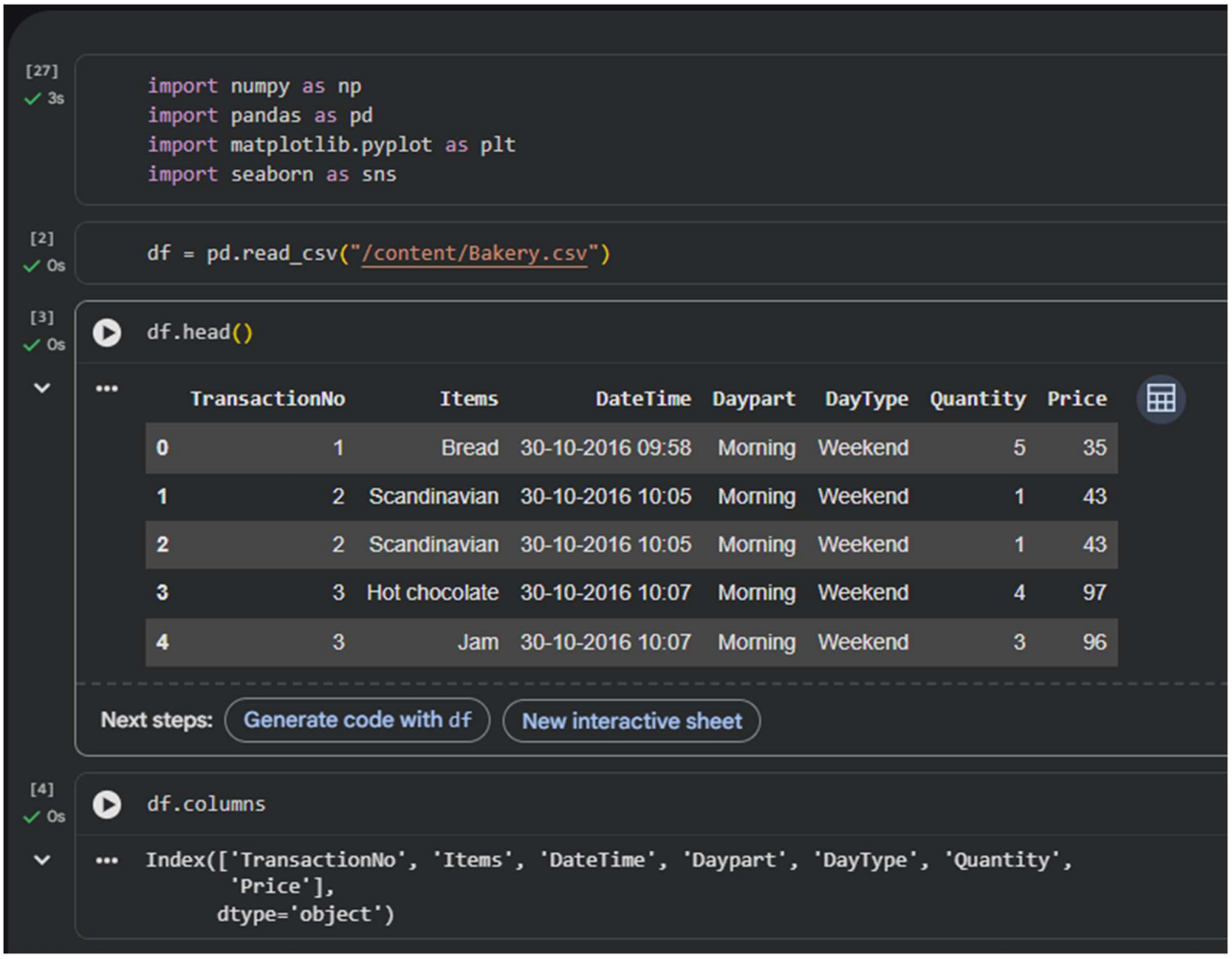
Bakery Sales Analysis using Python

End-to-End Python Data Analyst Project

Prepared by: Rushikesh Patil

Tools & Libraries Used

- Python
- Pandas
- Matplotlib
- Jupyter Notebook



```
[27] ✓ 3s
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

[2] ✓ 0s
df = pd.read_csv("/content/Bakery.csv")

[3] ✓ 0s
df.head()
```

	TransactionNo	Items	DateTime	Daypart	DayType	Quantity	Price
0	1	Bread	30-10-2016 09:58	Morning	Weekend	5	35
1	2	Scandinavian	30-10-2016 10:05	Morning	Weekend	1	43
2	2	Scandinavian	30-10-2016 10:05	Morning	Weekend	1	43
3	3	Hot chocolate	30-10-2016 10:07	Morning	Weekend	4	97
4	3	Jam	30-10-2016 10:07	Morning	Weekend	3	96

Next steps: Generate code with df | New interactive sheet

```
[4] ✓ 0s
df.columns

Index(['TransactionNo', 'Items', 'DateTime', 'Daypart', 'DayType', 'Quantity',
       'Price'],
      dtype='object')
```

Importing Required Libraries

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

Data Loading

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

The dataset is loaded using Pandas for further analysis

Data Cleaning & Feature Engineering

```
df['DateTime'] = pd.to_datetime(df['DateTime'], dayfirst=True)
df['Sales'] = df['Quantity'] * df['Price']
```

DateTime column was converted to datetime format. A new Sales column was created to calculate total revenue per transaction.

```
df.isnull().sum().sort_values(ascending=False)
```

	0
TransactionNo	0
Items	0
DateTime	0
Daypart	0
DayType	0
Quantity	0
Price	0

dtype: int64

```
df.duplicated(keep=False).sum()
```

```
np.int64(694)
```

```
df.drop_duplicates(keep="first").reset_index(drop=True)
```

	TransactionNo	Items	DateTime	Daypart	DayType	Quantity	Price
0	1	Bread	30-10-2016 09:58	Morning	Weekend	5	35
1	2	Scandinavian	30-10-2016 10:05	Morning	Weekend	1	43
2	3	Hot chocolate	30-10-2016 10:07	Morning	Weekend	4	97
3	3	Jam	30-10-2016 10:07	Morning	Weekend	3	96
4	3	Cookies	30-10-2016 10:07	Morning	Weekend	3	72
...	...	...	...	...	...	...	...
20154	9682	Coffee	04-09-2017 14:32	Afternoon	Weekend	1	43
20155	9682	Tea	04-09-2017 14:32	Afternoon	Weekend	3	39
20156	9683	Coffee	04-09-2017 14:57	Afternoon	Weekend	2	43
20157	9683	Pastry	04-09-2017 14:57	Afternoon	Weekend	4	54
20158	9684	Smoothies	04-09-2017 15:04	Afternoon	Weekend	1	99

20159 rows × 7 columns

```
revenue = df["Price"] * df["Quantity"]
print(revenue)
total_revenue = revenue.sum()
print("total_revenue:", total_revenue)
```

```
0        175
1         43
2         43
3        388
4        288
        ...
20502     43
20503    117
20504     86
20505    216
20506     99
Length: 20507, dtype: int64
total_revenue: 3139967
```

```
d2.duplicated().sum()
```

```
np.int64(348)
```

```
d2.drop_duplicates(keep="first").reset_index(drop=True)
```

	TransactionNo	Items	DateTime	Daypart	DayType	Quantity	Price
0	1	Bread	30-10-2016 09:58	Morning	Weekend	5	35
1	2	Scandinavian	30-10-2016 10:05	Morning	Weekend	1	43
2	3	Hot chocolate	30-10-2016 10:07	Morning	Weekend	4	97
3	3	Jam	30-10-2016 10:07	Morning	Weekend	3	96
4	3	Cookies	30-10-2016 10:07	Morning	Weekend	3	72
...	...	...	...	...	...	...	...
20154	9682	Coffee	04-09-2017 14:32	Afternoon	Weekend	1	43
20155	9682	Tea	04-09-2017 14:32	Afternoon	Weekend	3	39
20156	9683	Coffee	04-09-2017 14:57	Afternoon	Weekend	2	43
20157	9683	Pastry	04-09-2017 14:57	Afternoon	Weekend	4	54
20158	9684	Smoothies	04-09-2017 15:04	Afternoon	Weekend	1	99

20159 rows × 7 columns

```
df.duplicated().sum()
```

```
np.int64(348)
```

```
df.drop_duplicates(inplace=True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

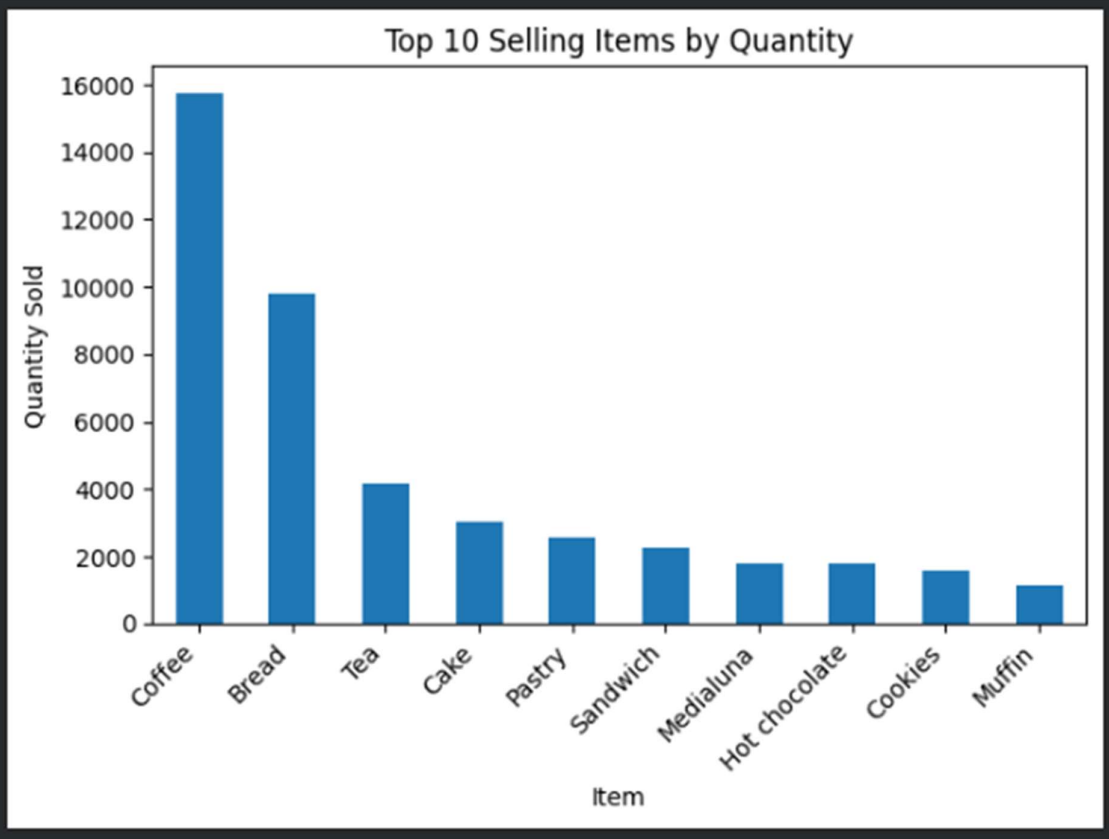
```
df['DateTime'] = pd.to_datetime(df['DateTime'], dayfirst=True)
df['Sales'] = df['Quantity'] * df['Price']
```

Exploratory Data Analysis & Visualizations

Top Selling Products

```
top_items = (
    df.groupby('Items')['Quantity']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

plt.figure()
top_items.plot(kind='bar')
plt.title('Top 10 Selling Items by Quantity')
plt.xlabel('Item')
plt.ylabel('Quantity Sold')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

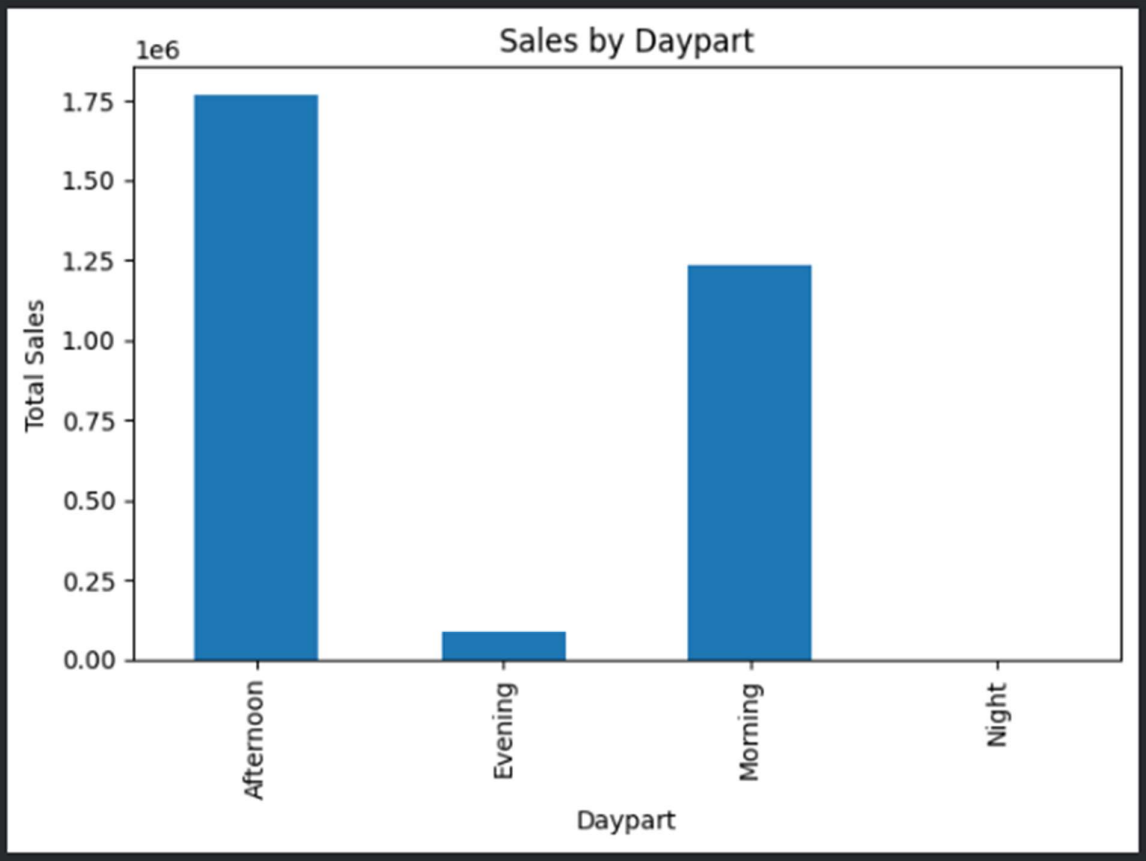


Coffee and Bread are the highest selling products and major contributors to revenue.

Sales by Daypart

```
sales_daypart = df.groupby('Daypart')['Sales'].sum()

plt.figure()
sales_daypart.plot(kind='bar')
plt.title('Sales by Daypart')
plt.xlabel('Daypart')
plt.ylabel('Total Sales')
plt.tight_layout()
plt.show()
```

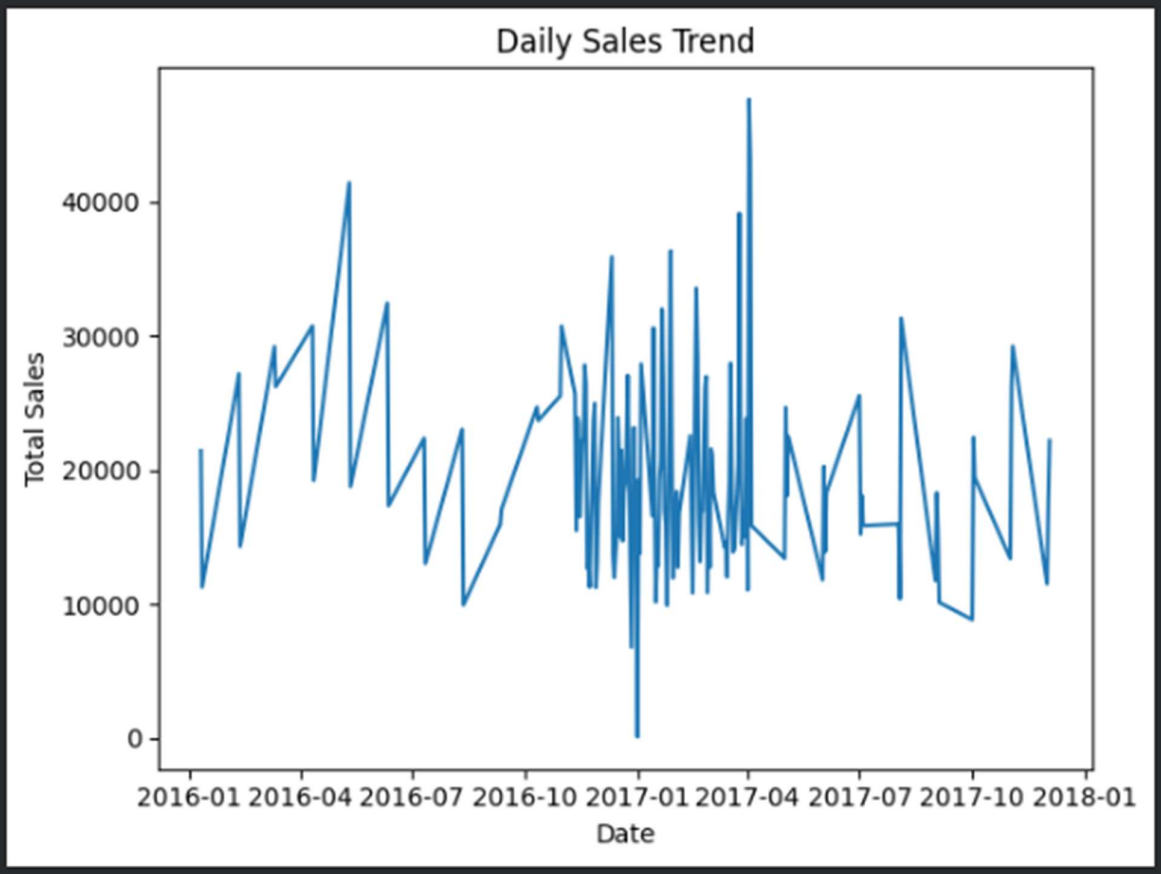


Afternoon time records the highest sales, indicating peak customer activity.

Daily Sales Trend

```
daily_sales = df.groupby(df['DateTime'].dt.date)['Sales'].sum()

plt.figure()
daily_sales.plot()
plt.title('Daily Sales Trend')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.tight_layout()
plt.show()
```

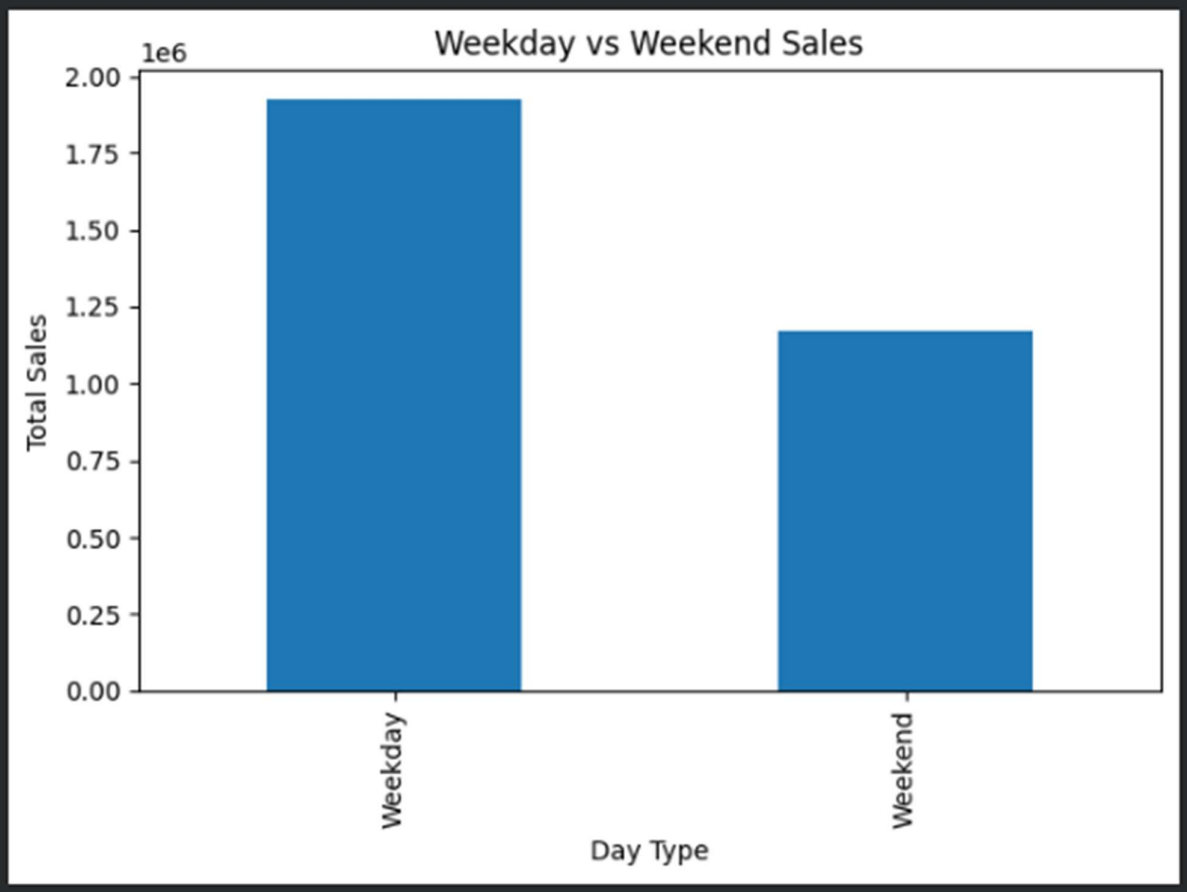


Daily sales fluctuate significantly, reflecting changes in customer demand.

Weekend vs Weekday Sales

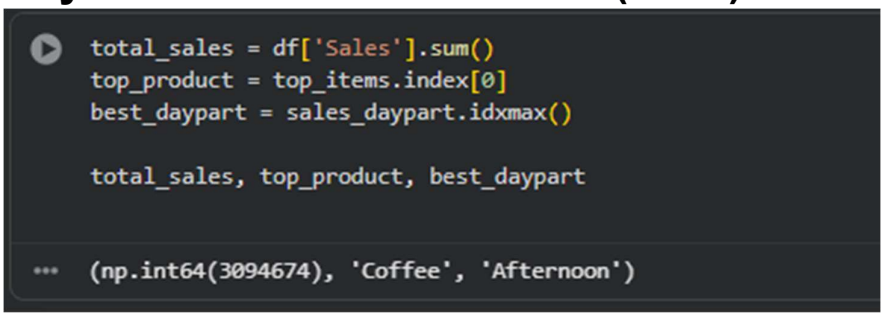
```
daytype_sales = df.groupby('DayType')['Sales'].sum()

plt.figure()
daytype_sales.plot(kind='bar')
plt.title('Weekday vs Weekend Sales')
plt.xlabel('Day Type')
plt.ylabel('Total Sales')
plt.tight_layout()
plt.show()
```



Weekday sales outperform weekend sales, suggesting higher weekday footfall.

Key Performance Indicators (KPIs)



```
total_sales = df['Sales'].sum()
top_product = top_items.index[0]
best_daypart = sales_daypart.idxmax()

total_sales, top_product, best_daypart
```

```
(np.int64(3094674), 'Coffee', 'Afternoon')
```

• Total Sales: 3139967
• Top Selling Product: Coffee
• Best Performing Daypart: Afternoon

Conclusion

This end-to-end Python analysis demonstrates the ability to clean data, perform exploratory analysis, create visualizations, and generate a professional report. The insights can be used for inventory planning, staffing optimization, and targeted promotions.