1. Importing pandas and loading the Excel file

import pandas as pd

salesDF = pd.read_excel("DemoSalesData.xlsx")

- **import pandas as pd**: Imports the pandas library, a powerful tool for data manipulation and analysis in Python.
- **pd.read_excel("DemoSalesData.xlsx")**: Reads data from an Excel file named DemoSalesData.xlsx into a pandas DataFrame called salesDF.

---

2. Basic inspection and slicing

python

Copy code

salesDF.head()

- Displays the first 5 rows of salesDF by default.

salesDF.tail()

- Displays the last 5 rows of salesDF by default.

salesDF.empty

- Checks if salesDF is empty. Returns True if empty, otherwise False.

salesDF[100:400]

- Selects rows from index 100 (inclusive) to 400 (exclusive).

---

3. Missing values analysis

salesDF.isnull().sum()

- Shows the total count of missing (NaN) values in each column.

python

Copy code

salesDF.isnull().any()

- Checks if there are any missing (NaN) values in each column. Returns True/False for each column.

---

4. Dataframe structure and summary

salesDF.shape

- Returns the shape (number of rows and columns) of salesDF as a tuple (rows, columns).

python

Copy code

salesDF.info()

- Displays detailed information about salesDF, such as column names, non-null counts, and data types.

---

5. Column access and slicing

salesDF['product']

- Selects and returns the product column.

python

Copy code

salesDF[['product', 'NetSales']][0:40]

- Selects the product and NetSales columns and retrieves the first 40 rows.

---

6. Columns and data types

salesDF.columns

- Returns the column labels of salesDF.

salesDF.dtypes

- Returns the data types of each column.

---

7. Filtering data

salesDF.loc[salesDF['product'] == 'ML Headset']

- Returns all rows where the product column equals 'ML Headset'.

---

8. Renaming columns

salesDF.rename(columns={"productcategory":"Product_Category"})

salesDF.rename(columns={"productsubcategory":"Product_SubCategory"})

salesDF.rename(columns={"Customer":"Client"})

- Renames columns in salesDF. For example:
    - Changes productcategory to Product_Category.
    - Changes Customer to Client.

---

9. Statistical analysis

salesDF["Sales"].std()

- Computes the standard deviation of the Sales column.

salesDF["Sales"].median()

- Computes the median of the Sales column.

salesDF["Sales"].mean()

- Computes the mean of the Sales column.

salesDF["Sales"].mode()

- Returns the mode(s) of the Sales column.

salesDF["Sales"].describe()

- Provides descriptive statistics (count, mean, std, min, 25%, 50%, 75%, max) for the Sales column.

---

10. Filtering with conditions

salesDF.loc[salesDF["NetSales"] < 0]

- Filters and returns rows where the NetSales column is less than 0.

---

11. Adding a new column

salesDF["profit"] = 99999999990

- Creates a new column profit in salesDF and assigns a constant value of 99999999990 to all rows.

---

12. Minimum and maximum values

salesDF["Sales"].min()

salesDF["Sales"].max()

- Returns the minimum and maximum values of the Sales column, respectively.

---

13. Handling missing values

print(salesDF["product"].isnull().sum())

- Prints the count of missing (NaN) values in the product column.

print(salesDF["product"].head())

- Prints the first 5 values of the product column.

salesDF["product"] = salesDF["product"].fillna("")

- Replaces missing values in the product column with an empty string.

salesDF.fillna({"product": "Unknown"}, inplace=True)

- Replaces missing values in the product column with "Unknown" and modifies salesDF in place.

---

14. Filtering and renaming columns

salesDF[salesDF["saleterritory"] == "United Kingdom"]

- Filters rows where the saleterritory column equals "United Kingdom".

salesDF.loc[salesDF["product"] == "ML Headset"]

- Same as above, filters rows where product equals "ML Headset".

---

15. Handling errors in code

alesDF.loc[salesDF["product"] == "ML Headset"]

- **Typo**: alesDF should be corrected to salesDF.

salesDF.rename(columns={"Customer":"Client"})

- This renames the Customer column to Client.