Basic Level Exercises

1. Load the dataset and display the first 5 rows.

```
import pandas as pd
df = pd.read_csv('sales.csv')
print(df.head())
```

2. Calculate the mean, median, and standard deviation of Item_MRP.

```
mean_mrp = df['Item_MRP'].mean()
median_mrp = df['Item_MRP'].median()
std_mrp = df['Item_MRP'].std()

print(f"Mean MRP: {mean_mrp}")
print(f"Median MRP: {median_mrp}")
print(f"Standard Deviation of MRP: {std_mrp}")
```

3. Filter and display rows where Item_Weight is greater than 10.

```
heavy_items = df[df['Item_Weight'] > 10]
print(heavy_items)
```

4. Check for missing values in the dataset.

```
missing_values = df.isnull().sum()
print(missing_values)
```

5. Count the occurrences of different Item_Fat_Content values.

```
fat_content_counts = df['Item_Fat_Content'].value_counts()
print(fat_content_counts)

fat_content_counts = df.groupby(by=['Item_Fat_Content'])['Item_Fat_Content'].count()
print(fat_content_counts)
```

6. Rename the Item_Outlet_Sales column to Sales.

```
df.rename(columns={'Item_Outlet_Sales': 'Sales'}, inplace=True)
print(df.head())
```

7. Drop the Outlet_Establishment_Year column from the dataset.

```
df.drop(columns=['Outlet_Establishment_Year'], inplace=True)
print(df.head())
```

8. Sort the dataset by Sales in descending order.

```
sorted_df = df.sort_values(by='Sales', ascending=False)
print(sorted_df.head())
```

9. Display the unique values in the Outlet_Type column.

```
unique_outlet_types = df['Outlet_Type'].unique()
print(unique_outlet_types)
```

10. Subset the dataset to include only the columns Item_Type, Item_MRP, and Sales.

```
subset_df = df[['Item_Type', 'Item_MRP', 'Sales']]
print(subset_df.head())
```

11. Create a new column Price_per_Weight by dividing Item_MRP by Item_Weight. Then, find the maximum and minimum values in this new column.

```
df['Price_per_Weight'] = df['Item_MRP'] / df['Item_Weight']
max_value = df['Price_per_Weight'].max()
min_value = df['Price_per_Weight'].min()
print(f"Max Price per Weight: {max_value}, Min Price per Weight: {min_value}")
```

12. Add a new column High_Sales that assigns True if Sales are greater than the average sales, and False otherwise.

```
average_sales = df['Sales'].mean()
df['High_Sales'] = df['Sales'] > average_sales
print(df.head())
```

13. Filter the dataset to include only rows where Item_Type contains the word "Snack."

```
snacks_df = df[df['Item_Type'].str.contains('Snack')]
print(snacks_df.head())
```

14. Print the shape (rows and columns) and size (total elements) of the dataset.

```
shape = df.shape
```

```
size = df.size
print(f"Shape: {shape}, Size: {size}")
```

15. Plot a simple line chart showing the first 50 rows of Sales.

```
df['Sales'].head(50).plot(kind='line')
plt.title('Sales for First 50 Rows')
plt.xlabel('Index')
plt.ylabel('Sales')
plt.show()
```

Intermediate Exercises

1. Group the dataset by Outlet_Type and calculate the average Sales for each group.

```
avg_sales_per_outlet = df.groupby('Outlet_Type')['Sales'].mean()
print(avg_sales_per_outlet)
```

2. Replace missing values in Item_Weight with the mean of the column.

```
df['Item_Weight'].fillna(df['Item_Weight'].mean(), inplace=True)
print(df.isnull().sum())
```

3. Create a new column Sales_per_Unit by dividing Sales by Item_Weight.

```
df['Sales_per_Unit'] = df['Sales'] / df['Item_Weight']
print(df.head())
```

4. Create a pivot table showing the sum of Sales for each Item_Type across different Outlet_Location_Type.

```
   sales_pivot = df.pivot_table(values='Sales', index='Item_Type', columns='Outlet_Location_Type',
aggfunc='sum')
   print(sales_pivot)
```

5. Datetime Conversion:
   Assume there is a date column and convert it to datetime format (If not available, you can skip this step).

```
   df['Date'] = pd.to_datetime(df['Date'])
   print(df.info())
```

6. Filter the dataset to display only Low Fat items with Sales greater than 1000.

```
   filtered_df = df[(df['Item_Fat_Content'] == 'Low Fat') & (df['Sales'] > 1000)]
   print(filtered_df)
```

7. Create a multi-index DataFrame using Outlet_Type and Item_Type, and then display the total Sales for each combination.

```
   multi_index_df = df.set_index(['Outlet_Type', 'Item_Type'])
   sales_multi_index = multi_index_df['Sales'].groupby(level=[0, 1]).sum()
   print(sales_multi_index)
```

8. Calculate the cumulative sum of Sales for each Outlet_Type and plot it.

```
   df['Cumulative_Sales'] = df.groupby('Outlet_Type')['Sales'].cumsum()
```

```
df.pivot_table(values='Cumulative_Sales', index='Index', columns='Outlet_Type').plot()
plt.title('Cumulative Sales by Outlet Type')
plt.xlabel('Index')
plt.ylabel('Cumulative Sales')
plt.show()
```

9. Filter the dataset to include only rows where Item_Visibility is above the median and Item_Fat_Content is 'Low Fat'.

```
median_visibility = df['Item_Visibility'].median()
filtered_df = df[(df['Item_Visibility'] > median_visibility) & (df['Item_Fat_Content'] == 'Low Fat')]
print(filtered_df)
```

10. If the dataset had a Date column, extract the year and month from the date and add them as new columns.

```
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
print(df.head())
```

Advanced Exercises

1. Assume the data has a Date column. Plot the Sales over time to identify trends.

```
df.set_index('Date', inplace=True)
df['Sales'].plot(figsize=(12, 6))
plt.title('Sales Over Time')
plt.ylabel('Sales')
plt.show()
```

2. Create a pivot table that shows both the average and total Sales for each Outlet_Type and Item_Type.

```
   advanced_pivot = df.pivot_table(values='Sales', index='Outlet_Type', columns='Item_Type',
aggfunc=['mean', 'sum'])
   print(advanced_pivot)
```

3. Group the dataset by Outlet_Type and Item_Type, and calculate the sum of Sales.

```
   grouped_df = df.groupby(['Outlet_Type', 'Item_Type'])['Sales'].sum()
   print(grouped_df)
```

4. Create a stacked bar plot showing the distribution of Sales across Item_Type for each Outlet_Type.

```
   df.groupby(['Item_Type', 'Outlet_Type'])['Sales'].sum().unstack().plot(kind='bar', stacked=True)
   plt.title('Sales Distribution by Item and Outlet Type')
   plt.ylabel('Sales')
   plt.show()
```

5. Aggregate the dataset to show the average Sales for each combination of Item_Fat_Content and Outlet_Location_Type.

```
   advanced_agg = df.groupby(['Item_Fat_Content', 'Outlet_Location_Type'])['Sales'].mean()
   print(advanced_agg)
```

6. Group the data by Outlet_Type and Item_Type, and calculate both the mean and standard deviation of Sales. Then, plot the results as a grouped bar chart.

```
grouped_stats = df.groupby(['Outlet_Type', 'Item_Type'])['Sales'].agg(['mean', 'std']).unstack()
grouped_stats.plot(kind='bar')
plt.title('Mean and Standard Deviation of Sales by Outlet and Item Type')
plt.xlabel('Outlet and Item Type')
plt.ylabel('Sales')
plt.show()
```

7. Write a custom function that categorizes Sales into bins (e.g., Low, Medium, High) based on specific thresholds, and apply it to create a new column.

```
def categorize_sales(sales):
    if sales < 500:
        return 'Low'
    elif 500 <= sales < 1500:
        return 'Medium'
    else:
        return 'High'

df['Sales_Category'] = df['Sales'].apply(categorize_sales)
print(df.head())
```