Sales Analysis Report

Table of contents

1 Setup the environment

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

1.1 Load and Inspect Data

```
# not cleand
sales_db=pd.read_csv(r'E:\_Projects\Python\1. Walmart Sales Analysis\Dataset\walmart.csv')

# cleaned data
sales=pd.read_pickle(r'E:\_Projects\Python\1. Walmart Sales Analysis\Dataset\sales_cleaned_data
```

2 Prepare the Data

2.1 Data Cleaning

```
sales_db.shape
```

```
(10051, 11)
```

```
sales_db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10051 entries, 0 to 10050
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   invoice_id      10051 non-null  int64
 1   Branch          10051 non-null  object
 2   City            10051 non-null  object
 3   category        10051 non-null  object
 4   unit_price      10020 non-null  object
 5   quantity        10020 non-null  float64
 6   date            10051 non-null  object
 7   time            10051 non-null  object
```

```
 8   payment_method  10051 non-null  object
 9   rating          10051 non-null  float64
 10  profit_margin   10051 non-null  float64
dtypes: float64(3), int64(1), object(7)
memory usage: 863.9+ KB
```

```
sales_db.head()
```

	invoice_id	Branch	City	category	unit_price	quantity	date	time
0	1	WALM003	San Antonio	Health and beauty	$74.69	7.0	05/01/19	13:08:00
1	2	WALM048	Harlingen	Electronic accessories	$15.28	5.0	08/03/19	10:29:00
2	3	WALM067	Haltom City	Home and lifestyle	$46.33	7.0	03/03/19	13:23:00
3	4	WALM064	Bedford	Health and beauty	$58.22	8.0	27/01/19	20:33:00
4	5	WALM013	Irving	Sports and travel	$86.31	7.0	08/02/19	10:37:00

```
sales_db['date'] = pd.to_datetime(sales_db['date'])
```

```
sales_db['unit_price'] = sales_db['unit_price'].str.replace('$', '').astype(float)
```

```
sales_db.duplicated().sum()
```

```
np.int64(51)
```

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

	invoice_id	Branch	City	category	unit_price	quantity	date	tim
0	1	WALM003	San Antonio	Health and beauty	74.69	7.0	2019-05-01	13:0
1	2	WALM048	Harlingen	Electronic accessories	15.28	5.0	2019-08-03	10:2
2	3	WALM067	Haltom City	Home and lifestyle	46.33	7.0	2019-03-03	13:2
3	4	WALM064	Bedford	Health and beauty	58.22	8.0	2019-01-27	20:3

	invoice_id	Branch	City	category	unit_price	quantity	date	tim
4	5	WALM013	Irving	Sports and travel	86.31	7.0	2019-08-02	10:3
...	...	...	...	...	...	...	...	...
9995	9996	WALM056	Rowlett	Fashion accessories	37.00	3.0	2023-03-08	10:1
9996	9997	WALM030	Richardson	Home and lifestyle	58.00	2.0	2021-02-22	14:2
9997	9998	WALM050	Victoria	Fashion accessories	52.00	3.0	2023-06-15	16:0
9998	9999	WALM032	Tyler	Home and lifestyle	79.00	2.0	2021-02-25	12:2
9999	10000	WALM069	Rockwall	Fashion accessories	62.00	3.0	2020-09-26	9:48

```
sales_db.isna().sum()
```

```
invoice_id         0
Branch             0
City               0
category           0
unit_price        31
quantity          31
date               0
time               0
payment_method     0
rating             0
profit_margin      0
dtype: int64
```

```
sales_db.dropna(inplace = True)
```

2.2 Add Columns

```
sales_db['total'] = sales_db['unit_price'] * sales_db['quantity']
```

```
sales_db['year'] = sales_db['date'].dt.year
```

```
sales_db['year'] = sales_db['year'].astype(int)
```

```
sales_db.head()
```

	invoice_id	Branch	City	category	unit_price	quantity	date	time
0	1	WALM003	San Antonio	Health and beauty	74.69	7.0	2019-05-01	13:08:0
1	2	WALM048	Harlingen	Electronic accessories	15.28	5.0	2019-08-03	10:29:0
2	3	WALM067	Haltom City	Home and lifestyle	46.33	7.0	2019-03-03	13:23:0
3	4	WALM064	Bedford	Health and beauty	58.22	8.0	2019-01-27	20:33:0
4	5	WALM013	Irving	Sports and travel	86.31	7.0	2019-08-02	10:37:0

2.3 Save the cleaned data

```
# sales_db.to_pickle(r'E:\_Projects\Python\1. Walmart Sales Analysis\Dataset\sales_cleaned_data
```

3 EDA

3.1 Load the cleaned data

```
sales = pd.read_pickle(r'E:\_Projects\Python\1. Walmart Sales Analysis\Dataset\sales_cleaned_da
sales.head()
```

	invoice_id	Branch	City	category	unit_price	quantity	date	time
0	1	WALM003	San Antonio	Health and beauty	74.69	7.0	2019-05-01	13:08:0
1	2	WALM048	Harlingen	Electronic accessories	15.28	5.0	2019-08-03	10:29:0
2	3	WALM067	Haltom City	Home and lifestyle	46.33	7.0	2019-03-03	13:23:0
3	4	WALM064	Bedford	Health and beauty	58.22	8.0	2019-01-27	20:33:0
4	5	WALM013	Irving	Sports and travel	86.31	7.0	2019-08-02	10:37:0

3.2 Generate a summary of key metrics

```
summary = pd.DataFrame({
    "Metric": ["Total Revenue", "Total Invoices", "Total Sales", "mean rating","First Date", "
    "Value" : [
        f"${sales['total'].sum():,.2f}",
        f"{sales['invoice_id'].count():,}",
        f"{sales['quantity'].sum():,}",
        f"{round(sales['rating'].mean(), 2)}",
        sales['date'].min().date(),
        sales['date'].max().date()
    ]
})

summary
```

	Metric	Value
0	Total Revenue	$1,209,726.38
1	Total Invoices	9,969
2	Total Sales	23,483.0
3	mean rating	5.83
4	First Date	2019-01-01
5	Last Date	2023-12-31

3.3 Display the first 10 unique cities and count the remaining ones

```
cities = sales['City'].unique()
for i, city in enumerate(cities[:10], start=1):
    print(f"{i}. {city}")
    print("*" * 10)
print(f"... , {len(cities)-10} Other cities")
```

```
1. San Antonio
**********
2. Harlingen
**********
3. Haltom City
**********
4. Bedford
**********
5. Irving
**********
6. Denton
**********
7. Cleburne
```

```
**********
8. Canyon
**********
9. Grapevine
**********
10. Texas City
**********
... , 88 Other cities
```

3.4 List all unique product categories with numbering

```python
for i, category in enumerate(sales['category'].unique(), start=1):
    print(f"{i}. {category}")
    print("*" * 10)
```

```
1. Health and beauty
**********
2. Electronic accessories
**********
3. Home and lifestyle
**********
4. Sports and travel
**********
5. Food and beverages
**********
6. Fashion accessories
**********
```

3.5 Display the first 10 unique branches and count the remaining ones

```python
branches = sales['Branch'].unique()
for i, branch in enumerate(branches[:10], start=1):
    print(f"{i}. {branch}")
    print("*" * 10)
print(f"... , {len(branches)-10} Other Branches")
```

```
1. WALM003
**********
2. WALM048
**********
3. WALM067
**********
4. WALM064
```

```
**********
5. WALM013
**********
6. WALM026
**********
7. WALM088
**********
8. WALM100
**********
9. WALM066
**********
10. WALM065
**********
... , 90 Other Branches
```

3.6 List all unique payment methods with numbering

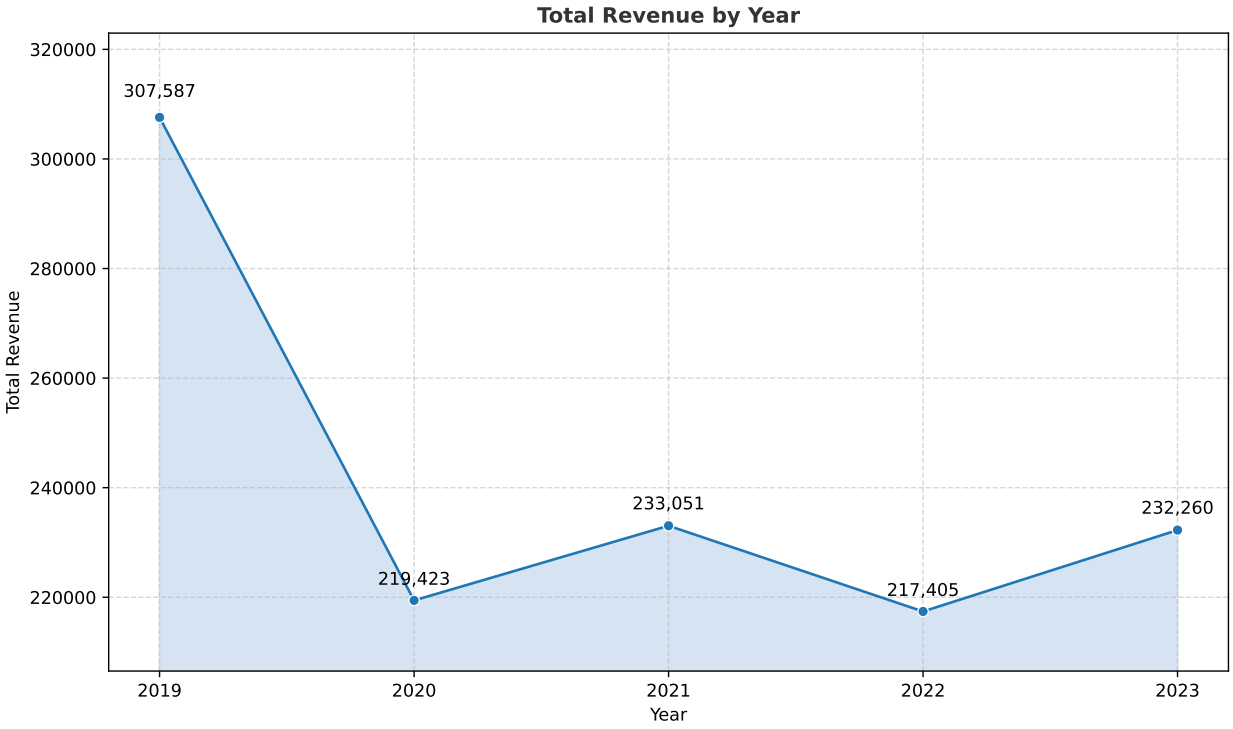
```python
for i, payment_method in enumerate(sales_db['payment_method'].unique(), start=1):
    print(f"{i}. {payment_method}")
    print("*" * 10)
```

```
1. Ewallet
**********
2. Cash
**********
3. Credit card
**********
```

4 Data Analsis Report

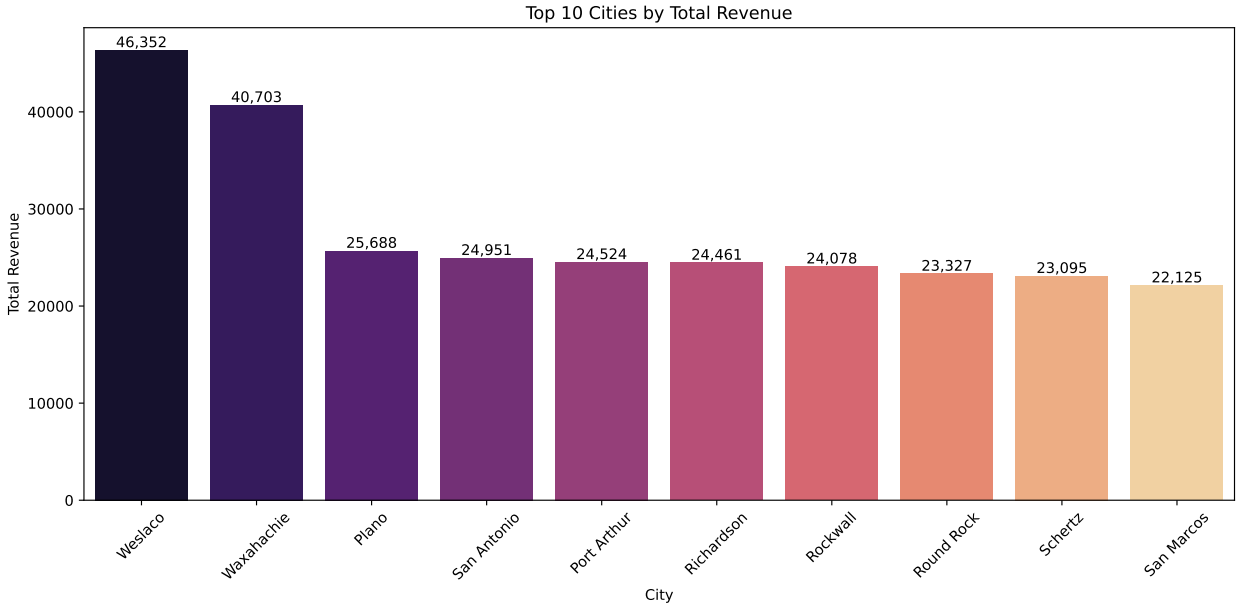
4.1 Plot total revenue by year using a line chart with value labels and shaded area

```
year_totals = sales.groupby('year')['total'].sum().reset_index()
min_total = year_totals['total'].min()
max_total = year_totals['total'].max()

plt.figure(figsize=(10,6))

sns.lineplot(
    x = 'year',
    y = 'total',
    data = year_totals,
    marker = 'o',
    color = '#1f77b4'
)

plt.fill_between(
    year_totals['year'],
    year_totals['total'],
    color = '#aec7e8',
    alpha = 0.5
)

for i, row in year_totals.iterrows():
    plt.text(
        row['year'],
        row['total'] + 0.01*row['total'],
        f"{row['total']:,.0f}",
        ha='center', va='bottom'
    )

plt.title("Total Revenue by Year", fontsize=12, fontweight='bold', color='#333333')
plt.xlabel("Year", fontsize=10)
plt.ylabel("Total Revenue", fontsize=10)
plt.xticks(year_totals['year'])
plt.grid(True, linestyle='--', alpha=0.5)

plt.ylim(min_total*0.95, max_total*1.05)
plt.tight_layout()
plt.show()
```


Total Revenue by Year
320000
300000
280000
260000
240000
220000
Total Revenue
307,587
219,423
233,051
217,405
232,260
2019
2020
2021
2022
2023
Year

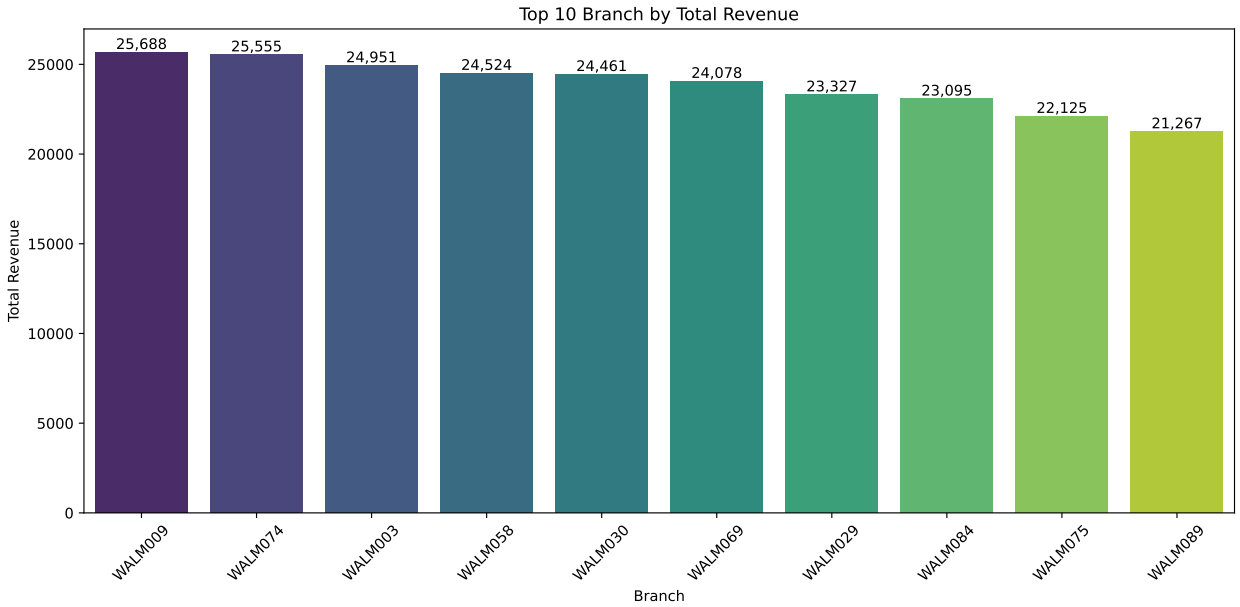
4.2 Top 10 cities by total revenue

```
top10 = sales.groupby('City')['total'].sum().sort_values(ascending=False).head(10).reset_index

plt.figure(figsize=(12,6))
ax = sns.barplot(
    x = 'City',
    y = 'total',
    data = top10,
    hue = 'City',
    palette = 'magma',
    dodge = False
)
for p in ax.patches:
    ax.annotate(f"{p.get_height():,.0f}",
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom')

plt.title("Top 10 Cities by Total Revenue")
plt.xlabel("City")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



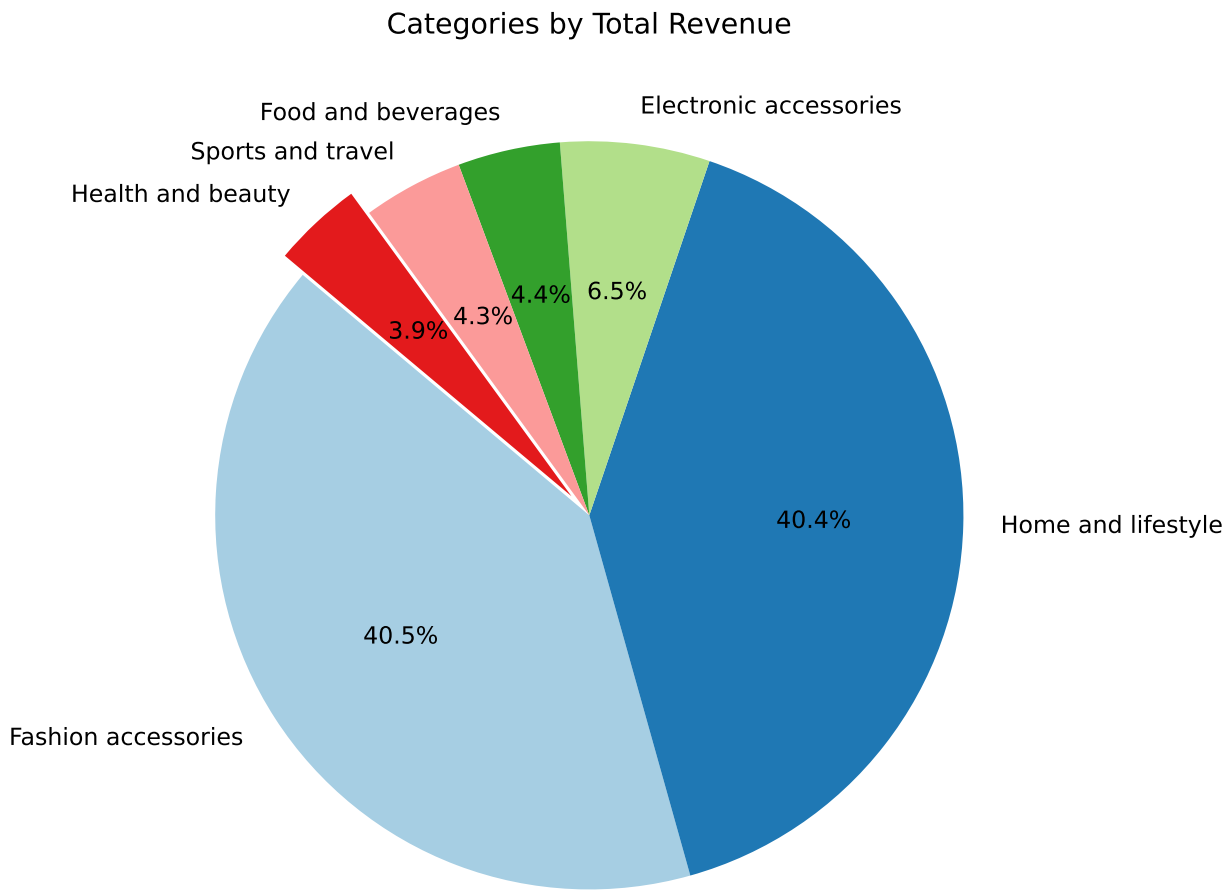
4.3 Top 10 Branches by total revenue

```
top10 = sales.groupby('Branch')['total'].sum().sort_values(ascending=False).head(10).reset_inde

plt.figure(figsize=(12,6))
ax = sns.barplot(
    x = 'Branch',
    y = 'total',
    data = top10,
    hue = 'Branch',
    palette = 'viridis',
    dodge = False
)

for p in ax.patches:
    ax.annotate(f"{p.get_height():,.0f}",
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom')

plt.title("Top 10 Branch by Total Revenue")
plt.xlabel("Branch")
plt.ylabel("Total Revenue")

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



4.4 Categories by total revenue

```
top10_category = sales.groupby('category')['total'].sum().sort_values(ascending=False)

plt.figure(figsize=(7,6))
explode = [0.07 if val == top10_category.min() else 0 for val in top10_category]

plt.pie(
    top10_category,
    labels = top10_category.index,
    autopct = '%1.1f%%',
    startangle = 140,
    colors = plt.cm.Paired.colors,
    explode = explode
)
plt.title("Categories by Total Revenue")
plt.tight_layout()
plt.show()
```



4.5 Payment methods by total revenue

```
top10_payment_method = sales.groupby('payment_method')['total'].sum().sort_values(ascending=Fal

plt.figure(figsize=(5,5))
explode = [0.05 if val == top10_payment_method.min() else 0 for val in top10_payment_method]

plt.pie(
    top10_payment_method.values,
    labels = top10_payment_method.index,
    autopct = '%1.1f%%',
    startangle = 140,
    colors = plt.cm.Paired.colors,
    explode = explode
)

plt.title("Payment Methods by Total Revenue")
plt.tight_layout()
plt.show()
```