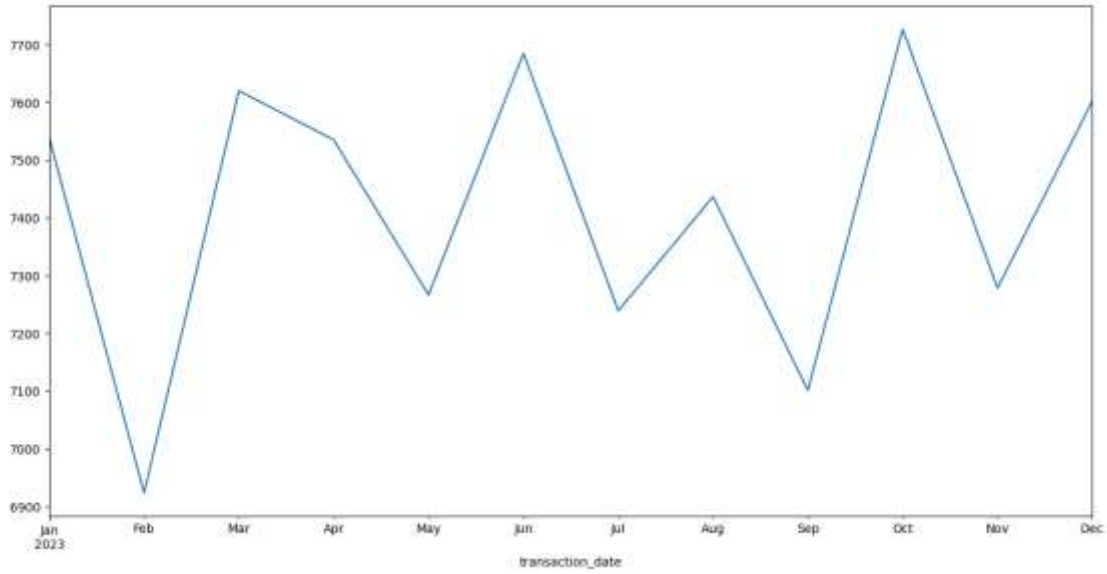
Data_cleaning project @vigneshwarreddy

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

#load the csv file
df =pd.read_csv ("C:/Users/tammi/Desktop/cafe_sales.csv")
# Standardize column names
df.columns = df.columns.str.strip().str.replace(' ', '_').str.lower()
print (df.info())
print(df.describe())
#Handle missing values and incorrect entries ("ERROR", "UNKNOWN")
# Replace "ERROR" and "UNKNOWN" with 0 for numeric columns
numeric_columns = ['quantity', 'price_per_unit_($)', 'total_spent_($)']
for col in numeric_columns:
    df[col] = df[col].replace(['ERROR', 'UNKNOWN'], 0)
    df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)

# Replace missing or "ERROR"/"UNKNOWN" in item, payment_method, location with "Unknown"

df['item'] = df['item'].replace(['ERROR', 'UNKNOWN'], 'Unknown').fillna('Unknown')
df['payment_method'] = df['payment_method'].replace(['ERROR', 'UNKNOWN'], 'Unknown').fillna('Unknown')
df['location'] = df['location'].replace(['ERROR', 'UNKNOWN'], 'Unknown').fillna('Unknown')


# transaction_date missing or null values filling with previous date
df['transaction_date'] = df['transaction_date'].ffill()
```

```
#print (df.info())
#print(df.to_string())

#converting datatypes
#---------------------

# Quantity  dtype change to float\integer
df['quantity'] = pd.to_numeric(df['quantity'], errors='coerce')

# Price_per_unit  dtype change to float
df['price_per_unit_($)'] = pd.to_numeric(df['price_per_unit_($)'], errors='coerce')

# Total_spent  dtype to change float
df['total_spent_($)'] = pd.to_numeric(df['total_spent_($)'], errors='coerce')

#transaction_date  dtype change to datetime
df['transaction_date'] = pd.to_datetime(df['transaction_date'], errors = 'coerce')

print (df.info())

#filling missing values in quantity, 'price_per_unit_($)', 'total_spent_($)'
#---------------------------------------------------------------------------

# Fill missing quantity
df['quantity'] = df['quantity'].replace(0, pd.NA)
df['quantity'] = df['quantity'].fillna(df['total_spent_($)'] / df['price_per_unit_($)'])

# Fill missing price_per_unit
df['price_per_unit_($)'] = df['price_per_unit_($)'].replace(0, pd.NA)
```

```
df['price_per_unit_($)'] = df['price_per_unit_($)'].fillna(df['total_spent_($)'] / df['quantity'])

# Fill missing total_spent
df['total_spent_($)'] = df['total_spent_($)'].replace(0, pd.NA)
df['total_spent_($)'] = df['total_spent_($)'].fillna(df['quantity'] * df['price_per_unit_($)'])

#missing or null values filling with previous date

df['transaction_date'] = df['transaction_date'].ffill()

print(df.to_string())



# Save cleaned DataFrame to same path
df.to_csv("C:/Users/tammi/Desktop/cafe_sales.csv", index=False)

# --- Correlation Analysis ---
# Select numeric columns for correlation
numeric_cols = ['quantity', 'price_per_unit_($)', 'total_spent_($)']
correlation_matrix = df[numeric_cols].corr()
print(correlation_matrix)

df.to_csv("C:/Users/tammi/Desktop/cafe_sales.csv", index=False)

#plot

#Monthly Revenue
df['transaction_date'] = pd.to_datetime(df['transaction_date'])
df.groupby(df['transaction_date'].dt.to_period("M"))['total_spent_($)'].sum().plot()
```



plt.show()

#items Sold (Top Selling Items)

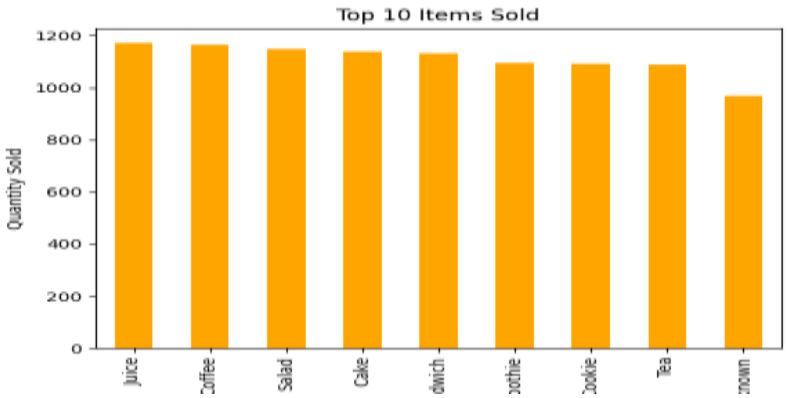
df['item'].value_counts().head(10).plot(kind='bar', color='orange')

plt.title("Top 10 Items Sold")

plt.ylabel("Quantity Sold")

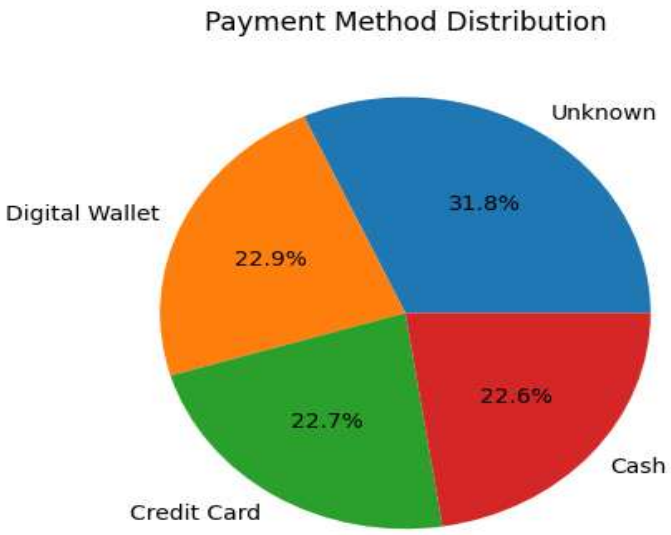
plt.xlabel("Item")

plt.show()



#Payment Methods Used

```
df['payment_method'].value_counts().plot(kind='pie', autopct='%1.1f%%')

plt.title("Payment Method Distribution")

plt.ylabel("")

plt.show()
```



--- Strategies and Insights to Improve Sales and Revenue ---

```
print("1. **Optimize Peak Periods**: Increase staffing and promotions during high-sales months and days (e.g., busiest days identified in the analysis)")

print("2. **Promote Best-Selling Items**: Focus marketing on top items (e.g., Sandwiches, Smoothies) through menu highlights or loyalty programs.")

print("3. **Introduce Bundles**: Create combo deals with high-revenue items (e.g., Sandwich + Juice) to increase average transaction value.")

print("4. **Target Low-Performing Items**: For items with low sales (e.g., Tea), consider recipe enhancements, promotions, or removal from the menu.")

print("5. **Seasonal Promotions**: Align offerings with seasonal trends (e.g., smoothies in summer, hot drinks in winter) based on monthly sales patterns.")
```