NUMPY TRAINING QUESTIONS

Review the following exploration of the given dataset (SALES.CSV)

1 How do you install NumPy?

In Command Prompt (CMD)

> pip install numpy

2 Load the dataset and preview the first few rows to understand its structure.

```
import numpy as np
data = np.genfromtxt('sales.csv', delimiter=',', names=True, dtype=None, encoding='utf-8')
print(data)
```

Output:

```
[('FDA15',  9.3 , 'Low Fat', 0.0160473 , 'Dairy', 249.8092, 'OUT049', 1999, 'Medium', 'Tier 1',
'Supermarket Type1', 3735.138 )
 ('DRC01',  5.92, 'Regular', 0.01927822, 'Soft Drinks',  48.2692, 'OUT018', 2009, 'Medium',
'Tier 3', 'Supermarket Type2',  443.4228)
 ('FDN15', 17.5 , 'Low Fat', 0.01676007, 'Meat', 141.618 , 'OUT049', 1999, 'Medium', 'Tier 1',
'Supermarket Type1', 2097.27  )
 ...
 ('NCJ29', 10.6 , 'Low Fat', 0.03518627, 'Health and Hygiene',  85.1224, 'OUT035', 2004,
'Small', 'Tier 2', 'Supermarket Type1', 1193.1136)
 ('FDN46',  7.21, 'Regular', 0.14522065, 'Snack Foods', 103.1332, 'OUT018', 2009, 'Medium',
'Tier 3', 'Supermarket Type2', 1845.5976)
 ('DRG01', 14.8 , 'Low Fat', 0.04487828, 'Soft Drinks',  75.467 , 'OUT046', 1997, 'Small', 'Tier
1', 'Supermarket Type1',  765.67  )]
```

3 Determine the number of rows and columns in the dataset.

```
print(data.shape)

num_rows = data.shape[0]
num_cols = len(data.dtype.names)
print(f"Number of rows: {num_rows}, Number of columns: {num_cols}")
```

Output:

```
(8523,)
Number of rows: 8523, Number of columns: 12
```

4 Calculate the minimum, maximum, mean, and standard deviation for numerical columns.

```
# I have manually given the columns with numerical values for this one sir
numerical_columns = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales']

for col in numerical_columns:
    print(f"{col} - Min: {np.nanmin(data[col])}, Max: {np.nanmax(data[col])}, Mean: {np.nanmean(data[col])}, Std: {np.nanstd(data[col])}")
```

Output:

```
Item_Weight
- Min: 4.555, Max: 21.35, Mean: 12.857645184135976, Std: 4.643127630847946
Item_Visibility
- Min: 0.0, Max: 0.328390948, Mean: 0.06613202877895108, Std: 0.051594795256961846
Item_MRP
- Min: 31.29, Max: 266.8884, Mean: 140.9927819781767, Std: 62.271413051361165
Item_Outlet_Sales
- Min: 33.29, Max: 13086.9648, Mean: 2181.288913575032, Std: 1706.3995013565955
```

5 Count the no. of unique values in a categorical column like Item_Type.

```
# I have printed the count & the names of it here sir
unique_item_types = np.unique(data['Item_Type'])

print(f"Number of unique Item Types: {len(unique_item_types)}")

print(unique_item_types,end='')
```

6 Filter and display all rows where Item_Fat_Content is Low Fat.

```
low_fat_rows = data[data['Item_Fat_Content'] == 'Low Fat']
print(low_fat_rows)
```

7 Extract only the Item_MRP and Item_Outlet_Sales columns.

```
print(data[['Item_MRP', 'Item_Outlet_Sales']])
```

8 Find the difference between the maximum and minimum item weights.

```
weight_diff = np.nanmax(data['Item_Weight']) - np.nanmin(data['Item_Weight'])
print(f"Difference between max and min Item Weights: {weight_diff}")
```

Output:
Difference between max and min Item Weights: 16.795

9 Count how many items are sold in each Outlet_Type.

```
unique_outlet_types, counts = np.unique(data['Outlet_Type'], return_counts=True)
for outlet_type, count in zip(unique_outlet_types, counts):
    print(f"{outlet_type}: {count}")
```

Output:
Grocery Store: 1083
Supermarket Type1: 5577
Supermarket Type2: 928
Grocery Store: 1083
Supermarket Type1: 5577
Supermarket Type2: 928
Supermarket Type2: 928
Supermarket Type3: 935
Supermarket Type3: 935

10 Identify the items with the highest and lowest Item_Outlet_Sales.

```
max_sales_item = data[np.argmax(data['Item_Outlet_Sales'])]
min_sales_item = data[np.argmin(data['Item_Outlet_Sales'])]
print(f"Item with highest sales: {max_sales_item}")
print(f"Item with lowest sales: {min_sales_item}")
```

Output:
Item with highest sales: ('NCE42', nan, 'Low Fat', 0.01055095, 'Household', 234.9958, 'OUT027', 1985, 'Medium', 'Tier 3', 'Supermarket Type3', 13086.9648)

Item with lowest sales: ('DRK12', 9.5, 'Low Fat', 0.0, 'Soft Drinks', 32.89, 'OUT010', 1998, '', 'Tier 3', 'Grocery Store', 33.29)

11 Check if any column has missing values.

Method 1:

```
missing_values = {col: np.isnan(data[col]).sum() for col in data.dtype.names if data[col].dtype.kind in 'f'}
print(missing_values)
```

Output:
{'Item_Weight': np.int64(1463), 'Item_Visibility': np.int64(0), 'Item_MRP': np.int64(0), 'Item_Outlet_Sales': np.int64(0)}

Method 2:

```
missing_values = {}

for col in data.dtype.names:
    if data[col].dtype.kind in 'f':  # Check if the column is of a float type
        missing_count = np.isnan(data[col]).sum()
        missing_values[col] = missing_count

for col, count in missing_values.items():
    print(f"{col}: {count} missing values")
```

Output:
Item_Weight: 1463 missing values
Item_Visibility: 0 missing values
Item_MRP: 0 missing values
Item_Outlet_Sales: 0 missing values

12 Calculate the total sales amount in the dataset.

```
total_sales = np.nansum(data['Item_Outlet_Sales'])
print(f"Total Sales Amount: {total_sales}")
```

Output: Total Sales Amount: 18591125.4104

13 Count the no. of rows with missing values in the Item_Weight column.

```
missing_weights = np.isnan(data['Item_Weight']).sum()
print(f"Number of rows with missing Item_Weight: {missing_weights}")
```

Output: Number of rows with missing Item_Weight: 1463

14 Find the items with the highest Item_MRP value.

```
max_mrp_item = data[np.argmax(data['Item_MRP'])]
print(f"Item with highest MRP: {max_mrp_item}")
```

Output: Item with highest MRP: ('FDS13', 6.465, 'Low Fat', 0.125210375, 'Canned', 266.8884, 'OUT017', 2007, '', 'Tier 2', 'Supermarket Type1', 1059.9536)

15 Calculate the average sales amount for each outlet.

```
unique_outlets = np.unique(data['Outlet_Identifier'])
for outlet in unique_outlets:
    avg_sales = np.nanmean(data['Item_Outlet_Sales'][data['Outlet_Identifier'] == outlet])
    print(f"Outlet {outlet} - Average Sales: {avg_sales}")
```

Output:
Outlet OUT010 - Average Sales: 339.351661981982
Outlet OUT013 - Average Sales: 2298.9952555793993
Outlet OUT017 - Average Sales: 2340.6752634989202
Outlet OUT018 - Average Sales: 1995.498739224138
Outlet OUT019 - Average Sales: 340.3297227272728
Outlet OUT027 - Average Sales: 3694.038557647059
Outlet OUT035 - Average Sales: 2438.841866021505
Outlet OUT045 - Average Sales: 2192.384797631862
Outlet OUT046 - Average Sales: 2277.8442668817206
Outlet OUT049 - Average Sales: 2348.3546346236562

16 Identify the top 5 outlets with the highest total sales.

Method 1:

```
unique_outlets = np.unique(data['Outlet_Identifier'])
total_sales_by_outlet = {outlet: np.nansum(data['Item_Outlet_Sales'][data['Outlet_Identifier']
== outlet]) for outlet in unique_outlets}
sorted_outlets = sorted(total_sales_by_outlet.items(), key=lambda x: x[1], reverse=True)[:5]
print(f"Top 5 outlets by total sales: {sorted_outlets}")
```

Output:
Top 5 outlets by total sales:
[(np.str_('OUT027'), np.float64(3453926.0514)), (np.str_('OUT035'), np.float64(2268122.9354)), (np.str_('OUT049'), np.float64(2183969.8102)), (np.str_('OUT017'), np.float64(2167465.294)), (np.str_('OUT013'), np.float64(2142663.5782000003))]

Method 2:

```
for outlet in np.unique(data['Outlet_Identifier']):
    total_sales_by_outlet[outlet] = np.nansum(data['Item_Outlet_Sales'][data['Outlet_Identifier'] == outlet])


sorted_outlets = sorted(total_sales_by_outlet.items(), key=lambda x: x[1], reverse=True)[:5]


for outlet, sales in sorted_outlets:
    print(f"Outlet {outlet} - Total Sales: {sales}")
```

Output:

```
Outlet OUT027 - Total Sales: 3453926.0514
Outlet OUT035 - Total Sales: 2268122.9354
Outlet OUT049 - Total Sales: 2183969.8102
Outlet OUT017 - Total Sales: 2167465.294
Outlet OUT013 - Total Sales: 2142663.5782000003
```

17 Analyze how sales have grown over the years.

```
unique_years = np.unique(data['Outlet_Establishment_Year'])
for year in unique_years:
    total_sales_year = np.nansum(data['Item_Outlet_Sales'][data['Outlet_Establishment_Year'] == year])
    print(f"Year {year} - Total Sales: {total_sales_year}")
```

Output:

```
Year 1985 - Total Sales: 3633620.145
Year 1987 - Total Sales: 2142663.5782000003
Year 1997 - Total Sales: 2118395.1682
Year 1998 - Total Sales: 188340.17239999998
Year 1999 - Total Sales: 2183969.8102
Year 2002 - Total Sales: 2036725.477
Year 2004 - Total Sales: 2268122.9354
Year 2007 - Total Sales: 2167465.294
Year 2009 - Total Sales: 1851822.83
```

18 Identify outliers in Item_MRP.

```
mrp_mean = np.nanmean(data['Item_MRP'])
mrp_std = np.nanstd(data['Item_MRP'])

upper_bound = mrp_mean + 3 * mrp_std
lower_bound = mrp_mean - 3 * mrp_std

outliers = data[(data['Item_MRP'] > upper_bound) | (data['Item_MRP'] < lower_bound)]

print("Outliers in Item MRP:")
print(outliers)
```

Output: [] => **No Outliers**

19 Calculate total sales based on Outlet_Location_Type.

```
unique_locations = np.unique(data['Outlet_Location_Type'])
for location in unique_locations:
    total_sales_location = np.nansum(data['Item_Outlet_Sales'][data['Outlet_Location_Type']
== location])
    print(f"Location {location} - Total Sales: {total_sales_location}")
```

Output:
Location Tier 1 - Total Sales: 4482059.072000001
Location Tier 2 - Total Sales: 6472313.7064000005
Location Tier 3 - Total Sales: 7636752.631999999

20 Calculate the average Item_Visibility for each Item_Type.

```
unique_item_types = np.unique(data['Item_Type'])
for item_type in unique_item_types:
    avg_visibility = np.nanmean(data['Item_Visibility'][data['Item_Type'] == item_type])
    print(f"Item Type {item_type} - Average Visibility: {avg_visibility}")
```

Output:

```
Item Type Baking Goods - Average Visibility: 0.06916929969598766
Item Type Breads - Average Visibility: 0.06625509788047809
Item Type Breakfast - Average Visibility: 0.08572300932727274
Item Type Canned - Average Visibility: 0.06812931539445301
Item Type Dairy - Average Visibility: 0.07242719825219941
Item Type Frozen Foods - Average Visibility: 0.06564523898481309
Item Type Fruits and Vegetables - Average Visibility: 0.06851294287337663
Item Type Hard Drinks - Average Visibility: 0.06494255579906542
Item Type Health and Hygiene - Average Visibility: 0.05521597975384616
Item Type Household - Average Visibility: 0.061322312856043955
Item Type Meat - Average Visibility: 0.06228381108705883
Item Type Others - Average Visibility: 0.06024103188165681
Item Type Seafood - Average Visibility: 0.074976079734375
Item Type Snack Foods - Average Visibility: 0.0668502227675
Item Type Soft Drinks - Average Visibility: 0.06397224782696628
Item Type Starchy Foods - Average Visibility: 0.0675635640337838
```

21 Calculate the percentage contribution of each item to the total sales.

```
total_sales = np.nansum(data['Item_Outlet_Sales'])

percentage_contributions = (data['Item_Outlet_Sales'] / total_sales) * 100

print("Percentage contribution of each item to total sales:")
for item, contribution in zip(data['Item_Identifier'], percentage_contributions):
    print(f"Item {item} contributes {contribution:.2f}% to total sales")
```

Output:

```
Item FDJ38 contributes 0.02% to total sales
Item FDG32 contributes 0.01% to total sales
Item DRQ35 contributes 0.02% to total sales
Item FDK25 contributes 0.00% to total sales
Item FDD58 contributes 0.01% to total sales
Item FDB15 contributes 0.02% to total sales
Item DRF60 contributes 0.03% to total sales
Item FDH04 contributes 0.00% to total sales
Item FDM04 contributes 0.01% to total sales…….
```

22 Analyze the total sales by different outlet types.

```
unique_outlet_types = np.unique(data['Outlet_Type'])
for outlet_type in unique_outlet_types:
    total_sales_outlet_type = np.nansum(data['Item_Outlet_Sales'][data['Outlet_Type'] ==
outlet_type])
    print(f"Outlet Type {outlet_type} - Total Sales: {total_sales_outlet_type}")
```

Output:

```
Outlet Type Grocery Store - Total Sales: 368034.266
Outlet Type Supermarket Type1 - Total Sales: 12917342.263
Outlet Type Supermarket Type2 - Total Sales: 1851822.83
Outlet Type Supermarket Type3 - Total Sales: 3453926.0514
```

23 Identify the item type with the highest total sales.

```
unique_item_types = np.unique(data['Item_Type'])
item_sales = {item_type: np.nansum(data['Item_Outlet_Sales'][data['Item_Type'] ==
item_type]) for item_type in unique_item_types}
highest_sales_item_type = max(item_sales, key=item_sales.get)
print(f"Item Type with highest sales: {highest_sales_item_type}")
```

Output:

```
Item Type with highest sales: Fruits and Vegetables
```

24 Analyze sales trends over months to identify any seasonal patterns.

Analyze sales trends over **years** - as months are not available, I have done using Years

```
years = np.unique(data['Outlet_Establishment_Year'])
sales_by_year = {year:
np.nansum(data['Item_Outlet_Sales'][data['Outlet_Establishment_Year'] == year]) for year in
years}

sorted_sales_by_year = sorted(sales_by_year.items())

print("Sales trends by Outlet Establishment Year:")
for year, sales in sorted_sales_by_year:
    print(f"Year {year}: Total Sales = {sales:.2f}")
```

Output:

```
Sales trends by Outlet Establishment Year:
Year 1985: Total Sales = 3633620.15
Year 1987: Total Sales = 2142663.58
Year 1997: Total Sales = 2118395.17
Year 1998: Total Sales = 188340.17
Year 1999: Total Sales = 2183969.81
Year 2002: Total Sales = 2036725.48
Year 2004: Total Sales = 2268122.94
Year 2007: Total Sales = 2167465.29
Year 2009: Total Sales = 1851822.83
```

25 Identify items with the highest profit margin if cost data is available.

Cost data not available so unable to find the highest profit margin

26 Calculate the average Item_Outlet_Sales for each Outlet_Size.

```
unique_outlet_sizes = np.unique(data['Outlet_Size'])
for outlet_size in unique_outlet_sizes:
    avg_sales_size = np.nanmean(data['Item_Outlet_Sales'][data['Outlet_Size'] == outlet_size])
    print(f"Outlet Size {outlet_size} - Average Sales: {avg_sales_size}")
```

Output:

```
Outlet Size  - Average Sales: 1822.6269474688797
Outlet Size High - Average Sales: 2298.9952555793993
Outlet Size Medium - Average Sales: 2681.603541568206
Outlet Size Small - Average Sales: 1912.1491613065327
```

27 Explore how Item_Visibility affects Item_Outlet_Sales.

Using Correlation to get the value between -1 and 1:

Positive correlation: Higher visibility might be associated with higher sales.

Negative correlation: Higher visibility might be associated with lower sales.

Near 0: No clear linear relationship.

```
correlation = np.corrcoef(data['Item_Visibility'], data['Item_Outlet_Sales'])[0, 1]
print(f"Correlation between Item Visibility and Item Outlet Sales: {correlation:.4f}")
```

Output:
Correlation between Item Visibility and Item Outlet Sales: -0.1286

Result : Negative correlation

28 Sum up the total sales for each Item_Type.

```
unique_item_types = np.unique(data['Item_Type'])
total_sales_by_item_type = {item_type:
np.nansum(data['Item_Outlet_Sales'][data['Item_Type'] == item_type]) for item_type in
unique_item_types}
for item_type, total_sales in total_sales_by_item_type.items():
    print(f"Item Type {item_type} - Total Sales: {total_sales}")
```

Output:
Item Type Baking Goods - Total Sales: 1265525.3421999998
Item Type Breads - Total Sales: 553237.1888
Item Type Breakfast - Total Sales: 232298.9516
Item Type Canned - Total Sales: 1444151.4926
Item Type Dairy - Total Sales: 1522594.0512
Item Type Frozen Foods - Total Sales: 1825734.7886
Item Type Fruits and Vegetables - Total Sales: 2820059.8168
Item Type Hard Drinks - Total Sales: 457793.42720000003
Item Type Health and Hygiene - Total Sales: 1045200.1378000001
Item Type Household - Total Sales: 2055493.7131999999
Item Type Meat - Total Sales: 917565.612
Item Type Others - Total Sales: 325517.6096
Item Type Seafood - Total Sales: 148868.2194
Item Type Snack Foods - Total Sales: 2732786.0870000003
Item Type Soft Drinks - Total Sales: 892897.7220000001
Item Type Starchy Foods - Total Sales: 351401.25039999996

29 Calculate the median Item_MRP for each Outlet_Type.

```
unique_outlet_types = np.unique(data['Outlet_Type'])
for outlet_type in unique_outlet_types:
    median_mrp = np.nanmedian(data['Item_MRP'][data['Outlet_Type'] == outlet_type])
    print(f"Outlet Type {outlet_type} - Median MRP: {median_mrp}")
```

Output:

```
Outlet Type Grocery Store - Median MRP: 143.9128
Outlet Type Supermarket Type1 - Median MRP: 143.1154
Outlet Type Supermarket Type2 - Median MRP: 140.5667
Outlet Type Supermarket Type3 - Median MRP: 143.7154
```

30 Find the most frequently occurring Item_Fat_Content category.

```
unique_fat_contents, counts = np.unique(data['Item_Fat_Content'], return_counts=True)
most_frequent_fat_content = unique_fat_contents[np.argmax(counts)]
print(f"Most frequently occurring Item Fat Content: {most_frequent_fat_content}")
```

Output:

```
Most frequently occurring Item Fat Content: Low Fat
```