Import Neccessary Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
```

Assign Path Variable to the Path of the Data

In [2]:
```
base_path = 'data_test.csv'
```

Read the File, and let's go through it.

In [3]:
```
df = pd.read_csv(base_path)
```

Undesrtanding the Data

- Top 10 rows

In [4]:
```
df.head(10)
```

Out[4]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT017	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	
5	FDH56	9.800	Regular	0.063817	Fruits and Vegetables	117.1492	OUT046	
6	FDL48	19.350	Regular	0.082602	Baking Goods	50.1034	OUT018	
7	FDC48	NaN	Low Fat	0.015782	Baking Goods	81.0592	OUT027	
8	FDN33	6.305	Regular	0.123365	Snack Foods	95.7436	OUT045	
9	FDA36	5.985	Low Fat	0.005698	Baking Goods	186.8924	OUT017	

- Find Shape

In [5]:
```
df.shape
```

Out[5]:
```
(5681, 11)
```

- Find Information

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            5681 non-null   object
 1   Item_Weight                4705 non-null   float64
 2   Item_Fat_Content           5681 non-null   object
 3   Item_Visibility            5681 non-null   float64
 4   Item_Type                  5681 non-null   object
 5   Item_MRP                   5681 non-null   float64
 6   Outlet_Identifier          5681 non-null   object
 7   Outlet_Establishment_Year  5681 non-null   int64
 8   Outlet_Size                4075 non-null   object
 9   Outlet_Location_Type       5681 non-null   object
 10  Outlet_Type                5681 non-null   object
dtypes: float64(3), int64(1), object(7)
memory usage: 488.3+ KB
```

- Identify Columns

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Item_Identifier', 'Item_Weight', 'Item_Fat_Content', 'Item_Visibility',
       'Item_Type', 'Item_MRP', 'Outlet_Identifier',
       'Outlet_Establishment_Year', 'Outlet_Size', 'Outlet_Location_Type',
       'Outlet_Type'],
      dtype='object')
```

- Find Data types

In [8]:
```
df.dtypes
```

Out[8]:
```
Item_Identifier               object
Item_Weight                  float64
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Size                   object
Outlet_Location_Type          object
Outlet_Type                   object
dtype: object
```

In [9]:
```
df.select_dtypes(include=['int64','float'])
```

Out[9]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
0	20.750	0.007565	107.8622	1999
1	8.300	0.038428	87.3198	2007
2	14.600	0.099575	241.7538	1998
3	7.315	0.015388	155.0340	2007
4	NaN	0.118599	234.2300	1985
...	...	...	...	...
5676	10.500	0.013496	141.3154	1997
5677	7.600	0.142991	169.1448	2009
5678	10.000	0.073529	118.7440	2002
5679	15.300	0.000000	214.6218	2007
5680	9.500	0.104720	79.7960	2002

5681 rows × 4 columns

In [10]:
```
df.select_dtypes(include=['object'])
```

Out[10]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_Type	Outlet_Type
0	FDW58	Low Fat	Snack Foods	OUT049	Medium	Tier 1	Supermarket Type1
1	FDW14	reg	Dairy	OUT017	NaN	Tier 2	Supermarket Type1
2	NCN55	Low Fat	Others	OUT010	NaN	Tier 3	Grocery Store
3	FDQ58	Low Fat	Snack Foods	OUT017	NaN	Tier 2	Supermarket Type1
4	FDY38	Regular	Dairy	OUT027	Medium	Tier 3	Supermarket Type3
...	...	...	...	...	...	...	...
5676	FDB58	Regular	Snack Foods	OUT046	Small	Tier 1	Supermarket Type1
5677	FDD47	Regular	Starchy Foods	OUT018	Medium	Tier 3	Supermarket Type2
5678	NCO17	Low Fat	Health and Hygiene	OUT045	NaN	Tier 2	Supermarket Type1
5679	FDJ26	Regular	Canned	OUT017	NaN	Tier 2	Supermarket Type1
5680	FDU37	Regular	Canned	OUT045	NaN	Tier 2	Supermarket Type1

5681 rows × 7 columns

Data Cleaning

Handling Missing Values

- There are missing values in the `Item_Weight` and `Outlet_Size` columns.
- Filling the missing values with mean and mode respectively.

`Item_Weight:`

Filling missing values with the mean weight of the items.

In [11]:
```
df['Item_Weight'].fillna(df['Item_Weight'].mean(),inplace=True)
```

`Outlet_Size:`

Since `Outlet_Size` is categorical, you can fill missing values with the most frequent category (mode).

In [12]:
```
df['Outlet_Size'].unique()
```

Out[12]:
```
array(['Medium', nan, 'Small', 'High'], dtype=object)
```

In [13]:
```
df['Outlet_Size'].fillna(df['Outlet_Size'].mode()[0],inplace=True)
```

In [14]:
```
df.isna().sum()
```

Out[14]:
```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
dtype: int64
```

Standardizing Values

`Item_Fat_Content:`

The `Item_Fat_Content` column has inconsistent values. We should standardize these values.

In [15]:
```
df['Item_Fat_Content'].unique()
```

Out[15]:
```
array(['Low Fat', 'reg', 'Regular', 'LF', 'low fat'], dtype=object)
```

In [16]:
```
df['Item_Fat_Content'] = df['Item_Fat_Content'].replace({'LF':'Low Fat','low fat':'Low Fat','reg':'Regular'})
```

In [17]:
```
df['Item_Fat_Content'].unique()
```

Out[17]:
```
array(['Low Fat', 'Regular'], dtype=object)
```

Exploratory Data Analysis

Summarised Statistics

- Summarised Stats for the numerical columns

In [18]:
```
df.describe().transpose().round(2)
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Item_Weight	5681.0	12.70	4.25	4.56	9.20	12.70	15.85	21.35
Item_Visibility	5681.0	0.07	0.05	0.00	0.03	0.05	0.09	0.32
Item_MRP	5681.0	141.02	61.81	31.99	94.41	141.42	186.03	266.59
Outlet_Establishment_Year	5681.0	1997.83	8.37	1985.00	1987.00	1999.00	2004.00	2009.00

Key Understandings from this:

- The average visibility of items is 0.07, with a significant variability (standard deviation of 0.05)
- The average weight of items is 12.70 units.
- The average MRP is 141.02, with a wide range of prices from 31.99 to 266.59.

- Summarised Stats for the numerical columns

In [19]:
```
df.describe(include=['O']).transpose()
```

Out[19]:

	count	unique	top	freq
Item_Identifier	5681	1543	DRF48	8
Item_Fat_Content	5681	2	Low Fat	3668
Item_Type	5681	16	Snack Foods	789
Outlet_Identifier	5681	10	OUT027	624
Outlet_Size	5681	3	Medium	3468
Outlet_Location_Type	5681	3	Tier 3	2233
Outlet_Type	5681	4	Supermarket Type1	3717

Key Understandings from this:

- The dataset includes 5681 entries.
- The items' fat content is predominantly **Low Fat.**
- The most common item types are **Snack Foods**.
- Most outlets are of **Medium size** and are located in **Tier 3** locations.
- **'Supermarket Type1'** is the most common outlet type.

Which items have the highest and lowest sales?

In [20]:
```
# Grouping by unique items throughout the data and finding the total sale of each unique item.
high_low_sorted = df.groupby('Item_Identifier')['Item_MRP'].sum()
high_low_sorted = high_low_sorted.reset_index()

# Re-defining the columns
high_low_sorted.columns = ['Item_Identifier', 'Total_sale']

# Sorting the unique items based on their total sale value
high_low_sorted = high_low_sorted.sort_values(by='Total_sale',ascending=False).reset_index().drop('index',axis=1)
```

Top 3 items by their sales

In [21]:
```
high_low_sorted.head(3)
```

Out[21]:

	Item_Identifier	Total_sale
0	FDP15	2055.5640
1	DRM11	1825.1946
2	NCQ29	1823.9946

Last 3 items by their sales

In [22]:
```
high_low_sorted.tail(3)
```

Out[22]:

	Item_Identifier	Total_sale
1540	NCA06	35.419
1541	FDR43	34.819
1542	DRK12	32.390

Items that made Highest and Lowest sale.

In [23]:
```
highest_sold_item = high_low_sorted['Item_Identifier'].iloc[0]

lowest_sold_item = high_low_sorted['Item_Identifier'].iloc[-1]

print(f"\nThe highest sold item is {highest_sold_item}. \nThe lowest sold item is {lowest_sold_item}\n")
```

```
The highest sold item is FDP15.
The lowest sold item is DRK12
```

In [ ]:

In [24]:
```
df[df['Item_Identifier']=='FDP15']
```

Out[24]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishm
1098	FDP15	15.200000	Low Fat	0.084073	Meat	255.533	OUT049	
1466	FDP15	15.200000	Low Fat	0.084285	Meat	254.933	OUT018	
2001	FDP15	12.695633	Low Fat	0.146973	Meat	256.333	OUT019	
3525	FDP15	15.200000	Low Fat	0.084113	Meat	257.733	OUT045	
3692	FDP15	15.200000	Low Fat	0.083927	Meat	257.133	OUT035	
4866	FDP15	15.200000	Low Fat	0.084417	Meat	258.233	OUT017	
5410	FDP15	15.200000	Low Fat	0.000000	Meat	258.233	OUT013	
5433	FDP15	15.200000	Low Fat	0.083943	Meat	257.433	OUT046	

Which outlets generate the most and least revenue?

In [25]:
```
# Grouping every Outlet Identifier by their total sale
outlet_sales = df.groupby('Outlet_Identifier')['Item_MRP'].sum(numeric_only=True).reset_index()

# Setting columns
outlet_sales.columns = ['Outlet_Identifier', 'Outlet_Total_Sale']

# Sorting by total sale
outlet_sales = outlet_sales.sort_values('Outlet_Total_Sale',ascending=False).reset_index(drop=True)

outlet_sales
```

Out[25]:

	Outlet_Identifier	Outlet_Total_Sale
0	OUT027	89123.5742
1	OUT017	88457.3936
2	OUT049	88326.0990
3	OUT013	87363.4336
4	OUT045	87104.3850
5	OUT018	86509.6182
6	OUT046	86160.7018
7	OUT035	85612.2960
8	OUT010	52441.1972
9	OUT019	50054.5176

Top 3 Outlets with Most Sales

In [26]:
```
outlet_sales.head(3)
```

Out[26]:

	Outlet_Identifier	Outlet_Total_Sale
0	OUT027	89123.5742
1	OUT017	88457.3936
2	OUT049	88326.0990

Bottom 3 Outlets with Least Sales

In [27]:
```
outlet_sales.tail(3)
```

Out[27]:

	Outlet_Identifier	Outlet_Total_Sale
7	OUT035	85612.2960
8	OUT010	52441.1972
9	OUT019	50054.5176

Outlets with Highest and Lowest Sale.

In [28]:
```
x = outlet_sales['Outlet_Identifier'].iloc[0]
x1 = outlet_sales['Outlet_Total_Sale'].iloc[0]
y = outlet_sales['Outlet_Identifier'].iloc[-1]
y1 = outlet_sales['Outlet_Total_Sale'].iloc[-1]
print(f"\nOutlet {x} made highest sale of {x1} \nOutlet {y} made lowest sale of {y1}\n<b>")
```

```
Outlet OUT027 made highest sale of 89123.5742
Outlet OUT019 made lowest sale of 50054.5176
<b>
```

How does the pricing of items affect their sales?

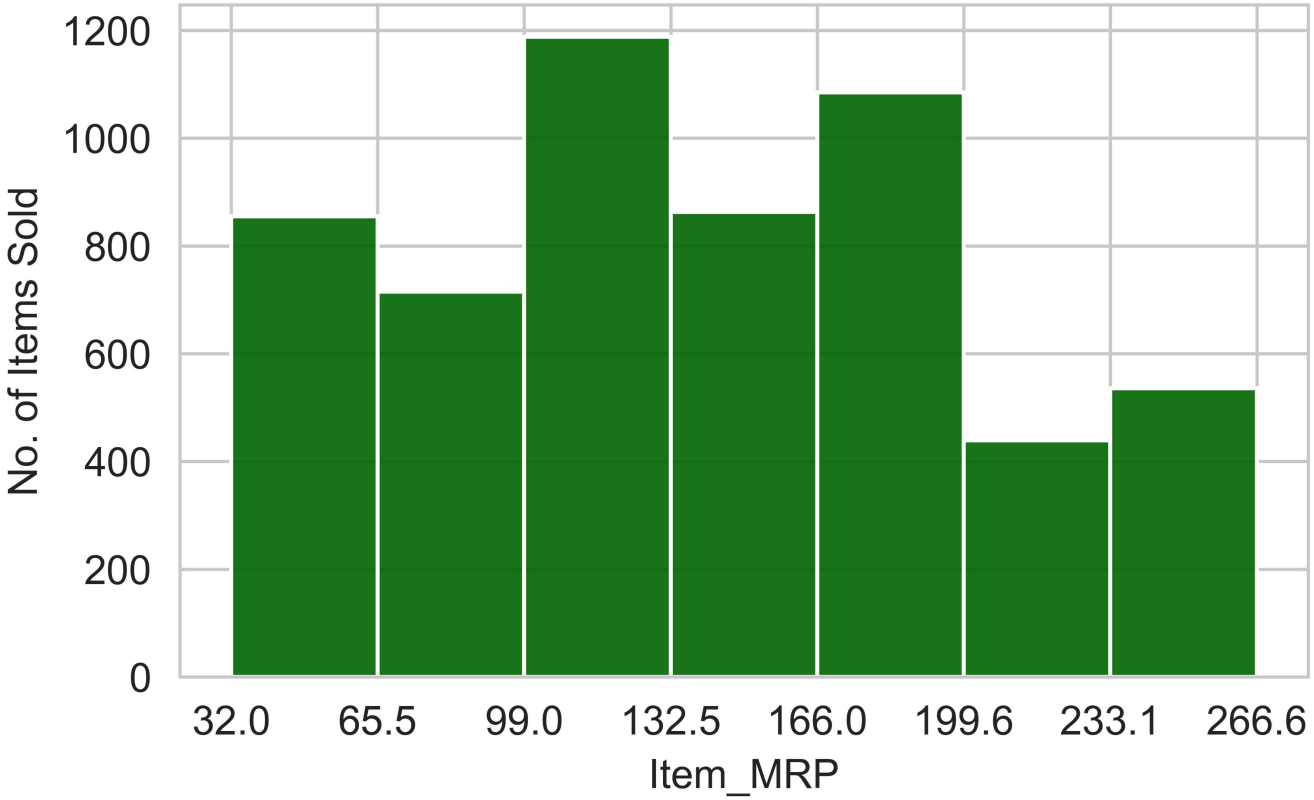
In [29]:
```
plt.figure(figsize=(5,3),dpi=1000)

hist = sns.histplot(data=df,x='Item_MRP',bins=7,color='darkgreen',alpha=0.9)

bin_edges = hist.patches[0].get_bbox().get_points()[:,0]
for patch in hist.patches:
    bin_edges = np.append(bin_edges,patch.get_bbox().get_points()[1,0])
plt.xticks(bin_edges)

plt.ylabel('No. of Items Sold');
```

```
C:\Users\Ram\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

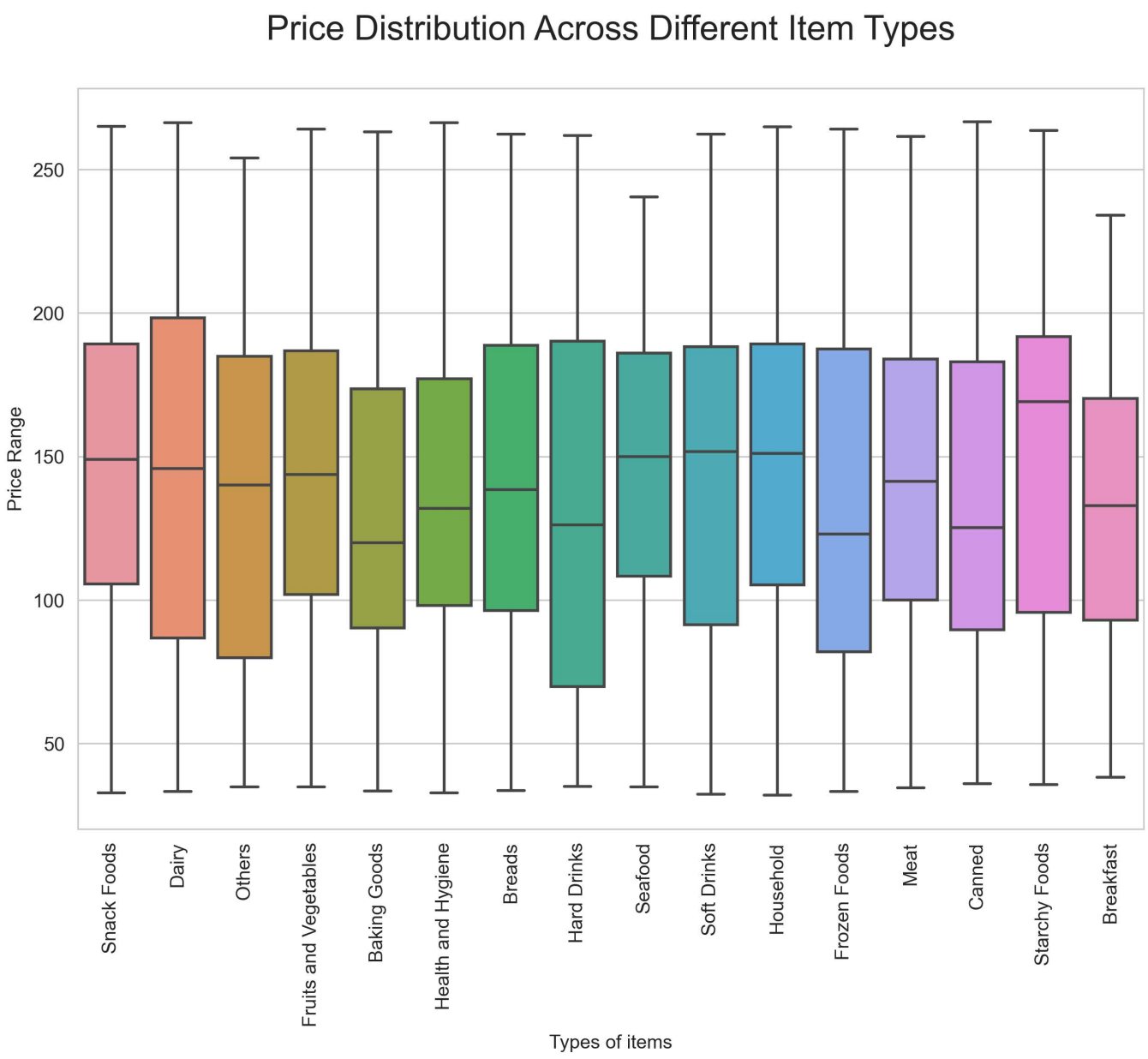


Key Understandings from this:

- The **highest number of items sold** is for items with MRP around 99 to 133. This range shows the tallest bars, indicating that items in this price range are more popular or in higher demand.
- Items with MRP around 200 and 267 have relatively **lower sales** compared to the adjacent price ranges.
- These observations suggest that mid-priced items (around 100 to 150 MRP) tend to sell the most, while very low or very high-priced items have less sales volume.

What is the price distribution across different item types?

In [30]:
```
plt.figure(figsize=(10, 7),dpi=300)
sns.boxplot(x='Item_Type', y='Item_MRP', data=df)
plt.xticks(rotation=90)
plt.title('Price Distribution Across Different Item Types\n',fontsize=18)
plt.xlabel('Types of items')
plt.ylabel('Price Range');
```

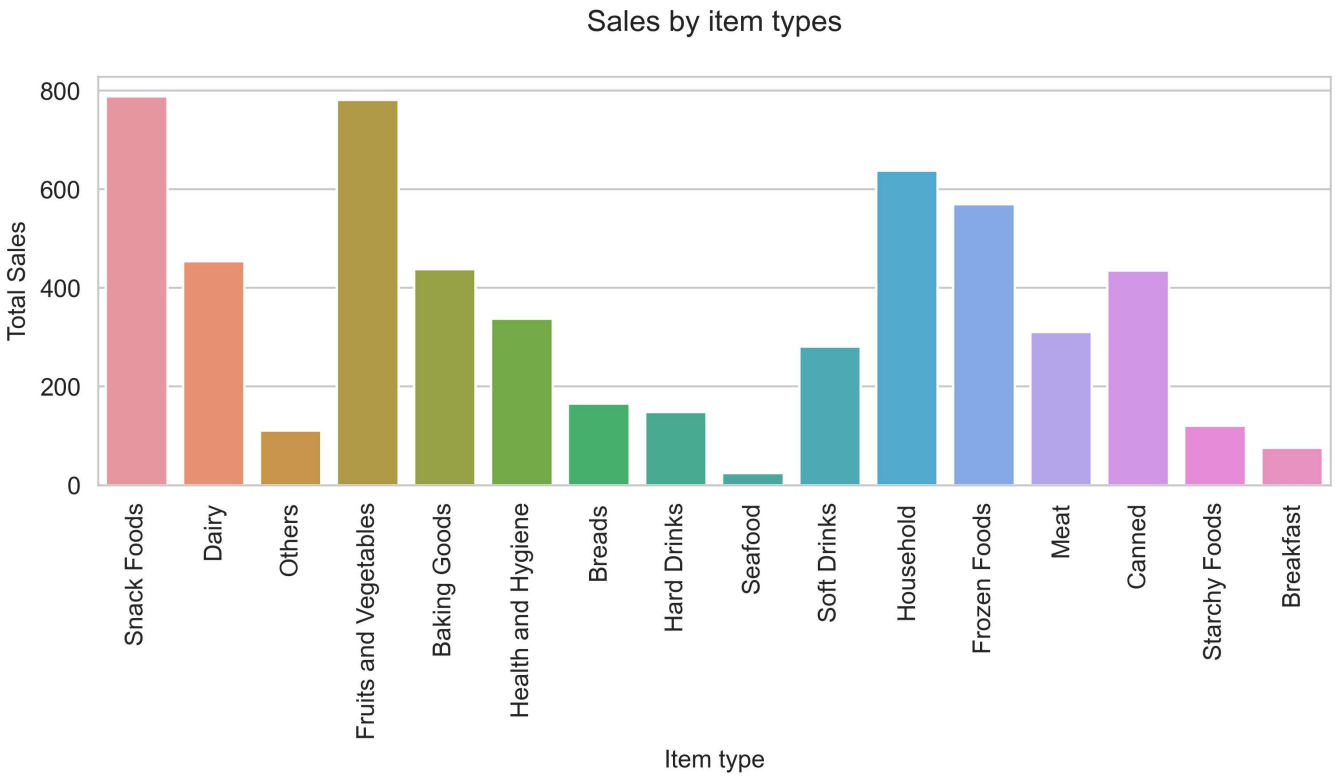


Key Understandings from this:

- The price distribution of items across different types is relatively consistent, with most categories showing a median price around 100-150.
- Categories like **Health and Hygiene** and **Fruits and Vegetables** have higher IQR(which represents the middle 50% of the data).

What are the popular item types among customers?

In [31]:
```
plt.figure(figsize=(9,3),dpi=400)
sns.countplot(data=df,x='Item_Type')
plt.xticks(rotation=90)
plt.xlabel('Item type')
plt.ylabel('Total Sales')
plt.title('Sales by item types\n');
```



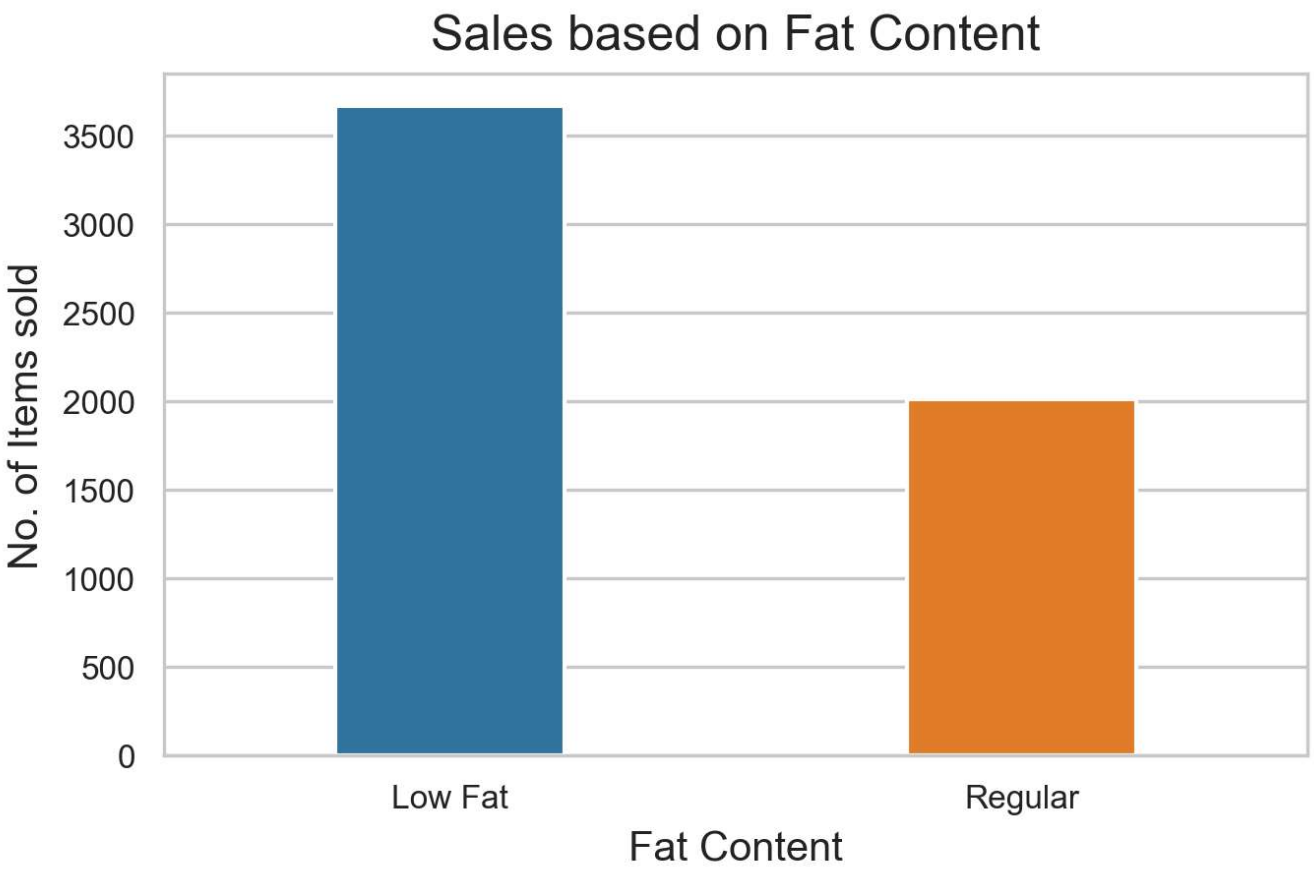
Key Understandings from this:

- The most popular item types among customers are **Snack Foods**, **Fruits and Vegetables**. These categories consistently show higher sales volumes, indicating strong customer preference and demand.
- While **Seafood** being less popular among the costumers

Do customers prefer low-fat or regular items?

In [32]:
```
plt.figure(dpi=300,figsize=(5,3))

sns.countplot(data=df,x='Item_Fat_Content',width=0.4)
plt.title('Sales based on Fat Content')
plt.xlabel('Fat Content')
plt.ylabel('No. of Items sold')
plt.tick_params(labelsize=8);
```



Key Understandings from this:

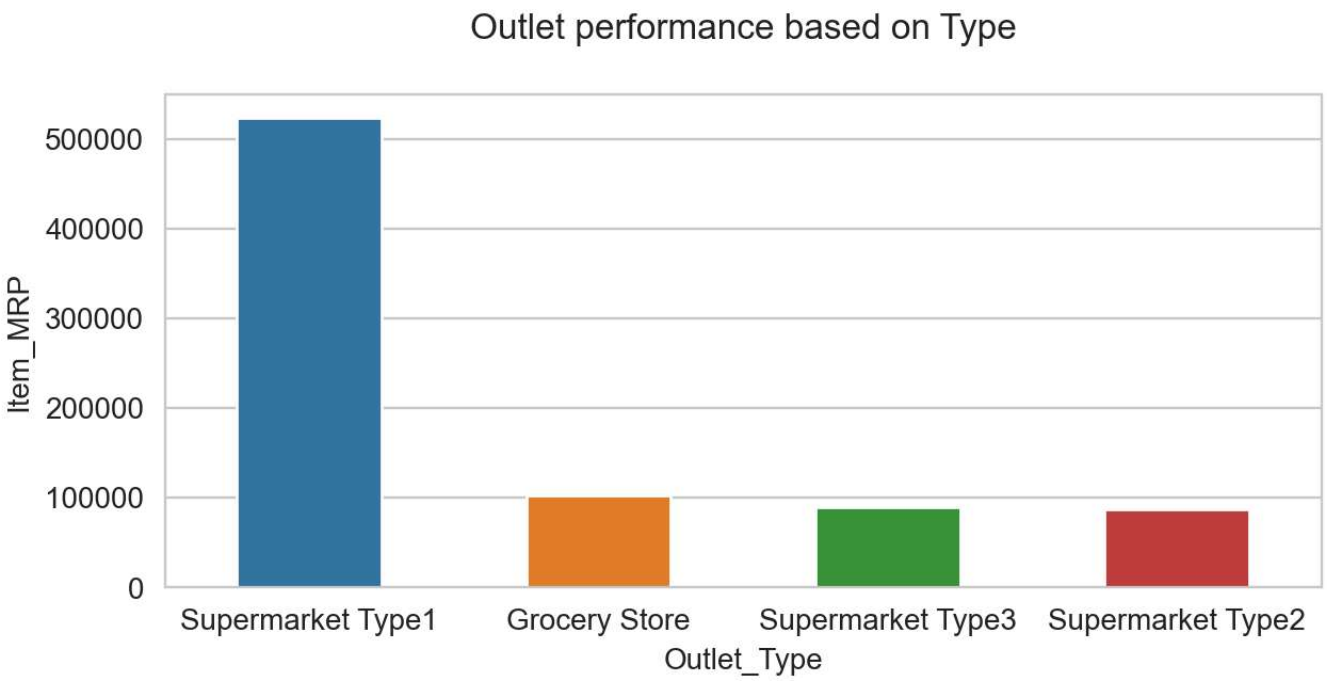
- It is clearly evident that constumers mostly prefer items with Low fat content.

How do different outlet types perform in terms of sales?

In [33]:
```
plt.figure(figsize=(7,3),dpi=200)

sns.barplot(data=df,x='Outlet_Type',y='Item_MRP',estimator=np.sum,errorbar=None,width=0.5)

plt.title("Outlet performance based on Type\n");
```



Key Understandings from this:

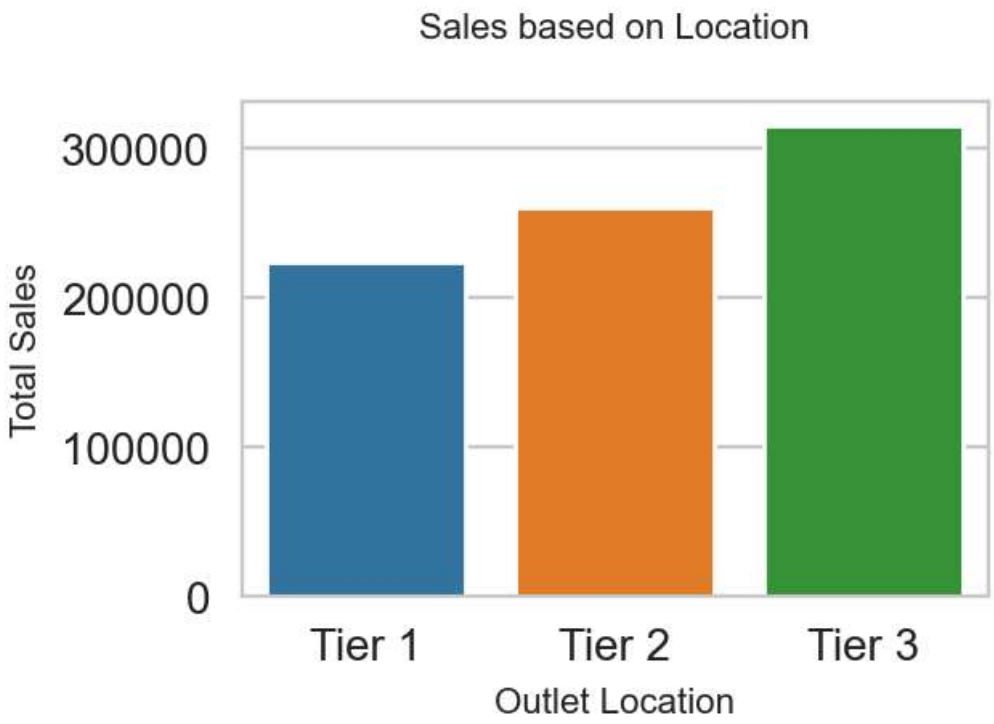
- The Supermarket Type 1 Outlets made the highest sales among all other outlet types.
- While the other three outlet types made relatively equal sale.

Does the location type of an outlet impact its sales?

In [34]:
```
plt.figure(figsize=(3,2),dpi=200)

sns.barplot(data=df,x='Outlet_Location_Type',y='Item_MRP',estimator=np.sum,errorbar=None)

plt.title('Sales based on Location\n',fontsize=8)
plt.ylabel('Total Sales',fontsize=8)
plt.xlabel('Outlet Location',fontsize=8);
```



Key Understandings from this:

- Outlets that are in Tier 3 cities show the best performance.
- While Tier 2 and Tier 1 follow respectively.

Which items have the highest visibility and how does it affect sales?

In [35]:
```
#Grouping by the item type to get it total sales and average visibilty.
x = df.groupby('Item_Identifier').agg({'Item_Visibility':'mean','Item_MRP':'sum'})
x = x.sort_values(by='Item_Visibility')
```

In [36]:
```
plt.figure(figsize=(3,3),dpi=300)

sns.scatterplot(data=x,y='Item_MRP',x='Item_Visibility',s=10)

plt.ylabel('Total Sale ',fontsize=8)
plt.xlabel('Item Visibility',fontsize=8)
plt.tick_params(labelsize=6)
plt.title('Relation between Visibility and Sales',fontsize=8);
```

- **No Strong Correlation:** The scatter plot does not show a strong correlation between Item Visibility and Sale of the Item. This suggests that the visibility of an item does not have a significant impact on its sales.
- **Further Analysis Needed:** While the scatter plot provides an initial visual understanding, further statistical analysis (like calculating the correlation coefficient) would be necessary to understand the relationship.

Finding the correlation between Item Visibility and its Total sale

In [37]:
```
x['Item_Visibility'].corr(x['Item_MRP'])
```

Out[37]:
```
-0.014590853436874081
```

Key Understandings from this:

- **Very Weak Negative Correlation:**

The correlation is very close to 0, indicating an extremely weak relationship between Item Visibility and it`s Sales. The negative sign indicates that there is a slight inverse relationship, but the magnitude is so small that it is almost negligible.

- Other factors might influence Total Sale more strongly than visibility. It's important to consider other variables, such as item type, outlet type, or promotional strategies, in a more comprehensive analysis.

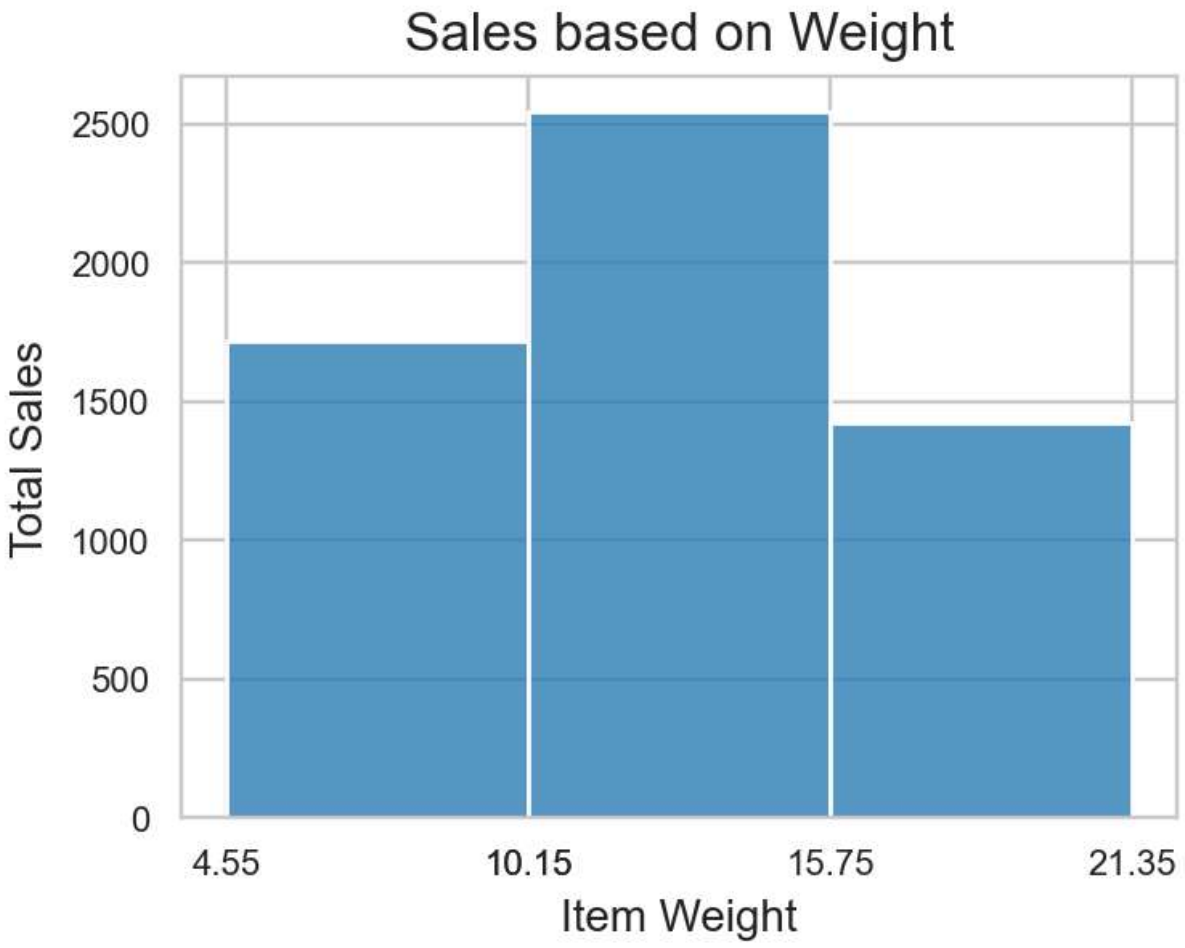
Are there any trends in item weight and sales?

In [38]:
```
plt.figure(figsize=(4,3),dpi=200)
hist = sns.histplot(data=df,x='Item_Weight',bins=3);

bin_edges = hist.patches[0].get_bbox().get_points()[:, 0]
for patch in hist.patches:
    bin_edges = np.append(bin_edges, patch.get_bbox().get_points()[1, 0])
plt.xticks(bin_edges);

plt.xlabel('Item Weight')
plt.ylabel('Total Sales')
plt.title('Sales based on Weight')
plt.tick_params(labelsize=8);
```

```
C:\Users\Ram\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated
and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Key Understandings from this:

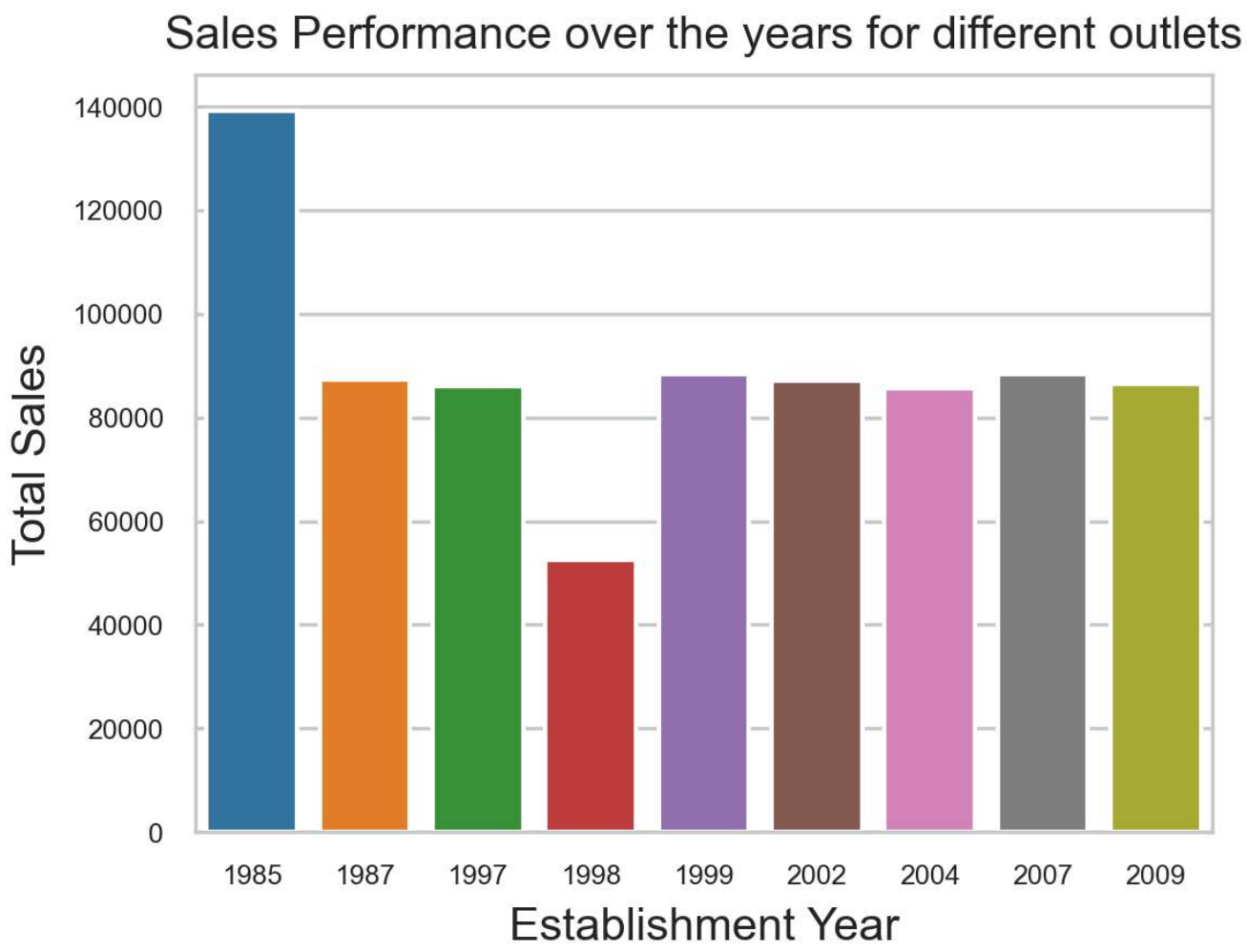
- Items that weigh in between 10 to 15 grams are mostly sold.
- This may be the preferreble amount for the costumer.

Sales Performance over the years for different outlets

In [39]:
```
plt.figure(figsize=(4,3),dpi=300)

sns.barplot(data=df,x='Outlet_Establishment_Year',y='Item_MRP',estimator=np.sum,errorbar=None)

plt.title('Sales Performance over the years for different outlets',fontsize=10)
plt.xlabel('Establishment Year')
plt.ylabel('Total Sales')
plt.tick_params(labelsize=6);
```



Key Understandings from this:

- Most of the Outlets have shown consistent sales.
- The oldest Outlet have made the highest sale of all time.
- While the Outlet the was established in 1998 is **underperforming**.

Recommendations for Sales Performance Improvement

Based on the key understandings from the analysis, here are some recommendations to improve sales performance:

---

1. **Improve the 1998 Outlet**:
   - **Check What's Wrong**: Take a closer look at the outlet from 1998 to see why it isn't doing well. Look into things like its location, the products it sells, prices, and what other stores are nearby.
   - **Advertise More**: Create special promotions and advertisements to attract more customers to this outlet. Consider discounts, special deals, and local ads to draw people in.

- **Ask Customers for Feedback**: Talk to customers and ask for their opinions about the outlet. Find out what they like and don't like, then make improvements based on their feedback.

---

2. **Use Success Strategies from Other Outlets**:
   - **Learn from the Best Outlets**: Find out what the successful outlets are doing right. Try to use these strategies in all outlets.
   - **Copy What Works**: Look at why the oldest outlet has the highest sales. Maybe it's because of its product selection, store layout, or customer service. Try to copy these successful strategies in other outlets.

---

3. **Make Products More Visible**:
   - **Better Displays**: Arrange products in a way that catches customers' eyes. Use attractive displays and put popular items in easy-to-see places.
   - **Run More Promotions**: Have more in-store promotions to make products stand out. This could include discounts, special offers, and events.

In [ ]: