CL-I DMV 7

July 16, 2025

[24]:
```
'''
NAME:Aher Swami Sandip
ROLL NO.01
COURSE: AI&DS
CLASS: BE
SUB:Computer Laboratory-I (DMV)
'''
```

[36]:
```
pip install --upgrade xlrd
```

```
Requirement already satisfied: xlrd in d:\anaconda3\lib\site-packages (2.0.2)
Note: you may need to restart the kernel to use updated packages.
```

0.1 Data Loading, Storage and File Formats

[2]:
```
# Problem Statement: Analyzing Sales Data from Multiple File Formats
# Dataset: Sales data in multiple file formats (e.g., CSV, Excel, JSON)
# Description: The goal is to load and analyze sales data from different file␣
 ↪formats, including
# CSV, Excel, and JSON, and perform data cleaning, transformation, and analysis␣
 ↪on the
# dataset.

# Tasks to Perform:
#  Obtain sales data files in various formats, such as CSV, Excel, and JSON.
```

0.1.1 1. Load the sales data from each file format into the appropriate data structures or dataframes.

[27]:
```
import pandas as pd
```

[39]:
```
# Load CSV data
csv_data = pd.read_csv("sales_data_sample.csv" ,encoding='ISO-8859-1')

# Load Excel data
excel_data = pd.read_excel("sales_data_sample.xls",engine='xlrd')

# Load JSON data
```

```python
import json
with open("sales_data_sample.json", 'r') as json_file:
    json_data = json.load(json_file)
json_data = pd.DataFrame(json_data)
```

[40]:

```python
# Print the first few rows of each dataframe
print("CSV Data:")
print(csv_data.head())

print("\nExcel Data:")
print(excel_data.head())

print("\nJSON Data:")
print(json_data.head())
```

```
CSV Data:
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107               30      95.70                2  2871.00
1        10121               34      81.35                5  2765.90
2        10134               41      94.74                2  3884.34
3        10145               45      83.26                6  3746.70
4        10159               49     100.00               14  5205.27

         ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0   2/24/2003 0:00  Shipped       1         2     2003  ...
1    5/7/2003 0:00  Shipped       2         5     2003  ...
2    7/1/2003 0:00  Shipped       3         7     2003  ...
3   8/25/2003 0:00  Shipped       3         8     2003  ...
4  10/10/2003 0:00  Shipped       4        10     2003  ...

                    ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0        897 Long Airport Avenue           NaN            NYC    NY
1             59 rue de l'Abbaye           NaN          Reims   NaN
2  27 rue du Colonel Pierre Avia           NaN          Paris   NaN
3             78934 Hillside Dr.           NaN       Pasadena    CA
4                7734 Strong St.           NaN  San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0      10022     USA       NaN              Yu             Kwai    Small
1      51100  France      EMEA         Henriot             Paul    Small
2      75508  France      EMEA        Da Cunha           Daniel   Medium
3      90003     USA       NaN           Young            Julie   Medium
4        NaN     USA       NaN           Brown            Julie   Medium

[5 rows x 25 columns]

Excel Data:
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107               30      95.70                2  2871.00
1        10121               34      81.35                5  2765.90
2        10134               41      94.74                2  3884.34
3        10145               45      83.26                6  3746.70
4        10159               49     100.00               14  5205.27

          ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  …  \
0   2/24/2003 0:00  Shipped       1         2     2003  …
1    5/7/2003 0:00  Shipped       2         5     2003  …
2    7/1/2003 0:00  Shipped       3         7     2003  …
3   8/25/2003 0:00  Shipped       3         8     2003  …
4  10/10/2003 0:00  Shipped       4        10     2003  …

                    ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0        897 Long Airport Avenue           NaN            NYC    NY
1             59 rue de l'Abbaye           NaN          Reims   NaN
2  27 rue du Colonel Pierre Avia           NaN          Paris   NaN
3             78934 Hillside Dr.           NaN       Pasadena    CA
4                7734 Strong St.           NaN  San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0      10022     USA       NaN              Yu             Kwai    Small
1      51100  France      EMEA         Henriot             Paul    Small
2      75508  France      EMEA        Da Cunha           Daniel   Medium
3      90003     USA       NaN           Young            Julie   Medium
4        NaN     USA       NaN           Brown            Julie   Medium

[5 rows x 25 columns]

JSON Data:
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107               30      95.70                2  2871.00
1        10121               34      81.35                5  2765.90
2        10134               41      94.74                2  3884.34
3        10145               45      83.26                6  3746.70
4        10159               49     100.00               14  5205.27

          ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  …  \
0   2/24/2003 0:00  Shipped       1         2     2003  …
1    5/7/2003 0:00  Shipped       2         5     2003  …
2    7/1/2003 0:00  Shipped       3         7     2003  …
3   8/25/2003 0:00  Shipped       3         8     2003  …
4  10/10/2003 0:00  Shipped       4        10     2003  …

                    ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0        897 Long Airport Avenue                          NYC    NY
1             59 rue de l'Abbaye                        Reims
2  27 rue du Colonel Pierre Avia                        Paris
```

```
3             78934 Hillside Dr.                     Pasadena    CA
4                7734 Strong St.                San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0      10022     USA        NA              Yu             Kwai    Small
1      51100  France      EMEA         Henriot             Paul    Small
2      75508  France      EMEA        Da Cunha           Daniel   Medium
3      90003     USA        NA           Young            Julie   Medium
4                USA        NA           Brown            Julie   Medium

[5 rows x 25 columns]
```

0.1.2 2. Explore the structure and content of the loaded data, identifying any inconsistencies,

0.1.3 missing values, or data quality issues.

[44]:

```python
# Check the structure of CSV data
print("Structure and Info of CSV Data:")
print(csv_data.info())

# Check for missing values in CSV data
print("\nMissing Values in CSV Data:")
print(csv_data.isnull().sum())

# Check the structure of Excel data
print("\nStructure and Info of Excel Data:")
print(excel_data.info())

# Check for missing values in Excel data
print("\nMissing Values in Excel Data:")
print(excel_data.isnull().sum())

# Check the structure of JSON data
print("\nStructure and Info of JSON Data:")
print(json_data.info())

# Check for missing values in JSON data
print("\nMissing Values in JSON Data:")
print(json_data.isnull().sum())
```

```
Structure and Info of CSV Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
None

Missing Values in CSV Data:
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
SALES                  0
ORDERDATE              0
STATUS                 0
QTR_ID                 0
MONTH_ID               0
YEAR_ID                0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
PHONE                  0
ADDRESSLINE1           0
ADDRESSLINE2        2521
CITY                   0
STATE               1486
```

```
POSTALCODE            76
COUNTRY                0
TERRITORY           1074
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64

Structure and Info of Excel Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      302 non-null    object
 17  CITY              2823 non-null   object
 18  STATE             1337 non-null   object
 19  POSTALCODE        2747 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         1749 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
None

Missing Values in Excel Data:
ORDERNUMBER            0
QUANTITYORDERED        0
PRICEEACH              0
ORDERLINENUMBER        0
```

```
SALES                  0
ORDERDATE              0
STATUS                 0
QTR_ID                 0
MONTH_ID               0
YEAR_ID                0
PRODUCTLINE            0
MSRP                   0
PRODUCTCODE            0
CUSTOMERNAME           0
PHONE                  0
ADDRESSLINE1           0
ADDRESSLINE2        2521
CITY                   0
STATE               1486
POSTALCODE            76
COUNTRY                0
TERRITORY           1074
CONTACTLASTNAME        0
CONTACTFIRSTNAME       0
DEALSIZE               0
dtype: int64

Structure and Info of JSON Data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ORDERNUMBER       2823 non-null   int64
 1   QUANTITYORDERED   2823 non-null   int64
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object
 6   STATUS            2823 non-null   object
 7   QTR_ID            2823 non-null   int64
 8   MONTH_ID          2823 non-null   int64
 9   YEAR_ID           2823 non-null   int64
 10  PRODUCTLINE       2823 non-null   object
 11  MSRP              2823 non-null   int64
 12  PRODUCTCODE       2823 non-null   object
 13  CUSTOMERNAME      2823 non-null   object
 14  PHONE             2823 non-null   object
 15  ADDRESSLINE1      2823 non-null   object
 16  ADDRESSLINE2      2823 non-null   object
 17  CITY              2823 non-null   object
 18  STATE             2823 non-null   object
 19  POSTALCODE        2823 non-null   object
 20  COUNTRY           2823 non-null   object
 21  TERRITORY         2823 non-null   object
 22  CONTACTLASTNAME   2823 non-null   object
 23  CONTACTFIRSTNAME  2823 non-null   object
 24  DEALSIZE          2823 non-null   object
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
None

Missing Values in JSON Data:
ORDERNUMBER         0
QUANTITYORDERED     0
PRICEEACH           0
ORDERLINENUMBER     0
SALES               0
ORDERDATE           0
STATUS              0
QTR_ID              0
MONTH_ID            0
YEAR_ID             0
PRODUCTLINE         0
MSRP                0
PRODUCTCODE         0
CUSTOMERNAME        0
PHONE               0
ADDRESSLINE1        0
ADDRESSLINE2        0
CITY                0
STATE               0
POSTALCODE          0
COUNTRY             0
TERRITORY           0
CONTACTLASTNAME     0
CONTACTFIRSTNAME    0
DEALSIZE            0
dtype: int64
```

0.1.4 3. Perform data cleaning operations, such as handling missing values, removing duplicates, or correcting inconsistencies.

[48]:

```
# Handling missing values
# Replace missing values with appropriate values or drop rows/columns with␣
 ↪missing data

# Replace missing values in CSV data with a default value (e.g., 0)
csv_data.fillna(0, inplace=True)
```

```
# Remove duplicates in CSV data based on all columns
csv_data.drop_duplicates(inplace=True)

# Handling missing values in Excel data
# Replace missing values in Excel data with a default value (e.g., 0)
excel_data.fillna(0, inplace=True)

# Remove duplicates in Excel data based on specific columns (e.g., first_name␣
 ↪and last_name)
excel_data.drop_duplicates(subset=['CONTACTLASTNAME', 'CONTACTFIRSTNAME'],␣
 ↪inplace=True)

# Handling missing values in JSON data
# Replace missing values in JSON data with a default value (e.g., 0)
json_data.fillna(0, inplace=True)

# Remove duplicates in JSON data based on specific columns (e.g., first_name␣
 ↪and last_name)
json_data.drop_duplicates(subset=['CONTACTLASTNAME', 'CONTACTFIRSTNAME'],␣
 ↪inplace=True)
```

0.1.5 4. Convert the data into a unified format, such as a common dataframe or data structure,to enable seamless analysis.

[50]:
```
# Combine the data into a common DataFrame
common_df = pd.concat([csv_data, excel_data, json_data], ignore_index=True)

# Optional: Reset index if needed
common_df.reset_index(drop=True, inplace=True)

# Print the unified DataFrame
print("Unified Data:")
print(common_df.head())
```

```
Unified Data:
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107               30      95.70                2  2871.00
1        10121               34      81.35                5  2765.90
2        10134               41      94.74                2  3884.34
3        10145               45      83.26                6  3746.70
4        10159               49     100.00               14  5205.27

         ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0   2/24/2003 0:00  Shipped       1         2     2003  ...
1    5/7/2003 0:00  Shipped       2         5     2003  ...
2    7/1/2003 0:00  Shipped       3         7     2003  ...
```

```
3   8/25/2003 0:00  Shipped       3         8     2003  …
4  10/10/2003 0:00  Shipped       4        10     2003  …

                     ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0         897 Long Airport Avenue             0            NYC    NY
1              59 rue de l'Abbaye             0          Reims     0
2   27 rue du Colonel Pierre Avia             0          Paris     0
3              78934 Hillside Dr.             0       Pasadena    CA
4                 7734 Strong St.             0  San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0      10022     USA         0              Yu             Kwai    Small
1      51100  France      EMEA         Henriot             Paul    Small
2      75508  France      EMEA        Da Cunha           Daniel   Medium
3      90003     USA         0           Young            Julie   Medium
4          0     USA         0           Brown            Julie   Medium

[5 rows x 25 columns]
```

0.1.6 5. Perform data transformation tasks, such as merging multiple datasets, splitting columns, or deriving new variables.

[52]:

```
# Check if columns are consistent across datasets
if not all(csv_data.columns == excel_data.columns) or not all(csv_data.columns␣
 ↪== json_data.columns):
    print("Columns are not consistent across datasets.")
else:
    # Merge the datasets
    common_df = pd.concat([csv_data, excel_data, json_data], ignore_index=True)

    # Split a column and create new variables
    common_df['ADDRESSLINE1'] = common_df['DEALSIZE'].str.extract(r'(\d+)')

    # Derive a new variable
    common_df['STATUS'] = common_df['MONTH_ID'] * common_df['QTR_ID']

    # Print the transformed DataFrame
    print("Transformed Data:")
    print(common_df.head())
```

```
Transformed Data:
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107               30      95.70                2  2871.00
1        10121               34      81.35                5  2765.90
2        10134               41      94.74                2  3884.34
3        10145               45      83.26                6  3746.70
4        10159               49     100.00               14  5205.27
```

```
         ORDERDATE  STATUS  QTR_ID  MONTH_ID  YEAR_ID  … ADDRESSLINE1  \
0   2/24/2003 0:00       2       1         2     2003  …          NaN
1     5/7/2003 0:00      10       2         5     2003  …          NaN
2     7/1/2003 0:00      21       3         7     2003  …          NaN
3   8/25/2003 0:00      24       3         8     2003  …          NaN
4  10/10/2003 0:00      40       4        10     2003  …          NaN

   ADDRESSLINE2            CITY STATE POSTALCODE COUNTRY TERRITORY  \
0             0             NYC    NY      10022     USA         0
1             0           Reims     0      51100  France      EMEA
2             0           Paris     0      75508  France      EMEA
3             0        Pasadena    CA      90003     USA         0
4             0  San Francisco    CA          0     USA         0

  CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0              Yu             Kwai    Small
1         Henriot             Paul    Small
2        Da Cunha           Daniel   Medium
3           Young            Julie   Medium
4           Brown            Julie   Medium

[5 rows x 25 columns]
```

0.1.7 6. Analyze the sales data by performing descriptive statistics, aggregating data by specific variables, or calculating metrics such as

0.1.8 total sales, average order value, or product category distribution.

[56]:

```
# Perform descriptive statistics
# You can use the `describe` method to get summary statistics for numeric␣
 ↪columns
desc_stats = common_df.describe()

# Aggregate data by specific variables
# For example, you can group data by 'STATUS' and calculate the total sales and␣
 ↪average order value
agg_data = common_df.groupby('STATUS').agg({'QTR_ID': 'sum', 'MONTH_ID':␣
 ↪'mean'})

# Calculate total sales
total_sales = common_df['PRICEEACH'].sum()

# Calculate average order value
average_order_value = common_df['PRICEEACH'].mean()

# Calculate product category distribution
# Assuming you have a 'STATUS' column in your DataFrame
product_distribution = common_df['STATUS'].value_counts()
```

```python
# Print the results
print("Descriptive Statistics:")
print(desc_stats)

print("\nAggregate Data by STATUS:")
print(agg_data)

print("\nTotal Sales: $", total_sales)

print("\nAverage Order Value: $", average_order_value)

print("\njob Category Distribution:")
print(product_distribution)
```

```
Descriptive Statistics:
        ORDERNUMBER  QUANTITYORDERED    PRICEEACH  ORDERLINENUMBER  \
count   3007.000000      3007.000000  3007.000000      3007.000000   
mean   10257.774194        35.181576    84.554889         6.413036   
std       91.718802         9.856334    19.898360         4.222717   
min    10100.000000         6.000000    26.880000         1.000000   
25%    10180.000000        27.000000    70.555000         3.000000   
50%    10262.000000        35.000000    98.000000         6.000000   
75%    10332.000000        43.000000   100.000000         9.000000   
max    10425.000000        97.000000   100.000000        18.000000   

              SALES       STATUS       QTR_ID     MONTH_ID      YEAR_ID  \
count   3007.000000  3007.000000  3007.000000  3007.000000  3007.000000   
mean    3660.248244    23.545394     2.715663     7.085467  2003.809112   
std     1916.208584    17.465654     1.203495     3.652897     0.697975   
min      482.130000     1.000000     1.000000     1.000000  2003.000000   
25%     2250.045000     8.000000     2.000000     4.000000  2003.000000   
50%     3267.250000    24.000000     3.000000     8.000000  2004.000000   
75%     4618.785000    44.000000     4.000000    11.000000  2004.000000   
max    14082.800000    48.000000     4.000000    12.000000  2005.000000   

              MSRP  
count  3007.000000  
mean    103.712670  
std      42.325638  
min      33.000000  
25%      70.000000  
50%      99.000000  
75%     132.000000  
max     214.000000  

Aggregate Data by STATUS:
        QTR_ID  MONTH_ID
```

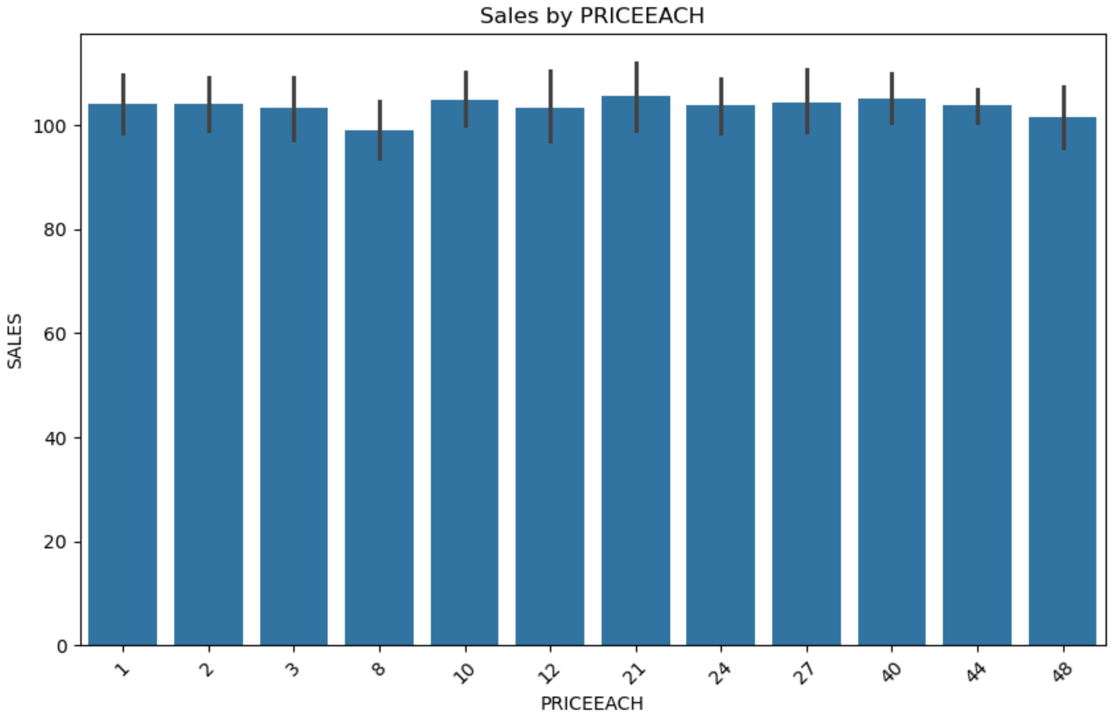
```
STATUS
1          243       1.0
2          242       2.0
3          224       3.0
8          380       4.0
10         532       5.0
12         286       6.0
21         447       7.0
24         615       8.0
27         549       9.0
40        1364      10.0
44        2532      11.0
48         752      12.0

Total Sales: $ 254256.55000000002

Average Order Value: $ 84.55488859328234

job Category Distribution:
STATUS
44    633
40    341
10    266
1     243
2     242
3     224
24    205
8     190
48    188
27    183
21    149
12    143
Name: count, dtype: int64
```

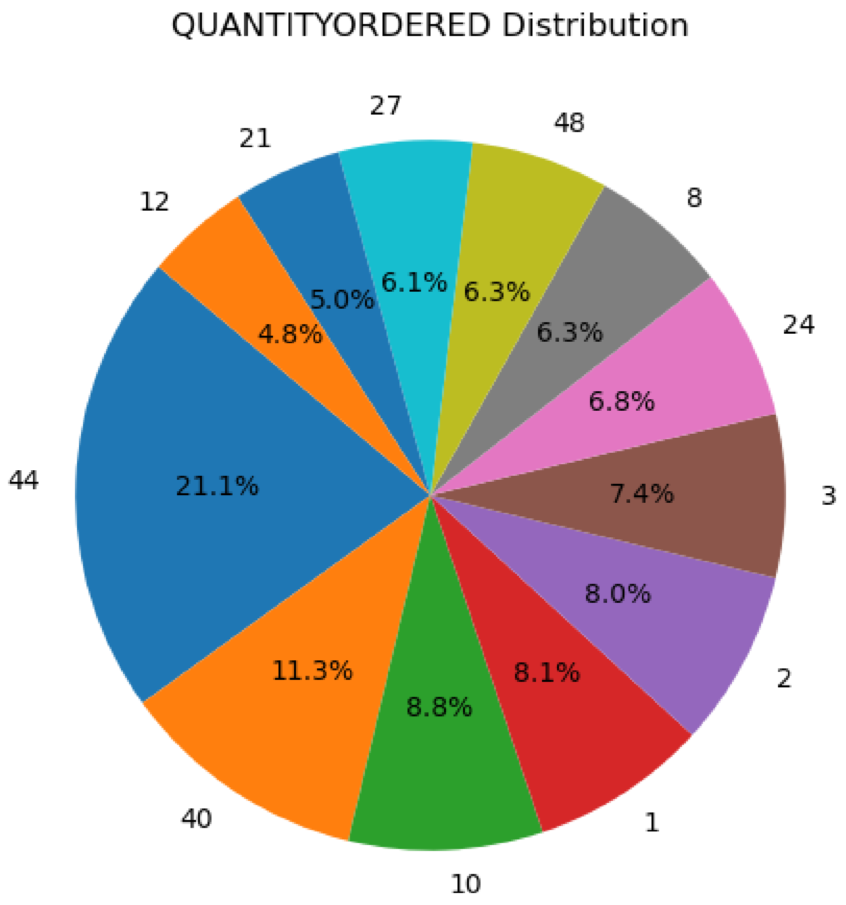
0.1.9 7. Create visualizations, such as bar plots, pie charts, or box plots, to represent the sales data

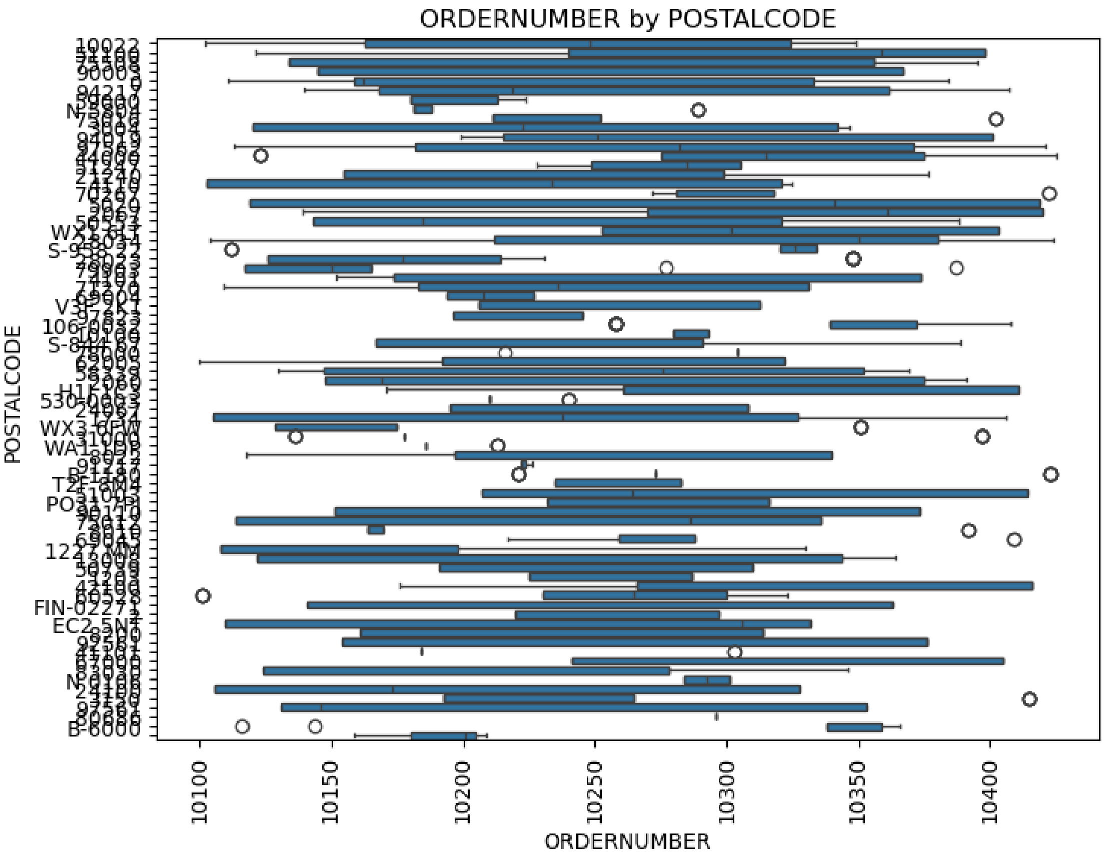
0.1.10 and gain insights into sales trends, customer behavior, or product performance.

[59]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Create a bar plot to represent sales by priceeach category
plt.figure(figsize=(10, 6))
sns.barplot(x='STATUS', y='MSRP', data=common_df)
plt.title('Sales by PRICEEACH')
plt.xlabel('PRICEEACH')
plt.ylabel('SALES')
plt.xticks(rotation=45)
plt.show()

# Create a pie chart to represent the distribution of quantity categories
QUANTITYORDERED = common_df['STATUS'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(QUANTITYORDERED, labels=QUANTITYORDERED.index, autopct='%1.1f%%', startangle=140)
plt.title('QUANTITYORDERED Distribution')
plt.show()

# Create a box plot to visualize the distribution of sales values
plt.figure(figsize=(8, 6))
sns.boxplot(x='ORDERNUMBER', y='POSTALCODE', data=common_df)
plt.title('ORDERNUMBER by POSTALCODE')
plt.xlabel('ORDERNUMBER')
plt.ylabel('POSTALCODE')
plt.xticks(rotation=90)
plt.show()
```




QUANTITYORDERED Distribution
44
21.1%
12
4.8%
21
5.0%
27
6.1%
48
6.3%
8
6.3%
24
6.8%
3
7.4%
2
8.0%
1
8.1%
10
8.8%
40
11.3%


ORDERNUMBER by POSTALCODE
POSTALCODE
10022
94217
WX1 6LT
S-958 22
V3F 2K1
106-0032
S-844 67
530-0003
WX3 6FW
WA1 1DP
T2F 8M4
1227 MM
FIN-02271
EC2 5NT
B-6000
10100
10150
10200
10250
10300
10350
10400
ORDERNUMBER

[ ]: