Walmart Sales Data Analysis

About

This project aims to explore the Walmart Sales data to understand top performing branches and products, sales trend of of different products, customer behaviour. The aims is to study how sales strategies can be improved and optimized. The dataset was obtained from the Kaggle Walmart Sales Forecasting Competition.

"In this recruiting competition, job-seekers are provided with historical sales data for 45 Walmart stores located in different regions. Each store contains many departments, and participants must project the sales for each department in each store. To add to the challenge, selected holiday markdown events are included in the dataset. These markdowns are known to affect sales, but it is challenging to predict which departments are affected and the extent of the impact." source

Purposes Of The Project

The major aim of thie project is to gain insight into the sales data of Walmart to understand the different factors that affect sales of the different branches.

About Data

The dataset was obtained from the Kaggle Walmart Sales Forecasting Competition. This dataset contains sales transactions from a three different branches of Walmart, respectively located in Mandalay, Yangon and Naypyitaw. The data contains 17 columns and 1000 rows:

Column	Description	Data Type
invoice_id	Invoice of the sales made	VARCHAR(30)
branch	Branch at which sales were made	VARCHAR(5)
city	The location of the branch	VARCHAR(30)
customer_type	The type of the customer	VARCHAR(30)
gender	Gender of the customer making purchase	VARCHAR(10)
product_line	Product line of the product solf	VARCHAR(100)
unit_price	The price of each product	DECIMAL(10, 2)
quantity	The amount of the product sold	INT
VAT	The amount of tax on the purchase	FLOAT(6, 4)
total	The total cost of the purchase	DECIMAL(10, 2)
date	The date on which the purchase was made	DATE
time	The time at which the purchase was made	TIMESTAMP
payment_method	The total amount paid	DECIMAL(10, 2)
cogs	Cost Of Goods sold	DECIMAL(10, 2)
gross_margin_percentage	Gross margin percentage	FLOAT(11, 9)

gross_income	Gross Income	DECIMAL(10, 2)
rating	Rating	FLOAT(2, 1)

Analysis List

1. Product Analysis

> Conduct analysis on the data to understand the different product lines, the products lines performing best and the product lines that need to be improved.

1. Sales Analysis

> This analysis aims to answer the question of the sales trends of product. The result of this can help use measure the effectiveness of each sales strategy the business applies and what modificatoins are needed to gain more sales.

1. Customer Analysis

> This analysis aims to uncover the different customers segments, purchase trends and the profitability of each customer segment.

Approach Used

1. **Data Wrangling:** This is the first step where inspection of data is done to make sure **NULL** values and missing values are detected and data replacement methods are used to replace, missing or **NULL** values.

> 1. Build a database
> 2. Create table and insert the data.
> 3. Select columns with null values in them. There are no null values in our database as in creating the tables, we set **NOT NULL** for each field, hence null values are filtered out.

1. **Feature Engineering:** This will help use generate some new columns from existing ones.

> 1. Add a new column named `time_of_day` to give insight of sales in the Morning, Afternoon and Evening. This will help answer the question on which part of the day most sales are made.
>
> 2. Add a new column named `day_name` that contains the extracted days of the week on which the given transaction took place (Mon, Tue, Wed, Thur, Fri). This will help answer the question on which week of the day each branch is busiest.
>
> 3. Add a new column named `month_name` that contains the extracted months of the year on which the given transaction took place (Jan, Feb, Mar). Help determine which month of the year has the most sales and profit.

1. **Exploratory Data Analysis (EDA):** Exploratory data analysis is done to answer the listed questions and aims of this project.

2. **Conclusion:**

Business Questions To Answer

Generic Question

1. How many unique cities does the data have?
2. In which city is each branch?

Product

1. How many unique product lines does the data have?
2. What is the most common payment method?
3. What is the most selling product line?
4. What is the total revenue by month?
5. What month had the largest COGS?
6. What product line had the largest revenue?
7. What is the city with the largest revenue?
8. What product line had the largest VAT?
9. Fetch each product line and add a column to those product line showing "Good", "Bad". Good if its greater than average sales
10. Which branch sold more products than average product sold?
11. What is the most common product line by gender?
12. What is the average rating of each product line?

Sales

1. Number of sales made in each time of the day per weekday
2. Which of the customer types brings the most revenue?
3. Which city has the largest tax percent/ VAT (**Value Added Tax**)?
4. Which customer type pays the most in VAT?

Customer

1. How many unique customer types does the data have?
2. How many unique payment methods does the data have?
3. What is the most common customer type?
4. Which customer type buys the most?
5. What is the gender of most of the customers?
6. What is the gender distribution per branch?
7. Which time of the day do customers give most ratings?
8. Which time of the day do customers give most ratings per branch?
9. Which day fo the week has the best avg ratings?
10. Which day of the week has the best average ratings per branch?

Revenue And Profit Calculations

$$COGS = unitsPrice * quantity$$

$$VAT = 5\% * COGS$$

VAT is added to the $COGS$ and this is what is billed to the customer.

$total(gross_sales) = VAT + COGS$

$grossProfit(grossIncome) = total(gross_sales) - COGS$

Gross Margin is gross profit expressed in percentage of the total(gross profit/revenue)

$$\text{Gross Margin} = \frac{\text{gross income}}{\text{total revenue}}$$

Example with the first row in our DB:

Data given:

- $\text{Unite Price} = 45.79$
- $\text{Quantity} = 7$

$COGS = 45.79 * 7 = 320.53$

$\text{VAT} = 5\% * COGS$
$= 5\%320.53 = 16.0265$

$total = VAT + COGS$
$= 16.0265 + 320.53 =$
336.5565$

$$\text{Gross Margin Percentage} = \frac{\text{gross income}}{\text{total revenue}}$$
$= \frac{16.0265}{336.5565} = 0.047619$
$\approx 4.7619\%$

INSTALL MYSQL DRIVER

In [1]:
```
!pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in /Users/tariniprasaddas/anaconda
3/lib/python3.11/site-packages (8.2.0)
Requirement already satisfied: protobuf<=4.21.12,>=4.21.1 in /Users/tariniprasaddas/anac
onda3/lib/python3.11/site-packages (from mysql-connector-python) (4.21.12)
```

IMPORT MYSQL CONNECTOR

In [5]:
```
import mysql.connector
```

CREATE CONNECTION

In [2]:
```
import mysql.connector
mydb=mysql.connector.connect(
    host='localhost',
    user='root',
    password='Kanha@8144',
    database='walmart_sales'

)
```

```
cursor=mydb.cursor()
```

CREATE CURSOR

In [7]:
```
cursor=mydb.cursor()
```

CREATE DATABASE(walmart_sales)

In [8]:
```
query="CREATE DATABASE IF NOT EXISTS walmart_sales "
cursor.execute(query)
```

In [9]:
```
query="SHOW DATABASES"
cursor.execute(query)
cursor.fetchall()
```

Out[9]:
```
[('imdb',),
 ('information_schema',),
 ('mysql',),
 ('performance_schema',),
 ('practiceDB',),
 ('sys',),
 ('testDB',),
 ('walmart_sales',),
 ('walmartsales',)]
```

CREATE TABLE

In [10]:
```
mydb=mysql.connector.connect(
    host='localhost',
    user='root',
    password='Kanha@8144',
    database='walmart_sales'

)

cursor=mydb.cursor()



query="""CREATE TABLE sales (
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    quantity INT NOT NULL,
    tax_5_pct DECIMAL(10, 4) NOT NULL,
    total DECIMAL(10, 4) NOT NULL,
    date DATE NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10, 2) NOT NULL,
    gross_margin_percentage DECIMAL(10, 9) NOT NULL,
    gross_income DECIMAL(10, 4) NOT NULL,
    rating DECIMAL(3, 1) NOT NULL
```

```
)
"""
cursor.execute(query)
```

In [11]:
```
query= "SHOW TABLES"
cursor.execute(query)
cursor.fetchall()
```

Out[11]:
```
[('sales',)]
```

In [ ]:
```

```

INSERT DATA INTO TABLE

In [17]:
```
import csv
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}

# CSV file path
csv_file_path = 'WalmartSalesData.csv'

# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

# Read the CSV file
with open(csv_file_path, 'r') as csv_file:
    csv_reader = csv.reader(csv_file)
    next(csv_reader)  # Skip the header row if it exists

    # Define the SQL query to insert data into the 'sales' table
    insert_query = """
    INSERT INTO sales (invoice_id, branch, city, customer_type, gender, product_line, un
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    """

    # Insert each row from the CSV into the MySQL table
    for row in csv_reader:
        # Make sure to convert numerical values to the appropriate data types
        row = (
            row[0], row[1], row[2], row[3], row[4], row[5],
            float(row[6]), int(row[7]), float(row[8]), float(row[9]),
            row[10], row[11], row[12],
            float(row[13]), float(row[14]), float(row[15]), float(row[16])
        )
        cursor.execute(insert_query, row)

# Commit the changes and close the connection
connection.commit()


print("Data has been successfully inserted into the 'sales' table.")
```

```
Data has been successfully inserted into the 'sales' table.
```

FETCHING DATA FROM DATABASE

In [20]:
```
query="SHOW TABLES"
cursor.execute(query)
cursor.fetchall()
```

Out[20]:
```
[('sales',)]
```

In [111…
```
query="SELECT * FROM sales LIMIT 10 "
cursor.execute(query)
result = cursor.fetchall()

for x in result:
    print(x)
```

```
('101-17-6199', 'A', 'Yangon', 'Normal', 'Male', 'Food and beverages', Decimal('45.79'), 7, Decimal('16.0265'), Decimal('336.5565'), datetime.date(2019, 3, 13), datetime.timedelta(seconds=71040), 'Credit card', Decimal('320.53'), Decimal('4.761904762'), Decimal('16.0265'), Decimal('7.0'), 'Evening', 'Wednesday', 'March')
('101-81-4070', 'C', 'Naypyitaw', 'Member', 'Female', 'Health and beauty', Decimal('62.82'), 2, Decimal('6.2820'), Decimal('131.9220'), datetime.date(2019, 1, 17), datetime.timedelta(seconds=45360), 'Ewallet', Decimal('125.64'), Decimal('4.761904762'), Decimal('6.2820'), Decimal('4.9'), 'Afternoon', 'Thursday', 'January')
('102-06-2002', 'C', 'Naypyitaw', 'Member', 'Male', 'Sports and travel', Decimal('25.25'), 5, Decimal('6.3125'), Decimal('132.5625'), datetime.date(2019, 3, 20), datetime.timedelta(seconds=64320), 'Cash', Decimal('126.25'), Decimal('4.761904762'), Decimal('6.3125'), Decimal('6.1'), 'Evening', 'Wednesday', 'March')
('102-77-2261', 'C', 'Naypyitaw', 'Member', 'Male', 'Health and beauty', Decimal('65.31'), 7, Decimal('22.8585'), Decimal('480.0285'), datetime.date(2019, 3, 5), datetime.timedelta(seconds=64920), 'Credit card', Decimal('457.17'), Decimal('4.761904762'), Decimal('22.8585'), Decimal('4.2'), 'Evening', 'Tuesday', 'March')
('105-10-6182', 'A', 'Yangon', 'Member', 'Male', 'Fashion accessories', Decimal('21.48'), 2, Decimal('2.1480'), Decimal('45.1080'), datetime.date(2019, 2, 27), datetime.timedelta(seconds=44520), 'Ewallet', Decimal('42.96'), Decimal('4.761904762'), Decimal('2.1480'), Decimal('6.6'), 'Afternoon', 'Wednesday', 'February')
('105-31-1824', 'A', 'Yangon', 'Member', 'Male', 'Sports and travel', Decimal('69.52'), 7, Decimal('24.3320'), Decimal('510.9720'), datetime.date(2019, 2, 1), datetime.timedelta(seconds=54600), 'Credit card', Decimal('486.64'), Decimal('4.761904762'), Decimal('24.3320'), Decimal('8.5'), 'Afternoon', 'Friday', 'February')
('106-35-6779', 'A', 'Yangon', 'Member', 'Male', 'Home and lifestyle', Decimal('44.34'), 2, Decimal('4.4340'), Decimal('93.1140'), datetime.date(2019, 3, 27), datetime.timedelta(seconds=41160), 'Cash', Decimal('88.68'), Decimal('4.761904762'), Decimal('4.4340'), Decimal('5.8'), 'Morning', 'Wednesday', 'March')
('109-28-2512', 'B', 'Mandalay', 'Member', 'Female', 'Fashion accessories', Decimal('97.61'), 6, Decimal('29.2830'), Decimal('614.9430'), datetime.date(2019, 1, 7), datetime.timedelta(seconds=54060), 'Ewallet', Decimal('585.66'), Decimal('4.761904762'), Decimal('29.2830'), Decimal('9.9'), 'Afternoon', 'Monday', 'January')
('109-86-4363', 'B', 'Mandalay', 'Member', 'Female', 'Sports and travel', Decimal('60.08'), 7, Decimal('21.0280'), Decimal('441.5880'), datetime.date(2019, 2, 14), datetime.timedelta(seconds=41760), 'Credit card', Decimal('420.56'), Decimal('4.761904762'), Decimal('21.0280'), Decimal('4.5'), 'Morning', 'Thursday', 'February')
('110-05-6330', 'C', 'Naypyitaw', 'Normal', 'Female', 'Food and beverages', Decimal('39.43'), 6, Decimal('11.8290'), Decimal('248.4090'), datetime.date(2019, 3, 25), datetime.timedelta(seconds=73080), 'Credit card', Decimal('236.58'), Decimal('4.761904762'), Decimal('11.8290'), Decimal('9.4'), 'Evening', 'Monday', 'March')
```

CONVERTING THE TABLE INTO PANDAS DATAFRAME

In [3]:
```
import mysql.connector
import pandas as pd
```

```
# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}

# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)

# Define the SQL query to select all data from the 'sales' table
select_query = "SELECT * FROM sales"

# Use pandas to read the SQL query result into a DataFrame
sales_df = pd.read_sql_query(select_query, connection)

sales_df
# Now, 'sales_df' contains the 'sales' table data as a DataFrame
```

```
/var/folders/rs/w1cppzrd0tg10lz07q7q90qm0000gn/T/ipykernel_1351/836903035.py:19: UserWar
ning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  sales_df = pd.read_sql_query(select_query, connection)
```

Out[3]:

	invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	tax_5_pct
0	101-17-6199	A	Yangon	Normal	Male	Food and beverages	45.79	7	16.0265
1	101-81-4070	C	Naypyitaw	Member	Female	Health and beauty	62.82	2	6.2820
2	102-06-2002	C	Naypyitaw	Member	Male	Sports and travel	25.25	5	6.3125
3	102-77-2261	C	Naypyitaw	Member	Male	Health and beauty	65.31	7	22.8585
4	105-10-6182	A	Yangon	Member	Male	Fashion accessories	21.48	2	2.1480
...	...	...	...	...	...	...	...	...	...
995	894-41-5205	C	Naypyitaw	Normal	Female	Food and beverages	43.18	8	17.2720
996	895-03-6665	B	Mandalay	Normal	Female	Fashion accessories	36.51	9	16.4295
997	895-66-0685	B	Mandalay	Member	Male	Food and beverages	18.08	3	2.7120
998	896-34-0956	A	Yangon	Normal	Male	Fashion accessories	21.32	1	1.0660
999	898-04-2717	A	Yangon	Normal	Male	Fashion accessories	76.40	9	34.3800

1000 rows × 20 columns

In [26]:
```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice_id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
 3   customer_type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product_line             1000 non-null   object
 6   unit_price               1000 non-null   float64
 7   quantity                 1000 non-null   int64
 8   tax_5_pct                1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   object
 11  time                     1000 non-null   timedelta64[ns]
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross_margin_percentage  1000 non-null   float64
 15  gross_income             1000 non-null   float64
 16  rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(8), timedelta64[ns](1)
memory usage: 132.9+ KB
```

FEATURE ENGINEERING

1.Add the time_of_day column

In [114…

```
import csv
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
SELECT
        time,
        (CASE
                WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
    END) AS time_of_day
FROM sales LIMIT 5
"""
cursor.execute(query)
cursor.fetchall()
```

Out[114]:

```
[(datetime.timedelta(seconds=71040), 'Evening'),
 (datetime.timedelta(seconds=45360), 'Afternoon'),
 (datetime.timedelta(seconds=64320), 'Evening'),
```

```
    (datetime.timedelta(seconds=64920), 'Evening'),
    (datetime.timedelta(seconds=44520), 'Afternoon')]
```

import mysql.connector # MySQL database connection parameters db_config = { 'host': 'localhost', 'user': 'root', 'password': 'Kanha@8144', 'database': 'walmart_sales', } # Connect to the MySQL database connection = mysql.connector.connect(**db_config) cursor = connection.cursor() query=""" ALTER TABLE sales ADD COLUMN time_of_day VARCHAR(20) """ cursor.execute(query) connection.commit()

In [8]:
```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
UPDATE sales
SET time_of_day = (
        CASE
                WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
    END
)
"""
cursor.execute(query)
connection.commit()
```

In [11]:
```
import mysql.connector
import pandas as pd

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}

# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)

# Define the SQL query to select all data from the 'sales' table
select_query = "SELECT * FROM sales"

# Use pandas to read the SQL query result into a DataFrame
sales_df = pd.read_sql_query(select_query, connection)

sales_df
# Now, 'sales_df' contains the 'sales' table data as a DataFrame
```

```
/var/folders/rs/w1cppzrd0tg10lz07q7q90qm0000gn/T/ipykernel_1763/836903035.py:19: UserWar
ning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  sales_df = pd.read_sql_query(select_query, connection)
```

Out[11]:

	invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	tax_5_pct
0	101-17-6199	A	Yangon	Normal	Male	Food and beverages	45.79	7	16.0265
1	101-81-4070	C	Naypyitaw	Member	Female	Health and beauty	62.82	2	6.2820
2	102-06-2002	C	Naypyitaw	Member	Male	Sports and travel	25.25	5	6.3125
3	102-77-2261	C	Naypyitaw	Member	Male	Health and beauty	65.31	7	22.8585
4	105-10-6182	A	Yangon	Member	Male	Fashion accessories	21.48	2	2.1480
...	...	...	...	...	...	...	...	...	...
995	894-41-5205	C	Naypyitaw	Normal	Female	Food and beverages	43.18	8	17.2720
996	895-03-6665	B	Mandalay	Normal	Female	Fashion accessories	36.51	9	16.4295
997	895-66-0685	B	Mandalay	Member	Male	Food and beverages	18.08	3	2.7120
998	896-34-0956	A	Yangon	Normal	Male	Fashion accessories	21.32	1	1.0660
999	898-04-2717	A	Yangon	Normal	Male	Fashion accessories	76.40	9	34.3800

1000 rows × 18 columns

2.Add day_name column

In [108…

```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
SELECT
        date,
        DAYNAME(date)
FROM sales LIMIT 10
"""
cursor.execute(query)

cursor.fetchall()
```

Out[108]:
```
[(datetime.date(2019, 3, 13), 'Wednesday'),
 (datetime.date(2019, 1, 17), 'Thursday'),
 (datetime.date(2019, 3, 20), 'Wednesday'),
 (datetime.date(2019, 3, 5), 'Tuesday'),
 (datetime.date(2019, 2, 27), 'Wednesday'),
 (datetime.date(2019, 2, 1), 'Friday'),
 (datetime.date(2019, 3, 27), 'Wednesday'),
 (datetime.date(2019, 1, 7), 'Monday'),
 (datetime.date(2019, 2, 14), 'Thursday'),
 (datetime.date(2019, 3, 25), 'Monday')]
```

In [ ]:
```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
ALTER TABLE sales ADD COLUMN day_name VARCHAR(10)
"""
cursor.execute(query)
```

In [4]:
```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
UPDATE sales
SET day_name = DAYNAME(date)
"""
cursor.execute(query)
connection.commit()
```

In [5]:
```
import mysql.connector
import pandas as pd

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
```

```
}

# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)

# Define the SQL query to select all data from the 'sales' table
select_query = "SELECT * FROM sales "

# Use pandas to read the SQL query result into a DataFrame
sales_df = pd.read_sql_query(select_query, connection)

sales_df
# Now, 'sales_df' contains the 'sales' table data as a DataFrame
```

```
/var/folders/rs/w1cppzrd0tg10lz07q7q90qm0000gn/T/ipykernel_1994/836903035.py:19: UserWar
ning: pandas only supports SQLAlchemy connectable (engine/connection) or database string
URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider u
sing SQLAlchemy.
  sales_df = pd.read_sql_query(select_query, connection)
```

Out[5]:

	invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	tax_5_pct
0	101-17-6199	A	Yangon	Normal	Male	Food and beverages	45.79	7	16.0265
1	101-81-4070	C	Naypyitaw	Member	Female	Health and beauty	62.82	2	6.2820
2	102-06-2002	C	Naypyitaw	Member	Male	Sports and travel	25.25	5	6.3125
3	102-77-2261	C	Naypyitaw	Member	Male	Health and beauty	65.31	7	22.8585
4	105-10-6182	A	Yangon	Member	Male	Fashion accessories	21.48	2	2.1480
...	...	...	...	...	...	...	...	...	...
995	894-41-5205	C	Naypyitaw	Normal	Female	Food and beverages	43.18	8	17.2720
996	895-03-6665	B	Mandalay	Normal	Female	Fashion accessories	36.51	9	16.4295
997	895-66-0685	B	Mandalay	Member	Male	Food and beverages	18.08	3	2.7120
998	896-34-0956	A	Yangon	Normal	Male	Fashion accessories	21.32	1	1.0660
999	898-04-2717	A	Yangon	Normal	Male	Fashion accessories	76.40	9	34.3800

1000 rows × 19 columns

3.Add month_name column

In [109…

```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
```

```
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}



# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query="""
SELECT
        date,
        MONTHNAME(date)
FROM sales LIMIT 10
"""
cursor.execute(query)
cursor.fetchall()
```

Out[109]:
```
[(datetime.date(2019, 3, 13), 'March'),
 (datetime.date(2019, 1, 17), 'January'),
 (datetime.date(2019, 3, 20), 'March'),
 (datetime.date(2019, 3, 5), 'March'),
 (datetime.date(2019, 2, 27), 'February'),
 (datetime.date(2019, 2, 1), 'February'),
 (datetime.date(2019, 3, 27), 'March'),
 (datetime.date(2019, 1, 7), 'January'),
 (datetime.date(2019, 2, 14), 'February'),
 (datetime.date(2019, 3, 25), 'March')]
```

In [4]:
```
import mysql.connector

# MySQL database connection parameters
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': 'Kanha@8144',
    'database': 'walmart_sales',
}

# Connect to the MySQL database
connection = mysql.connector.connect(**db_config)
cursor = connection.cursor()

query = """
ALTER TABLE sales
ADD COLUMN month_name VARCHAR(10)
"""

cursor.execute(query)
```

In [5]:
```
query="""
UPDATE sales
SET month_name = MONTHNAME(date)
"""

cursor.execute(query)
connection.commit()
```

In [7]:
```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 20 columns):
```

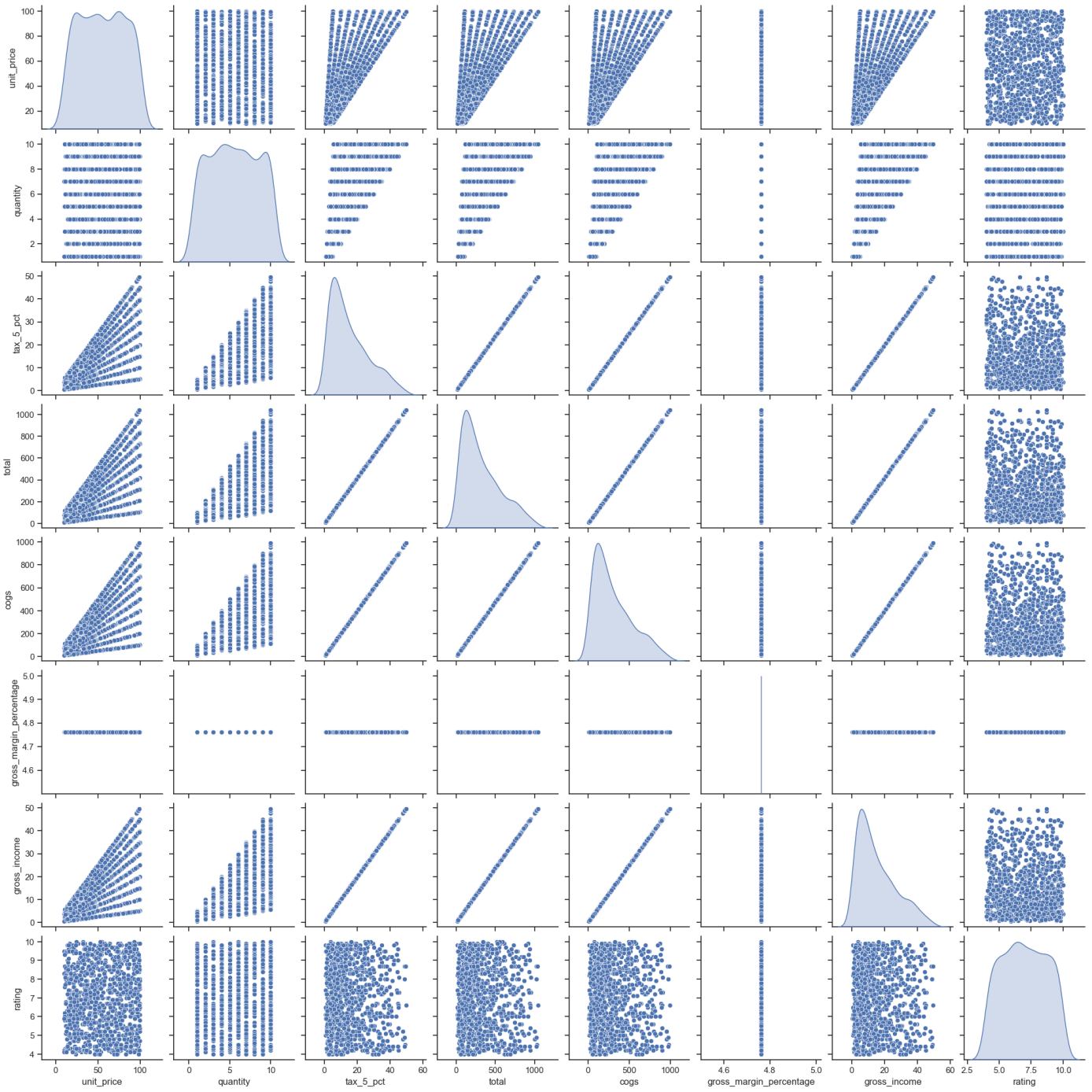
```
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice_id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
 3   customer_type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product_line             1000 non-null   object
 6   unit_price               1000 non-null   float64
 7   quantity                 1000 non-null   int64
 8   tax_5_pct                1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   object
 11  time                     1000 non-null   timedelta64[ns]
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross_margin_percentage  1000 non-null   float64
 15  gross_income             1000 non-null   float64
 16  rating                   1000 non-null   float64
 17  time_of_day              1000 non-null   object
 18  day_name                 1000 non-null   object
 19  month_name               1000 non-null   object
dtypes: float64(7), int64(1), object(11), timedelta64[ns](1)
memory usage: 156.4+ KB
```

MULTIVARIATE ANALYSIS OF COMPLETE DATASET

In [13]:

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Select columns you want to include in the pair plot
selected_columns = ['unit_price', 'quantity', 'tax_5_pct', 'total', 'cogs', 'gross_margi

# Create a pair plot
sns.set(style="ticks")
sns.pairplot(sales_df[selected_columns], diag_kind='kde', kind='scatter')
plt.show()
```

```
/Users/tariniprasaddas/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:118: U
serWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



GENERIC QUESTIONS

1. How many unique cities does the data have?

In [10]:
```
sales_df['city'].unique()
```

Out[10]:
```
array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)
```

In [11]:
```
sales_df['city'].value_counts()
```

Out[11]:
```
city
Yangon       340
Mandalay     332
Naypyitaw    328
Name: count, dtype: int64
```

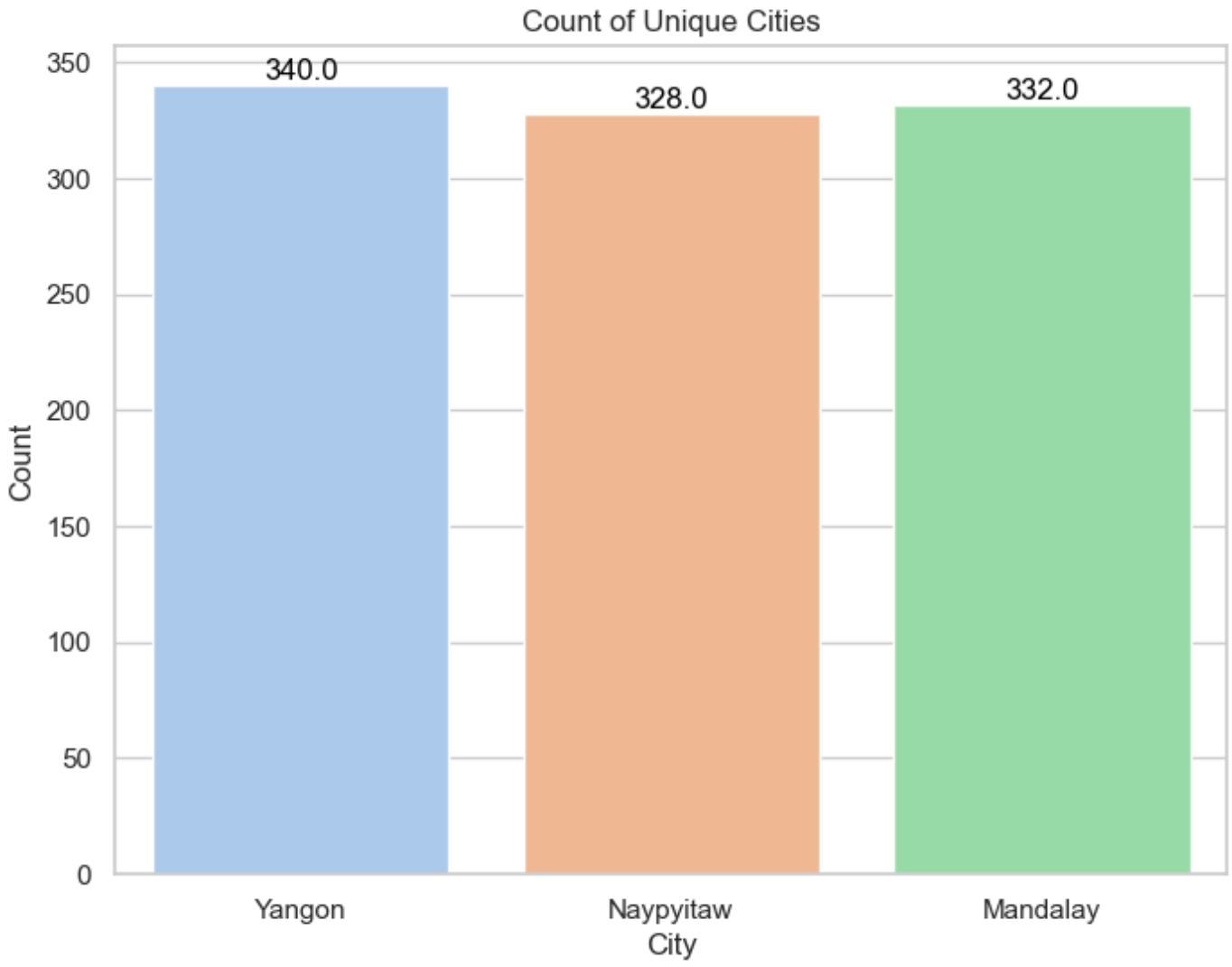
In [8]:
```
unique_cities = sales_df['city'].nunique()
```

```
print(f"The data has {unique_cities} unique cities.")
```

```
The data has 3 unique cities.
```

In [108…

```
import seaborn as sns
import matplotlib.pyplot as plt



plt.figure(figsize=(8, 6))
ax = sns.countplot(data=sales_df, x='city', palette='pastel')
plt.title("Count of Unique Cities")
plt.xlabel("City")
plt.ylabel("Count")

# Add count labels above each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()), h

plt.show()
```



2.In which city is each branch?

In [13]:

```
sales_df['branch'].value_counts()
```

Out[13]:

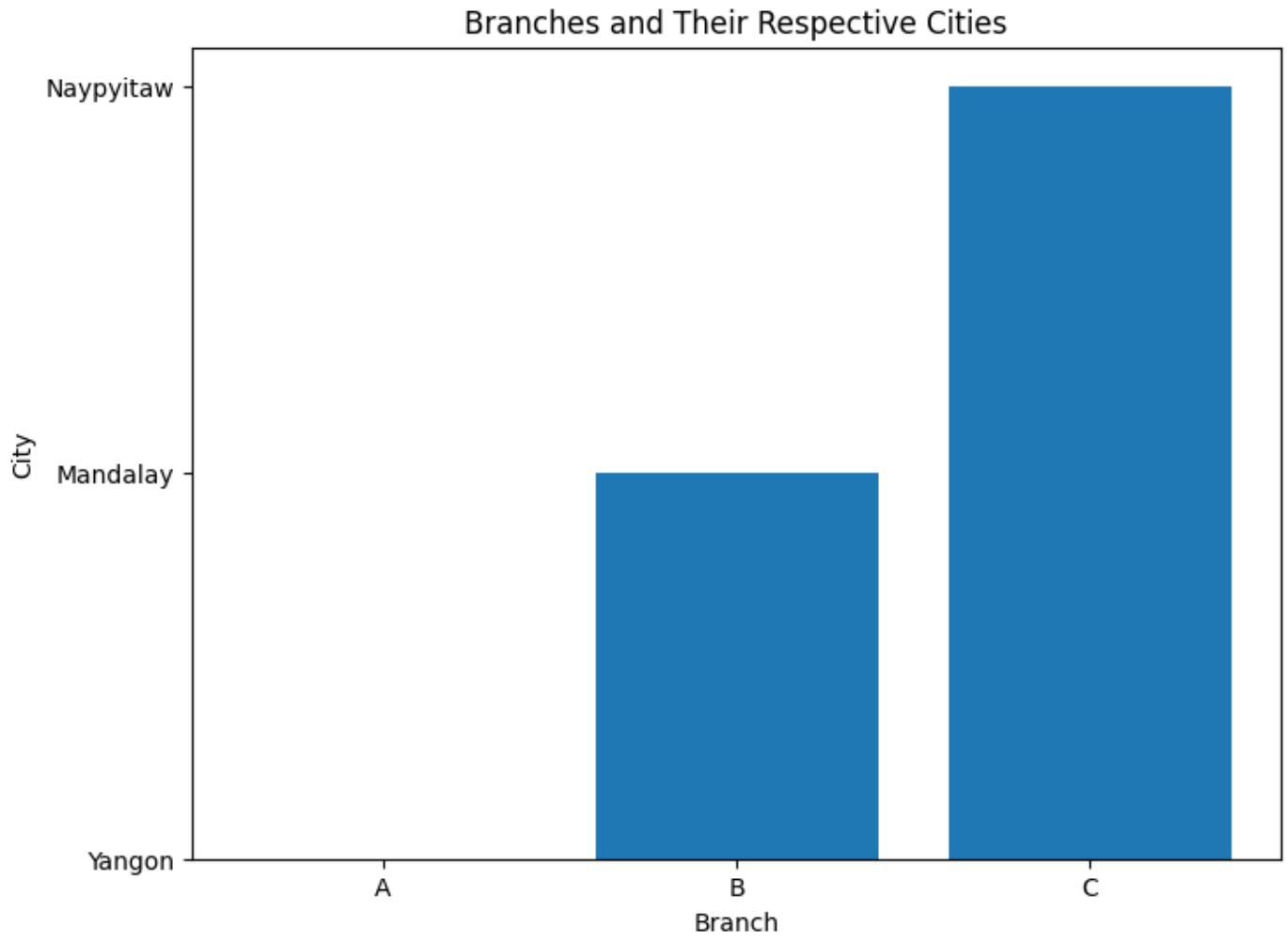
```
branch
A    340
B    332
```

```
C    328
Name: count, dtype: int64
```

In [16]:
```python
# Group the data by 'branch' and get the first value of 'city' within each group
branch_city_mapping = sales_df.groupby('branch')['city'].first().reset_index()

print(branch_city_mapping)
```

```
  branch       city
0      A     Yangon
1      B   Mandalay
2      C  Naypyitaw
```

In [17]:
```python
import matplotlib.pyplot as plt



# Create a bar plot
plt.figure(figsize=(8, 6))
plt.bar(branch_city_mapping['branch'], branch_city_mapping['city'])
plt.title("Branches and Their Respective Cities")
plt.xlabel("Branch")
plt.ylabel("City")
plt.show()
```



PRODUCT BASED QUESTIONS

1. How many unique product lines does the data have?

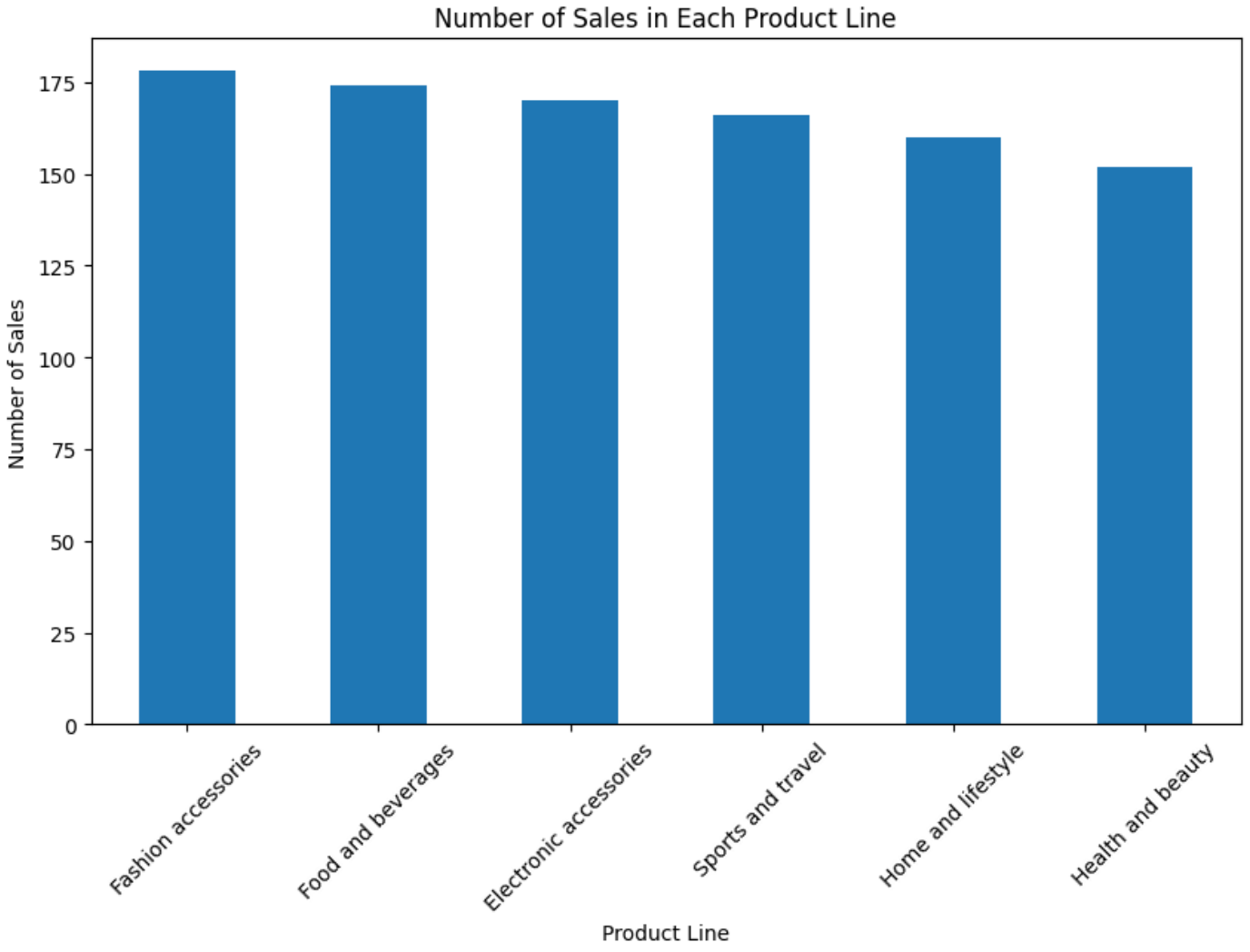
In [19]:
```
sales_df['product_line'].unique()
```

Out[19]:
```
array(['Food and beverages', 'Health and beauty', 'Sports and travel',
       'Fashion accessories', 'Home and lifestyle',
       'Electronic accessories'], dtype=object)
```

In [20]:
```
sales_df.product_line.value_counts()
```

Out[20]:
```
product_line
Fashion accessories       178
Food and beverages        174
Electronic accessories    170
Sports and travel         166
Home and lifestyle        160
Health and beauty         152
Name: count, dtype: int64
```

In [21]:
```
import matplotlib.pyplot as plt

# Group the data by 'product_line' and count the number of occurrences in each product l
product_line_counts = sales_df['product_line'].value_counts()

# Create a bar plot
plt.figure(figsize=(10, 6))
product_line_counts.plot(kind='bar')
plt.title("Number of Sales in Each Product Line")
plt.xlabel("Product Line")
plt.ylabel("Number of Sales")
plt.xticks(rotation=45)
plt.show()
```



2. What is the most common payment method?

In [23]:
```
sales_df['payment'].value_counts()
```

Out[23]:
```
payment
Ewallet        345
Cash           344
Credit card    311
Name: count, dtype: int64
```

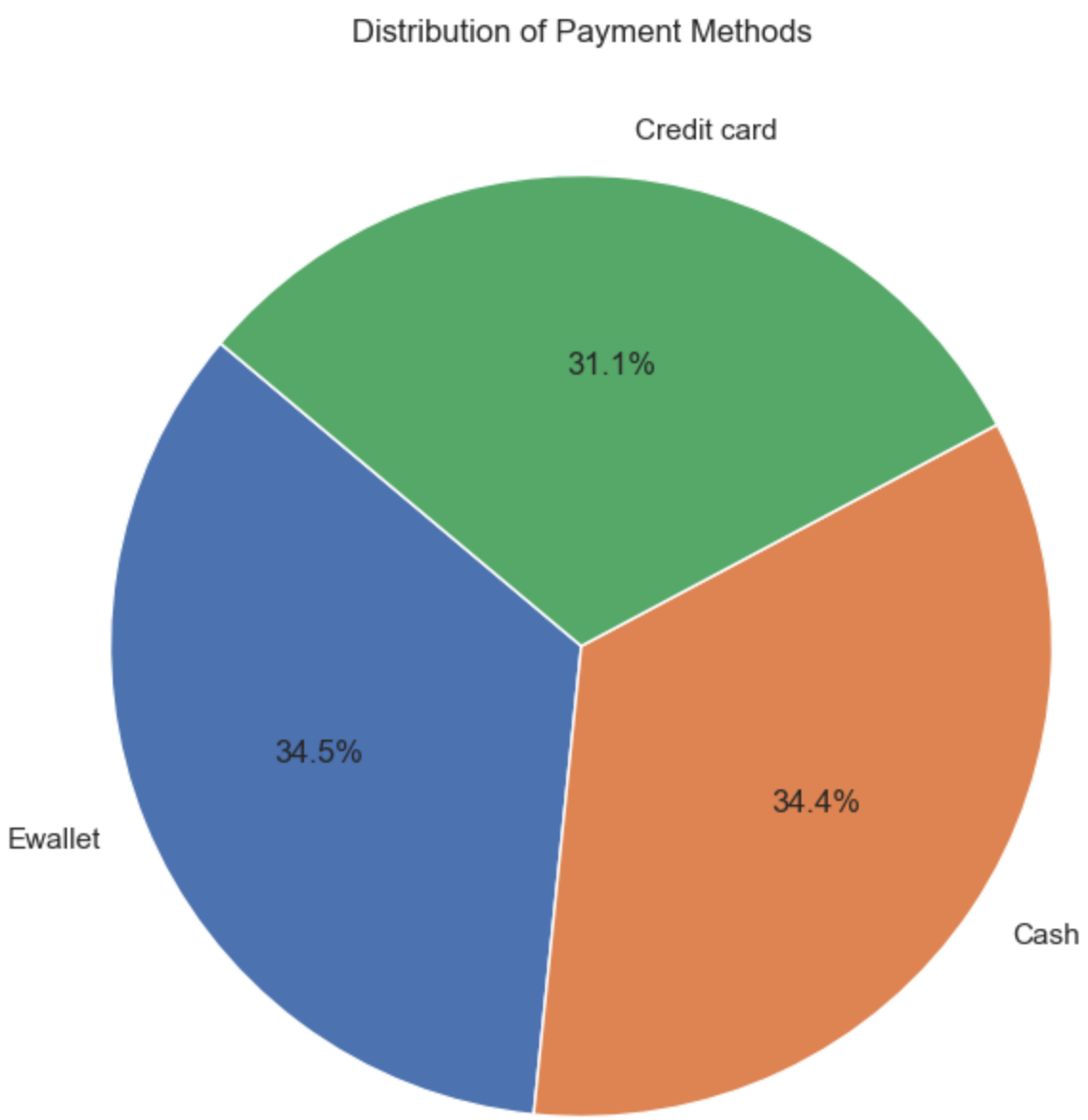
In [27]:
```
sales_df['payment'].mode()
```

Out[27]:
```
0    Ewallet
Name: payment, dtype: object
```

In [22]:
```
# Calculate the most common payment method
most_common_payment_method = sales_df['payment'].mode()[0]

print(f"The most common payment method is: {most_common_payment_method}")
```

```
The most common payment method is: Ewallet
```

In [29]:
```
import seaborn as sns
import matplotlib.pyplot as plt


# Step 2: Create a pie chart to visualize payment methods
payment_counts = sales_df['payment'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(payment_counts, labels=payment_counts.index, autopct='%1.1f%%', startangle=140)
plt.title("Distribution of Payment Methods")
plt.show()
```



3. What is the most selling product line?

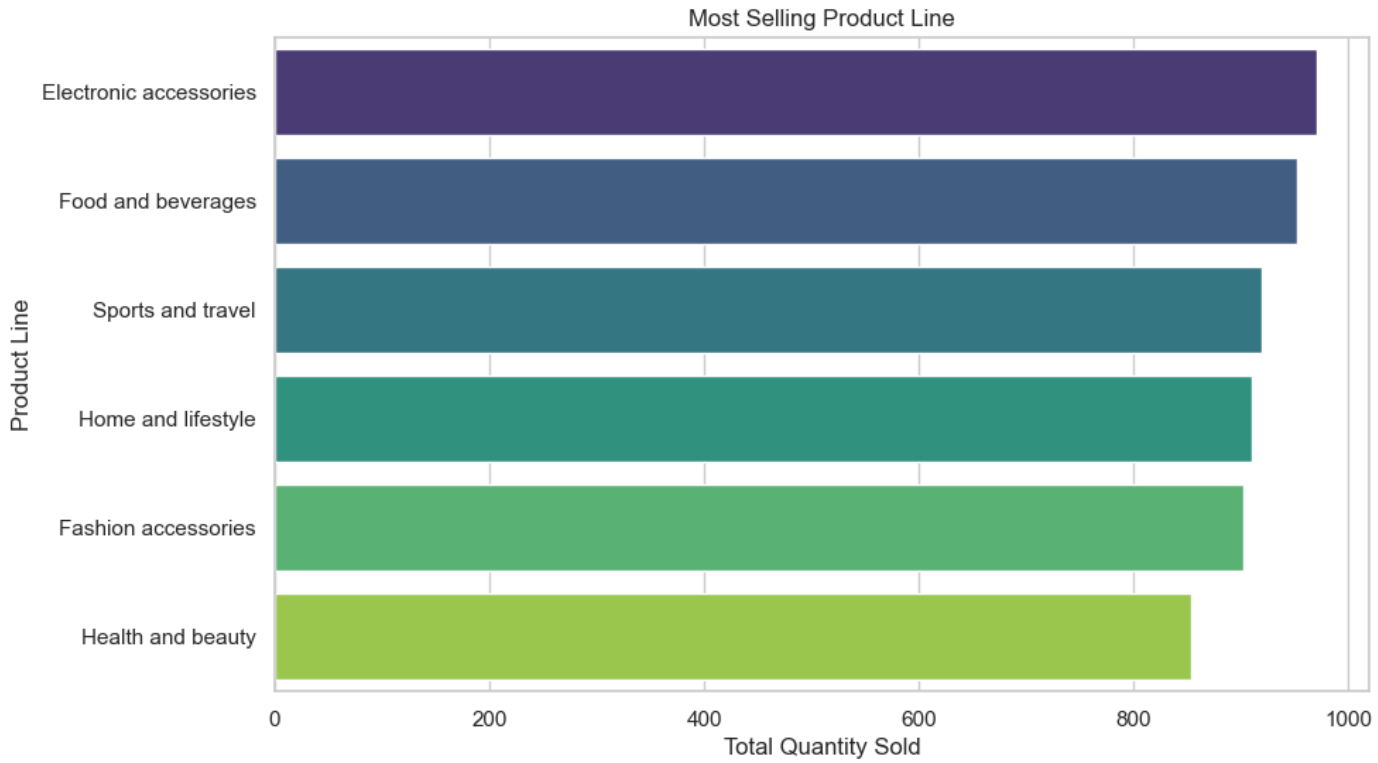
In [31]:
```
sales_df.groupby('product_line')['quantity'].sum()
```

Out[31]:
```
product_line
Electronic accessories    971
Fashion accessories       902
Food and beverages        952
Health and beauty         854
Home and lifestyle        911
Sports and travel         920
Name: quantity, dtype: int64
```

In [30]:
```
# Group the data by 'product_line' and sum the quantities sold in each product line
product_line_sales = sales_df.groupby('product_line')['quantity'].sum()

# Find the product line with the highest total quantity sold
most_selling_product_line = product_line_sales.idxmax()

print(f"The most selling product line is: {most_selling_product_line}")
```

```
The most selling product line is: Electronic accessories
```

In [32]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Group the data by 'product_line' and sum the quantities sold in each product line
product_line_sales = sales_df.groupby('product_line')['quantity'].sum().reset_index()

# Sort the product lines by total quantity sold in descending order
product_line_sales = product_line_sales.sort_values(by='quantity', ascending=False)

# Create a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x='quantity', y='product_line', data=product_line_sales, palette='viridis')
plt.title("Most Selling Product Line")
plt.xlabel("Total Quantity Sold")
plt.ylabel("Product Line")
plt.show()
```



4. What is the total revenue by month?

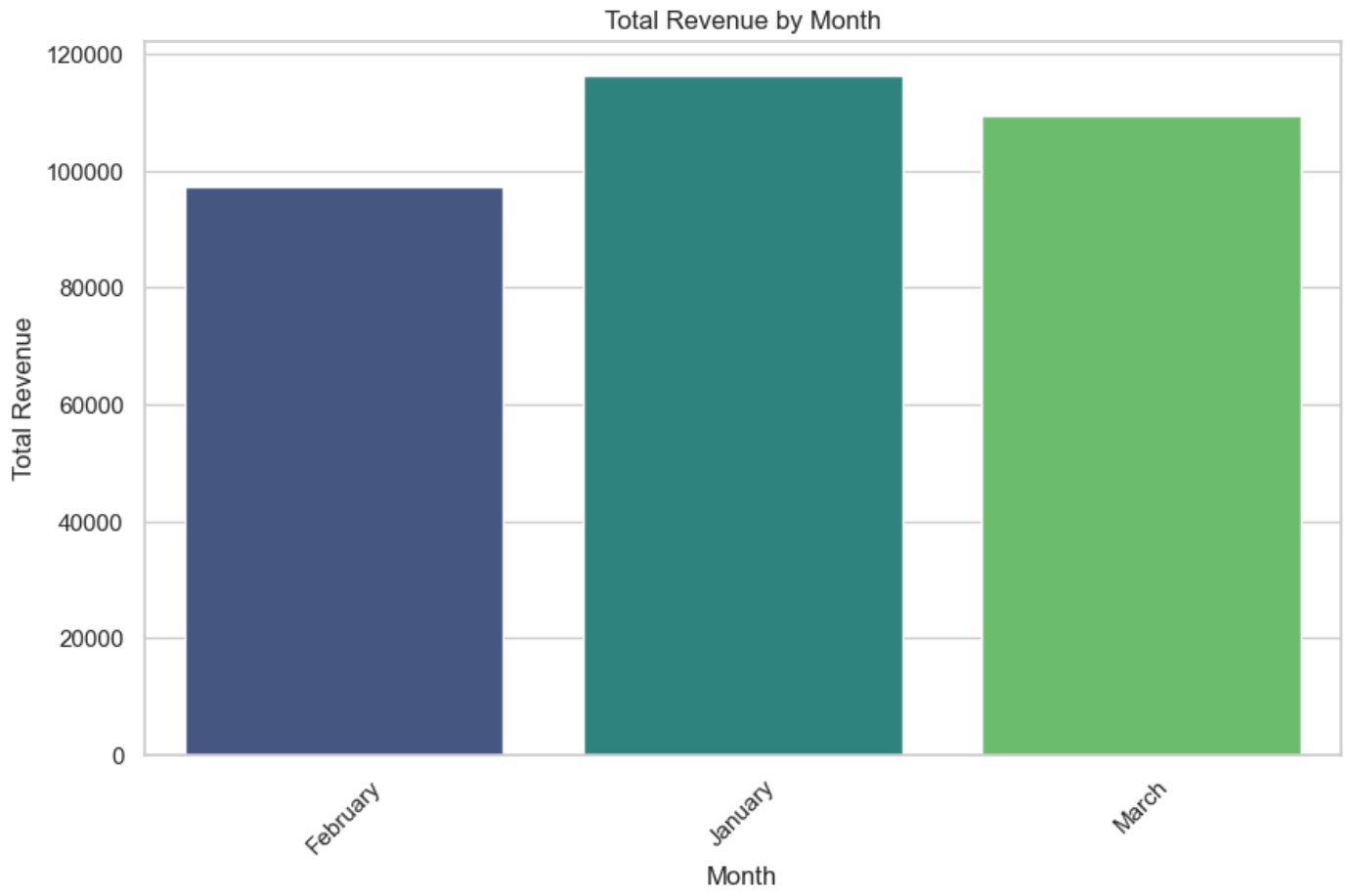
In [37]:
```
monthly_revenue=sales_df.groupby('month_name')['total'].sum().reset_index()
monthly_revenue
```

Out[37]:

	month_name	total
0	February	97219.374
1	January	116291.868
2	March	109455.507

In [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt


# Create a bar plot using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x='month_name', y='total', data=monthly_revenue, palette='viridis')
plt.title("Total Revenue by Month")
```

```
plt.xlabel("Month")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)
plt.show()
```



5. What month had the largest COGS?

In [44]:
```
monthly_cogs = sales_df.groupby('month_name')['cogs'].sum()

monthly_cogs
```

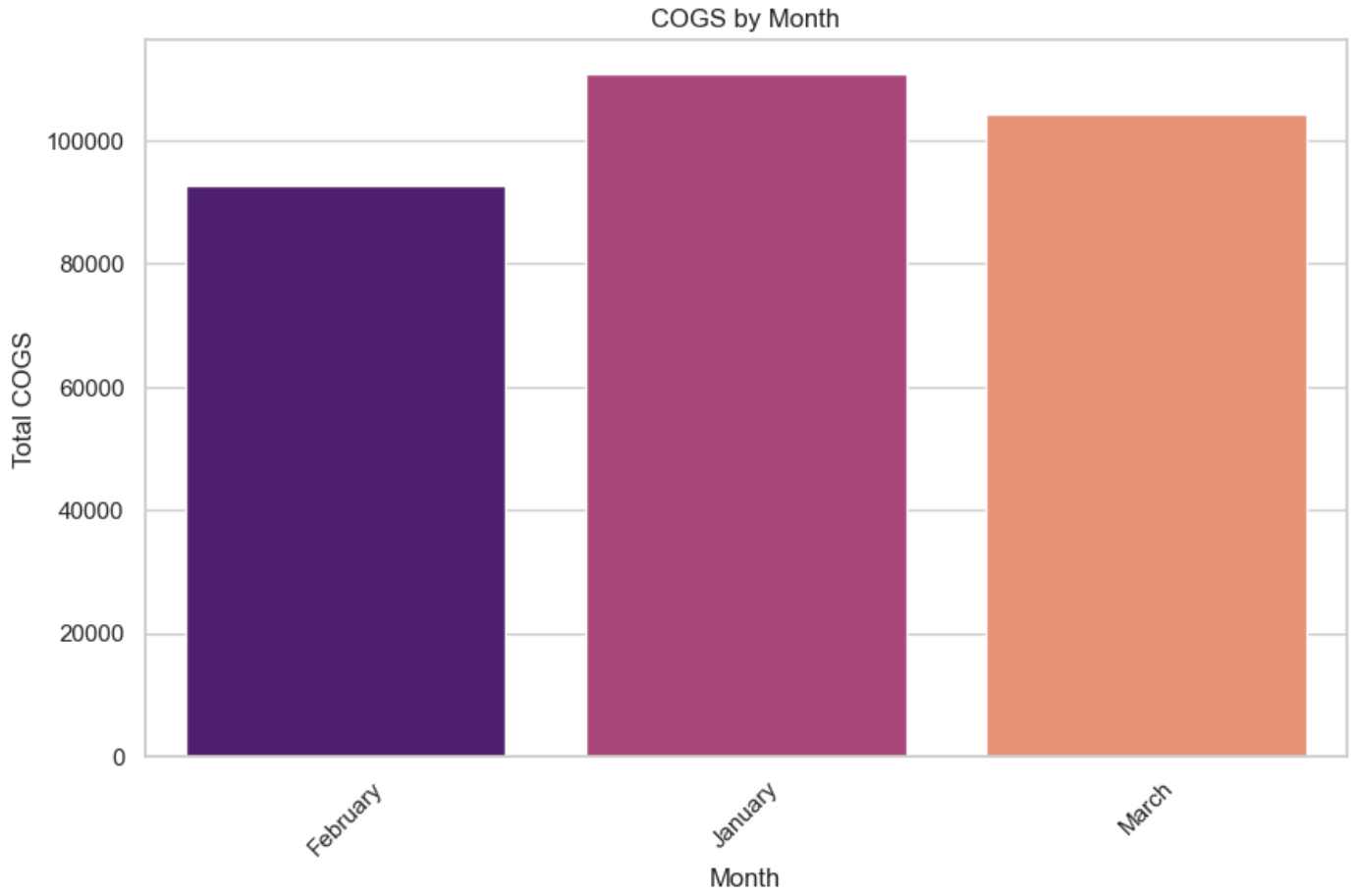
Out[44]:
```
month_name
February     92589.88
January     110754.16
March       104243.34
Name: cogs, dtype: float64
```

In [47]:
```
# Find the month with the largest COGS
largest_cogs_month = monthly_cogs.idxmax()

print(f"The month with the largest COGS is: {largest_cogs_month}")
```

```
The month with the largest COGS is: January
```

In [48]:
```
import seaborn as sns
import matplotlib.pyplot as plt



# Create a bar plot using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=monthly_cogs.index, y=monthly_cogs.values, palette='magma')
plt.title("COGS by Month")
plt.xlabel("Month")
```

```
plt.ylabel("Total COGS")
plt.xticks(rotation=45)
plt.show()
```



6. What product line had the largest revenue?

In [53]:
```
# Group the data by 'product_line' and sum the total revenue in each product line
product_line_revenue = sales_df.groupby('product_line')['total'].sum()
product_line_revenue
```

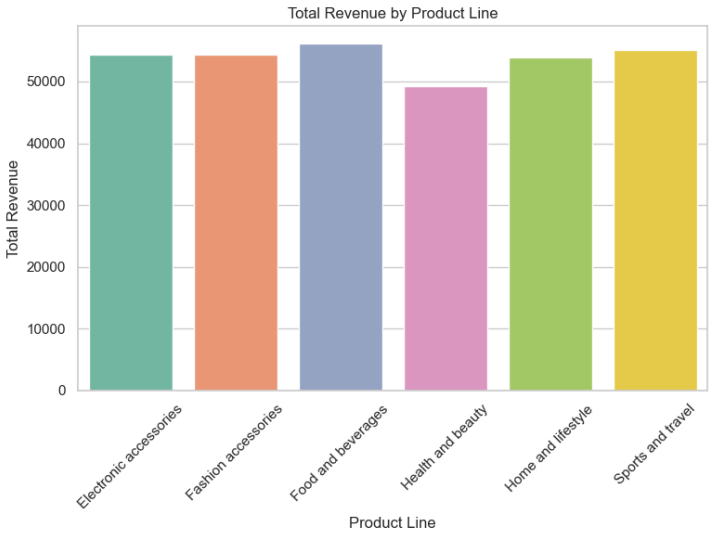
Out[53]:
```
product_line
Electronic accessories    54337.5315
Fashion accessories       54305.8950
Food and beverages        56144.8440
Health and beauty         49193.7390
Home and lifestyle        53861.9130
Sports and travel         55122.8265
Name: total, dtype: float64
```

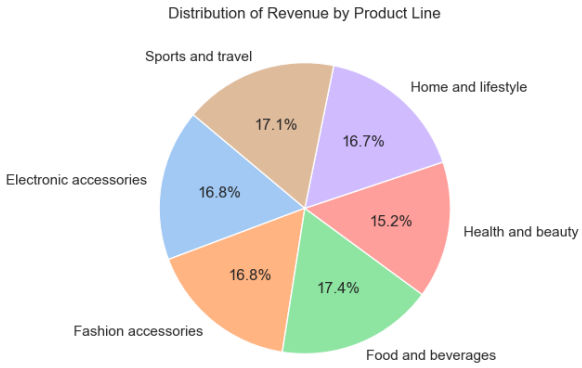
In [54]:
```
# Find the product line with the largest revenue
largest_revenue_product_line = product_line_revenue.idxmax()

print(f"The product line with the largest revenue is: {largest_revenue_product_line}")
```

```
The product line with the largest revenue is: Food and beverages
```

In [56]:
```
import seaborn as sns
import matplotlib.pyplot as plt



# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
```

```
# Plot the bar plot on the first subplot
sns.barplot(x=product_line_revenue.index, y=product_line_revenue.values, palette='Set2',
axes[0].set_title("Total Revenue by Product Line")
axes[0].set_xlabel("Product Line")
axes[0].set_ylabel("Total Revenue")
axes[0].tick_params(axis='x', rotation=45)

# Plot the pie chart on the second subplot
colors = sns.color_palette('pastel')
axes[1].pie(product_line_revenue, labels=product_line_revenue.index, autopct='%1.1f%%',
axes[1].set_title("Distribution of Revenue by Product Line")

# Adjust the layout to prevent overlapping
plt.tight_layout()

# Show the combined plot
plt.show()
```





7. What is the city with the largest revenue?

In [58]:
```
# Group the data by 'city' and sum the total revenue in each city
city_revenue = sales_df.groupby('city')['total'].sum()
city_revenue
```

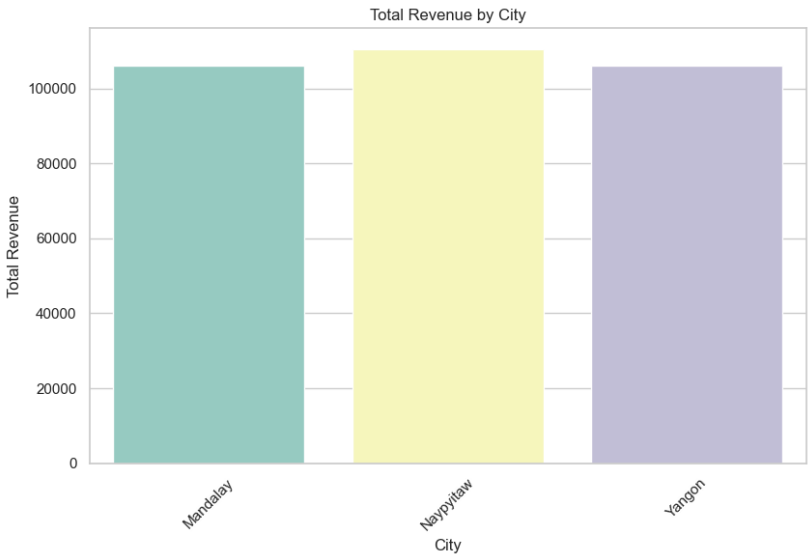
Out[58]:
```
city
Mandalay      106197.6720
Naypyitaw     110568.7065
Yangon        106200.3705
Name: total, dtype: float64
```

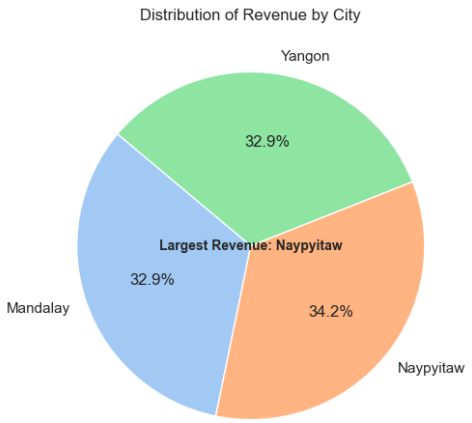
In [57]:
```
# Find the city with the largest revenue
largest_revenue_city = city_revenue.idxmax()

print(f"The city with the largest revenue is: {largest_revenue_city}")
```

```
The city with the largest revenue is: Naypyitaw
```

In [59]:
```
import seaborn as sns
import matplotlib.pyplot as plt



# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot the bar plot on the first subplot
sns.barplot(x=city_revenue.index, y=city_revenue.values, palette='Set3', ax=axes[0])
```

```
axes[0].set_title("Total Revenue by City")
axes[0].set_xlabel("City")
axes[0].set_ylabel("Total Revenue")
axes[0].tick_params(axis='x', rotation=45)

# Plot the pie chart on the second subplot
colors = sns.color_palette('pastel')
axes[1].pie(city_revenue, labels=city_revenue.index, autopct='%1.1f%%', startangle=140,
axes[1].set_title("Distribution of Revenue by City")

# Add a text annotation for the city with the largest revenue
axes[1].text(0, 0, f"Largest Revenue: {largest_revenue_city}", horizontalalignment='cent

# Adjust the layout to prevent overlapping
plt.tight_layout()

# Show the combined plot
plt.show()
```





8. What product line had the largest VAT?

In [61]:
```
# Group the data by 'product_line' and sum the VAT in each product line
product_line_vat = sales_df.groupby('product_line')['tax_5_pct'].sum()
product_line_vat
```

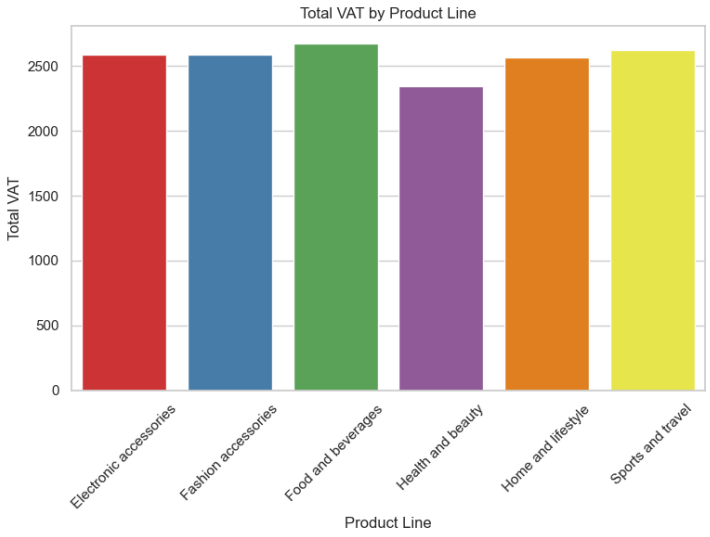
Out[61]:
```
product_line
Electronic accessories    2587.5015
Fashion accessories       2585.9950
Food and beverages        2673.5640
Health and beauty         2342.5590
Home and lifestyle        2564.8530
Sports and travel         2624.8965
Name: tax_5_pct, dtype: float64
```

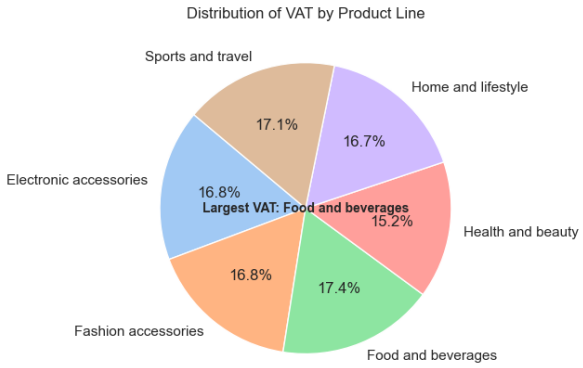
In [60]:
```
# Find the product line with the largest VAT
largest_vat_product_line = product_line_vat.idxmax()

print(f"The product line with the largest VAT is: {largest_vat_product_line}")
```

```
The product line with the largest VAT is: Food and beverages
```

In [63]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Create subplots
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Plot the bar plot on the first subplot
sns.barplot(x=product_line_vat.index, y=product_line_vat.values, palette='Set1', ax=axes
axes[0].set_title("Total VAT by Product Line")
axes[0].set_xlabel("Product Line")
axes[0].set_ylabel("Total VAT")
axes[0].tick_params(axis='x', rotation=45)

# Plot the pie chart on the second subplot
colors = sns.color_palette('pastel')
axes[1].pie(product_line_vat, labels=product_line_vat.index, autopct='%1.1f%%', startang
axes[1].set_title("Distribution of VAT by Product Line")

# Add a text annotation for the product line with the largest VAT
axes[1].text(0, 0, f"Largest VAT: {largest_vat_product_line}", horizontalalignment='cent

# Adjust the layout to prevent overlapping
plt.tight_layout()

# Show the combined plot
plt.show()
```





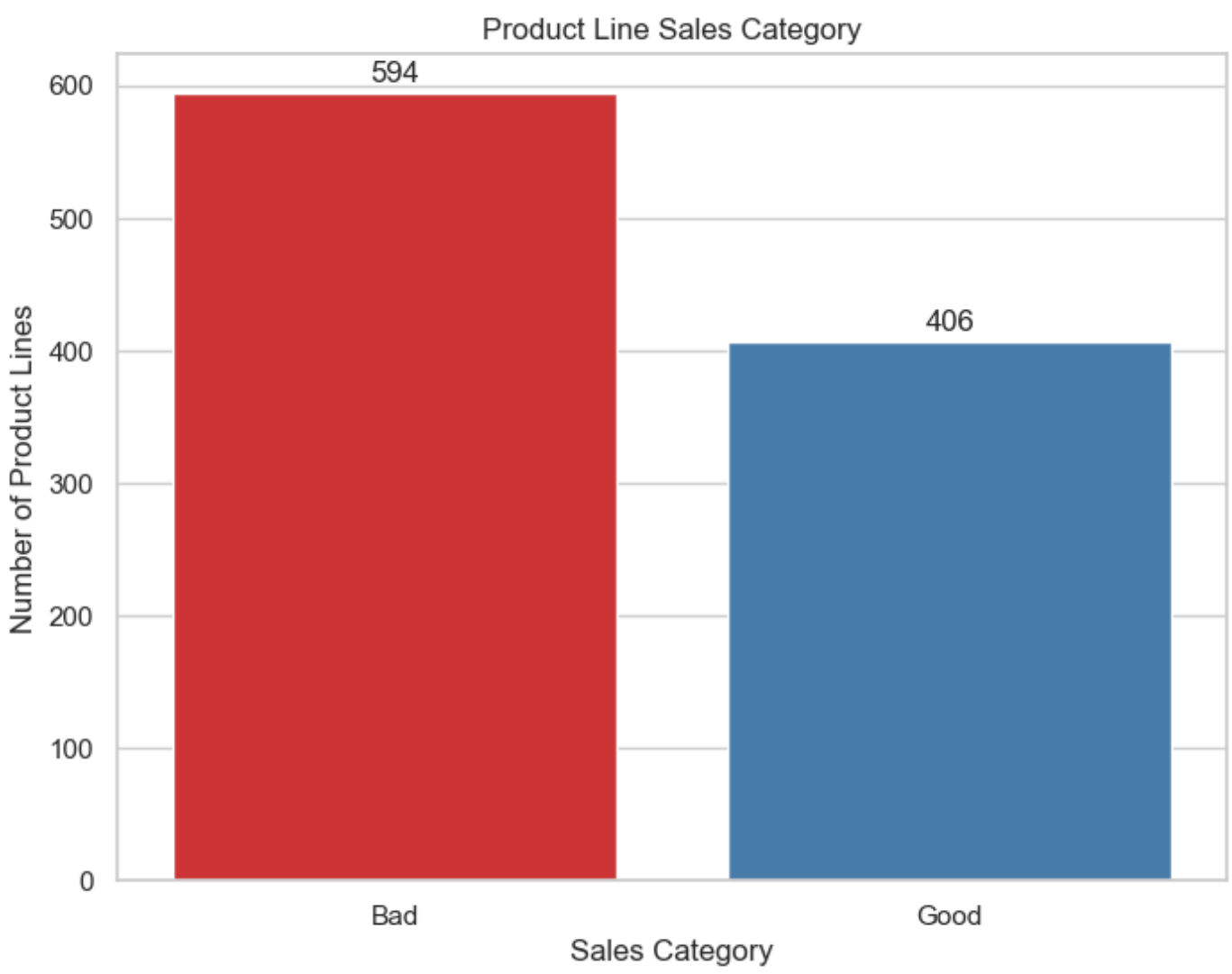
9. Fetch each product line and add a column to those product line showing "Good", "Bad". Good if its greater than average sales

In [64]:
```
import pandas as pd



# Calculate the average sales for all product lines
average_sales = sales_df.groupby('product_line')['total'].mean().reset_index()
average_sales.rename(columns={'total': 'average_sales'}, inplace=True)

# Merge the average sales with the original DataFrame
sales_df = pd.merge(sales_df, average_sales, on='product_line')

# Add a new column 'sales_category' based on comparison with average sales
sales_df['sales_category'] = sales_df.apply(lambda row: 'Good' if row['total'] > row['av

# Display the updated DataFrame
print(sales_df[['product_line', 'total', 'average_sales', 'sales_category']])
```

```
                product_line     total  average_sales sales_category
0         Food and beverages  336.5565     322.671517           Good
1         Food and beverages  248.4090     322.671517            Bad
2         Food and beverages  151.5150     322.671517            Bad
3         Food and beverages  305.5500     322.671517            Bad
4         Food and beverages  609.1680     322.671517           Good
..                       ...       ...            ...            ...
995  Electronic accessories   78.4350     319.632538            Bad
996  Electronic accessories  771.4350     319.632538           Good
997  Electronic accessories  603.8760     319.632538           Good
998  Electronic accessories  247.8735     319.632538            Bad
999  Electronic accessories  470.6730     319.632538           Good

[1000 rows x 4 columns]
```

In [65]:

```python
import seaborn as sns
import matplotlib.pyplot as plt


# Count the number of product lines in each category
category_counts = sales_df['sales_category'].value_counts()

# Create a bar plot using seaborn
plt.figure(figsize=(8, 6))
sns.barplot(x=category_counts.index, y=category_counts.values, palette='Set1')
plt.title("Product Line Sales Category")
plt.xlabel("Sales Category")
plt.ylabel("Number of Product Lines")

# Add value counts on top of the bars
for i, count in enumerate(category_counts):
    plt.text(i, count + 5, str(count), ha='center', va='bottom', fontsize=12)

plt.show()
```



10. Which branch sold more products than average product sold?

In [67]:
```
# Calculate the average product quantity sold for each branch
average_quantity = sales_df.groupby('branch')['quantity'].mean()
average_quantity
```

Out[67]:
```
branch
A    5.467647
B    5.481928
C    5.582317
Name: quantity, dtype: float64
```

In [69]:
```
average_sale=sales_df['quantity'].mean()
average_sale
```

Out[69]:
```
5.51
```

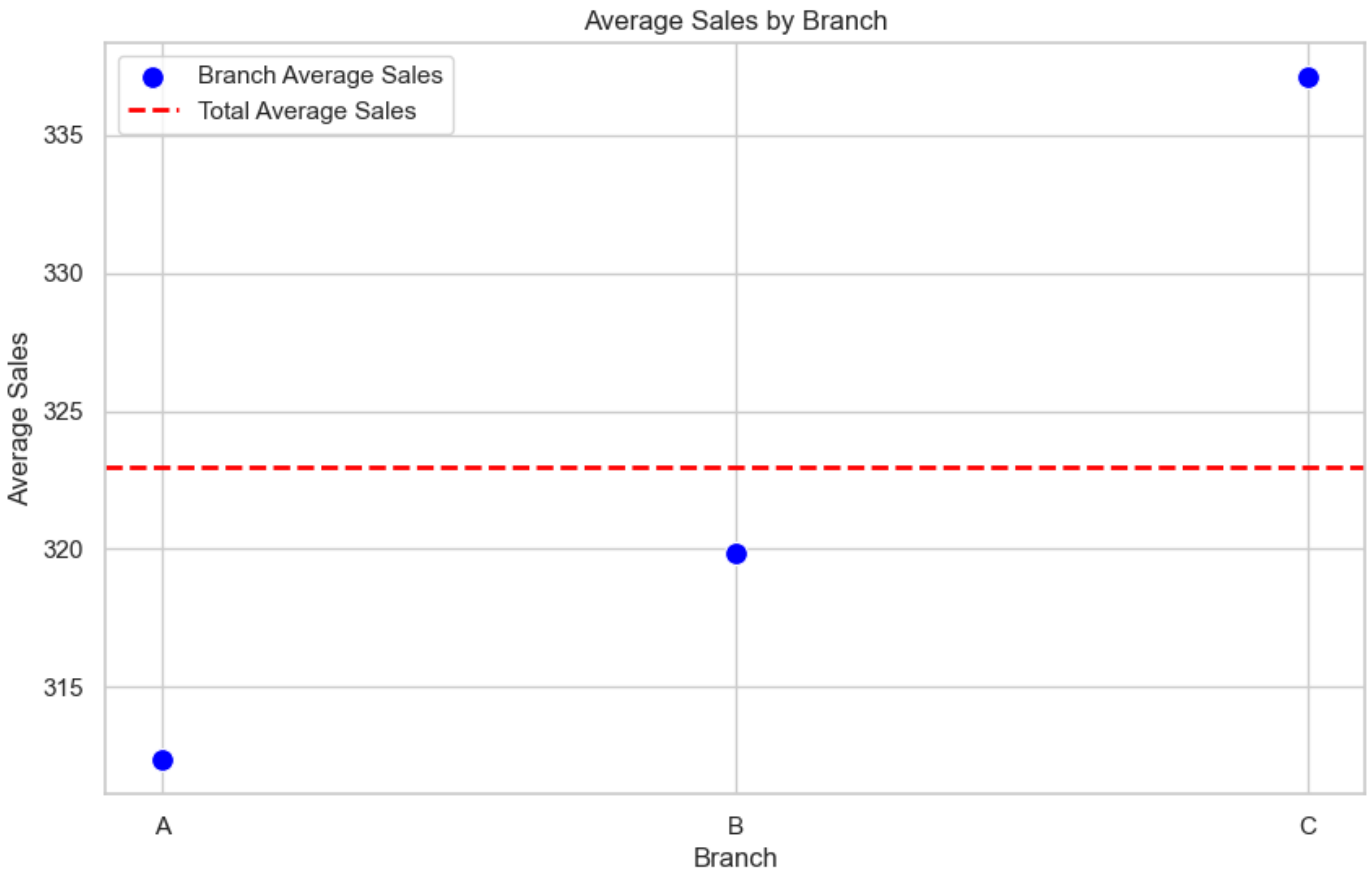
In [79]:
```
# Determine which branches sold more products than the average
branches_above_average = average_quantity[average_quantity > average_sale]
branches_above_average
```

Out[79]:
```
branch
C    5.582317
Name: quantity, dtype: float64
```

In [80]:
```
print(f"The branch(es) that sold more products than the average are: {', '.join(branches
```

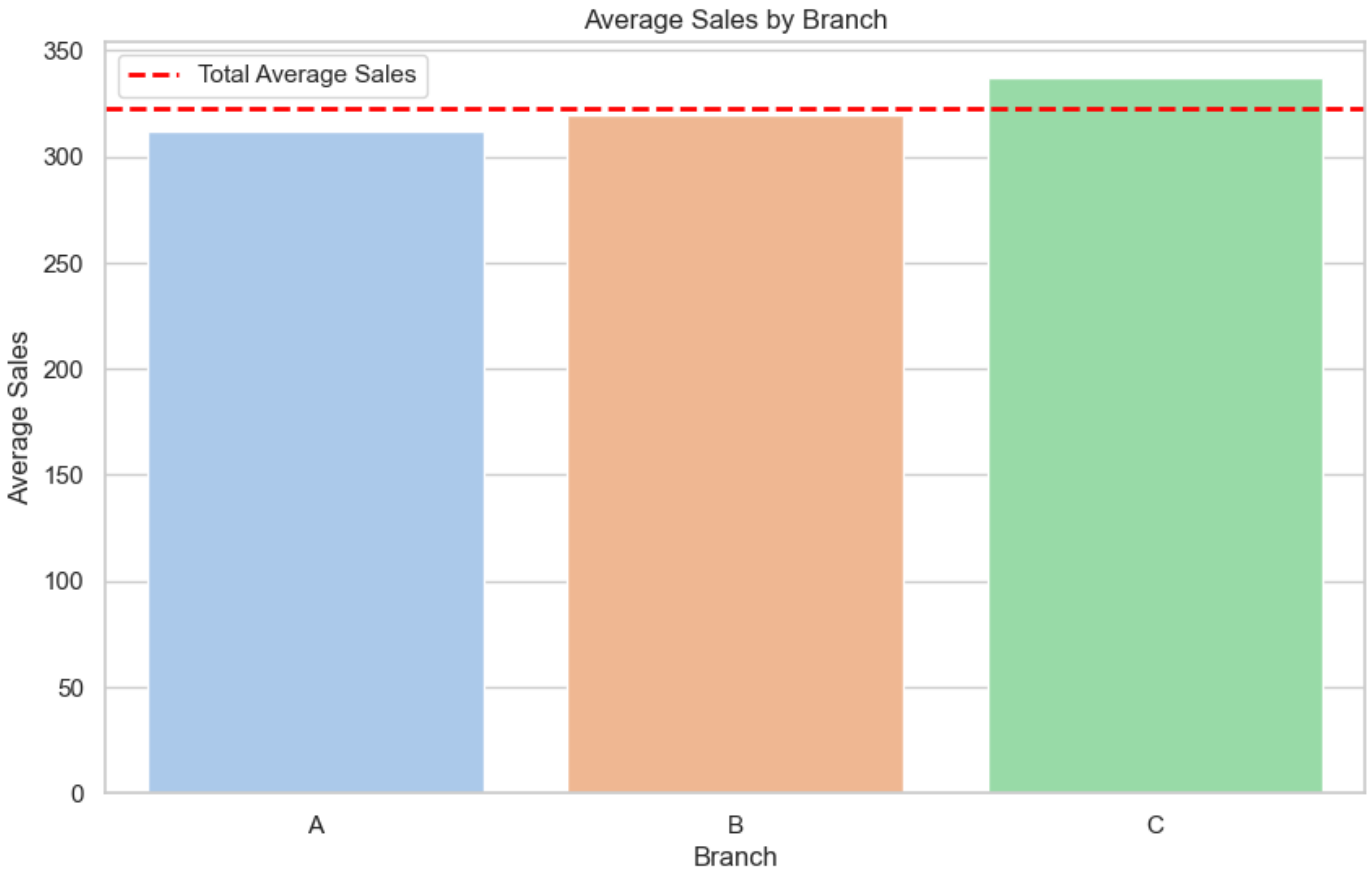
```
The branch(es) that sold more products than the average are: C
```

In [86]:
```
import seaborn as sns
import matplotlib.pyplot as plt


# Calculate the average sales for each branch
average_sales_by_branch = sales_df.groupby('branch')['total'].mean()
total_average_sales = sales_df['total'].mean()

# Create a scatter plot using seaborn
plt.figure(figsize=(10, 6))
sns.scatterplot(x=average_sales_by_branch.index, y=average_sales_by_branch.values, s=100
plt.axhline(total_average_sales, color='red', linestyle='--', label="Total Average Sales
plt.title("Average Sales by Branch")
plt.xlabel("Branch")
plt.ylabel("Average Sales")

# Change the position of the legend to upper left
plt.legend(loc='upper left')

plt.show()
```



In [87]:
```
import seaborn as sns
import matplotlib.pyplot as plt



# Calculate the average sales for each branch
average_sales_by_branch = sales_df.groupby('branch')['total'].mean()
total_average_sales = sales_df['total'].mean()

# Create a bar plot using seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=average_sales_by_branch.index, y=average_sales_by_branch.values, palette="
plt.axhline(total_average_sales, color='red', linestyle='--', label="Total Average Sales
plt.title("Average Sales by Branch")
```

```
plt.xlabel("Branch")
plt.ylabel("Average Sales")

# Change the position of the legend to upper left
plt.legend(loc='upper left')

plt.show()
```



11. What is the most common product line by gender?

In [95]:
```
sales_df.groupby(['gender', 'product_line'])['product_line'].count()
```

Out[95]:
```
gender  product_line
Female  Electronic accessories    84
        Fashion accessories       96
        Food and beverages        90
        Health and beauty         64
        Home and lifestyle        79
        Sports and travel         88
Male    Electronic accessories    86
        Fashion accessories       82
        Food and beverages        84
        Health and beauty         88
        Home and lifestyle        81
        Sports and travel         78
Name: product_line, dtype: int64
```

In [91]:
```
most_common_product_by_gender = sales_df.groupby(['gender', 'product_line'])['product_li
most_common_product_by_gender
```

Out[91]:
```
('Female', 'Fashion accessories')
```

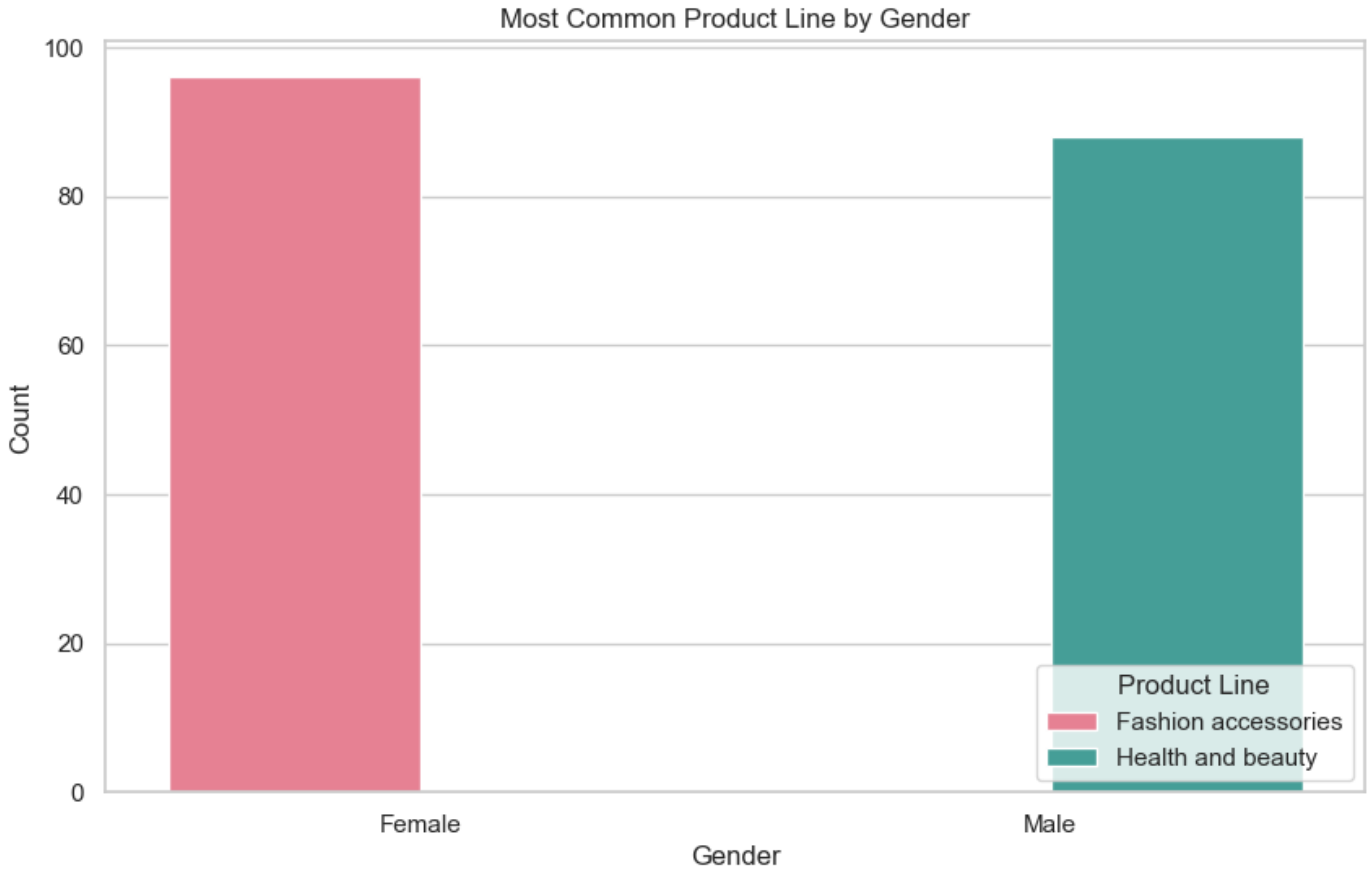
In [98]:
```
import pandas as pd
```

```
# Group the data by 'gender' and 'product_line', then count the occurrences
common_product_line_by_gender = sales_df.groupby(['gender', 'product_line']).size().rese

# Find the most common product line for each gender
most_common_product_line_by_gender = common_product_line_by_gender.loc[common_product_li

# Display the result
print(most_common_product_line_by_gender)
```

```
   gender         product_line  count
1  Female  Fashion accessories     96
9    Male    Health and beauty     88
```

In [100…

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


plt.figure(figsize=(10, 6))
sns.barplot(data=most_common_product_line_by_gender, x='gender', y='count', hue='product
plt.title("Most Common Product Line by Gender")
plt.xlabel("Gender")
plt.ylabel("Count")

plt.legend(title="Product Line", title_fontsize='12', loc='lower right')

plt.show()
```



12. What is the average rating of each product line?

In [101…

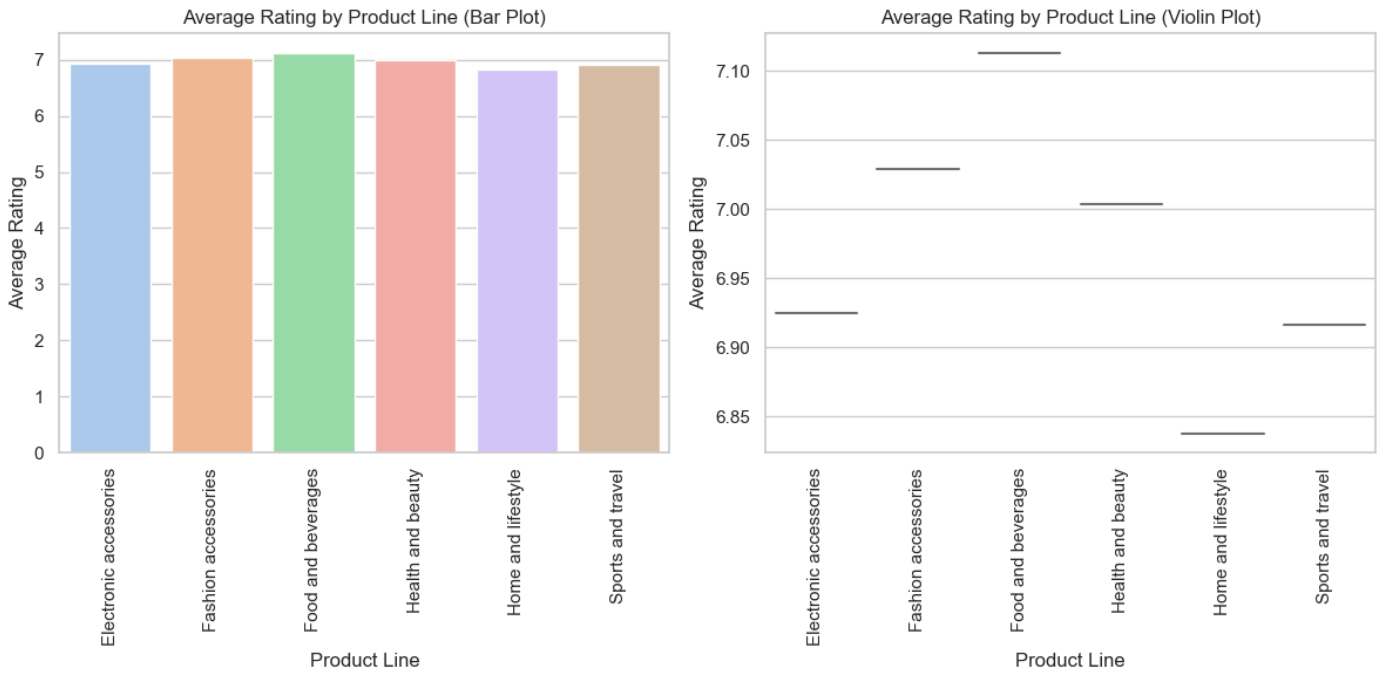
```
import pandas as pd


# Group the data by 'product_line' and calculate the mean rating
```

```
average_rating_by_product_line = sales_df.groupby('product_line')['rating'].mean().reset

# Display the result
print(average_rating_by_product_line)
```

```
             product_line    rating
0  Electronic accessories  6.924706
1     Fashion accessories  7.029213
2      Food and beverages  7.113218
3       Health and beauty  7.003289
4      Home and lifestyle  6.837500
5       Sports and travel  6.916265
```

In [106…

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



plt.figure(figsize=(12, 6))

# Create a bar plot
plt.subplot(1, 2, 1)
sns.barplot(data=average_rating_by_product_line, x='product_line', y='rating', palette='
plt.title("Average Rating by Product Line (Bar Plot)")
plt.xlabel("Product Line")
plt.ylabel("Average Rating")
plt.xticks(rotation=90)

# Create a violin plot
plt.subplot(1, 2, 2)
sns.violinplot(data=average_rating_by_product_line, x='product_line', y='rating', palett
plt.title("Average Rating by Product Line (Violin Plot)")
plt.xlabel("Product Line")
plt.ylabel("Average Rating")
plt.xticks(rotation=90)

plt.tight_layout()

plt.show()
```



SALES BASED QUESTIONS

1. Number of sales made in each time of the day per weekday

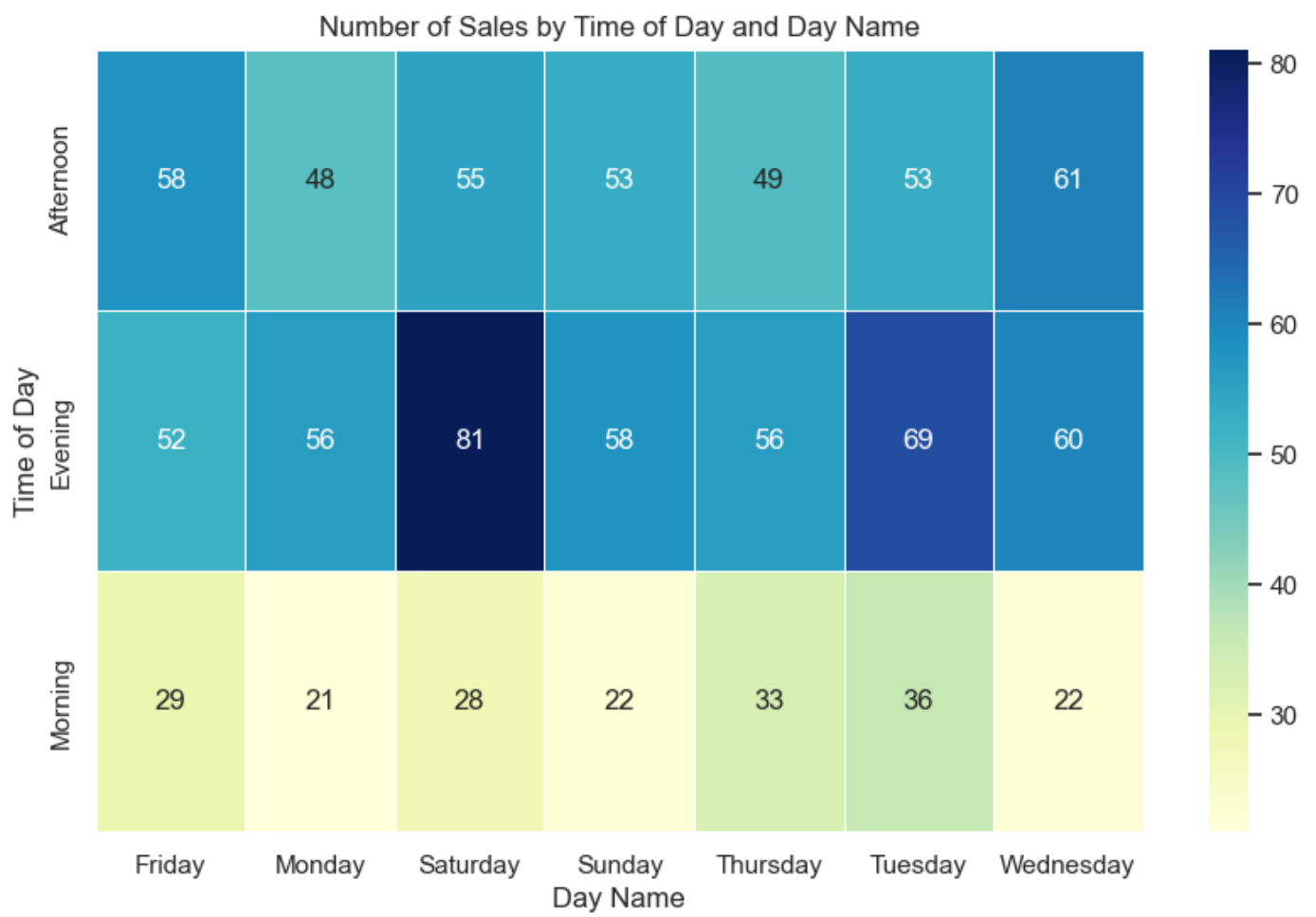
In [109…

```
import pandas as pd



# Group the data by 'day_name' and 'time_of_day' and count the number of sales
sales_count_by_time_and_day = sales_df.groupby(['day_name', 'time_of_day']).size().reset

# Display the result
print(sales_count_by_time_and_day)
```

```
     day_name time_of_day  sales_count
0      Friday   Afternoon           58
1      Friday     Evening           52
2      Friday     Morning           29
3      Monday   Afternoon           48
4      Monday     Evening           56
5      Monday     Morning           21
6    Saturday   Afternoon           55
7    Saturday     Evening           81
8    Saturday     Morning           28
9      Sunday   Afternoon           53
10     Sunday     Evening           58
11     Sunday     Morning           22
12   Thursday   Afternoon           49
13   Thursday     Evening           56
14   Thursday     Morning           33
15    Tuesday   Afternoon           53
16    Tuesday     Evening           69
17    Tuesday     Morning           36
18  Wednesday   Afternoon           61
19  Wednesday     Evening           60
20  Wednesday     Morning           22
```

In [111…

```
import seaborn as sns
import matplotlib.pyplot as plt


# Pivot the DataFrame to create a heatmap
heatmap_data = sales_count_by_time_and_day.pivot(index='time_of_day', columns='day_name'

plt.figure(figsize=(10, 6))
sns.heatmap(heatmap_data, cmap="YlGnBu", annot=True, fmt='d', linewidths=.5)
plt.title("Number of Sales by Time of Day and Day Name")
plt.xlabel("Day Name")
plt.ylabel("Time of Day")

plt.show()
```



2. Which of the customer types brings the most revenue?

In [114…
```
import pandas as pd


# Group the data by 'customer_type' and calculate the sum of total revenue
revenue_by_customer_type = sales_df.groupby('customer_type')['total'].sum().reset_index(
revenue_by_customer_type
```

Out[114]:

	customer_type	total
0	Member	164223.444
1	Normal	158743.305

In [115…
```
# Find the customer type with the highest total revenue
most_revenue_customer_type = revenue_by_customer_type.loc[revenue_by_customer_type['tota
most_revenue_customer_type
```

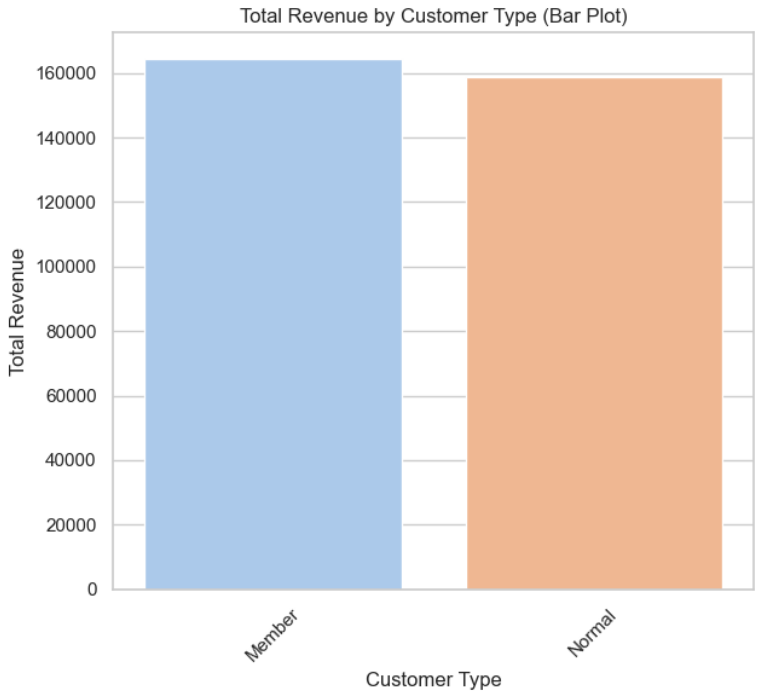
Out[115]:
```
customer_type        Member
total            164223.444
Name: 0, dtype: object
```

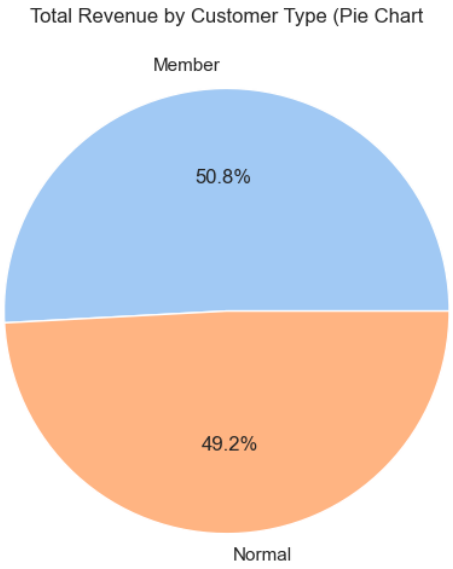
In [116…
```
# Display the result
print("Customer Type with the Most Revenue:", most_revenue_customer_type['customer_type'
print("Total Revenue:", most_revenue_customer_type['total'])
```

```
Customer Type with the Most Revenue: Member
Total Revenue: 164223.444
```

In [117…
```
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Assuming 'revenue_by_customer_type' is the DataFrame with total revenue by customer ty

plt.figure(figsize=(12, 6))

# Create a bar plot
plt.subplot(1, 2, 1)
sns.barplot(data=revenue_by_customer_type, x='customer_type', y='total', palette='pastel
plt.title("Total Revenue by Customer Type (Bar Plot)")
plt.xlabel("Customer Type")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)

# Create a pie chart
plt.subplot(1, 2, 2)
plt.pie(revenue_by_customer_type['total'], labels=revenue_by_customer_type['customer_typ
plt.title("Total Revenue by Customer Type (Pie Chart")

plt.tight_layout()

plt.show()
```





3. Which city has the largest tax percent/ VAT (**Value Added Tax**)?

In [123…

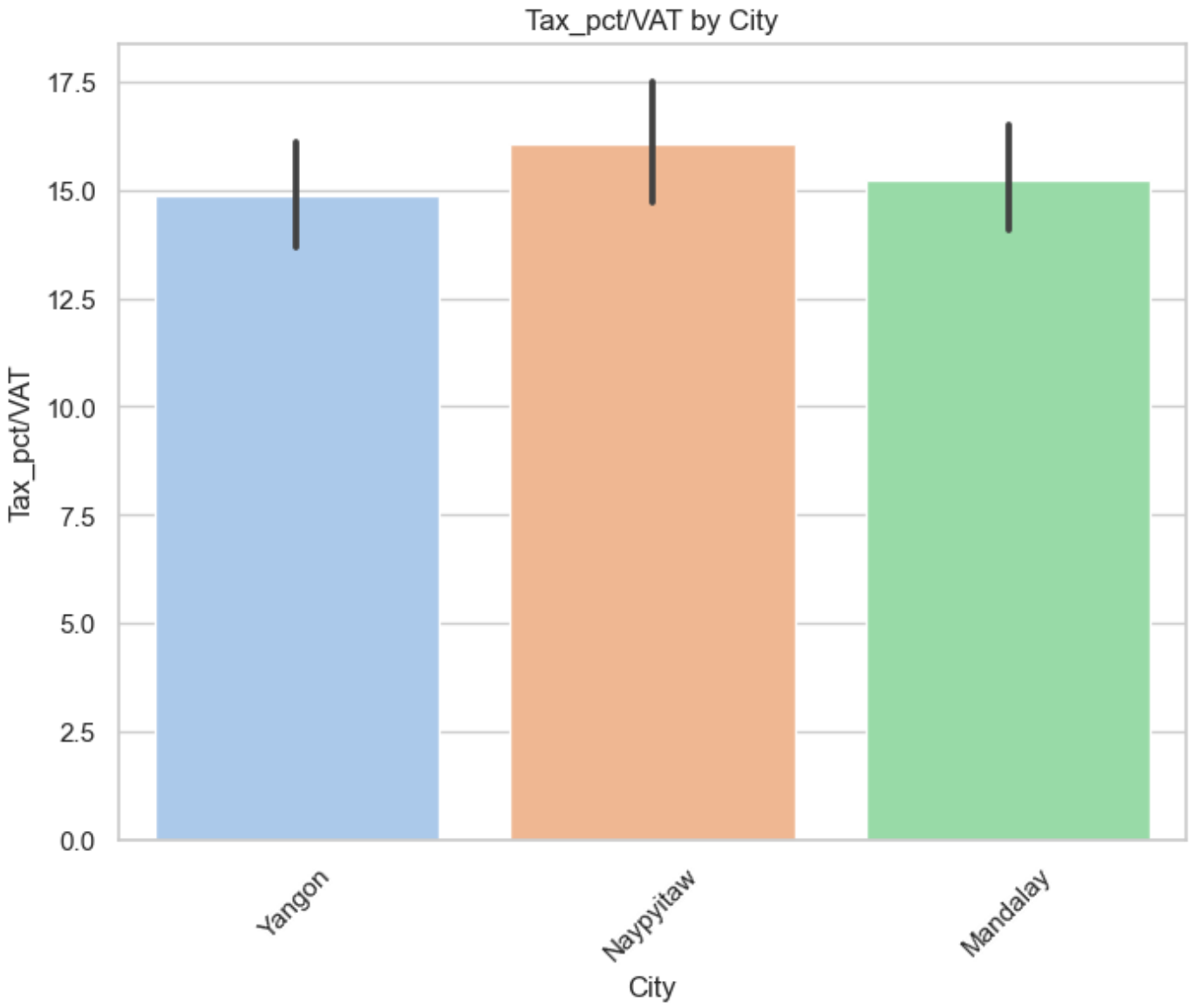
```
import pandas as pd


# Find the row with the largest tax_pct
largest_tax_pct_city = sales_df.loc[sales_df['tax_5_pct'].idxmax()]

# Display the result
print("City with the Largest tax_pct:", largest_tax_pct_city['city'])
print("Largest tax_pct:", largest_tax_pct_city['tax_5_pct'])
```

```
City with the Largest tax_pct: Naypyitaw
Largest tax_pct: 49.65
```

In [124…

```
# Create a bar plot to visualize the largest tax_pct/VAT by city
plt.figure(figsize=(8, 6))
```

```
sns.barplot(data=sales_df, x='city', y='tax_5_pct', palette='pastel')
plt.title("Tax_pct/VAT by City")
plt.xlabel("City")
plt.ylabel("Tax_pct/VAT")
plt.xticks(rotation=45)

plt.show()
```

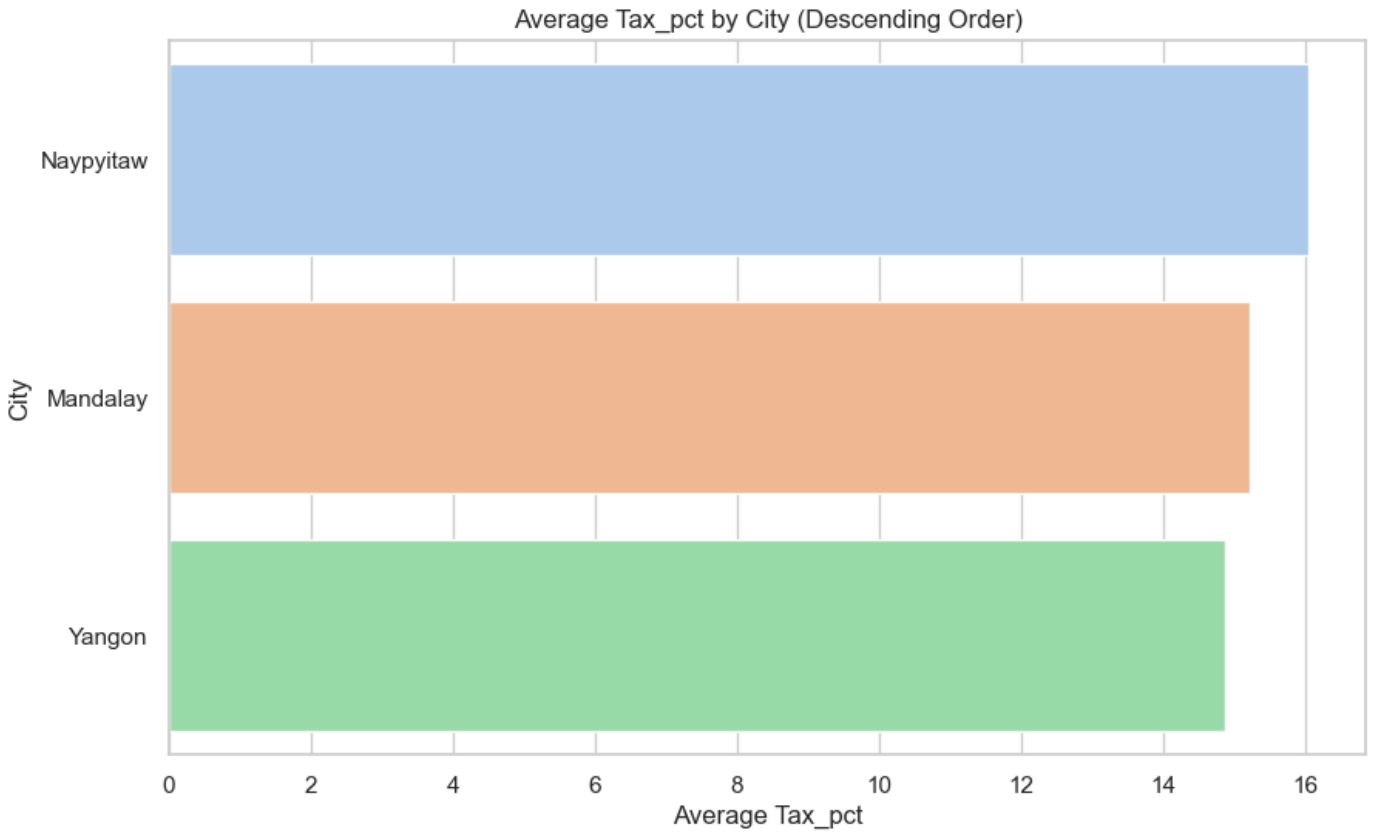


In [122…
```
import pandas as pd



# Group by 'city' and calculate the average tax_pct, rounding to 2 decimal places
result = sales_df.groupby('city')['tax_5_pct'].agg(avg_tax_pct='mean').round(2)

# Sort the result in descending order by 'avg_tax_pct'
result = result.reset_index().sort_values(by='avg_tax_pct', ascending=False)

print(result)
```

```
        city  avg_tax_pct
1  Naypyitaw        16.05
0   Mandalay        15.23
2     Yangon        14.87
```

In [125…
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=result, x='avg_tax_pct', y='city', palette='pastel')
plt.title("Average Tax_pct by City (Descending Order)")
plt.xlabel("Average Tax_pct")
plt.ylabel("City")

plt.show()
```



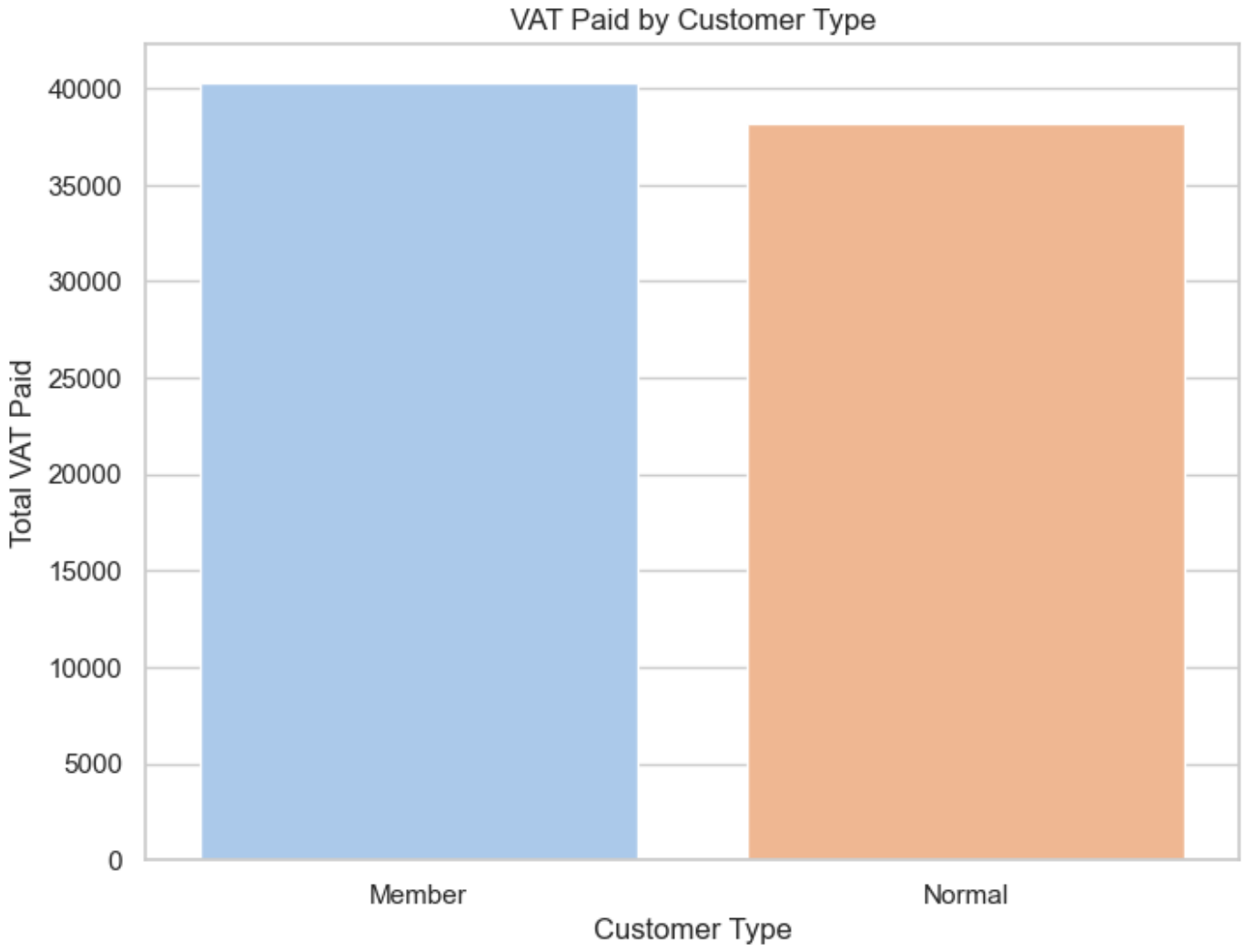
4. Which customer type pays the most in VAT?

In [126…

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



# Calculate the total VAT paid by each customer type
sales_df['vat_paid'] = sales_df['total'] * (sales_df['tax_5_pct'] / 100)

# Group by 'customer_type' and calculate the sum of VAT paid
vat_by_customer_type = sales_df.groupby('customer_type')['vat_paid'].sum().reset_index()

# Find the customer type that pays the most VAT
most_vat_customer_type = vat_by_customer_type.loc[vat_by_customer_type['vat_paid'].idxma

# Display the result
print("Customer Type Paying the Most VAT:", most_vat_customer_type['customer_type'])
print("Total VAT Paid:", most_vat_customer_type['vat_paid'])

# Create a bar plot to visualize VAT paid by customer type
plt.figure(figsize=(8, 6))
sns.barplot(data=vat_by_customer_type, x='customer_type', y='vat_paid', palette='pastel'
plt.title("VAT Paid by Customer Type")
```

```
plt.xlabel("Customer Type")
plt.ylabel("Total VAT Paid")

plt.show()
```

```
Customer Type Paying the Most VAT: Member
Total VAT Paid: 40276.89285081
```



In [4]:
```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   invoice_id               1000 non-null   object
 1   branch                   1000 non-null   object
 2   city                     1000 non-null   object
 3   customer_type            1000 non-null   object
 4   gender                   1000 non-null   object
 5   product_line             1000 non-null   object
 6   unit_price               1000 non-null   float64
 7   quantity                 1000 non-null   int64
 8   tax_5_pct                1000 non-null   float64
 9   total                    1000 non-null   float64
 10  date                     1000 non-null   object
 11  time                     1000 non-null   timedelta64[ns]
 12  payment                  1000 non-null   object
 13  cogs                     1000 non-null   float64
 14  gross_margin_percentage  1000 non-null   float64
 15  gross_income             1000 non-null   float64
 16  rating                   1000 non-null   float64
```

```
 17  time_of_day              1000 non-null   object
 18  day_name                 1000 non-null   object
 19  month_name               1000 non-null   object
dtypes: float64(7), int64(1), object(11), timedelta64[ns](1)
memory usage: 156.4+ KB
```

Customer Based Questions

1. How many unique customer types does the data have?

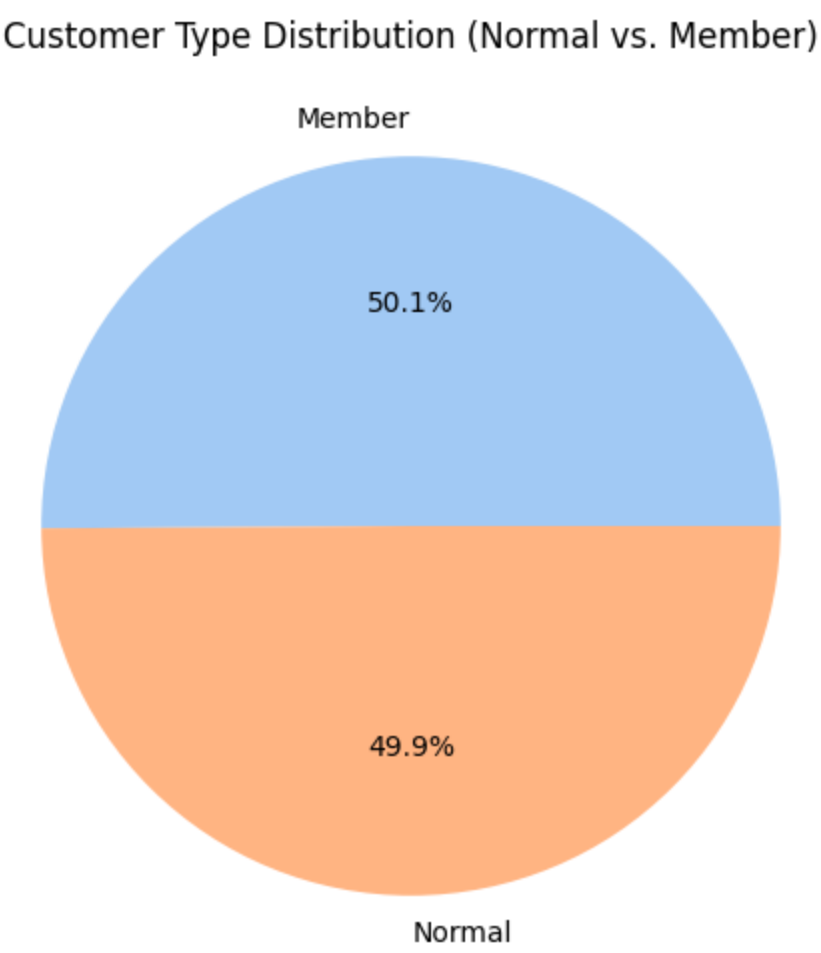
In [6]:
```
sales_df.customer_type.unique()
```

Out[6]:
```
array(['Normal', 'Member'], dtype=object)
```

In [7]:
```
unique_customer_types = sales_df['customer_type'].nunique()
print("Number of Unique Customer Types:", unique_customer_types)
```

```
Number of Unique Customer Types: 2
```

In [11]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



# Filter the data for 'customer_type' equal to 'Normal' and 'Member'
filtered_data = sales_df[sales_df['customer_type'].isin(['Normal', 'Member'])]

# Calculate the count of each customer type
customer_type_counts = filtered_data['customer_type'].value_counts()

# Create a pie chart using Seaborn and Matplotlib
plt.figure(figsize=(6, 6))
plt.pie(customer_type_counts, labels=customer_type_counts.index, autopct='%1.1f%%', colo
plt.title("Customer Type Distribution (Normal vs. Member)")

plt.show()
```



2. What is the most common customer type?

In [21]:
```
sales_df.customer_type.value_counts()
```

Out[21]:
```
customer_type
Member    501
Normal    499
Name: count, dtype: int64
```

In [23]:
```
sales_df.customer_type.value_counts().idxmax()
```

Out[23]:
```
'Member'
```

3. How many unique payment methods does the data have?

In [14]:
```
sales_df['payment'].unique()
```

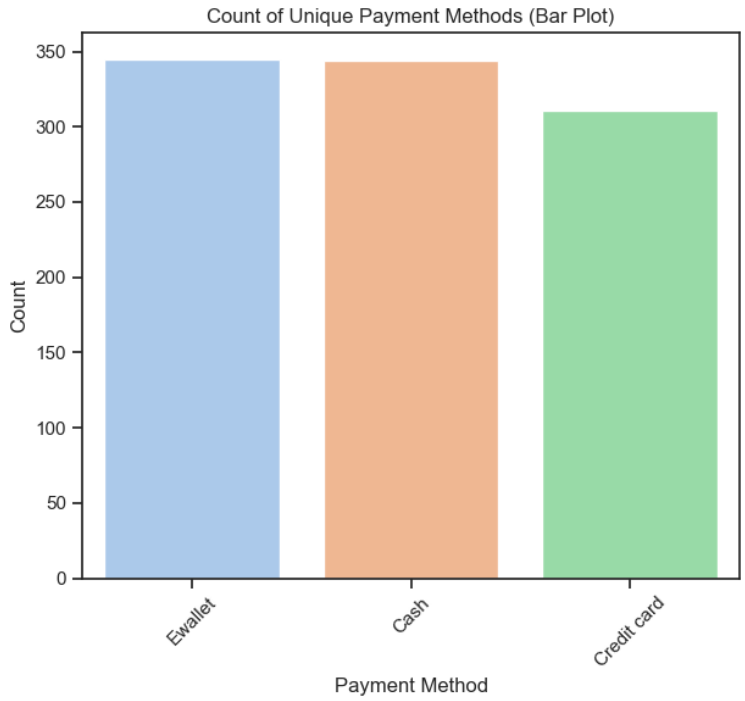
Out[14]:
```
array(['Credit card', 'Ewallet', 'Cash'], dtype=object)
```

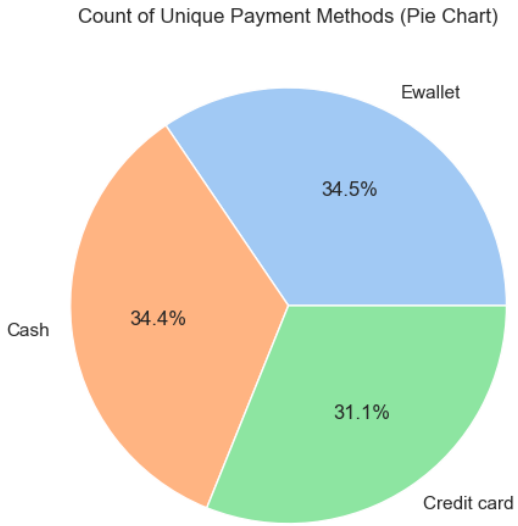
In [15]:
```
unique_payment_methods = sales_df['payment'].nunique()

print("Number of Unique Payment Methods:", unique_payment_methods)
```

```
Number of Unique Payment Methods: 3
```

In [18]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Calculate the count of each unique payment method
payment_counts = sales_df['payment'].value_counts()

# Create a figure with two subplots
plt.figure(figsize=(12, 6))

# Create the bar graph
plt.subplot(1, 2, 1)
sns.barplot(x=payment_counts.index, y=payment_counts.values, palette='pastel')
plt.title("Count of Unique Payment Methods (Bar Plot)")
plt.xlabel("Payment Method")
plt.ylabel("Count")
plt.xticks(rotation=45)

# Create the pie chart
plt.subplot(1, 2, 2)
plt.pie(payment_counts, labels=payment_counts.index, autopct='%1.1f%%', colors=sns.color
plt.title("Count of Unique Payment Methods (Pie Chart)")

plt.tight_layout()

plt.show()
```





4. Which customer type buys the most?

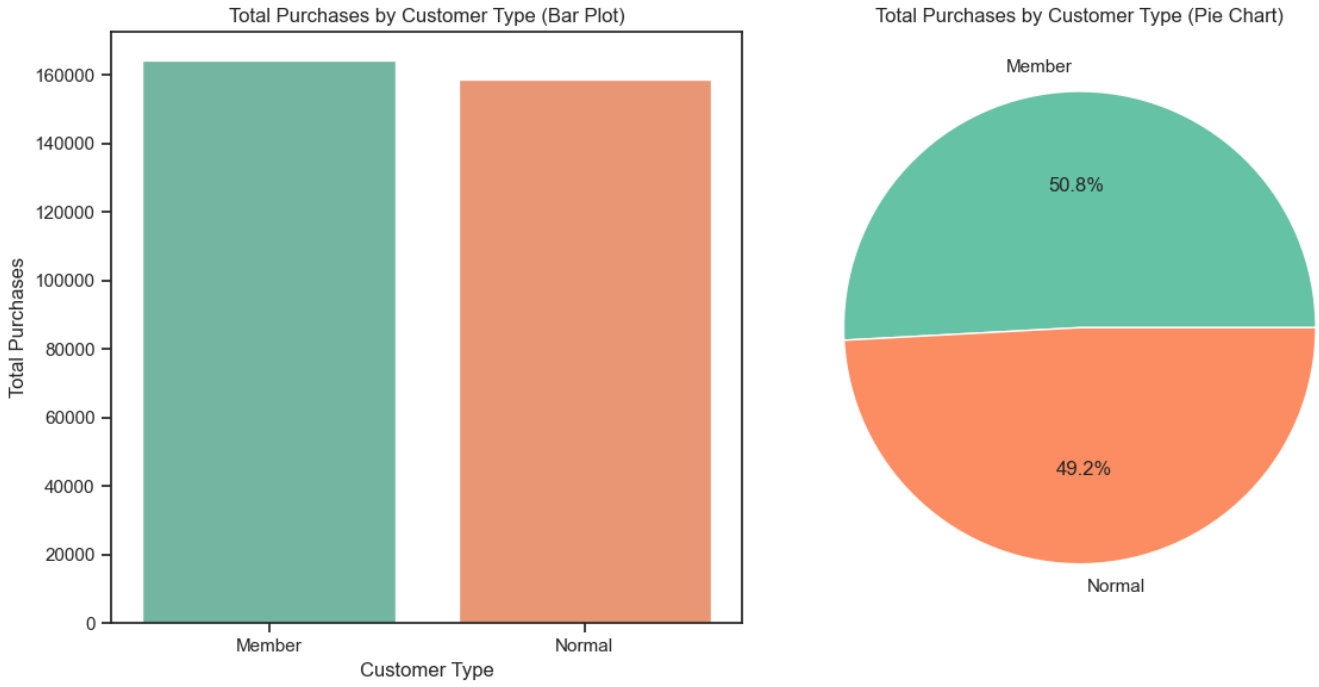
In [30]:
```
sales_df.groupby('customer_type')['total'].sum().reset_index()
```

Out[30]:

	customer_type	total
0	Member	164223.444
1	Normal	158743.305

In [33]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Group by customer type and sum the total purchases
customer_type_totals = sales_df.groupby('customer_type')['total'].sum().reset_index()

# Create a figure with two subplots
plt.figure(figsize=(12, 6))

# Create the bar graph with a different color palette
plt.subplot(1, 2, 1)
sns.barplot(data=customer_type_totals, x='customer_type', y='total', palette='Set2')
plt.title("Total Purchases by Customer Type (Bar Plot)")
plt.xlabel("Customer Type")
plt.ylabel("Total Purchases")
plt.xticks(rotation=0)

# Create the pie chart with a different color palette
plt.subplot(1, 2, 2)
plt.pie(customer_type_totals['total'], labels=customer_type_totals['customer_type'], aut
plt.title("Total Purchases by Customer Type (Pie Chart)")

plt.tight_layout()

plt.show()
```



5. What is the gender of most of the customers?

In [38]:
```
# Group by gender and count the number of customer types
gender_customer_counts = sales_df.groupby('gender')['customer_type'].count().reset_index
gender_customer_counts
```

Out[38]:

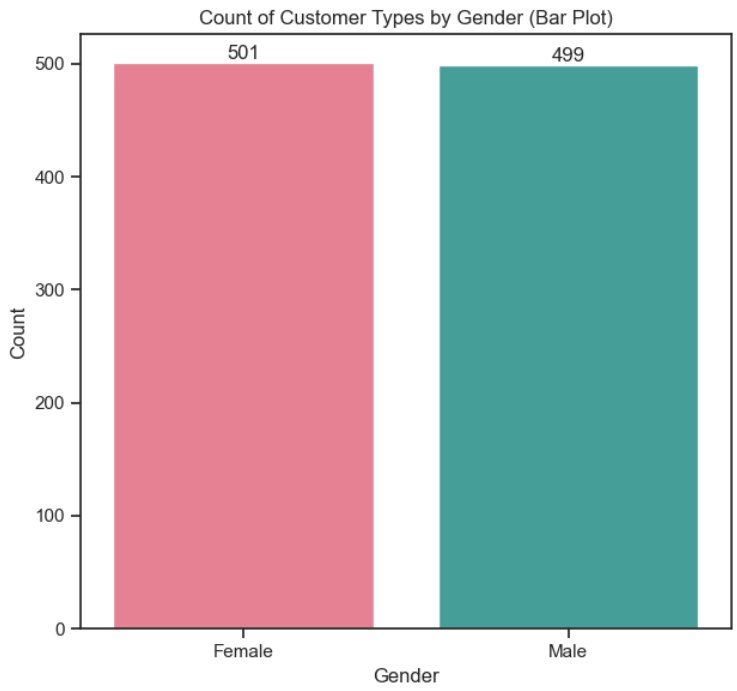
	gender	customer_type
0	Female	501
1	Male	499

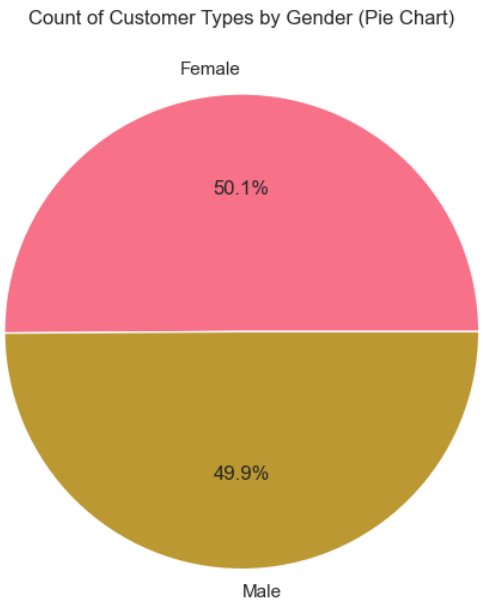
In [36]:
```
import pandas as pd

most_common_gender = sales_df['gender'].mode()[0]

print("Gender of Most Customers:", most_common_gender)
```

```
Gender of Most Customers: Female
```

In [42]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'sales_df' is your DataFrame

# Group by gender and count the number of customer types
gender_customer_counts = sales_df.groupby('gender')['customer_type'].count().reset_index

# Create a unique color palette
unique_colors = sns.color_palette('husl', n_colors=len(gender_customer_counts))

# Create a single figure with subplots
plt.figure(figsize=(12, 6))

# Create the bar chart on the left
plt.subplot(1, 2, 1)
ax = sns.barplot(data=gender_customer_counts, x='gender', y='customer_type', palette=uni

# Add count labels to the bars
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2, p.get_height()

plt.title("Count of Customer Types by Gender (Bar Plot)")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.xticks(rotation=0)

# Create the pie chart on the right
plt.subplot(1, 2, 2)
plt.pie(gender_customer_counts['customer_type'], labels=gender_customer_counts['gender']
plt.title("Count of Customer Types by Gender (Pie Chart)")

plt.tight_layout()

plt.show()
```





6. What is the gender distribution per branch?

In [43]:
```
sales_df.groupby(['gender','branch'])['branch'].count()
```

Out[43]:
```
gender  branch
Female  A         161
        B         162
        C         178
Male    A         179
        B         170
        C         150
Name: branch, dtype: int64
```

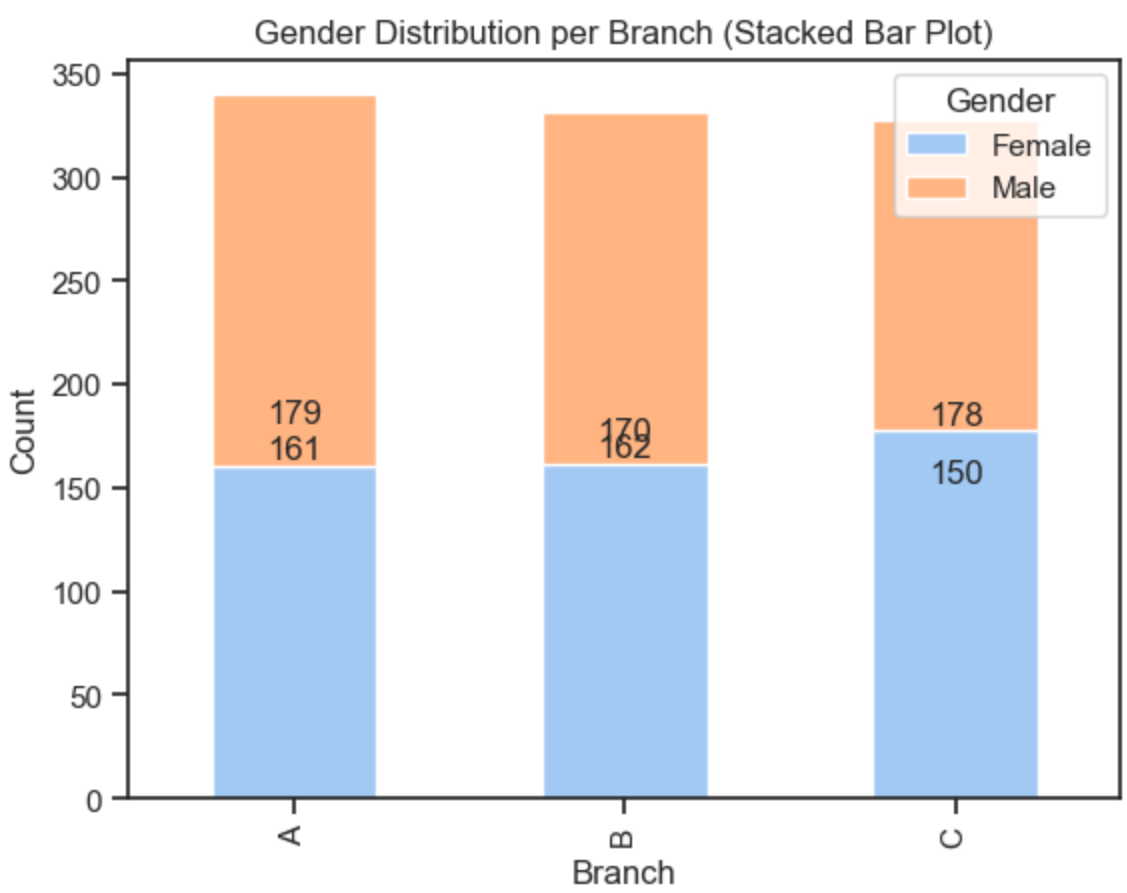
In [47]:
```
import pandas as pd


# Group by both 'branch' and 'gender' and count the number of occurrences
gender_distribution_per_branch = sales_df.groupby(['branch', 'gender']).size().unstack(f

gender_distribution_per_branch
```

Out[47]:

gender	Female	Male
branch		
A	161	179
B	162	170
C	178	150

In [51]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Group by both 'branch' and 'gender' and count the number of occurrences
gender_distribution_per_branch = sales_df.groupby(['branch', 'gender']).size().unstack(f

# Create a stacked bar plot
plt.figure(figsize=(10, 6))
sns.set_palette("pastel")  # Use a pastel color palette

ax = gender_distribution_per_branch.plot(kind='bar', stacked=True)

# Add count labels to the bars
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{int(height)}', (p.get_x() + p.get_width() / 2, height), ha='center',

plt.title("Gender Distribution per Branch (Stacked Bar Plot)")
plt.xlabel("Branch")
plt.ylabel("Count")
plt.legend(title='Gender', labels=['Female', 'Male'], loc='upper right')

plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



7. Which time of the day do customers give most ratings?

In [59]:
```
# Group by 'time_of_day' and calculate the average rating
average_rating_by_time = sales_df.groupby('time_of_day')['rating'].mean()
average_rating_by_time
```

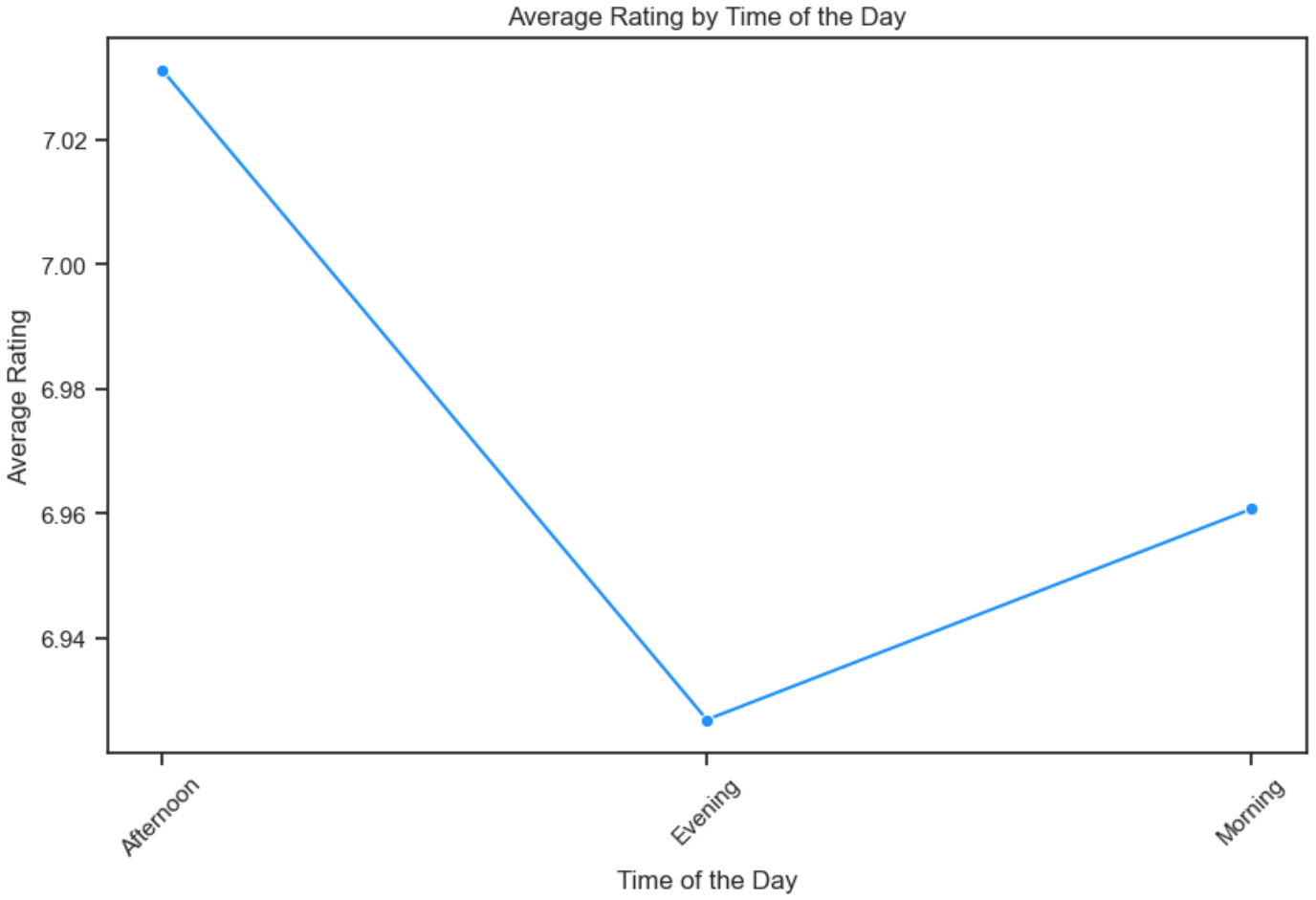
Out[59]:
```
time_of_day
Afternoon    7.031300
Evening      6.926852
Morning      6.960733
Name: rating, dtype: float64
```

In [61]:
```
import pandas as pd



# Group by 'time_of_day' and calculate the average rating
average_rating_by_time = sales_df.groupby('time_of_day')['rating'].mean().reset_index()

# Find the time of the day with the highest average rating
most_rated_time = average_rating_by_time[average_rating_by_time['rating'] == average_rat

print(most_rated_time)
```

```
  time_of_day  rating
0   Afternoon  7.0313
```

In [62]:
```
import seaborn as sns
import matplotlib.pyplot as plt



# Create a line plot to visualize the average rating by time of the day
plt.figure(figsize=(10, 6))
sns.lineplot(data=average_rating_by_time, x='time_of_day', y='rating', marker='o', color
```

```
plt.title("Average Rating by Time of the Day")
plt.xlabel("Time of the Day")
plt.ylabel("Average Rating")
plt.xticks(rotation=45)

plt.show()
```



8. Which time of the day do customers give most ratings per branch?

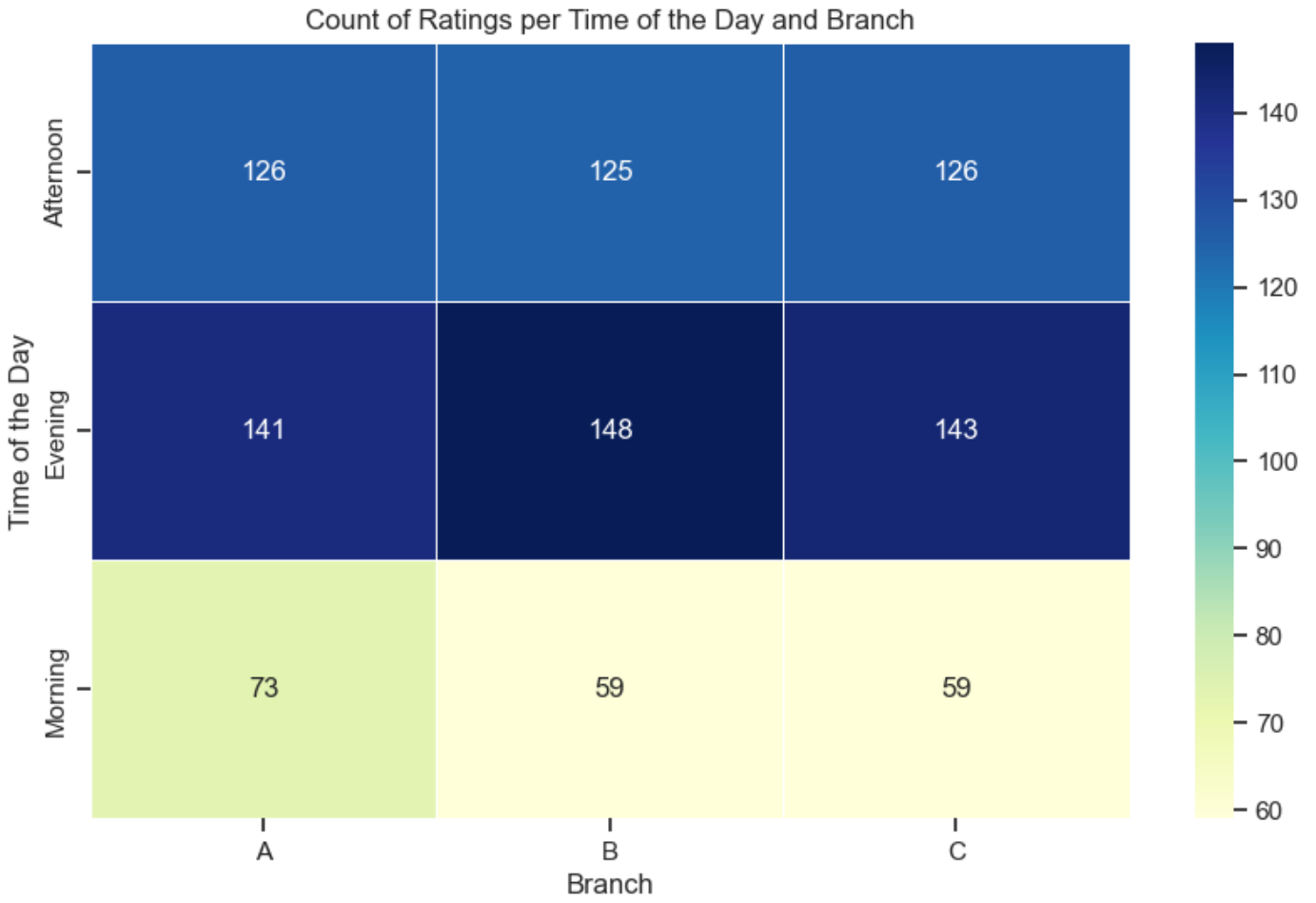
In [64]:
```
sales_df.groupby(["time_of_day",'branch'])['rating'].count().unstack(fill_value=0)
```

Out[64]:

time_of_day	branch	A	B	C
Afternoon		126	125	126
Evening		141	148	143
Morning		73	59	59

In [74]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Create a heatmap to visualize the count of ratings
plt.figure(figsize=(10, 6))
sns.heatmap(data=rating_counts, cmap="YlGnBu", annot=True, fmt='d', linewidths=.5)
plt.title("Count of Ratings per Time of the Day and Branch")
plt.xlabel("Branch")
```

```
plt.ylabel("Time of the Day")

plt.show()
```



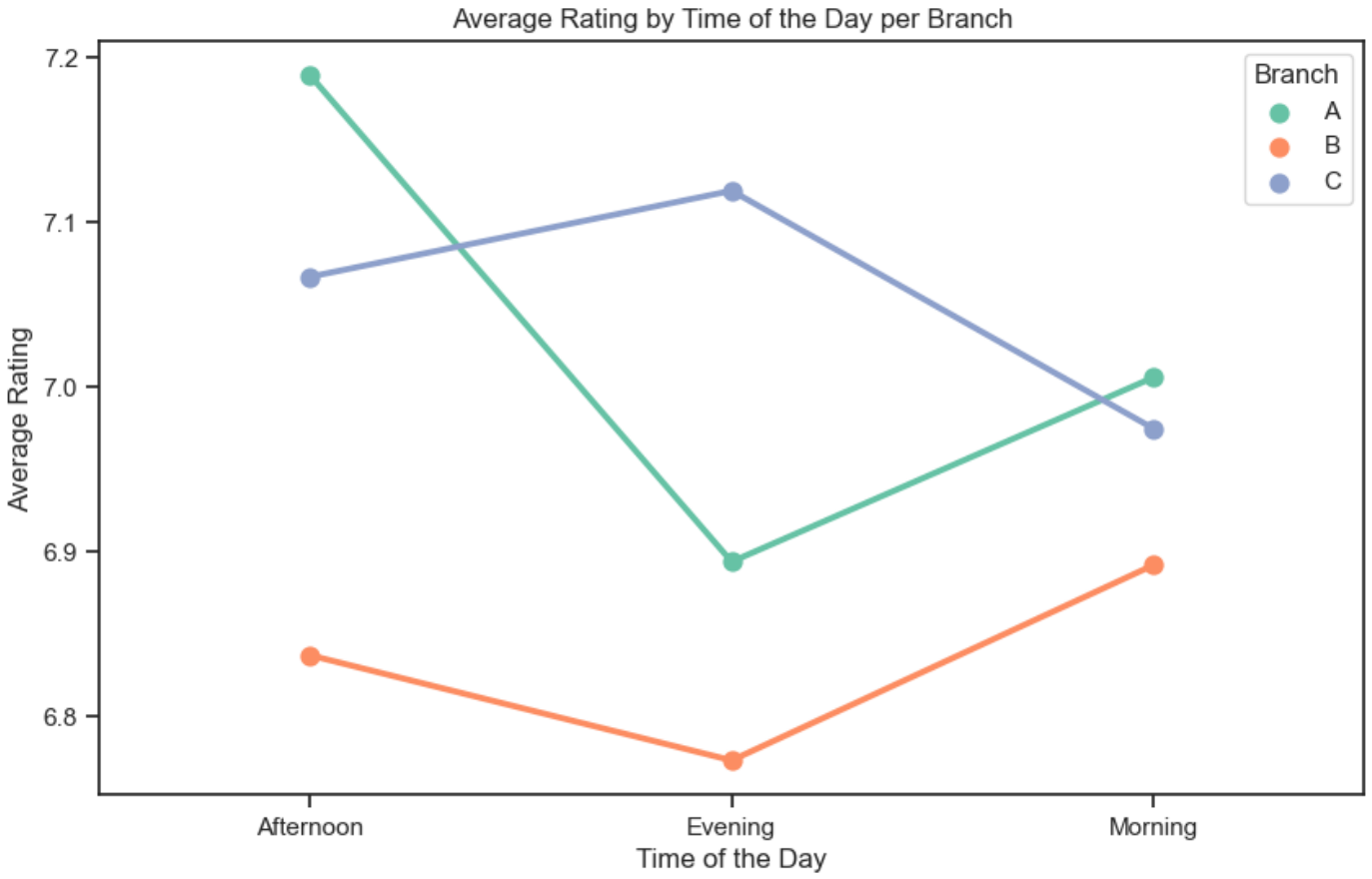
In [65]:
```
import pandas as pd


# Group by both 'branch' and 'time_of_day' and calculate the average rating
average_rating_by_branch_time = sales_df.groupby(['branch', 'time_of_day'])['rating'].me

# Find the time of the day with the highest average rating per branch
most_rated_time_per_branch = average_rating_by_branch_time.groupby('branch').apply(lambd

most_rated_time_per_branch
```

Out[65]:

branch		branch	time_of_day	rating
A	**0**	A	Afternoon	7.188889
B	**5**	B	Morning	6.891525
C	**7**	C	Evening	7.118881

In [76]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Group by both 'branch' and 'time_of_day' and calculate the average rating
average_rating_by_branch_time = sales_df.groupby(['branch', 'time_of_day'])['rating'].me
```

```
# Create a point plot to visualize the average rating by time of the day per branch
plt.figure(figsize=(10, 6))
sns.pointplot(data=average_rating_by_branch_time, x='time_of_day', y='rating', hue='bran
plt.title("Average Rating by Time of the Day per Branch")
plt.xlabel("Time of the Day")
plt.ylabel("Average Rating")
plt.legend(title='Branch', loc='upper right')

plt.show()
```



9. Which day fo the week has the best avg ratings?

In [87]:
```
sales_df.groupby('day_name')['rating'].mean().reset_index()
```

Out[87]:

	day_name	rating
0	Friday	7.076259
1	Monday	7.153600
2	Saturday	6.901829
3	Sunday	7.011278
4	Thursday	6.889855
5	Tuesday	7.003165
6	Wednesday	6.805594

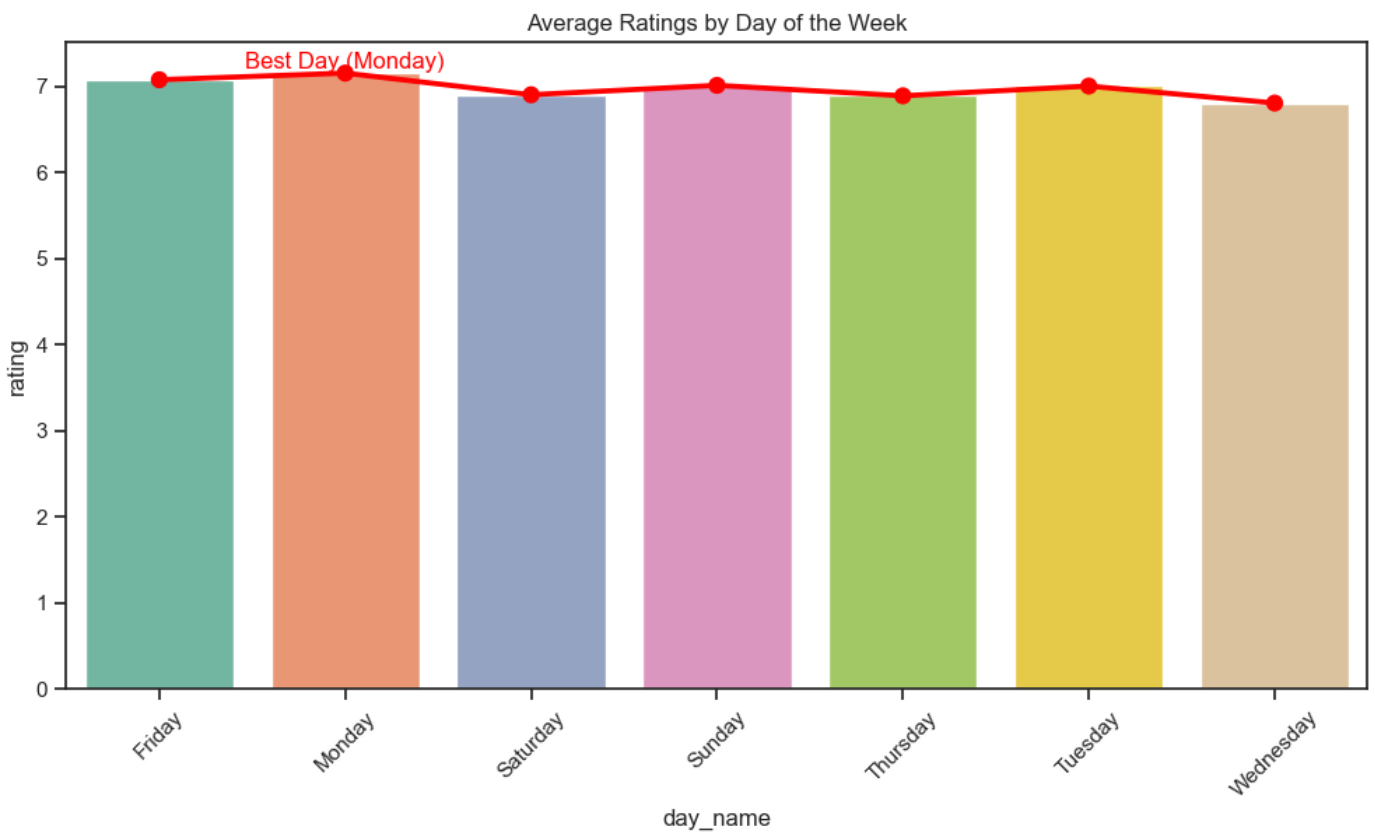
In [92]:
```
average_rating_by_day.loc[average_rating_by_day['rating'].idxmax()]
```

Out[92]:
```
day_name    Monday
rating      7.1536
Name: 1, dtype: object
```

In [89]:
```
import pandas as pd


# Group by 'day_name' and calculate the average rating for each day
average_rating_by_day = sales_df.groupby('day_name')['rating'].mean().reset_index()

# Find the day_name with the best average ratings
best_day_for_ratings = average_rating_by_day.loc[average_rating_by_day['rating'].idxmax(

print(best_day_for_ratings)
```

```
day_name    Monday
rating      7.1536
Name: 1, dtype: object
```

In [99]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



# Define a custom color palette for the days
custom_palette = sns.color_palette("Set2", len(sales_df['day_name'].unique()))


plt.figure(figsize=(12, 6))

# Create a bar plot with custom colors for different days
sns.barplot(data=average_rating_by_day, x='day_name', y='rating', palette=custom_palette
plt.title("Average Ratings by Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Average Rating")

# Highlight the best day with the highest rating using an annotation
plt.text(best_day_for_ratings.name, best_day_for_ratings["rating"],
         f'Best Day ({best_day_for_ratings["day_name"]})', ha='center', va='bottom', col

# Create a point plot to overlay the same data for better visualization
sns.pointplot(data=average_rating_by_day, x='day_name', y='rating', color='red')

plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility

plt.show()
```



10. Which day of the week has the best average ratings per branch?

In [100…

```
import pandas as pd



# Group by both 'branch' and 'day_name' and calculate the average rating for each combin
average_rating_by_branch_day = sales_df.groupby(['branch', 'day_name'])['rating'].mean()

# Find the day of the week with the best average ratings per branch
best_day_by_branch = average_rating_by_branch_day.groupby('branch')['rating'].idxmax()
best_days_for_ratings = average_rating_by_branch_day.loc[best_day_by_branch]

best_days_for_ratings
```

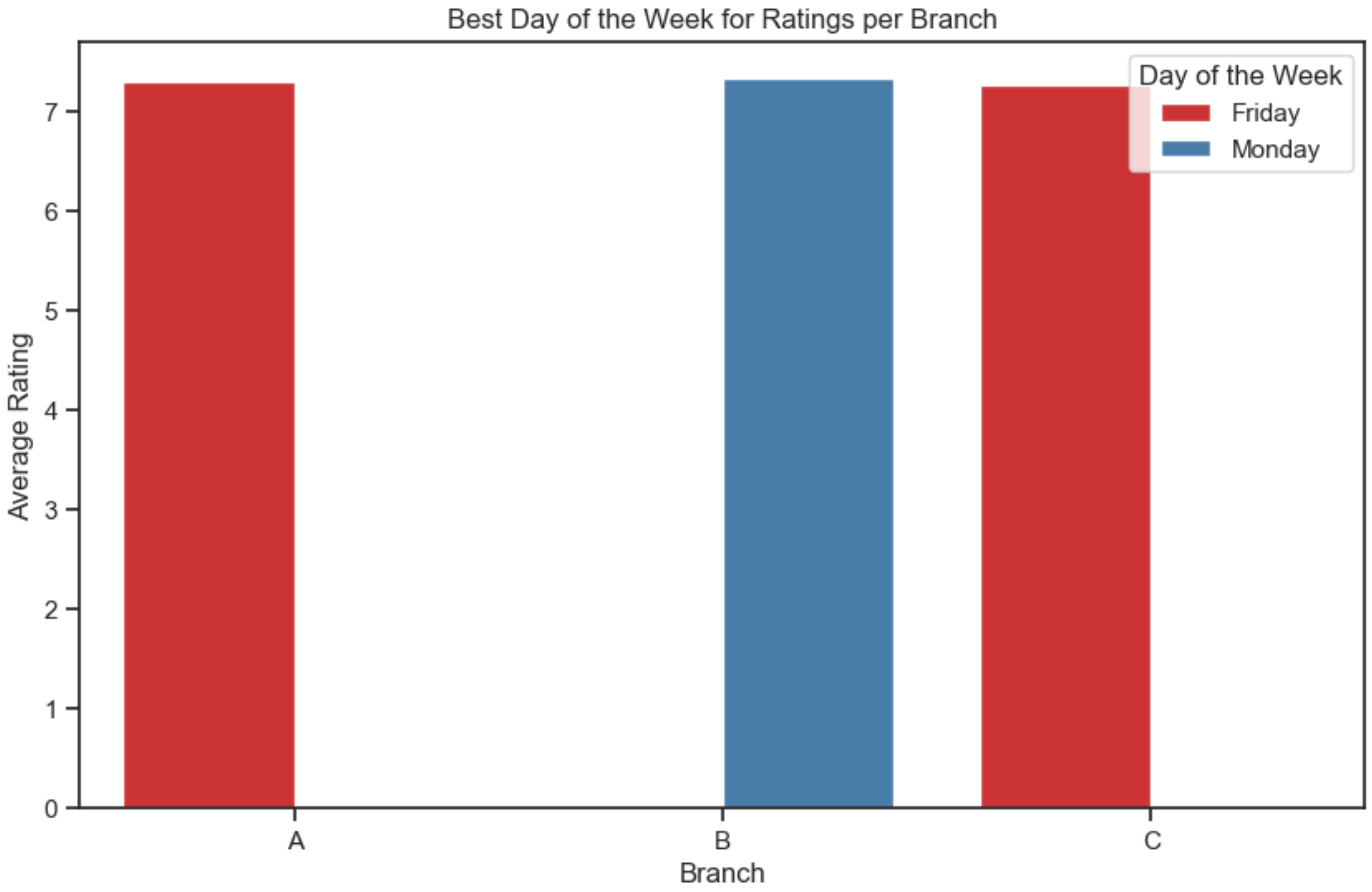
Out[100]:

	branch	day_name	rating
0	A	Friday	7.312000
8	B	Monday	7.335897
14	C	Friday	7.278947

In [106…

```
import seaborn as sns
import matplotlib.pyplot as plt



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

# Create a bar plot to visualize the best days of the week for ratings per branch
sns.barplot(data=best_days_for_ratings, x='branch', y='rating', hue='day_name', palette=
plt.title("Best Day of the Week for Ratings per Branch")
plt.xlabel("Branch")
```

```
plt.ylabel("Average Rating")
plt.legend(title='Day of the Week', loc='upper right')

plt.show()
```



In [112…
```
!jupyter nbconvert --to webpdf --allow-chromium-download python_mysql_walmart_Data_Analy
```

```
[NbConvertApp] Converting notebook python_mysql_walmart_data_analysis.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 2130948 bytes to python_mysql_walmart_data_analysis.pdf
```

In [ ]: