Experiment No. 07

Aim : Data Loading, Storage and File Formats .

Problem Statement : Analyzing Sales Data from MultipleFile Formats. Perform Following tasks : 1.Load the Dataset. 2. Explore the structure and content of the loaded data. 3. Perform data cleaning operations. 4. Convert the data into a unified format. 5. Perform data transformation tasks. 6. Analyze the sales data by performing descriptive statistics. 7. Create visualizations.

Dataset: Sales data in multiple file formats (e.g., CSV, Excel, JSON)

Software Requirements : Jupyter Notebook.

Hardware Requirements : 6GB free disk space, 2GB RAM plus additional RAM for virtual machines, Intel 64 and AMD 64 Architectures.

Objectives : Efficiently load diverse sales data, Ensuring integrity, scalability, security, and analysis readiness across multiple file formats for seamless insights extraction.

Theory : Analysing sales data from multiple file formats can be a common task in business and data analysis. Sales data can come in various formats, including spreedsheets [Excel, CSV], databases and even text files.

Here's a step by step guide on how to analyze sales data from multiple file formats :

1. **Gather the Data -** Collect all the sales data files you need from various sources and formats.
2. **File format Identification –** Determine the formats of the data files you have common formats include Excel(.xlsx), CSV(.CSV), JSON(JSON), SQL databases and text files(.txt).
3. **Data Prepration –** If the data is not a format that a you can work with directly, you might need to clean and preprocess it. This includes removing duplicates, handling missing values, and standardizing data formats. It's often beneficial to have a consistent data structure across all files. This means having the same coloumns and data types in each model.
4. **Choose Analysis Tools –** Depending on your data format and the analysis you want to perform, choose appropriate analysis tools.
   For Example – 1. For Excel files, you can use Microsoft excel or google sheets.

2. For CSV files or databases, you might use python or R with libraries like pandas or SQL for database querying.
3. For text files or other custom formats, you may need to write custom parsing scripts.

5. Load Data – Import the data into your choosen analysis tool. This often involves reading the data from the file and loading it into data structures like dataframes or database tables.

6. Data Consolidation – Data consolidation refers to the process of combining data from different sources for or formats into a unified dataset. Data consolidation is specifically used for analysing sales for the integration of data sources, creating a single source of truth , data cleaning and transformation, Enhanced analysis capabilities, scalabilty and efficiency.

7. Visualization and Reporting - Create a visualizations such as charts, graphs, and dashboards to communicate key findings effectively. Design reports that summarizes the analysis results, including insigts, trends and actionable recommendations.

8. Interpretation and Actionable Insights – Interpret the analysis results in the context of business goals and objectives. Identify actionable insights and recommendations based on the findings to optimize sales strategies, improve efficiency, or address challenges.

9. Documentation and Iteration – Document the analysis process, methodologies used and assumptions made for transperancy and reproducibility. Iterate on the analysis as nedded, incorporating feedback new data, or challenging business requirements to refine insights and strategies over time.

By following this steps, you can effectively analyse sales data from multiple file formats and derive valuable insights to drive business decisions.

Implementation :

Step 1 - Load the Sales Dataset.

```
import pandas as pd

df = pd.read_csv(r"C:\Users\saira\Downloads\supermarket_sales - Sheet1.csv")

df.head()
```

Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating

0 750-67-8428 A Yangon Member Female Health and beauty 74.69 7 26.1415 548.9715 1/5/2019 13:08 Ewallet 522.83 4.761905 26.1415 9.1
1 226-31-3081 C Naypyitaw Normal Female Electronic accessories 15.28 5 3.8200 80.2200 3/8/2019 10:29 Cash 76.40 4.761905 3.8200 9.6
2 631-41-3108 A Yangon Normal Male Home and lifestyle 46.33 7 16.2155 340.5255 3/3/2019 13:23 Credit card 324.31 4.761905 16.2155 7.4
3 123-19-1176 A Yangon Member Male Health and beauty 58.22 8 23.2880 489.0480 1/27/2019 20:33 Ewallet 465.76 4.761905 23.2880 8.4
4 373-73-7910 A Yangon Normal Male Sports and travel 86.31 7 30.2085 634.3785 2/8/2019 10:37 Ewallet 604.17 4.761905 30.2085 5.3

df.tail()

Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	1/29/2019	13:46	Ewallet	40.35	4.761905	2.0175	6.2
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	3/2/2019	17:16	Ewallet	973.80	4.761905	48.6900	4.4
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	33.4320	2/9/2019	13:22	Cash	31.84	4.761905	1.5920	7.7
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	69.1110	2/22/2019	15:33	Cash	65.82	4.761905	3.2910	4.1
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	649.2990	2/18/2019	13:28	Cash	618.38	4.761905	30.9190	6.6

Step 2- Explore the Structure and Content.

df.describe()

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.00000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	0.000000	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905	49.650000	10.00000

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%                  1000 non-null   float64
 9   Total                   1000 non-null   float64
 10  Date                    1000 non-null   object
 11  Time                    1000 non-null   object
 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(9)
memory usage: 132.9+ KB
df.isnull().sum()
Invoice ID              0
Branch                  0
City                    0
Customer type           0
Gender                  0
Product line            0
```

Unit price 0

Quantity 0

Tax 5% 0

Total 0

Date 0

Time 0

Payment 0

cogs 0

gross margin percentage 0

gross income 0

Rating 0

dtype: int64

df.duplicated().sum()

0

Step 4 - Convert the Data into a Unified Format.

df.columns = [col.lower() for col in df.columns]

df['date'] = pd.to_datetime(df['date'])

df['unit price'] = df['unit price'].astype(float)

df['quantity'] = df['quantity'].astype(int)

df['total'] = df['total'].astype(float)

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 18 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   int32
 8   tax 5%                  1000 non-null   float64
 9   total                   1000 non-null   float64
 10  date                    1000 non-null   datetime64[ns]
 11  time                    1000 non-null   object
 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  total sales             1000 non-null   float64
dtypes: datetime64[ns](1), float64(8), int32(1), object(8)
memory usage: 136.8+ KB
```

Step 5 - Perform Data Transformation and Analyze the Data.

```
df['total sales'] = df['unit price'] * df['quantity']

df
```

	invoice id	branch	city	customer type	gender	product line	unit price	quantity	tax 5%	total	date	time	payment	cogs	gross margin percentage	gross income	rating	total sales
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01-05	13:08	Ewallet	522.83	4.761905	26.1415	9.1	522.83
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	2019-03-08	10:29	Cash	76.40	4.761905	3.8200	9.6	76.40
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03-03	13:23	Credit card	324.31	4.761905	16.2155	7.4	324.31
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	2019-01-27	20:33	Ewallet	465.76	4.761905	23.2880	8.4	465.76
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02-08	10:37	Ewallet	604.17	4.761905	30.2085	5.3	604.17
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	42.3675	2019-01-29	13:46	Ewallet	40.35	4.761905	2.0175	6.2	40.35
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1022.4900	2019-03-02	17:16	Ewallet	973.80	4.761905	48.6900	4.4	973.80

997 727-02-1313 A Yangon Member Male Food and beverages 31.84 1 1.5920 33.4320 2019-02-09 13:22 Cash 31.84 4.761905 1.5920 7.7 31.84

998 347-56-2442 A Yangon Normal Male Home and lifestyle 65.82 1 3.2910 69.1110 2019-02-22 15:33 Cash 65.82 4.761905 3.2910 4.1 65.82

999 849-09-3807 A Yangon Member Female Fashion accessories 88.34 7 30.9190 649.2990 2019-02-18 13:28 Cash 618.38 4.761905 30.9190 6.6 618.38

1000 rows × 18 columns

total_sales.sum()

307587.38

average_order_value = df['total sales'].mean()

average_order_value

307.58738

category_sales = df.groupby('product line')['total sales'].sum()

category_sales

product line

Electronic accessories 51750.03

Fashion accessories 51719.90

Food and beverages 53471.28

Health and beauty 46851.18

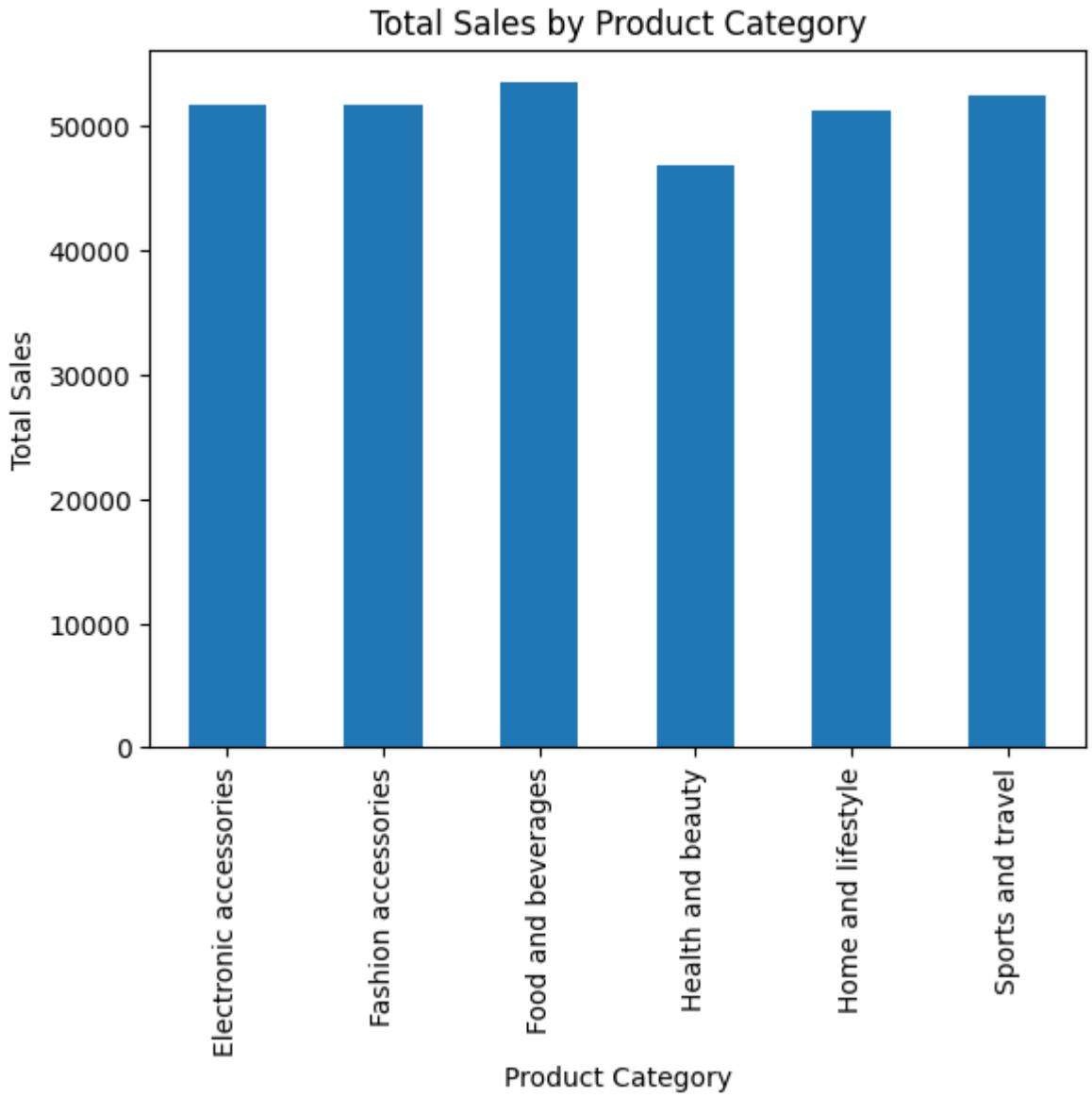
Home and lifestyle 51297.06

Sports and travel 52497.93

Name: total sales, dtype: float64

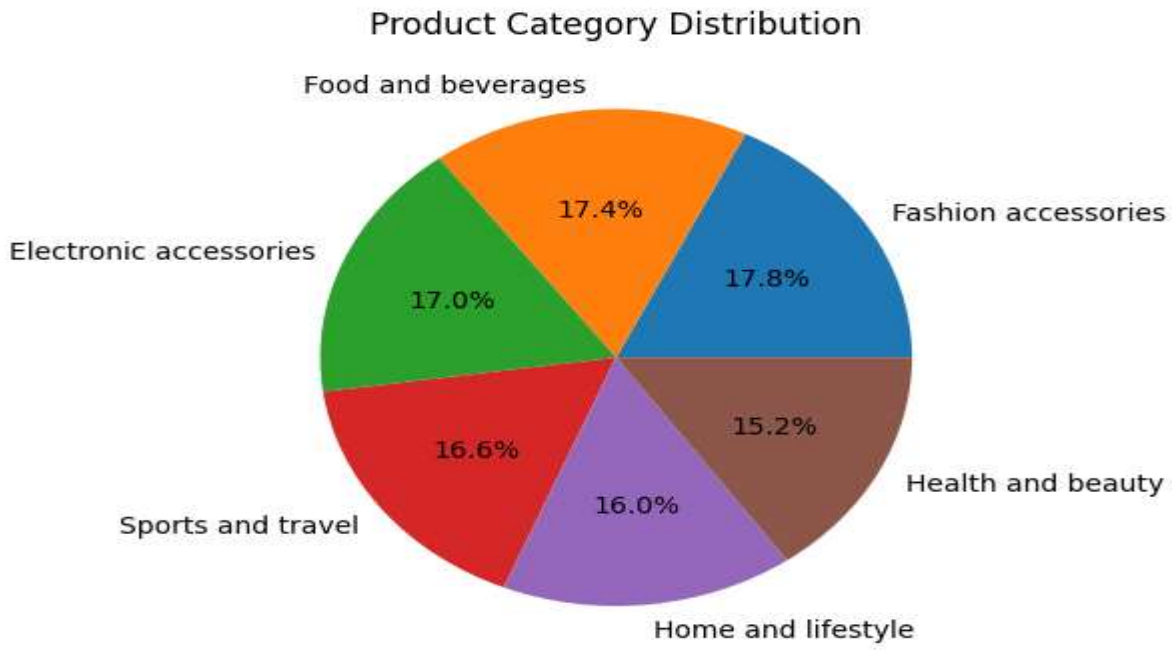
Step 6 -Create Visualizations.

```
import matplotlib.pyplot as plt

category_sales.plot(kind='bar', title='Total Sales by Product Category')

plt.xlabel('Product Category')

plt.ylabel('Total Sales')

plt.show()
```

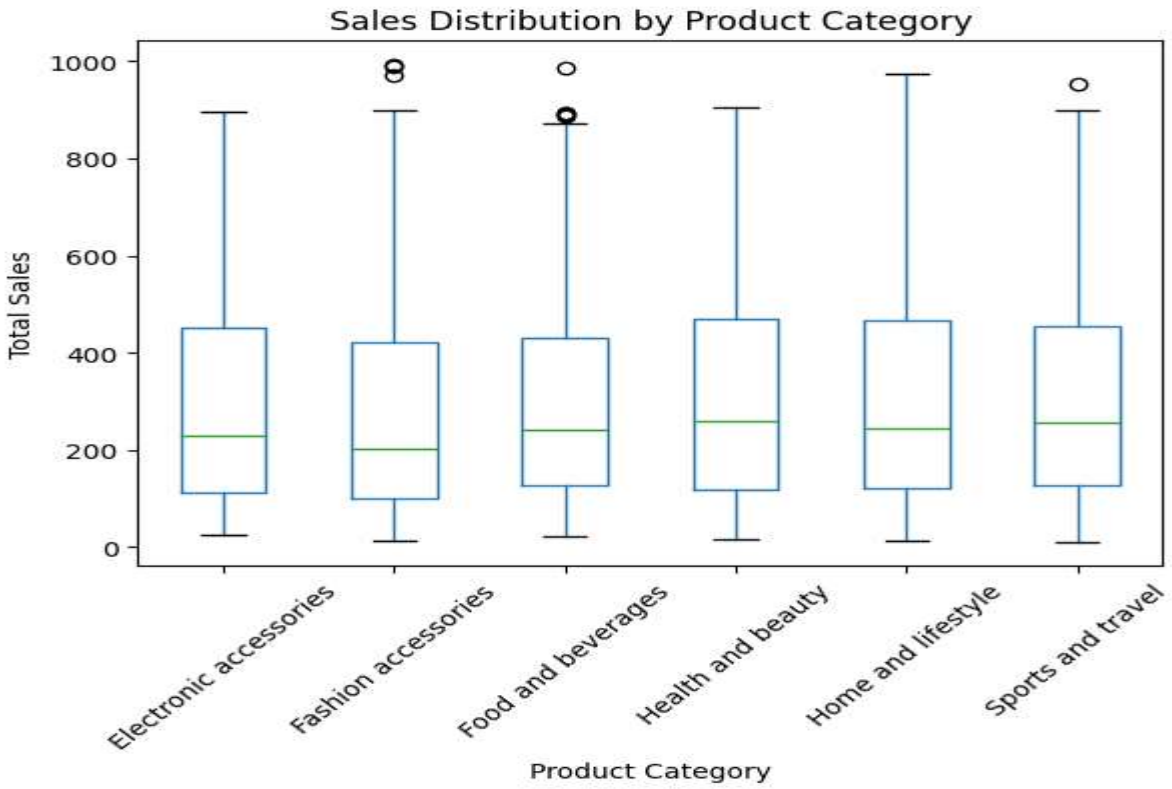



```
category_distribution = df['product line'].value_counts()
```

```
category_distribution.plot(kind='pie', title='Product Category Distribution', autopct='%1.1f%%')
plt.ylabel('')
plt.show()
```




```
df.boxplot(column='total sales', by='product line', grid=False, rot=45)
plt.title('Sales Distribution by Product Category')
plt.suptitle('')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```



Conclusion – We can successfully step by step analyze sales data from CSV File formats.