Experiment 1

Aim: Introduction to Data science and Data preparation using Pandas steps.

Theory:

Data Science is a multidisciplinary field that involves collecting, processing, analyzing, and visualizing data to extract meaningful insights. It uses techniques from statistics, machine learning, and programming to solve real-world problems. One of the essential steps in data science is data preparation, which ensures that raw data is cleaned and structured for analysis.

Key Steps in Data Preparation Using Pandas:

1. Loading Data
2. Exploring Data
3. Handling Missing Values
4. Data Transformation
5. Feature Engineering

Dataset Overview:

The dataset consists of 12 columns, each providing key insights into retail products, their pricing, visibility, and sales performance across different outlets. Below is a breakdown of the dataset's columns and their significance:

- **Item_Identifier:** A unique code assigned to each product, helping distinguish different items in the inventory.
- **Item_Weight:** The weight of a product, which can influence logistics, storage, and customer purchasing decisions.
- **Item_Fat_Content:** Categorizes products as Low Fat or Regular Fat, reflecting their nutritional composition and target consumers.
- **Item_Visibility:** Measures how prominently an item is displayed on store shelves, impacting its likelihood of being purchased.
- **Item_Type:** Classifies products into broad categories like Dairy, Beverages, or Snacks, aiding in trend analysis across different product segments.
- **Item_MRP:** The maximum retail price, which plays a crucial role in determining affordability and customer demand.
- **Outlet_Identifier:** A unique code assigned to each retail outlet, linking products to specific store locations.
- **Outlet_Establishment_Year:** The year an outlet was established, useful for analyzing how store maturity influences sales performance.
- **Outlet_Size:** Defines store sizes as Small, Medium, or Large, affecting customer foot traffic and product demand.

- **Outlet_Location_Type:** Indicates whether a store is located in an Urban, Suburban, or Tier 3 area, capturing demographic influences on sales.
- **Outlet_Type:** Differentiates between store formats, such as Grocery Stores and Supermarkets, highlighting variations in business models.
- **Item_Outlet_Sales:** The target variable, representing the total revenue generated by a product at a particular outlet.

Problem Statement:

The given dataset provides comprehensive details about retail product sales, focusing on the relationship between product attributes, outlet characteristics, and sales performance. This analysis aims to address the following key objectives:

- **Product Performance**: Identifying products or product types that drive the highest sales and those underperforming.
- **Outlet Insights**: Understanding the impact of outlet size, location, and type on overall sales performance.
- **Pricing Analysis**: Investigating how pricing strategies, such as maximum retail price (MRP), influence customer purchasing behavior.
- **Possible sales Prediction**: Developing models to predict item-level sales and provide actionable insights for inventory and pricing strategies.

By preprocessing the dataset and applying statistical analysis, the goal is to extract meaningful patterns that can guide data-driven decisions in retail operations.

Steps:

1. Load Data in Pandas
   The first step is to load our excel sheet or dataset using read_csv method.

Load Data in Pandas

```
import pandas as pd
df = pd.read_csv("/content/market_data.csv")
df
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Me
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Me
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Me
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	
...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	S
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Me
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	S

2. Description of the dataset
   df.info() provides information about the dataset like what are the different columns or features present in the dataset, what are their respective data types etc.

Description of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

```
df.describe()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

The describe method helps us understand the instances or the value in every row. It gives information like number of values, mean, max value for every column.

3. Drop columns that aren't useful
   It may not be necessary that all features present in our dataset will contribute to our analysis. There would be columns which are not required and which increase the size of data unnecessarily. In such cases we drop entire such columns. In our dataset the feature "Outlet_Establishment_Year" is not useful, so we drop it.

Drop columns that aren't useful

```
[5] cols = ["Outlet_Establishment_Year"]
    df = df.drop(cols, axis=1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Item_Identifier       8523 non-null   object
 1   Item_Weight           7060 non-null   float64
 2   Item_Fat_Content      8523 non-null   object
 3   Item_Visibility       8523 non-null   float64
 4   Item_Type             8523 non-null   object
 5   Item_MRP              8523 non-null   float64
 6   Outlet_Identifier     8523 non-null   object
 7   Outlet_Size           6113 non-null   object
 8   Outlet_Location_Type  8523 non-null   object
 9   Outlet_Type           8523 non-null   object
 10  Item_Outlet_Sales     8523 non-null   float64
dtypes: float64(4), object(7)
memory usage: 732.6+ KB
```

4. Drop rows with maximum missing values.
   In our dataset there are instances that have some missing values. We want a proper dataset where there are no missing values. In order to do this, we simply remove the rows with null values in them. To do this we use the dropna method. After deleting all null values rows, we observe that the number of non null values for every feature is now the same.

Drop rows with maximum missing values

```
[7] df = df.dropna()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4650 entries, 0 to 8522
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            4650 non-null   object
 1   Item_Weight                4650 non-null   float64
 2   Item_Fat_Content           4650 non-null   object
 3   Item_Visibility            4650 non-null   float64
 4   Item_Type                  4650 non-null   object
 5   Item_MRP                   4650 non-null   float64
 6   Outlet_Identifier          4650 non-null   object
 7   Outlet_Size                4650 non-null   object
 8   Outlet_Location_Type       4650 non-null   object
 9   Outlet_Type                4650 non-null   object
 10  Item_Outlet_Sales          4650 non-null   float64
dtypes: float64(4), object(7)
memory usage: 435.9+ KB
```

5. Take care of missing values

```
print(df["Item_Weight"].isnull().sum())  # Should show 0 missing values
```

```
1463
```

Analysis of our dataset shows that “Item_Weight” has 1463 missing values. This can cause problems in our analysis. So to address this, we will first categorize items by their types(as similar items may have similar weights), then we find their respective mean and fill null values with the mean.

```
df['Item_Weight'] = df.groupby('Item_Type')['Item_Weight'].transform(lambda x: x.fillna(x.mean()))
print(df["Item_Weight"].isnull().sum())  # Should show 0 missing values
```

```
0
```

6. Find out outliers (manually)
   We will use the IQR method to calculate outliers for “Item_Outlet_Sale”:
   First Quartile Q1 = 834.2474
   Second Quartile Q2 = 3101.2964

IQR = Q3-Q1 = 2267.049

Lower Bound = 834.2474 - 1.5x2267.049 = -2566.3261
Upper Bound = 3101.2964 - 1.5x2267.049 = 6501.8699
So any value less than -2566.3261 or greater than 6501.8699 can be considered as outliers. (Last column represents Item Outlet Sales)

Item_Iden	Item_Weig	Item_Fat_	Item_Visib	Item_Type	Item_MRP	Outlet_Ide	Outlet_Est	Outlet_Siz	Outlet_Lo	Outlet_Typ	Item_Outlet_Sales
FDA15	9.3	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermark	3735.138
FDA46	13.6	Low Fat	0.117818	Snack Foo	192.9136	OUT049	1999	Medium	Tier 1	Supermark	2527.377
FDC02	21.35	Low Fat	0.069103	Canned	259.9278	OUT018	2009	Medium	Tier 3	Supermark	6768.523
FDL58	13.15	Regular	0.043378	Canned	136.5046	OUT013	1987	High	Tier 3	Supermark	[illegible]
NCP30	20.5	Low Fat	0.032835	Household	40.2822	OUT045	2002		Tier 2	Supermark	707.0796
FDY25		Low Fat	0.03381	Canned	180.5976	OUT027	1985	Medium	Tier 3	Supermark	7968.294
NCH54	13.5	Low Fat	0.072669	Household	160.292	OUT046	1997	Small	Tier 1	Supermark	1438.128
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
NCR53		Low Fat	0.144338	Health anc	224.4404	OUT027	1985	Medium	Tier 3	Supermark	6976.252
FDS52	8.89	Low Fat	0.009163	Frozen Foo	101.7016	OUT010	1998		Tier 3	Grocery St	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
FDY56	16.35	Regular	0.062764	Fruits and	227.6062	OUT017	2007		Tier 2	Supermark	7222.598
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
FDH19		Low Fat	0.032928	Meat	173.1738	OUT027	1985	Medium	Tier 3	Supermark	7298.5
FDY55	16.75	Low Fat	0.081253	Fruits and	256.4988	OUT013	1987	High	Tier 3	Supermark	7452.965
FDY03	17.6	Regular	0.076552	Meat	110.5202	OUT017	2007		Tier 2	Supermark	450.0808
FDO23	17.85	Low Fat	0.147024	Breads	93.7436	OUT018	2009	Medium	Tier 3	Supermark	1134.523
DRE60	9.395	Low Fat	0.159658	Soft Drinks	224.972	OUT045	2002		Tier 2	Supermark	7696.648
DRP47	15.75	Low Fat	0.141399	Hard Drink	250.5382	OUT017	2007		Tier 2	Supermark	2775.72
FDU55	16.2	Low Fat	0.035984	Fruits and	260.6278	OUT045	2002		Tier 2	Supermark	4425.573
FDN58		Regular	0.056597	Snack Foo	230.9984	OUT027	1985	Medium	Tier 3	Supermark	9267.936
FDC41		Low Fat	0.2047	Frozen Foo	76.867	OUT019	1985	Small	Tier 1	Grocery St	[illegible]
FDI44	16.1	Low Fat	0.100389	Fruits and	76.0328	OUT049	1999	Medium	Tier 1	Supermark	1853.587
FDW56		Low Fat	0.070557	Fruits and	191.2162	OUT027	1985	Medium	Tier 3	Supermark	7504.232
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
FDE45	12.1	Low Fat	0.040522	Fruits and	178.5002	OUT018	2009	Medium	Tier 3	Supermark	3552.106
FDR35		Low Fat	0.020597	Breads	200.0742	OUT027	1985	Medium	Tier 3	Supermark	8958.339
FDT58		Low Fat	0.085528	Snack Foo	160.2816	OUT027	1985	Medium	Tier 3	Supermark	[illegible]

7. Standardization and Normalization
   Standardization and normalization are crucial in data preprocessing because they bring features to a common scale, improving model performance and convergence. Standardization (Z-score scaling) centers data around zero with unit variance, while Normalization (Min-Max scaling) rescales data to a fixed range (e.g., 0 to 1), ensuring fair weight distribution across features.

Standardization and Normalization of columns

```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/market_data.csv")

numerical_columns = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales']

# Initialize scalers
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

# Standardization of the data
standardized_data = standard_scaler.fit_transform(df[numerical_columns])
df_standardized = pd.DataFrame(standardized_data, columns=numerical_columns)

# Normalization of the data
normalized_data = minmax_scaler.fit_transform(df[numerical_columns])
df_normalized = pd.DataFrame(normalized_data, columns=numerical_columns)


print("Standardized Data:")
print(df_standardized.head())

print("\nNormalized Data:")
print(df_normalized.head())
```

```
Standardized Data:
   Item_Weight  Item_Visibility  Item_MRP  Item_Outlet_Sales
0    -0.766217        -0.970732  1.747454           0.910601
1    -1.494175        -0.908111 -1.489023          -1.018440
2     0.999834        -0.956917  0.010040          -0.049238
3     1.365966        -1.281758  0.660050          -0.849103
4    -0.845905        -1.281758 -1.399220          -0.695373

Normalized Data:
   Item_Weight  Item_Visibility  Item_MRP  Item_Outlet_Sales
0     0.282525         0.048866  0.927507           0.283587
1     0.081274         0.058705  0.072068           0.031419
2     0.770765         0.051037  0.468288           0.158115
3     0.871986         0.000000  0.640093           0.053555
4     0.260494         0.000000  0.095805           0.073651
```

Conclusion:

In this experiment, we explored the fundamental steps of data preparation using Pandas in Data Science. We loaded and analyzed a retail dataset, identified missing values, and handled them effectively. We also detected and removed outliers using the IQR method to ensure clean and reliable data. Additionally, we discussed feature selection, data transformation, and the importance of standardization and normalization in preparing data for further analysis.