In this project, we are working with two CSV files that contain coffee sales transaction data from different time periods.

So lets deep dive into it

In [282…

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

Question 1: Load, Validate, and Combine Sales Data

In [283…

```
# Lets load both the CSV files
df1 = pd.read_csv("index_001.csv")
df2 = pd.read_csv("index_002.csv")
```

In [284…

```
df1.head()
```

Out[284…

	date	datetime	cash_type	card	money	coffee_name
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.7	Latte
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.7	Hot Chocolate
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.7	Hot Chocolate
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.9	Americano
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.7	Latte

In [285…

```
df2.tail()
```

Out[285…

	date	datetime	cash_type	money	coffee_name
257	2025-03-23	2025-03-23 14:55:46	cash	30.0	Cappuccino
258	2025-03-23	2025-03-23 15:15:36	card	25.0	Irish whiskey
259	2025-03-23	2025-03-23 17:59:25	card	28.0	Super chocolate
260	2025-03-23	2025-03-23 18:01:33	card	28.0	Vanilla with Irish whiskey
261	2025-03-23	2025-03-23 21:23:11	card	29.0	Coffee with Irish whiskey

In [286…
```
# Basic Information of file 1
```

In [287…
```
# No. of rows
df1.shape[0]
```

Out[287…
```
3636
```

In [288…
```
# No. of columns
df1.shape[1]
```

Out[288…
```
6
```

In [289…
```
# Column names
df1.columns
```

Out[289…
```
Index(['date', 'datetime', 'cash_type', 'card', 'money', 'coffee_name'], dtype
='object')
```

In [290…
```
# Basic Information of file 2
```

In [291…
```
# No. of rows
df2.shape[0]
```

Out[291…
```
262
```

In [292…
```
# No. of columns
df2.shape[1]
```

Out[292…
```
5
```

In [293…
```
# Column names
df2.columns
```

Out[293…
```
Index(['date', 'datetime', 'cash_type', 'money', 'coffee_name'], dtype='object')
```

In [294…
```
# lets check if we can merge the two files
```

In [295…
```
if list(df1.columns) == list(df2.columns):
    print("We can combine the 2 datasets")
else:
    print("Cant combine")
```

```
Cant combine
```

In [296…
```
missing_in_df1 = set(df2.columns) - set(df1.columns)
```

In [297…
```
print(missing_in_df1)
```
```
set()
```

In [298…
```
missing_in_df2 = set(df1.columns) - set(df2.columns)
```

In [299…
```
print(missing_in_df2)
```
```
{'card'}
```

In [300…
```
# lets add a column name card in df2
```

In [301…
```
df2_aligned = df2.reindex(columns=df1.columns)
```

In [302…
```
df2_aligned
```

Out[302…

	date	datetime	cash_type	card	money	coffee_name
0	2025-02-08	2025-02-08 14:26:04	cash	NaN	15.0	Tea
1	2025-02-08	2025-02-08 14:28:26	cash	NaN	15.0	Tea
2	2025-02-08	2025-02-08 14:33:04	card	NaN	20.0	Espresso
3	2025-02-08	2025-02-08 15:51:04	card	NaN	30.0	Chocolate with coffee
4	2025-02-08	2025-02-08 16:35:01	cash	NaN	27.0	Chocolate with milk
...	...	...	...	...	...	...
257	2025-03-23	2025-03-23 14:55:46	cash	NaN	30.0	Cappuccino
258	2025-03-23	2025-03-23 15:15:36	card	NaN	25.0	Irish whiskey
259	2025-03-23	2025-03-23 17:59:25	card	NaN	28.0	Super chocolate
260	2025-03-23	2025-03-23 18:01:33	card	NaN	28.0	Vanilla with Irish whiskey
261	2025-03-23	2025-03-23 21:23:11	card	NaN	29.0	Coffee with Irish whiskey

262 rows × 6 columns

In [303…
```
# Now lets combine both the datasets
```

In [304…
```
combined_df = pd.concat([df1,df2_aligned])
```

In [305…
```
combined_df
```

Out[305…

	date	datetime	cash_type	card	money	coffee_name
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.7	Latte
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.7	Hot Chocolate
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.7	Hot Chocolate
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.9	Americano
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.7	Latte
...	...	...	...	...	...	...
257	2025-03-23	2025-03-23 14:55:46	cash	NaN	30.0	Cappuccino
258	2025-03-23	2025-03-23 15:15:36	card	NaN	25.0	Irish whiskey
259	2025-03-23	2025-03-23 17:59:25	card	NaN	28.0	Super chocolate
260	2025-03-23	2025-03-23 18:01:33	card	NaN	28.0	Vanilla with Irish whiskey
261	2025-03-23	2025-03-23 21:23:11	card	NaN	29.0	Coffee with Irish whiskey

3898 rows × 6 columns

In [306…
```
combined_df.shape
```
Out[306…
```
(3898, 6)
```

In [307…
```
combined_df.columns
```
Out[307…
```
Index(['date', 'datetime', 'cash_type', 'card', 'money', 'coffee_name'], dtype
='object')
```

In [ ]:
```

```

In [308…
```
# Now lets check the datatypes of all the columns
```

In [309…
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3898 entries, 0 to 261
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   date         3898 non-null   object 
 1   datetime     3898 non-null   object 
 2   cash_type    3898 non-null   object 
 3   card         3547 non-null   object 
 4   money        3898 non-null   float64
 5   coffee_name  3898 non-null   object 
dtypes: float64(1), object(5)
memory usage: 213.2+ KB
```

Question 2: Clean and Prepare the Dataset for Analysis

In [310…
```
# lets convert date column to datetime
```

In [311…
```
combined_df['datetime'] = pd.to_datetime(combined_df['datetime'], errors = 'coerc
```

In [312…
```
combined_df['date'] = pd.to_datetime(combined_df['date'], errors = 'coerce')
```

In [313…
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3898 entries, 0 to 261
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   date         3898 non-null   datetime64[ns]
 1   datetime     3636 non-null   datetime64[ns]
 2   cash_type    3898 non-null   object        
 3   card         3547 non-null   object        
 4   money        3898 non-null   float64       
 5   coffee_name  3898 non-null   object        
dtypes: datetime64[ns](2), float64(1), object(3)
memory usage: 213.2+ KB
```

In [314…
```
# now lets check missing values
```

In [315…
```
print(combined_df.isnull().sum())
```

```
date             0
datetime       262
cash_type        0
card           351
money            0
coffee_name      0
dtype: int64
```

In [316…
```
# lets fill the null values
```

In [317…
```
combined_df['datetime'] = combined_df['datetime'].fillna(combined_df['date'])
```

In [318…
```
combined_df['card'] = combined_df['card'].fillna('No Card')
```

In [319…

```
print(combined_df.isnull().sum())
```

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

In [320…

```
combined_df.shape
```

Out[320…

```
(3898, 6)
```

In [321…

```
combined_df
```

Out[321…

	date	datetime	cash_type	card	money	coffee_name
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.7	Latte
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.7	Hot Chocolate
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.7	Hot Chocolate
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.9	Americano
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.7	Latte
...	...	...	...	...	...	...
257	2025-03-23	2025-03-23 00:00:00.000	cash	No Card	30.0	Cappuccino
258	2025-03-23	2025-03-23 00:00:00.000	card	No Card	25.0	Irish whiskey
259	2025-03-23	2025-03-23 00:00:00.000	card	No Card	28.0	Super chocolate
260	2025-03-23	2025-03-23 00:00:00.000	card	No Card	28.0	Vanilla with Irish whiskey
261	2025-03-23	2025-03-23 00:00:00.000	card	No Card	29.0	Coffee with Irish whiskey

3898 rows × 6 columns

Question 3: Analyze Sales Performance and Customer Behavior

In [322…

```
# Now lets create a new column called Payment_category which will tell if the tra
```

In [323…

```
def classify_payment(payment_method):
    if payment_method == 'cash':
        return 'cash'
    else:
        return 'digital'
```

In [324…
```
combined_df['payment_category'] = combined_df['cash_type'].apply(classify_payment
```

In [325…
```
combined_df.head()
```

Out[325…

	date	datetime	cash_type	card	money	coffee_name	payment_category
0	2024-03-01	2024-03-01 10:15:50.520	card	ANON-0000-0000-0001	38.7	Latte	digital
1	2024-03-01	2024-03-01 12:19:22.539	card	ANON-0000-0000-0002	38.7	Hot Chocolate	digital
2	2024-03-01	2024-03-01 12:20:18.089	card	ANON-0000-0000-0002	38.7	Hot Chocolate	digital
3	2024-03-01	2024-03-01 13:46:33.006	card	ANON-0000-0000-0003	28.9	Americano	digital
4	2024-03-01	2024-03-01 13:48:14.626	card	ANON-0000-0000-0004	38.7	Latte	digital

In [326…
```
combined_df.shape
```

Out[326…
```
(3898, 7)
```

In [327…
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3898 entries, 0 to 261
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   date              3898 non-null   datetime64[ns]
 1   datetime          3898 non-null   datetime64[ns]
 2   cash_type         3898 non-null   object
 3   card              3898 non-null   object
 4   money             3898 non-null   float64
 5   coffee_name       3898 non-null   object
 6   payment_category  3898 non-null   object
dtypes: datetime64[ns](2), float64(1), object(4)
memory usage: 243.6+ KB
```

In [328…
```
# Now our data is cleaned and ready for further analysis
```

In [ ]:

In [329…
```
# Now lets calculate Total revenue generated
```

In [330…
```
Total_revenue_generated = combined_df['money'].sum()
```

In [331…
```
print('Total Revenue Generated is:',round(Total_revenue_generated,2))
```

```
Total Revenue Generated is: 122321.58
```

In [ ]:

In [332…
```
# Now lets calculate Average transaction value
```

In [333…
```
average_transaction = combined_df['money'].mean()
```

In [334…
```
print("Average Transaction Value is:", round(average_transaction, 2))
```

```
Average Transaction Value is: 31.38
```

In [ ]:

In [335…
```
# Now lets see Most sold Coffee products
```

In [336…
```
top_products = combined_df['coffee_name'].value_counts()
```

In [337…
```
print("Top Selling Coffee Products:\n")
print(top_products.head())
```

```
Top Selling Coffee Products:

coffee_name
Americano with Milk    824
Latte                  806
Americano              593
Cappuccino             517
Cortado                292
Name: count, dtype: int64
```

In [ ]:

In [338…
```
# Sales difference between Cash and Digital Payments
```

In [339…
```
sales_by_payment = combined_df.groupby('payment_category')['money'].sum()
```

In [340…
```
print("Sales by Payment Category:\n")
print(sales_by_payment)
```

```
Sales by Payment Category:

payment_category
cash         5207.00
digital    117114.58
Name: money, dtype: float64
```

Question 4: Build a Reusable Sales Analysis Module Using OOPS

In [341…
```
# Now lets build  reusable Sales Analysis Module using OOPS
```

In [342…
```
class SalesAnalysis:
    def __init__(self,dataframe):
        self.df = dataframe

    def total_revenue(self):
        return self.df['money'].sum()

    def average_transaction_value(self):
        return self.df['money'].mean()

    def product_popularity(self):
        return self.df['coffee_name'].value_counts()

    def payment_trends(self):
        return self.df.groupby('payment_category')['money'].sum()
```

In [343…
```
# lets create object of the class
```

In [344…
```
analysis = SalesAnalysis(combined_df)

total_rev = analysis.total_revenue()
avg_trans  = analysis.average_transaction_value()
top_products = analysis.product_popularity()
payment_sales = analysis.payment_trends()
```

In [345…
```
print(f"Total revenue is: {round(total_rev,2)}")
```
```
Total revenue is: 122321.58
```

In [346…
```
print(f"Average transaction value is:{round(avg_trans,2)}")
```
```
Average transaction value is:31.38
```

In [347…
```
print(f"Top 5 products are:\n{top_products.head()}")
```
```
Top 5 products are:
coffee_name
Americano with Milk    824
Latte                  806
Americano              593
Cappuccino             517
Cortado                292
Name: count, dtype: int64
```

In [348…
```
 print(f"Sales by payment category:\n{payment_sales}")
```
```
Sales by payment category:
payment_category
cash         5207.00
digital    117114.58
Name: money, dtype: float64
```

Question 5: Visualize Insights and Handle Real-World Errors

In [349…
```
# Now lets create a folder and save visuals inside the folder
```

In [350…
```
def generate_report(file_path):
    try:
        if not os.path.exists(file_path):
            print("Error: File not found.")
            return

        df = pd.read_csv(file_path)

        if df.empty:
            print("Error: Dataset is empty.")
            return

        if not os.path.exists("reports"):
            os.makedirs("reports")

        df['datetime'] = pd.to_datetime(df['datetime'], errors='coerce')

        #Top products
        top_products = df['coffee_name'].value_counts().head(5)
        plt.figure(figsize=(8,5))
        top_products.plot(kind='bar')
        plt.savefig("reports/top_products.png")
        plt.close()

        #Revenue trend
        revenue_trend = df.groupby(df['datetime'].dt.date)['money'].sum()
        plt.figure(figsize=(8,5))
        revenue_trend.plot()
        plt.savefig("reports/revenue_trend.png")
        plt.close()

        #Payment modes
        payment_modes = df.groupby('payment_category')['money'].sum()
        plt.figure(figsize=(6,4))
        payment_modes.plot(kind='bar')
        plt.savefig("reports/payment_modes.png")
        plt.close()

        print("Report generated successfully inside 'reports' folder.")

    except Exception as e:
        print("An error occurred:", e)
```

In [351…
```
combined_df.to_csv("cleaned_coffee_sales.csv", index=False)
```

In [352…
```
generate_report("cleaned_coffee_sales.csv")
```

```
Report generated successfully inside 'reports' folder.
```

In [353…
```
# END
```