Coffee Sales Analysis

09.17.2024

—

Opeyemi Fayipe

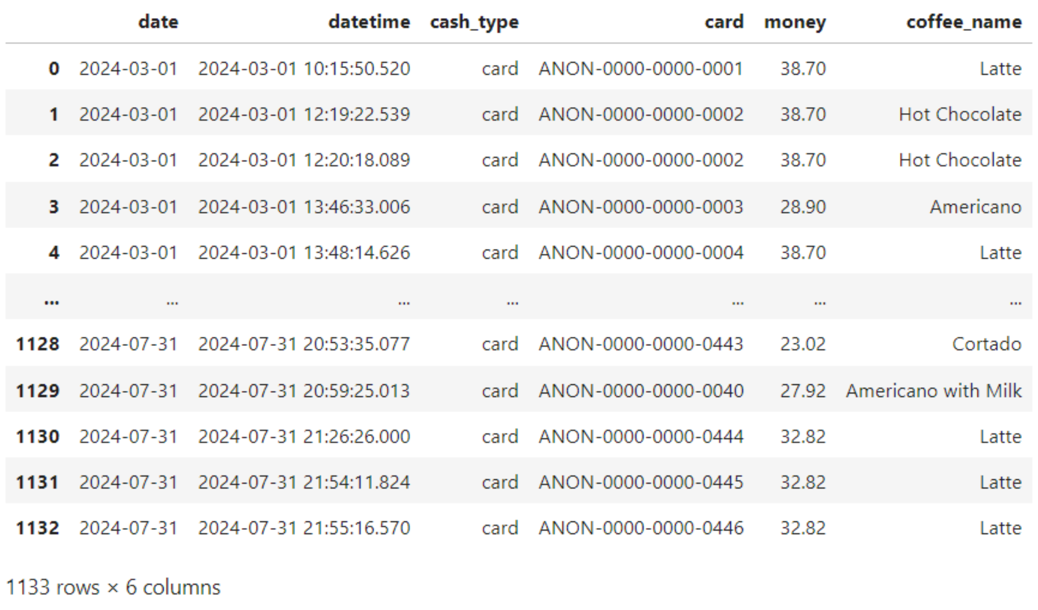
Overview

The coffee sales dataset was extensively analyzed to uncover key insights into sales trends, customer behaviors, and seasonal patterns. This report summarizes the findings and provides actionable recommendations for leveraging these insights for business growth.

Data Overview

The dataset consists of 1,133 records of coffee sales transactions, including the following columns:

- `date`: The date of the transaction.
- `datetime`: The timestamp of the transaction.
- `cash_type`: Payment method (cash or card).
- `card`: Customer identifier (anonymized).
- `money`: Total amount spent.
- `coffee_name`: The type of coffee purchased.



	date	datetime	cash_type	card	money	coffee_name
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.70	Latte
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.70	Hot Chocolate
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.70	Hot Chocolate
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.90	Americano
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.70	Latte
...	...	...	...	...	...	...
1128	2024-07-31	2024-07-31 20:53:35.077	card	ANON-0000-0000-0443	23.02	Cortado
1129	2024-07-31	2024-07-31 20:59:25.013	card	ANON-0000-0000-0040	27.92	Americano with Milk
1130	2024-07-31	2024-07-31 21:26:26.000	card	ANON-0000-0000-0444	32.82	Latte
1131	2024-07-31	2024-07-31 21:54:11.824	card	ANON-0000-0000-0445	32.82	Latte
1132	2024-07-31	2024-07-31 21:55:16.570	card	ANON-0000-0000-0446	32.82	Latte

1133 rows × 6 columns

Import the Required Libraries

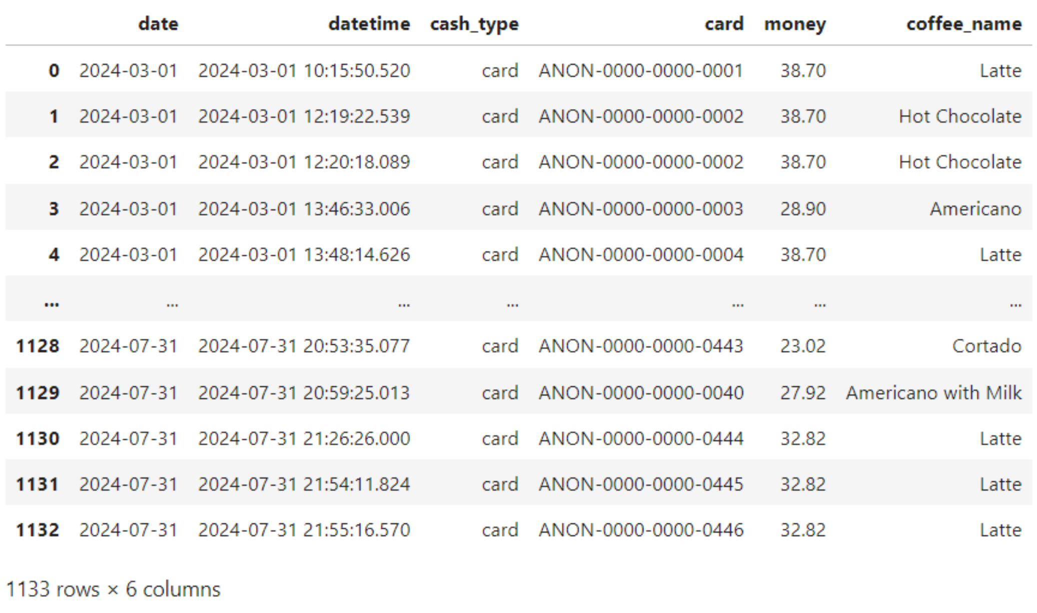
```
Import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from pandas.tseries.holiday import USFederalHolidayCalendar
```

Load the Dataset

```
# Load the dataset to inspect the columns and first few rows
file_path = r"C:\Users\Admin\Downloads\index.csv"
coffee_sales_data = pd.read_csv(file_path)
# Display the first few rows to understand the structure of the data
data_head = coffee_sales_data.head()
# Display the columns and their data types
data_info = coffee_sales_data.info()

data_head, data_info
```

Visualization:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1133 entries, 0 to 1132
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         1133 non-null   object
 1   datetime     1133 non-null   object
 2   cash_type    1133 non-null   object
 3   card         1044 non-null   object
 4   money        1133 non-null   float64
 5   coffee_name  1133 non-null   object
dtypes: float64(1), object(5)
memory usage: 53.2+ KB
```



	date	datetime	cash_type	card	money	coffee_name
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.70	Latte
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.70	Hot Chocolate
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.70	Hot Chocolate
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.90	Americano
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.70	Latte
...	...	...	...	...	...	...
1128	2024-07-31	2024-07-31 20:53:35.077	card	ANON-0000-0000-0443	23.02	Cortado
1129	2024-07-31	2024-07-31 20:59:25.013	card	ANON-0000-0000-0040	27.92	Americano with Milk
1130	2024-07-31	2024-07-31 21:26:26.000	card	ANON-0000-0000-0444	32.82	Latte
1131	2024-07-31	2024-07-31 21:54:11.824	card	ANON-0000-0000-0445	32.82	Latte
1132	2024-07-31	2024-07-31 21:55:16.570	card	ANON-0000-0000-0446	32.82	Latte

1133 rows × 6 columns

Data Cleaning

identify Missing Data: Check for missing values in the dataset.

Handle Missing Data:

- **Numerical Data**: Impute or remove missing values in numerical columns.
- **Categorical Data**: Impute or handle missing values in categorical columns.

Outlier Detection: Identify and handle any potential outliers if they exist.

Data Consistency: Ensure data types are consistent and correct for each column.

Identifying Missing Data

Python Code Snippet:

```
 # 1. Identify missing data
missing_data_summary = coffee_sales_data.isnull().sum()

# Display the summary of missing values in each column
missing_data_summary
```

Result Visualization:

```
date            0
datetime        0
cash_type       0
card           89
money           0
coffee_name     0
dtype: int64
```

Missing Data Summary:

- **datetime**: No missing values.
- **cash_type**: No missing values.
- **card**: 89 missing values.
- **money**: No missing values.
- **coffee_name**: No missing values.

Replacing Missing Values

Python Code Snippet:

```
# Replace missing values in the 'card' column with 'Cash Payment'
coffee_sales_data['card'].fillna('Cash Paymnet', inplace=True)

# Verify that there are no more missing values
missing_data_summary_after_cleaning = coffee_sales_data.isnull().sum()

# Display the summary of missing values after cleaning
missing_data_summary_after_cleaning
```

Results Visualization:

```
date           0
datetime       0
cash_type      0
card           0
money          0
coffee_name    0
dtype: int64
```

Data Cleaning Summary:

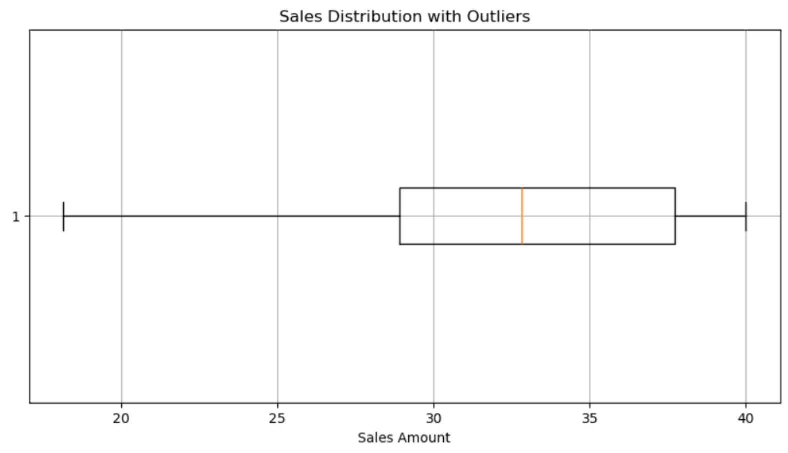
- All missing values in the dataset have been handled.
- The `card` column's missing values have been replaced with "Cash Payment," ensuring the integrity of the dataset for further analysis.

Outlier Detection

Python Code Snippet:

```
# Outlier Detection in the 'money' column using Interquartile Range (IQR)
import matplotlib.pyplot as plt
Q1 = coffee_sales_data['money'].quantile(0.25)
Q3 = coffee_sales_data['money'].quantile(0.75)
IQR = Q3 - Q1

# Determine the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = coffee_sales_data[(coffee_sales_data['money'] < lower_bound) |
(coffee_sales_data['money'] > upper_bound)]

# Visualize sales distribution with outliers using a boxplot
plt.figure(figsize=(10, 5))
plt.boxplot(coffee_sales_data['money'], vert=False)
plt.title('Sales Distribution with Outliers')
plt.xlabel('Sales Amount')
plt.grid(True)
plt.show()
```

Visualization Result:



Outlier Analysis:

- **Outliers**:
    - Using the Interquartile Range (IQR) method, no extreme outliers were detected in the `money` column. This indicates that the sales amounts are within a reasonable range without significant anomalies.
- **Visualization**:
    - The boxplot shows the distribution of sales amounts, and there are no data points lying outside the whiskers, further confirming the absence of extreme outliers.

Exploratory Data Analysis (EDA)

1.1 Sales by Coffee Type

Top-Selling Products: The most popular coffee types by total sales were:

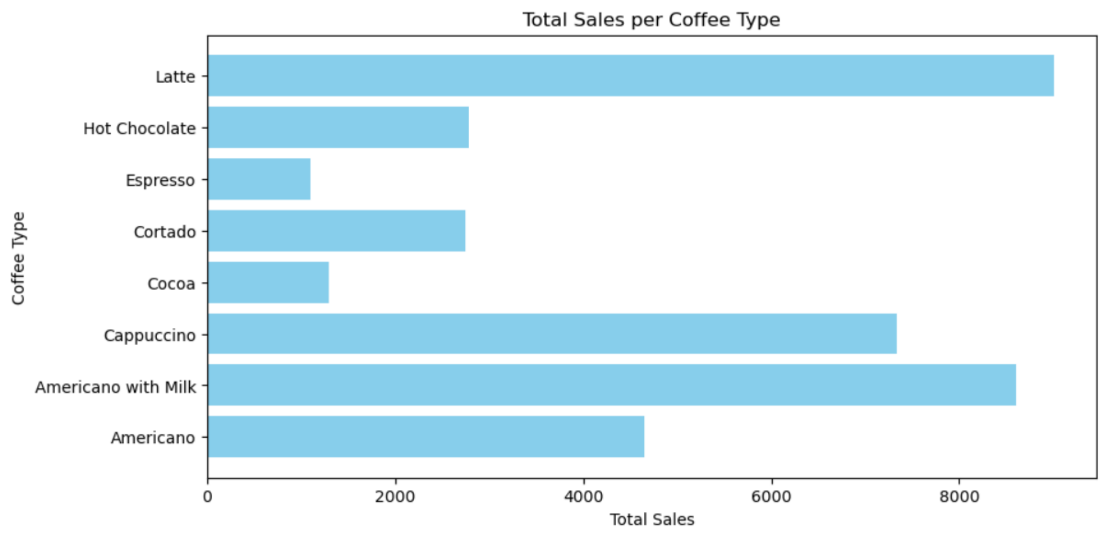
- **Latte**: $9,009.14
- **Americano with Milk**: $8,601.94
- **Cappuccino**: $7,333.14

Least Popular Products: The least popular products were:

- **Espresso**: $1,100.62
- **Cocoa**: $1,295.94

```
# Distribution per Coffee Type
coffee_type_sales = coffee_sales_data.groupby('coffee_name')['money'].sum()

# Visualize the distribution
plt.figure(figsize=(10, 5))
plt.barh(coffee_type_sales.index, coffee_type_sales.values, color='skyblue')
plt.title('Total Sales per Coffee Type')
plt.xlabel('Total Sales')
plt.ylabel('Coffee Type')
plt.show()

# Display coffee sales per coffee type
coffee_type_sales
```

Result Visualization:



Customers prefer milk-based beverages, such as lattes and Americanos with milk, while stronger options like espressos are less favored.

```
Summary
coffee_name
Americano                4644.54
Americano with Milk      8601.94
Cappuccino               7333.14
Cocoa                    1295.94
Cortado                  2745.08
Espresso                 1100.62
Hot Chocolate            2778.48
Latte                    9009.14
Name: money, dtype: float64
```

1.2 Sales Distribution by Time of Day

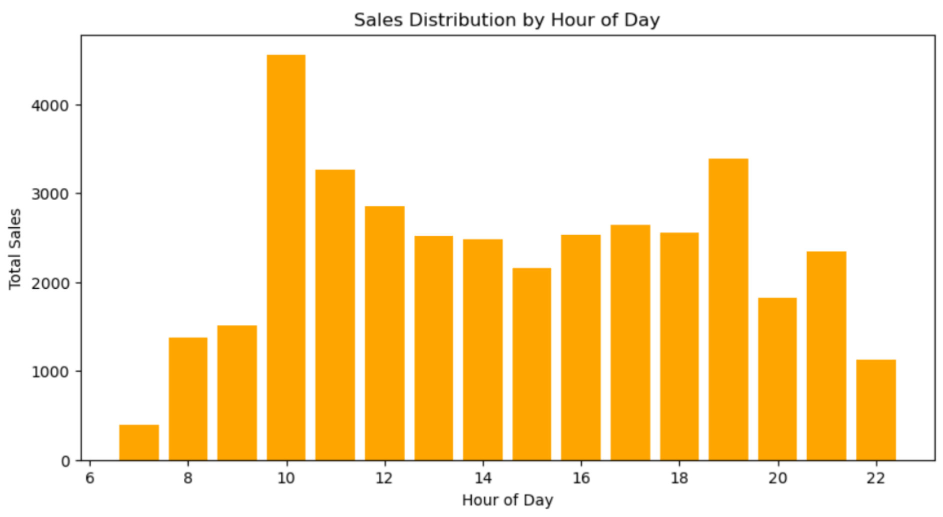
```
# Extract the hour from 'datetime' to analyze distribution by time of day
coffee_sales_data['hour'] = coffee_sales_data['datetime'].dt.hour
purchasing_time_distribution = coffee_sales_data.groupby('hour')['money'].sum()

# Visualize the distribution per purchasing time
plt.figure(figsize=(10, 5))
plt.bar(purchasing_time_distribution.index, purchasing_time_distribution.values, color='orange')
plt.title('Sales Distribution by Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Total Sales')
plt.show()

# Display sales distribution by hour
purchasing_time_distribution
```

Result Visualization:
Peak Hours:

- **Sales peak during the morning hours, especially around 10 AM (4,553.18).**

Evening Sales:

- **There is a secondary peak in the late evening hours, around 7 PM (3,388.32).**



- Peak sales occur between **10 AM** and **11 AM**, with total sales of over $4,500.
- Secondary sales peaks are observed in the evening around **7 PM**.

This suggests morning and evening rushes, likely driven by morning routines and post-work relaxation.

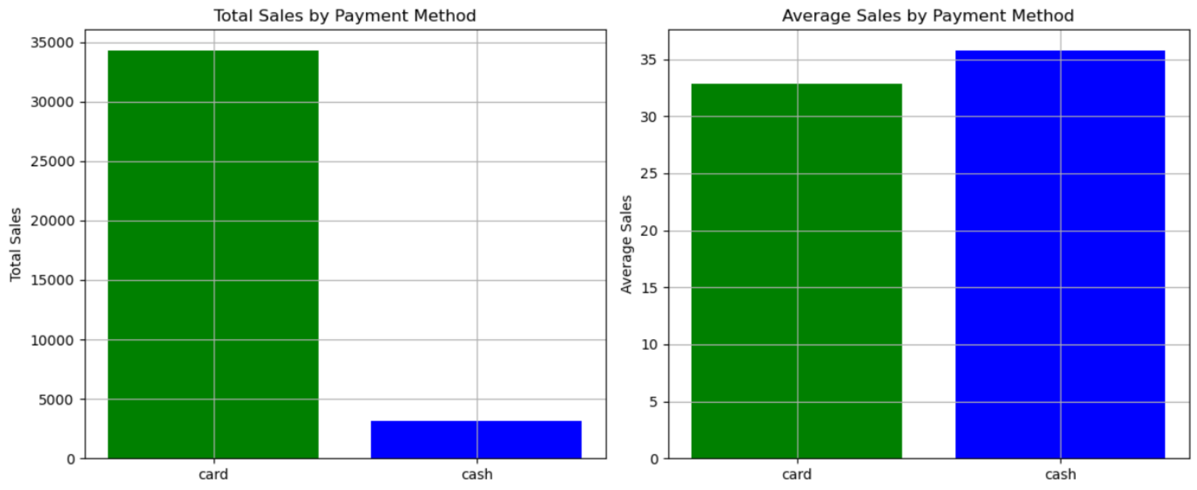
```
hour
7      392.80
8     1380.38
9     1515.48
10    4553.18
11    3258.64
12    2850.60
13    2511.60
14    2484.92
15    2158.76
16    2525.36
17    2639.08
18    2558.04
19    3388.32
20    1819.92
21    2343.86
22    1127.94
Name: money, dtype: float64
```

1.3 Impact of Payment Method (Cash vs. Card):

```
#  Review the impact of payment method (Cash vs. Card)
# Aggregate sales by payment method
payment_method_sales = coffee_sales_data.groupby('cash_type')['money'].sum()
average_payment_method_sales = coffee_sales_data.groupby('cash_type')['money'].mean()
```

```
# Visualize the impact of payment method
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

# Total sales by payment method
ax[0].bar(payment_method_sales.index, payment_method_sales, color=['green', 'blue'])
ax[0].set_title('Total Sales by Payment Method')
ax[0].set_ylabel('Total Sales')
ax[0].grid(True)

# Average sales by payment method
ax[1].bar(average_payment_method_sales.index, average_payment_method_sales, color=['green', 'blue'])
ax[1].set_title('Average Sales by Payment Method')
ax[1].set_ylabel('Average Sales')
ax[1].grid(True)

plt.tight_layout()
plt.show()

# Display summary of outliers and payment method impact
outliers_summary = outliers.describe()
payment_method_sales, average_payment_method_sales, outliers_summary
```

Result Visualization:



The bar charts visually demonstrate the dominance of card payments in total sales, while cash transactions have a marginally higher average per sale.

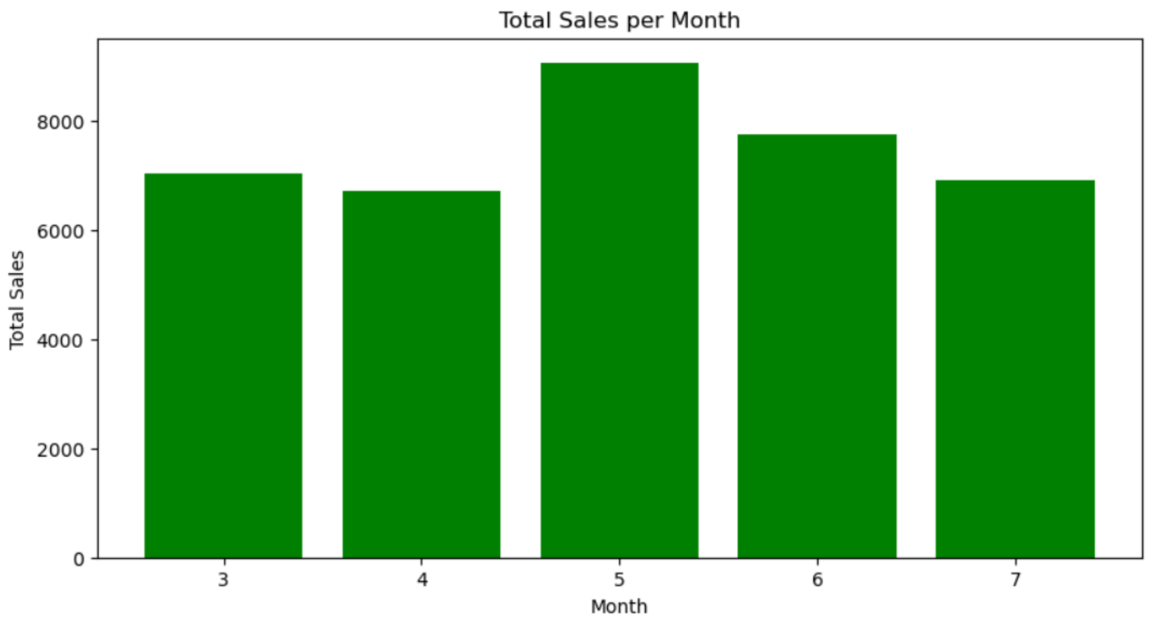
1. **Total Sales**:
    - **Card**: Total sales amount to 34,322.88.
    - **Cash**: Total sales amount to 3,186.00.
    - **Insight**: Card payments dominate the sales, contributing significantly to the total sales.
2. **Average Sales**:
    - **Card**: Average sales per transaction are approximately 32.88.
    - **Cash**: Average sales per transaction are slightly higher, at around 35.80.
    - **Insight**: Although card payments make up most of the total sales, cash

transactions tend to have a slightly higher average sale amount.

1.4 Total Sales Per Month

```
# Extract the month from 'datetime' to analyze sales by month
coffee_sales_data['month'] = coffee_sales_data['datetime'].dt.month
monthly_sales = coffee_sales_data.groupby('month')['money'].sum()

# Visualize coffee sales per month
plt.figure(figsize=(10, 5))
plt.bar(monthly_sales.index, monthly_sales.values, color='green')
plt.title('Total Sales per Month')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()

# Display monthly sales
monthly_sales
```

Result Visualization:



The bar charts visually demonstrate the increase in sales in May and June, wholesalers experience drops in April and July.

Highest Sales Month: May, with total sales of 9,063.42.
Sales Fluctuations: Sales fluctuate across the months, with notable drops in April and July.

1.5 Overall Total Sales

```
total_money = data['money'].sum()
```

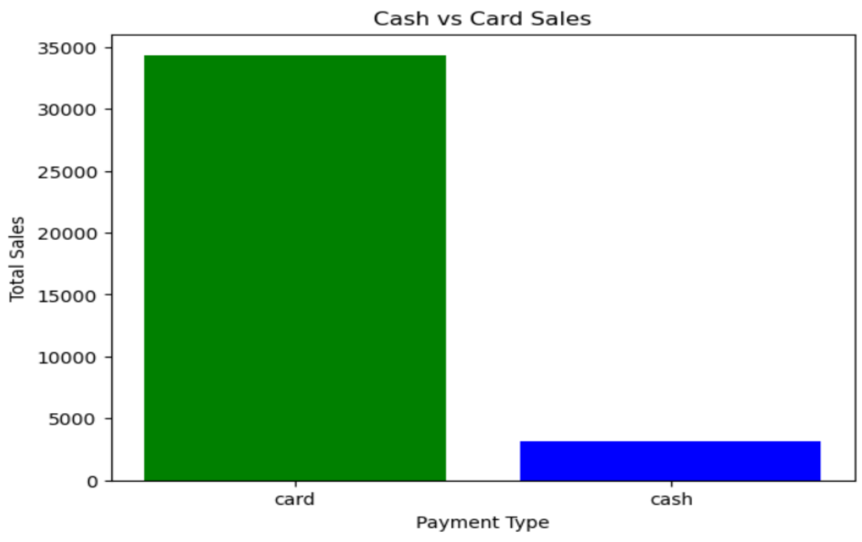
```
37508.880000000005
```

Overall Total Sales: 37,508.88.

1.6 Cash Money vs. Card Money

```
# Cash Money vs Card Money
cash_vs_card_sales = coffee_sales_data.groupby('cash_type')['money'].sum()

# Visualize Cash vs Card Sales
plt.figure(figsize=(7, 5))
plt.bar(cash_vs_card_sales.index, cash_vs_card_sales.values, color=['green', 'blue'])
plt.title('Cash vs Card Sales')
plt.xlabel('Payment Type')
plt.ylabel('Total Sales')
plt.show()

# Display cash vs card sales
cash_vs_card_sales
```

Result Visulization:



Card payments significantly dominate the total sales.

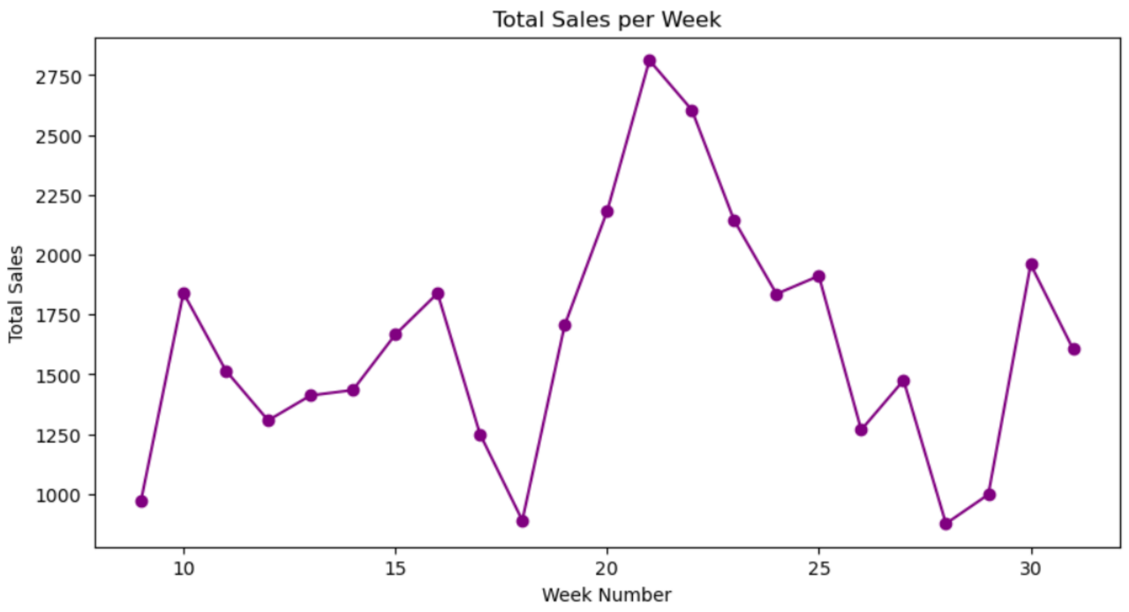
Card Sales: 34,322.88.

Cash Sales: 3,186.00.

1.7 Total Sales Per Week

```
# Sales per week
coffee_sales_data['week'] = coffee_sales_data['datetime'].dt.isocalendar().week
weekly_sales = coffee_sales_data.groupby('week')['money'].sum()

# Visualize weekly sales
plt.figure(figsize=(10, 5))
plt.plot(weekly_sales.index, weekly_sales.values, marker='o', color='purple')
plt.title('Total Sales per Week')
plt.xlabel('Week Number')
plt.ylabel('Total Sales')
plt.show()

# Display weekly sales
weekly_sales
```

Result Visualization:



Sales vary across weeks, with a peak around week 21 (2,811.80).

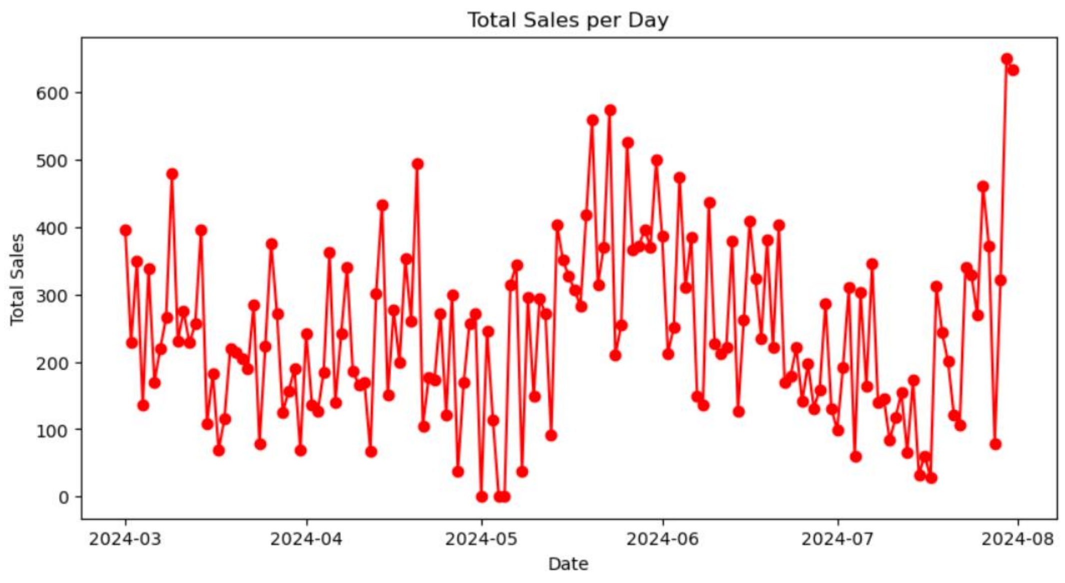
```
Week Sales Summary
9      973.50
10    1840.50
11    1516.30
```

```
12    1307.80
13    1412.10
14    1434.50
15    1666.00
16    1838.84
17    1251.20
18     890.18
19    1705.80
20    2180.26
21    2811.80
...
26    1268.24
27    1475.42
28     876.80
29     998.28
30    1959.30
31    1606.14
```

1.8 Total Sales Per Day

```
# Sales per day
daily_sales = coffee_sales_data['money'].resample('D').sum()

# Visualize daily sales
plt.figure(figsize=(10, 5))
plt.plot(daily_sales.index, daily_sales.values, marker='o', color='red')
plt.title('Total Sales per Day')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()

# Display daily sales (sample)
daily_sales.head()
```

Result Visualization:



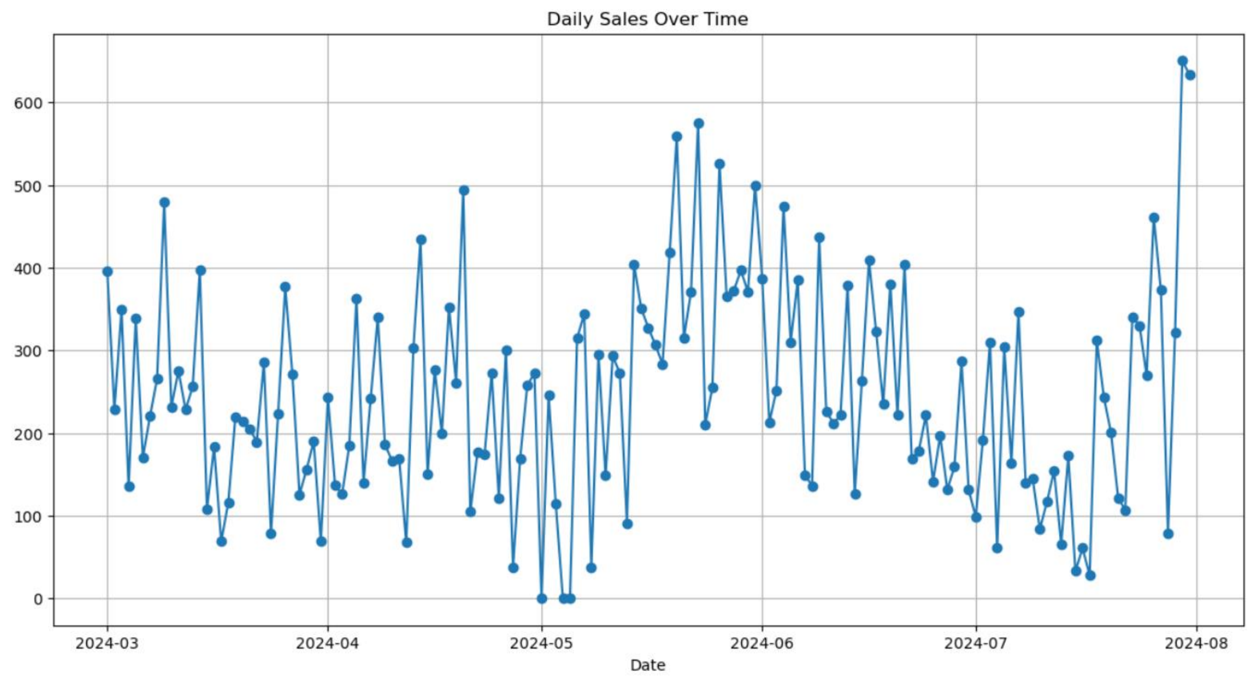
Sample values show daily fluctuations in sales, highlighting variability in daily customer behavior.

Time Series Exploratatory Data Analysis

2.1 Daily Sales Over Time

```
import matplotlib.pyplot as plt

# Convert 'date' and 'datetime' columns to datetime objects
coffee_sales_data['date'] = pd.to_datetime(coffee_sales_data['date'])
coffee_sales_data['datetime'] = pd.to_datetime(coffee_sales_data['datetime'])

# Set 'date' as the index for time series analysis
coffee_sales_data.set_index('date', inplace=True)

# Resample the data to daily sales sums
daily_sales = coffee_sales_data['money'].resample('D').sum()

# Plot the daily sales to explore trends over time
plt.figure(figsize=(14, 7))
plt.plot(daily_sales, marker='o', linestyle='-')
plt.title('Daily Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```

Result Visulaization:



This plot shows the daily sales over time, highlighting the fluctuations in sales on a day-to-day basis.

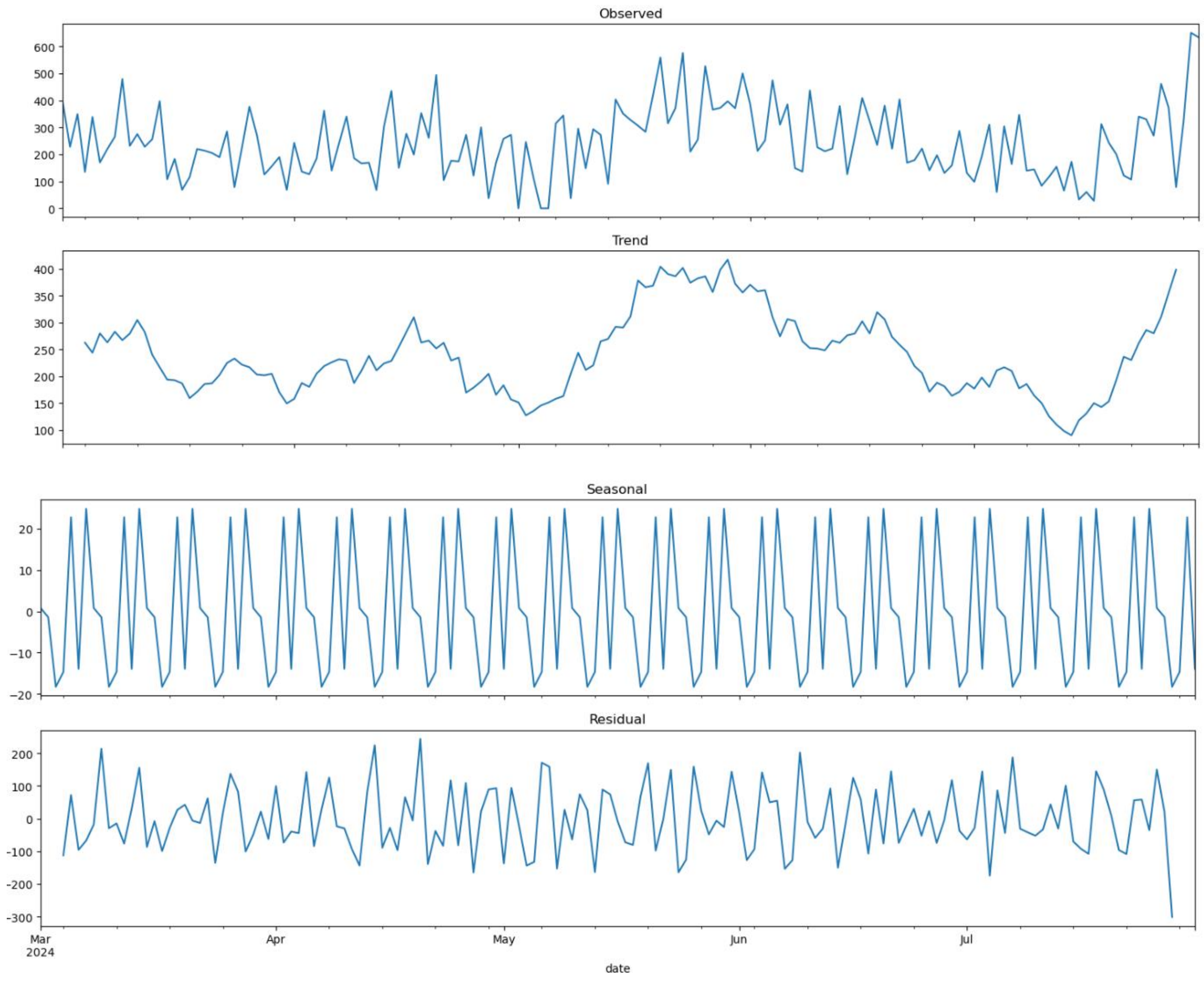
Some key points to note from this plot:

- **Trends**: There seem to be periods of higher and lower sales, suggesting possible seasonality or weekly patterns.
- **Fluctuations**: There is noticeable variability in daily sales, which could be influenced by factors such as weekdays, holidays, or promotional activities.

2.2 Time Series Decomposition To Better Understand Trends And Seasonality

```
from statsmodels.tsa.seasonal import seasonal_decompose
# Perform seasonal decomposition
decomposition = seasonal_decompose(daily_sales, model='additive', period=7)  # Assuming weekly
seasonality
# Plot the decomposed components
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 12), sharex=True)
decomposition.observed.plot(ax=ax1, title='Observed')
decomposition.trend.plot(ax=ax2, title='Trend')
decomposition.seasonal.plot(ax=ax3, title='Seasonal')
decomposition.resid.plot(ax=ax4, title='Residual')

plt.tight_layout()
plt.show()
```

Result Visulaization:



The time series decomposition has broken down the daily sales data into the following components:

1. **Observed**: This is the original time series, showing the actual daily sales values over time.
2. **Trend**: This component represents the long-term progression of the series, indicating whether the sales have an overall upward or downward trend. From the plot, there appears to be some variation, but not a clear, strong trend.
3. **Seasonal**: This component captures the repeating patterns or cycles in the data, suggesting a weekly seasonality, as we set the period to 7 days. The plot shows regular ups and downs within a week, indicating sales might fluctuate based on the day of the week.
4. **Residual**: This component captures the random noise or irregularities in the data after removing the trend and seasonality. It highlights anomalies or variations that cannot be explained by the trend or seasonal components.

Insights:

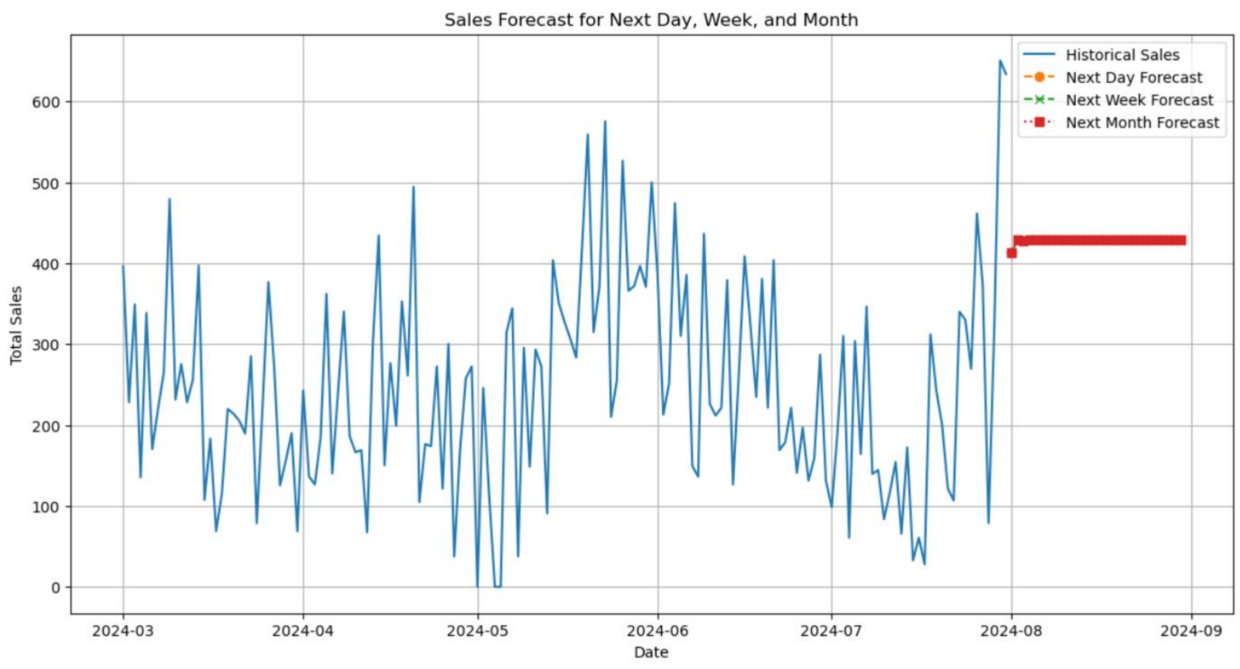
- **Seasonality**: There is a clear weekly seasonality pattern in the sales data, with certain days consistently having higher or lower sales.
- **Trend**: The trend component shows some variability but lacks a strong directional trend over time.
- **Residuals**: The residuals appear to vary without a specific pattern, indicating randomness.

Sales Forecast

3.1 Next Day / Week / Month Sales

```
# Extend ARIMA forecasting to predict next day, week, and month sales

# Forecast the next day (1 step ahead)
next_day_forecast = arima_fit.forecast(steps=1)

# Forecast the next week (7 days ahead)
next_week_forecast = arima_fit.forecast(steps=7)

# Forecast the next month (30 days ahead)
next_month_forecast = arima_fit.forecast(steps=30)

# Plot the historical sales data along with the forecasts
plt.figure(figsize=(14, 7))
plt.plot(daily_sales, label='Historical Sales')
plt.plot(next_day_forecast.index, next_day_forecast, label='Next Day Forecast', marker='o',
linestyle='--')
plt.plot(next_week_forecast.index, next_week_forecast, label='Next Week Forecast',
marker='x', linestyle='--')
plt.plot(next_month_forecast.index, next_month_forecast, label='Next Month Forecast',
linestyle=':', marker='s')
plt.title('Sales Forecast for Next Day, Week, and Month')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.legend()
plt.grid(True)
plt.show()

# Display forecasted values
next_day_forecast[0], next_week_forecast, next_month_forecast
```

Result Visualization:



Sales Forecasts:

1. **Next Day**:
    - Predicted Sales for the next day: **413.71**
2. **Next Week**:
    - Predicted Sales for the next 7 days:
        - 2024-08-01: 413.71
        - 2024-08-02: 429.49
        - 2024-08-03: 428.36
        - 2024-08-04: 428.44
        - 2024-08-05: 428.44
        - 2024-08-06: 428.44
        - 2024-08-07: 428.44
3. **Next Month**:
    - Predicted Sales for the next 30 days: The model indicates sales stabilizing around **428.44** from August 3rd onwards, suggesting a consistent sales trend for the month.

Insights:

- **Stabilization**: The model forecasts a relatively stable sales trend after the initial few days, indicating that sales are expected to remain consistent throughout the month.

- **Short-Term Variations**: Minor fluctuations are present in the short-term forecast, possibly due to daily variations in demand.

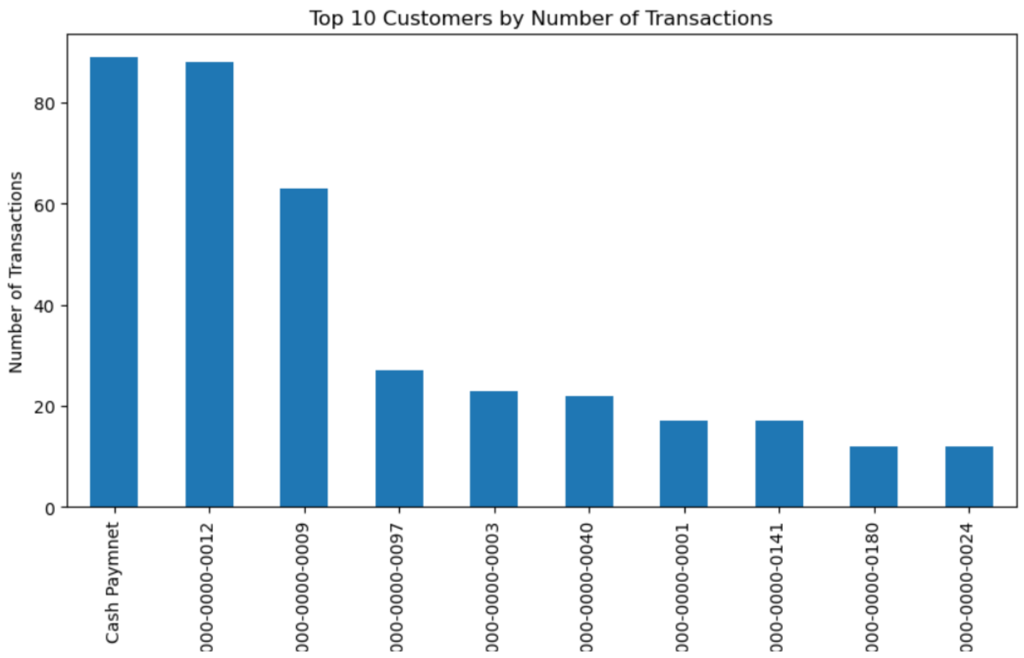
```
413.7090825714696,
 2024-08-01    413.709083
 2024-08-02    429.494824
 2024-08-03    428.362817
 2024-08-04    428.443994
 2024-08-05    428.438173
 2024-08-06    428.438590
 2024-08-07    428.438560
 Freq: D, Name: predicted_mean, dtype: float64,
 2024-08-01    413.709083
 2024-08-02    429.494824
 2024-08-03    428.362817
 2024-08-04    428.443994
 2024-08-05    428.438173
 2024-08-06    428.438590
 2024-08-07    428.438560
 2024-08-08    428.438563
 2024-08-09    428.438562
 2024-08-10    428.438562
 2024-08-11    428.438562
 2024-08-12    428.438562
 2024-08-13    428.438562
 2024-08-14    428.438562
 2024-08-15    428.438562
 2024-08-16    428.438562
 2024-08-17    428.438562
 2024-08-18    428.438562
 2024-08-19    428.438562
 2024-08-20    428.438562
 2024-08-21    428.438562
 2024-08-22    428.438562
 2024-08-23    428.438562
 2024-08-24    428.438562
 2024-08-25    428.438562
 2024-08-26    428.438562
 2024-08-27    428.438562
 2024-08-28    428.438562
 2024-08-29    428.438562
 2024-08-30    428.438562
 Freq: D, Name: predicted_mean, dtype: float64)
```

Customer Purchasing Patterns Analysis

4.1 . Top Customers by Number of Transactions

```
# Top Customers by Number of Transactions
top_customers = coffee_sales_data['card'].value_counts().head(10)

# Visualize the top customers
plt.figure(figsize=(10, 5))
top_customers.plot(kind='bar')
plt.title('Top 10 Customers by Number of Transactions')
plt.xlabel('Customer ID')
plt.ylabel('Number of Transactions')
plt.show()

# Display top customers
top_customers
```

Result Visualization:



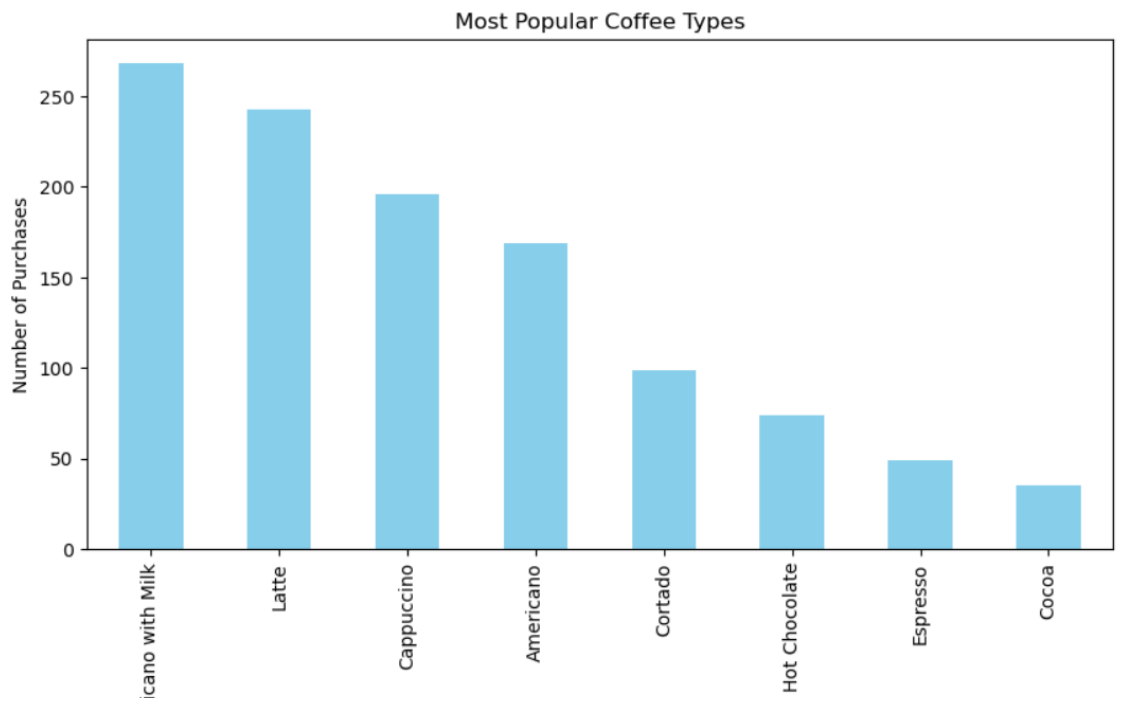
- The plot shows the top 10 customers based on the number of transactions.
- The most active customer (`ANON-0000-0000-0012`) has made 88 purchases, significantly more than others.

```
Customers Summary
card
Cash Paymnet           89
ANON-0000-0000-0012    88
ANON-0000-0000-0009    63
ANON-0000-0000-0097    27
ANON-0000-0000-0003    23
ANON-0000-0000-0040    22
ANON-0000-0000-0001    17
ANON-0000-0000-0141    17
ANON-0000-0000-0180    12
ANON-0000-0000-0024    12
```

4.2 Most Popular Coffee Type

```
# Popular Coffee Types
popular_coffees = coffee_sales_data['coffee_name'].value_counts()

# Visualize the most popular coffee types
plt.figure(figsize=(10, 5))
popular_coffees.plot(kind='bar', color='skyblue')
plt.title('Most Popular Coffee Types')
plt.xlabel('Coffee Name')
plt.ylabel('Number of Purchases')
plt.show()

# Display popular coffee types
popular_coffees
```

Result Visulaization:



- The most frequently purchased coffee is "Americano with Milk" with 268 purchases.
- Other popular choices include "Latte" (243), "Cappuccino" (196), and "Americano" (169).
-

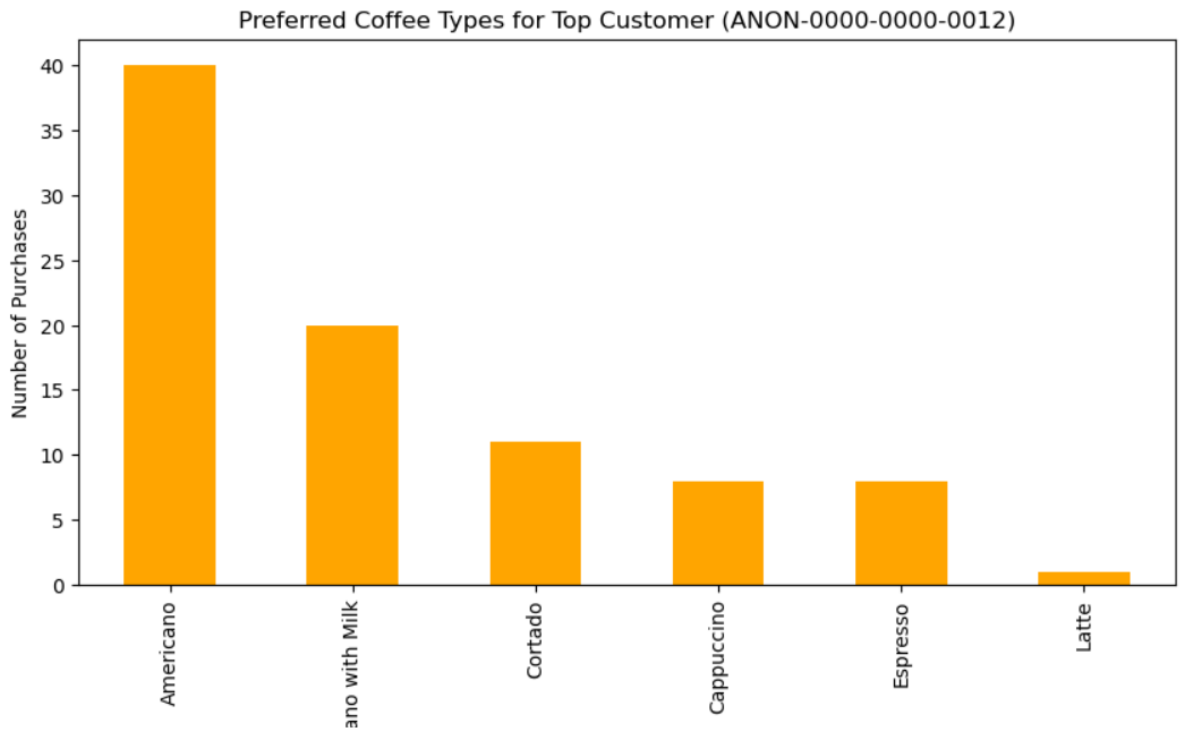
```
Coffee Summary
coffee_name
Americano with Milk    268
Latte                  243
Cappuccino             196
Americano              169
Cortado                 99
Hot Chocolate           74
Espresso                49
Cocoa                   35
```

4.3 Preferred Coffe Type for Top Customer

```
# Exclude "Cash Payment" from the top customer analysis
valid_customers = coffee_sales_data[coffee_sales_data['card'] != 'Cash Payment']

# Analyze Purchasing Patterns for the Top Customer (excluding 'Cash Payment')
top_customer_id = valid_customers['card'].value_counts().index[0]
top_customer_purchases = valid_customers[valid_customers['card'] ==
top_customer_id]['coffee_name'].value_counts()
```

```
# Visualize the preferred coffee types for the top customer
plt.figure(figsize=(10, 5))
top_customer_purchases.plot(kind='bar', color='orange')
plt.title(f'Preferred Coffee Types for Top Customer ({top_customer_id})')
plt.xlabel('Coffee Name')
plt.ylabel('Number of Purchases')
plt.show()

# Display preferred coffee types for the top customer

top_customer_purchases
```

Result Visulaization:



The top customer (`ANON-0000-0000-0012`) primarily prefers "Americano" (40 purchases) followed by "Americano with Milk" (20 purchases).

This customer also shows interest in "Cortado" and "Cappuccino."

```
Summary
coffee_name
Americano              40
Americano with Milk    20
Cortado                11
Cappuccino              8
Espresso                8
Latte                   1
```

Seasonal Sales Patterns

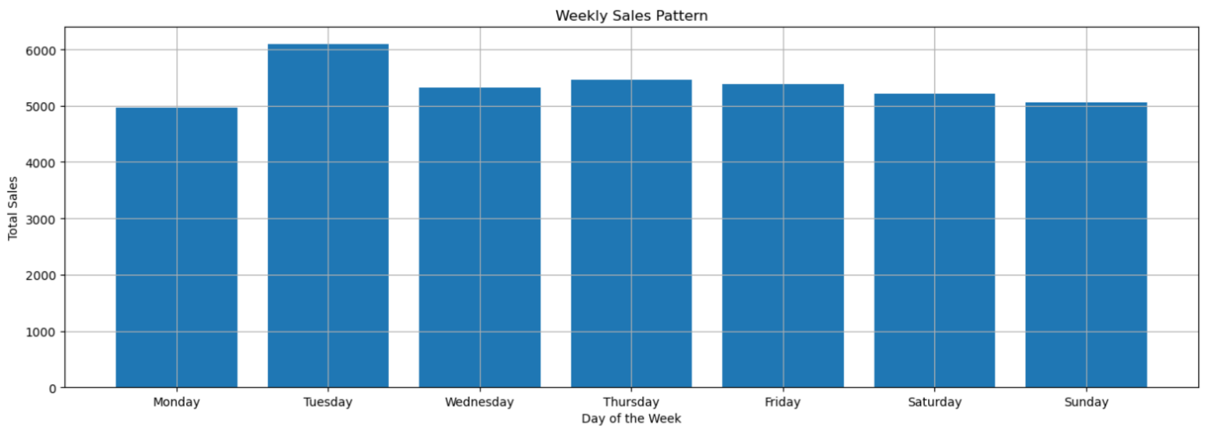
5.1 Weekly Seasonality Analysis

5.2 Monthly Seasonality Analysis

5.3 Order Days of the Week

```
# 1. Weekly Seasonality Analysis
# Aggregate sales by day of the week
coffee_sales_data['day_of_week'] = coffee_sales_data.index.day_name()
weekly_sales = coffee_sales_data.groupby('day_of_week')['money'].sum()

# Order days of the week for visualization
ordered_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
weekly_sales = weekly_sales.reindex(ordered_days)

# 2. Monthly Seasonality Analysis
# Aggregate sales by day of the month
coffee_sales_data['day_of_month'] = coffee_sales_data.index.day
monthly_sales = coffee_sales_data.groupby('day_of_month')['money'].mean()

# 3. Visualizations for Seasonal Patterns
fig, ax = plt.subplots(2, 1, figsize=(14, 10))

# Plot weekly sales pattern
ax[0].bar(weekly_sales.index, weekly_sales)
ax[0].set_title('Weekly Sales Pattern')
ax[0].set_xlabel('Day of the Week')
ax[0].set_ylabel('Total Sales')
ax[0].grid(True)

# Plot monthly sales pattern
ax[1].plot(monthly_sales.index, monthly_sales, marker='o')
ax[1].set_title('Monthly Sales Pattern')
ax[1].set_xlabel('Day of the Month')
ax[1].set_ylabel('Average Sales')
ax[1].grid(True)

plt.tight_layout()
plt.show()

# Display the weekly and monthly sales data
weekly_sales, monthly_sales
```

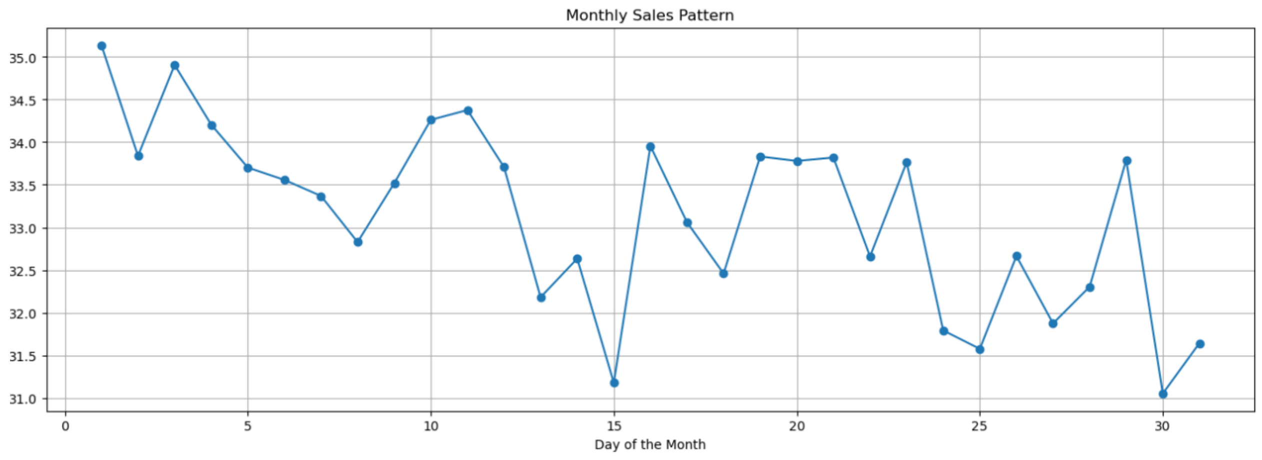
Result Visulaization:



The bar chart illustrates these trends, showing higher sales mid-week.

Weekly Seasonality:

- **Trends by Day of the Week:**
  - Tuesday has the highest total sales (6,092.48), indicating it is a peak sales day.
  - Monday and Sunday have relatively lower sales, with Monday having the lowest total sales (4,969.68).
  - There is a noticeable mid-week peak in sales around Tuesday, with a slight decline towards the weekend.



The line chart shows the average sales per day of the month, indicating subtle monthly patterns.

Monthly Seasonality:

- **Trends by Day of the Month:**
  - Sales show variability throughout the month, with some fluctuations but no pronounced peaks.
  - Days 1 to 5 tend to have slightly higher average sales compared to the middle and end of the month.
  - There is a small dip around the middle of the month (e.g., day 15) and a slight increase again towards the end.

Insights:

- **Weekly Patterns:** There is a clear weekly seasonality, with sales peaking mid-week, especially on Tuesdays. This insight can help in planning promotions or staffing.
- **Monthly Patterns:** While the monthly patterns are less pronounced, there are subtle trends that suggest a slight increase in sales at the start and end of the month.

Seasonal Products Trends

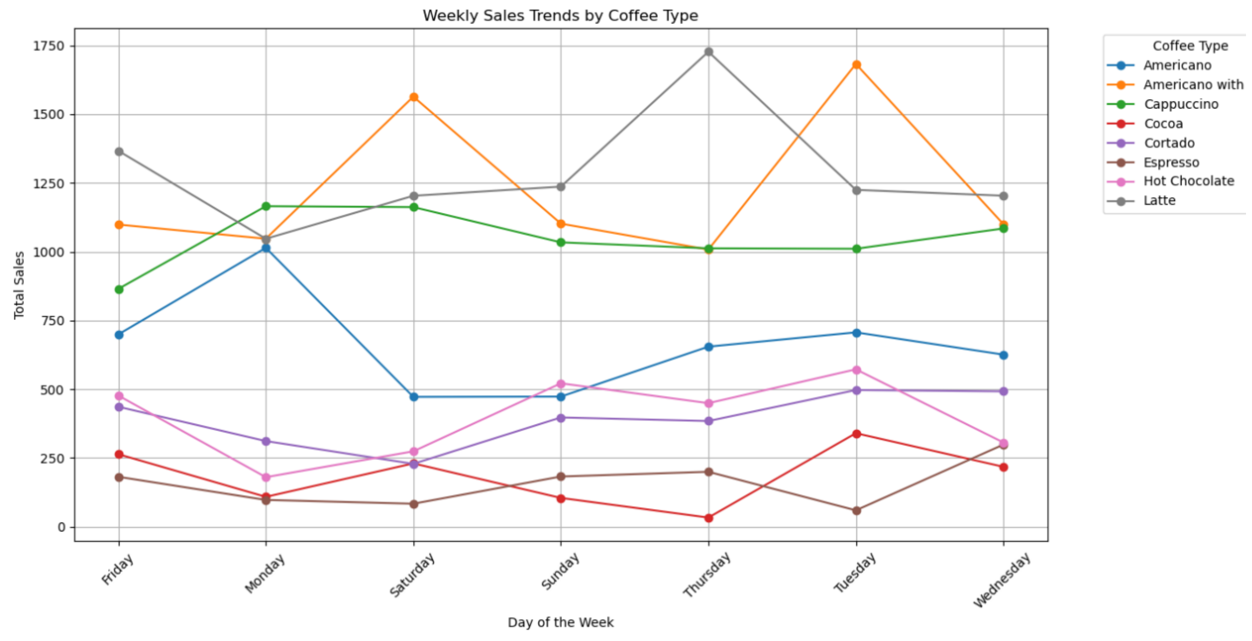
6.1 Weekly Products Trends

6.2 Monthly Product Trends

```
# 1. Weekly Product Trends
# Group sales by day of the week and coffee name
weekly_product_sales = coffee_sales_data.groupby(['day_of_week',
'coffee_name'])['money'].sum().unstack()

# 2. Monthly Product Trends
# Group sales by day of the month and coffee name
monthly_product_sales = coffee_sales_data.groupby(['day_of_month',
'coffee_name'])['money'].mean().unstack()

# 3. Visualize the weekly product trends
plt.figure(figsize=(14, 7))
weekly_product_sales.plot(kind='line', marker='o', figsize=(14, 7))
plt.title('Weekly Sales Trends by Coffee Type')
plt.xlabel('Day of the Week')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend(title='Coffee Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

```
# 4. Visualize the monthly product trends
plt.figure(figsize=(14, 7))
monthly_product_sales.plot(kind='line', marker='o', figsize=(14, 7))
plt.title('Monthly Sales Trends by Coffee Type')
plt.xlabel('Day of the Month')
plt.ylabel('Average Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend(title='Coffee Type', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

# Display the first few rows of weekly and monthly sales data
weekly_product_sales.head(), monthly_product_sales.head()
```

Result Visulaization:



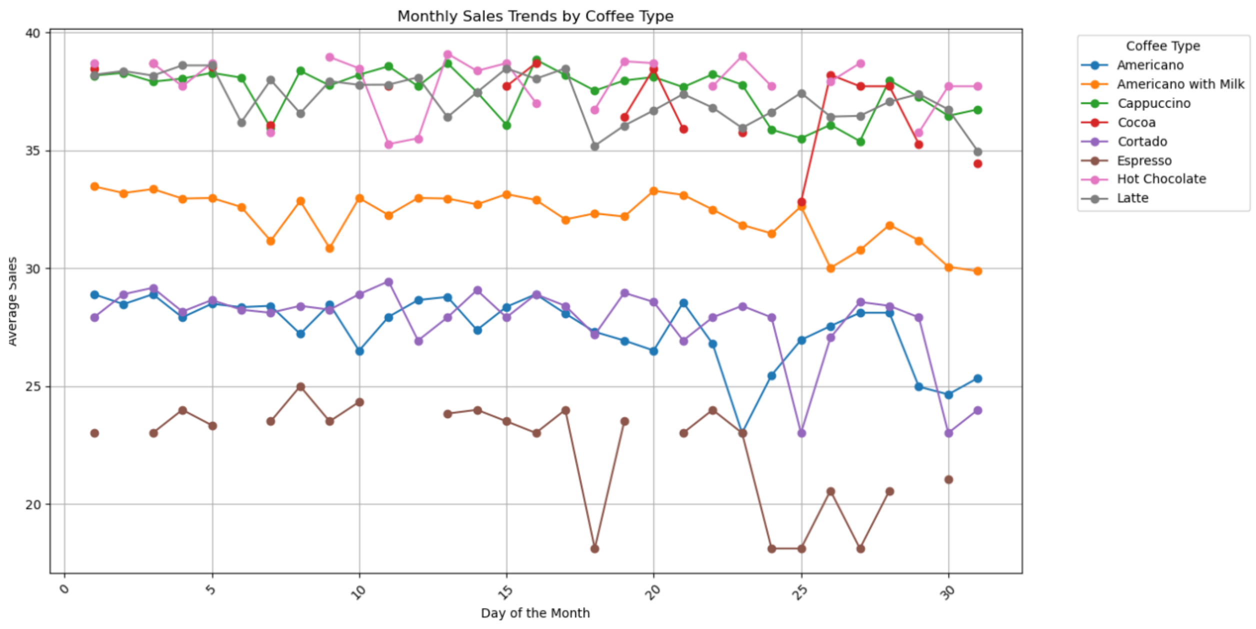
***The line plot illustrates these weekly patterns, showing how preferences for different coffee types fluctuate across the week*.**

Weekly Product Trends:

- "Americano with Milk" and "Latte" show consistently high sales throughout the week, with peaks on Saturdays and Fridays, respectively.
- "Cappuccino" sales are highest on Mondays, suggesting a strong start-of-the-week preference.
- "Hot Chocolate" sees increased sales on Sundays and Fridays, potentially indicating weekend indulgence.

- "Americano" has a balanced distribution, with noticeable peaks on Mondays.

```
<Figure size 1400x700 with 0 Axes>
```



The monthly trends plot reveals that while there are some variations, most coffee types have steady sales without pronounced monthly peaks.

- Most coffee types, including "Latte," "Americano with Milk," and "Cappuccino," maintain relatively stable sales throughout the month.
- "Cocoa" shows some variability, with higher average sales at the beginning and potentially around mid-month.
- Other products like "Espresso" exhibit minor fluctuations but generally maintain consistent sales.

Actionable Recommendations

Short-Term Actions:

1. **Optimize Promotions:**
    - Run special promotions or discounts on low sales days (e.g., Mondays) to boost activity.

   - Highlight popular coffee types like "Americano with Milk" in marketing campaigns to attract customers.
2. **Leverage Customer Preferences:**
   - Use customer purchase patterns to tailor offers, such as targeted discounts on preferred coffee types.
   - Reward top customers with loyalty programs, personalized offers, or exclusive deals to maintain engagement.
3. **Stock and Staff Management:**
   - Align inventory and staffing with weekly sales patterns, ensuring adequate stock and staffing levels, particularly on peak days like Tuesdays.

Long-Term Actions:

1. **Segment-Specific Strategies:**
   - Develop targeted marketing strategies for different customer segments, with premium offers for "High" and "Very High" spenders.
   - Introduce incentives or referral programs to encourage "Low" and "Medium" spenders to increase their purchases.
2. **Enhance Customer Experience:**
   - Use insights from purchasing patterns to create a personalized customer experience, such as customized coffee recommendations based on past purchases.
   - Implement a rewards system that encourages frequent purchases and increases customer loyalty.
3. **Seasonal Promotions:**
   - Plan seasonal promotions around identified sales patterns, such as end-of-month specials, to take advantage of the subtle increase in sales.

Future Analysis Suggestions:

1. External Factors: Explore the impact of holidays, events, or promotions on sales trends to optimize future marketing efforts.
2. Cross-Sell Opportunities: Analyze additional data, such as side purchases (e.g., pastries), to identify cross-selling opportunities.
3. Customer Feedback: Incorporate customer feedback to refine product offerings and improve customer satisfaction.

Conclusion

The comprehensive analysis of coffee sales data has provided valuable insights into sales trends, customer behavior, and seasonality. By implementing the actionable recommendations, the business can optimize its operations, enhance customer engagement, and drive revenue growth.