# MaxAB Test

July 4, 2024

### 0.0.1 Introduction

*In this project we're going to do a pricing data analysis, we aim to answer the following questions:* * How much is the total sales for delivered orders per day? * What's the prediction of the total sales for the 4th week given the 3 weeks of data? * Based on the previous prediction, what will be the contribution of Wednesday and Friday in the 4th week? * Based on the 4 weeks of data, which week has the highest sales and which day is usually the highest per week in terms of sales? * How much is the margin percentage per product? * How much is the gross profit per product? * What are the top 3 and bottom 3 products in terms of gross profit? * What do we recommend to further increase the gross profit? * Write a SQL Query to create 1 table that has the total number of unique delivered products and the total number of unique canceled products per day for the first week and create a flag that is equal to 1 if that day had more than than 5 unique products delivere and 0 else wise.

**Importing Python Modules**

[49]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns

import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
```

**Loading the datasets into 3 separate dataframes**

[80]:
```
data = pd.read_excel('Data.xlsx', sheet_name = [1, 2, 3])
sales_orders = data[1]
cost_of_goods = data[2]
product_sales_order = data[3]

# Changing the format the float numbers are viewed
pd.options.display.float_format = '{:.2f}'.format
```

*Inspecting the first dataset.*

[51]:
```
sales_orders.head()
```

[51]:
| | ORDER_ID | DATE | DAY_NAME | Order_status | SALES |
|---|---|---|---|---|---|
| 0 | 6387833 | 2023-10-01 | Sun | Delivered | 285.50 |
| 1 | 6385549 | 2023-10-01 | Sun | Delivered | 1512.25 |
| 2 | 6387475 | 2023-10-01 | Sun | Delivered | 197.50 |
| 3 | 6389331 | 2023-10-01 | Sun | Delivered | 67.50 |
| 4 | 6390122 | 2023-10-01 | Sun | Delivered | 118.00 |

*Doing a quick descriptive analysis to check for outliers and missing values*

[52]:
```
sales_orders.describe(include = 'all')
```

[52]:
| | ORDER_ID | DATE | DAY_NAME | Order_status | SALES |
|---|---|---|---|---|---|
| count | 17163.00 | 17163 | 17163 | 17163 | 17162.00 |
| unique | NaN | NaN | 7 | 2 | NaN |
| top | NaN | NaN | Wed | Delivered | NaN |
| freq | NaN | NaN | 2959 | 15768 | NaN |
| mean | 6444772.53 | 2023-10-10 09:19:32.312532736 | NaN | NaN | -185.88 |
| min | 6383207.00 | 2023-10-01 00:00:00 | NaN | NaN | -10000000.00 |
| 25% | 6414099.00 | 2023-10-05 00:00:00 | NaN | NaN | 159.00 |
| 50% | 6442612.00 | 2023-10-10 00:00:00 | NaN | NaN | 235.50 |
| 75% | 6475501.50 | 2023-10-16 00:00:00 | NaN | NaN | 426.00 |
| max | 6524009.00 | 2023-10-21 00:00:00 | NaN | NaN | 9568.75 |
| std | 35855.09 | NaN | NaN | NaN | 76338.51 |

*Checking missing values and the index of them*

[53]:
```
print(sales_orders.SALES.isna().sum())
sales_orders[sales_orders.SALES.isna() == True]
```

1

[53]:
| | ORDER_ID | DATE | DAY_NAME | Order_status | SALES |
|---|---|---|---|---|---|
| 29 | 6383668 | 2023-10-01 | Sun | Delivered | NaN |

*Getting the index of the outlier*

[81]:
```
sales_orders[sales_orders.SALES == -10000000.00]
```

[81]:

| | ORDER_ID | DATE | DAY_NAME | Order_status | SALES |
|---|---|---|---|---|---|
| 4713 | 6418568 | 2023-10-06 | Fri | Delivered | -10000000.00 |

*Dropping the rows that has missing values/outliers, and double checking the dataset with another descriptive analysis*

[55]:
```
sales_orders.dropna(inplace = True)
sales_orders.drop(sales_orders[sales_orders.SALES == -10000000.00].index,␣
 ↪inplace = True)
sales_orders.describe(include = 'all')
```

[55]:

| | ORDER_ID | DATE | DAY_NAME | Order_status | SALES |
|---|---|---|---|---|---|
| count | 17161.00 | 17161 | 17161 | 17161 | 17161.00 |
| unique | NaN | NaN | 7 | 2 | NaN |
| top | NaN | NaN | Wed | Delivered | NaN |
| freq | NaN | NaN | 2959 | 15766 | NaN |
| mean | 6444777.62 | 2023-10-10 09:20:41.675892992 | NaN | NaN | 396.83 |
| min | 6383207.00 | 2023-10-01 00:00:00 | NaN | NaN | 16.00 |
| 25% | 6414101.00 | 2023-10-05 00:00:00 | NaN | NaN | 159.00 |
| 50% | 6442630.00 | 2023-10-10 00:00:00 | NaN | NaN | 235.50 |
| 75% | 6475502.00 | 2023-10-16 00:00:00 | NaN | NaN | 426.00 |
| max | 6524009.00 | 2023-10-21 00:00:00 | NaN | NaN | 9568.75 |
| std | 35853.59 | NaN | NaN | NaN | 528.14 |

*Calculating the Total Sales per Day for delivered orders only*

[56]:
```
total_sales_per_day = sales_orders[sales_orders.Order_status == 'Delivered']\
                        .drop(['Order_status', 'ORDER_ID'], axis = 1).
 ↪groupby(['DATE', 'DAY_NAME']).sum().reset_index()
total_sales_per_day
```

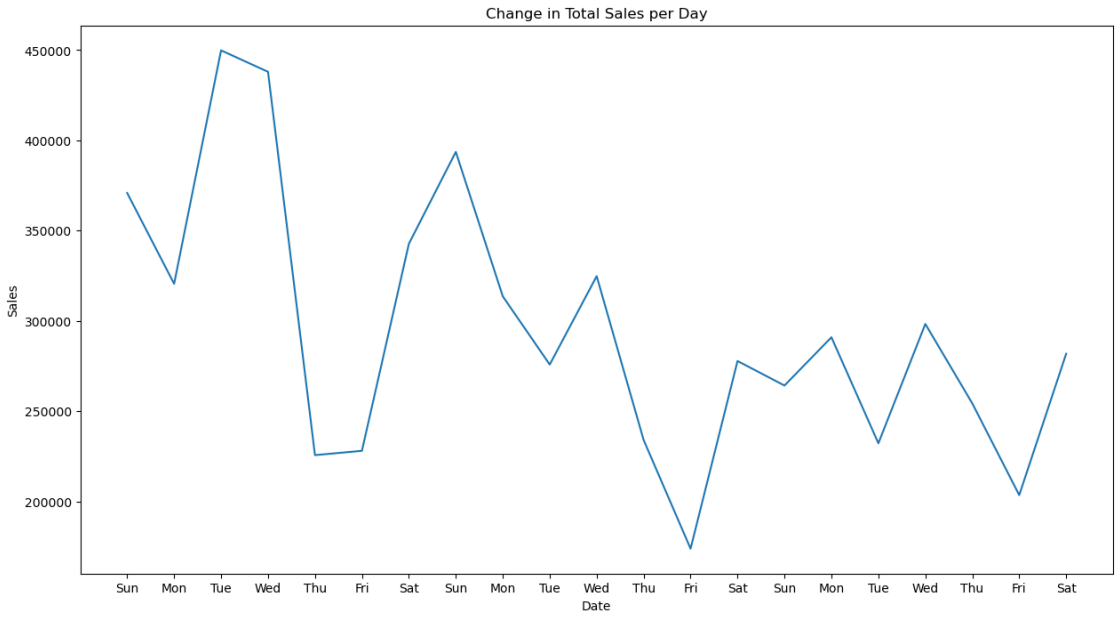
[56]:

| | DATE | DAY_NAME | SALES |
|---|---|---|---|
| 0 | 2023-10-01 | Sun | 370872.25 |

```
1  2023-10-02      Mon 320557.93
2  2023-10-03      Tue 449848.73
3  2023-10-04      Wed 437969.17
4  2023-10-05      Thu 225666.14
5  2023-10-06      Fri 228052.52
6  2023-10-07      Sat 342714.68
7  2023-10-08      Sun 393627.03
8  2023-10-09      Mon 313617.20
9  2023-10-10      Tue 275830.06
10 2023-10-11      Wed 324794.78
11 2023-10-12      Thu 234056.89
12 2023-10-13      Fri 173854.80
13 2023-10-14      Sat 277795.64
14 2023-10-15      Sun 264212.10
15 2023-10-16      Mon 290943.01
16 2023-10-17      Tue 232227.59
17 2023-10-18      Wed 298317.21
18 2023-10-19      Thu 254330.92
19 2023-10-20      Fri 203526.50
20 2023-10-21      Sat 281832.08
```

*Visulaization Showing the change in Total Sales per Day for delivered orders only*

[57]:
```
plt.figure(figsize = (15,8))
ax = plt.subplot()
sns.lineplot(x = 'DATE', y = 'SALES', data = total_sales_per_day)
plt.title('Change in Total Sales per Day')
plt.xlabel('Date')
plt.ylabel("Sales")
ax.set_xticks(total_sales_per_day.DATE)
ax.set_xticklabels(total_sales_per_day.DAY_NAME)
plt.show()
plt.clf()
```




```
<Figure size 640x480 with 0 Axes>
```

**Using Linear Regression to Predict the Total Sales of the 4th Week** *Getting the coefficients of the prediction equation using the Day Name as a predictor*
*The equation is as follows:*

$$\begin{aligned}\text{Total Sales} = 201811.27 &+ 106561.44 * \text{DAY\_NAME[T.Mon]} + 98969.52 * \text{DAY\_NAME[T.Sat]} \\ &+ 141092.52 * \text{DAY\_NAME[T.Sun]} + 36206.71 * \text{DAY\_NAME[T.Thu]} \\ &+ 117490.85 * \text{DAY\_NAME[T.Tue]} + 151882.45 * \text{DAY\_NAME[T.Wed]}\end{aligned}$$

*Notice how all days' coefficients are there except Friday, that's because Friday is the reference category for this model.*
*Let's Calculate the Total Sales for Monday as an example:*

$$\text{Total Sales (Monday)} = 201811.27 + 106561.44 * (1) + 98969.52 * (0) + 141092.52 * (0) + 36206.71 * (0) + 117490.85 * (0) + 151882.45 * (0) = 201811.27 + 106561.44 = 308372.71$$

[58]:
```
import statsmodels.api as sm
model = sm.OLS.from_formula('SALES ~ DAY_NAME', total_sales_per_day).fit()
print(model.params)
```

```
Intercept          201811.27
DAY_NAME[T.Mon]    106561.44
DAY_NAME[T.Sat]     98969.52
DAY_NAME[T.Sun]    141092.52
DAY_NAME[T.Thu]     36206.71
DAY_NAME[T.Tue]    117490.85
DAY_NAME[T.Wed]    151882.45
dtype: float64
```

*A for loop that calculates the predictions for the 4th week*

[59]:
```
predictions = []
for i in range(7):
    if i == 0:
        predictions.append(np.round(model.params[0], 2))
    else:
        predictions.append(np.round(model.params[0] + model.params[i], 2))
predictions
```

[59]:
```
[201811.27, 308372.71, 300780.8, 342903.79, 238017.98, 319302.13, 353693.72]
```

*Sorting the predictions so it can be added easily into a dataframe later*

[60]:
```
predictions_sorted = [predictions[3], predictions[1], predictions[5],␣
 ↪predictions[6], predictions[4], predictions[0], predictions[2]]
predictions_sorted
```

[60]:
```
[342903.79, 308372.71, 319302.13, 353693.72, 238017.98, 201811.27, 300780.8]
```

*Creating a new dataframe with the dates, day names, and predicted total sales for the 4th week*

[61]:
```
from datetime import datetime

start_date = '2023-10-22'
end_date = '2023-10-28'
date_range = pd.date_range(start_date, end_date)

day_names = [datetime.strftime(date, '%a') for date in date_range]

prediction_df = pd.DataFrame({
    'DATE': date_range,
    'DAY_NAME': day_names,
    'SALES': predictions_sorted
})
prediction_df
```

[61]:

| | DATE | DAY_NAME | SALES |
|---|---|---|---|
| 0 | 2023-10-22 | Sun | 342903.79 |
| 1 | 2023-10-23 | Mon | 308372.71 |
| 2 | 2023-10-24 | Tue | 319302.13 |
| 3 | 2023-10-25 | Wed | 353693.72 |
| 4 | 2023-10-26 | Thu | 238017.98 |
| 5 | 2023-10-27 | Fri | 201811.27 |
| 6 | 2023-10-28 | Sat | 300780.80 |

*Showing the contribution of Wednesday and Friday in the 4th week*

[62]:
```
print('The Contribution of Wednesday in the 4th week is {}'.
 ↪format(prediction_df.SALES[3]))
```

```
print('The Contribution of Friday in the 4th week is {}'.format(prediction_df.
  ↪SALES[5]))
print('The contribution of Wednesday and Friday in the 4th week is {}'.
  ↪format(prediction_df.SALES[3] + prediction_df.SALES[5]))
```

```
The Contribution of Wednesday in the 4th week is 353693.72
The Contribution of Friday in the 4th week is 201811.27
The contribution of Wednesday and Friday in the 4th week is 555504.99
```

*Merging the 4th week with the 3 week dataset for further analysis*

[63]:
```
total_sales_per_day = pd.concat([total_sales_per_day, prediction_df],␣
  ↪ignore_index = True)
total_sales_per_day
```

[63]:

| | DATE | DAY_NAME | SALES |
|---|---|---|---|
| 0 | 2023-10-01 | Sun | 370872.25 |
| 1 | 2023-10-02 | Mon | 320557.93 |
| 2 | 2023-10-03 | Tue | 449848.73 |
| 3 | 2023-10-04 | Wed | 437969.17 |
| 4 | 2023-10-05 | Thu | 225666.14 |
| 5 | 2023-10-06 | Fri | 228052.52 |
| 6 | 2023-10-07 | Sat | 342714.68 |
| 7 | 2023-10-08 | Sun | 393627.03 |
| 8 | 2023-10-09 | Mon | 313617.20 |
| 9 | 2023-10-10 | Tue | 275830.06 |
| 10 | 2023-10-11 | Wed | 324794.78 |
| 11 | 2023-10-12 | Thu | 234056.89 |
| 12 | 2023-10-13 | Fri | 173854.80 |
| 13 | 2023-10-14 | Sat | 277795.64 |
| 14 | 2023-10-15 | Sun | 264212.10 |
| 15 | 2023-10-16 | Mon | 290943.01 |
| 16 | 2023-10-17 | Tue | 232227.59 |
| 17 | 2023-10-18 | Wed | 298317.21 |
| 18 | 2023-10-19 | Thu | 254330.92 |
| 19 | 2023-10-20 | Fri | 203526.50 |
| 20 | 2023-10-21 | Sat | 281832.08 |
| 21 | 2023-10-22 | Sun | 342903.79 |
| 22 | 2023-10-23 | Mon | 308372.71 |
| 23 | 2023-10-24 | Tue | 319302.13 |
| 24 | 2023-10-25 | Wed | 353693.72 |
| 25 | 2023-10-26 | Thu | 238017.98 |
| 26 | 2023-10-27 | Fri | 201811.27 |
| 27 | 2023-10-28 | Sat | 300780.80 |

*Adding a Week column to the dataset so the data can be grouped by week*

[64]:
```
values = ['Week 1', 'Week 2', 'Week 3', 'Week 4']
pattern_length = 7

weeks = []
for value in values:
    weeks.extend([value] * pattern_length)
total_sales_per_day['WEEK'] = weeks
total_sales_per_day
```

[64]:

| | DATE | DAY_NAME | SALES | WEEK |
|---|---|---|---|---|
| 0 | 2023-10-01 | Sun | 370872.25 | Week 1 |
| 1 | 2023-10-02 | Mon | 320557.93 | Week 1 |
| 2 | 2023-10-03 | Tue | 449848.73 | Week 1 |
| 3 | 2023-10-04 | Wed | 437969.17 | Week 1 |
| 4 | 2023-10-05 | Thu | 225666.14 | Week 1 |
| 5 | 2023-10-06 | Fri | 228052.52 | Week 1 |
| 6 | 2023-10-07 | Sat | 342714.68 | Week 1 |
| 7 | 2023-10-08 | Sun | 393627.03 | Week 2 |
| 8 | 2023-10-09 | Mon | 313617.20 | Week 2 |
| 9 | 2023-10-10 | Tue | 275830.06 | Week 2 |
| 10 | 2023-10-11 | Wed | 324794.78 | Week 2 |
| 11 | 2023-10-12 | Thu | 234056.89 | Week 2 |
| 12 | 2023-10-13 | Fri | 173854.80 | Week 2 |
| 13 | 2023-10-14 | Sat | 277795.64 | Week 2 |
| 14 | 2023-10-15 | Sun | 264212.10 | Week 3 |
| 15 | 2023-10-16 | Mon | 290943.01 | Week 3 |
| 16 | 2023-10-17 | Tue | 232227.59 | Week 3 |
| 17 | 2023-10-18 | Wed | 298317.21 | Week 3 |
| 18 | 2023-10-19 | Thu | 254330.92 | Week 3 |
| 19 | 2023-10-20 | Fri | 203526.50 | Week 3 |
| 20 | 2023-10-21 | Sat | 281832.08 | Week 3 |
| 21 | 2023-10-22 | Sun | 342903.79 | Week 4 |
| 22 | 2023-10-23 | Mon | 308372.71 | Week 4 |
| 23 | 2023-10-24 | Tue | 319302.13 | Week 4 |
| 24 | 2023-10-25 | Wed | 353693.72 | Week 4 |
| 25 | 2023-10-26 | Thu | 238017.98 | Week 4 |
| 26 | 2023-10-27 | Fri | 201811.27 | Week 4 |
| 27 | 2023-10-28 | Sat | 300780.80 | Week 4 |

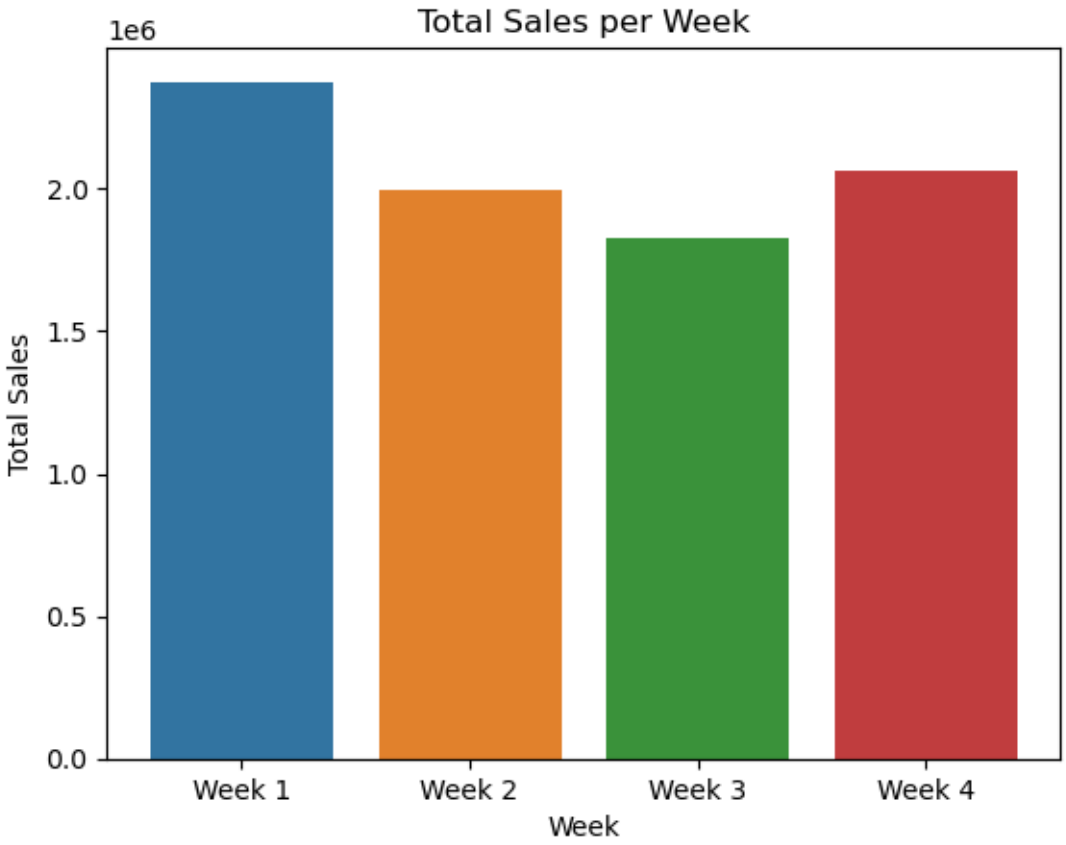
**Grouping the Data by Week and Creating a Visualization that shows Which Week has the Highest Total Sales** *Week 1 has the highest total sales*

[65]:
```
total_sales_by_week = total_sales_per_day.drop(['DATE', 'DAY_NAME'], axis = 1).
 ↪groupby('WEEK').sum().reset_index()
total_sales_by_week
```

[65]:

| | WEEK | SALES |
|---|---|---|
| 0 | Week 1 | 2375681.42 |
| 1 | Week 2 | 1993576.39 |
| 2 | Week 3 | 1825389.40 |
| 3 | Week 4 | 2064882.40 |

[66]:
```
sns.barplot(x = 'WEEK', y = 'SALES', data = total_sales_by_week)
plt.xlabel('Week')
plt.ylabel('Total Sales')
plt.title('Total Sales per Week')
plt.plot()
plt.show()
```



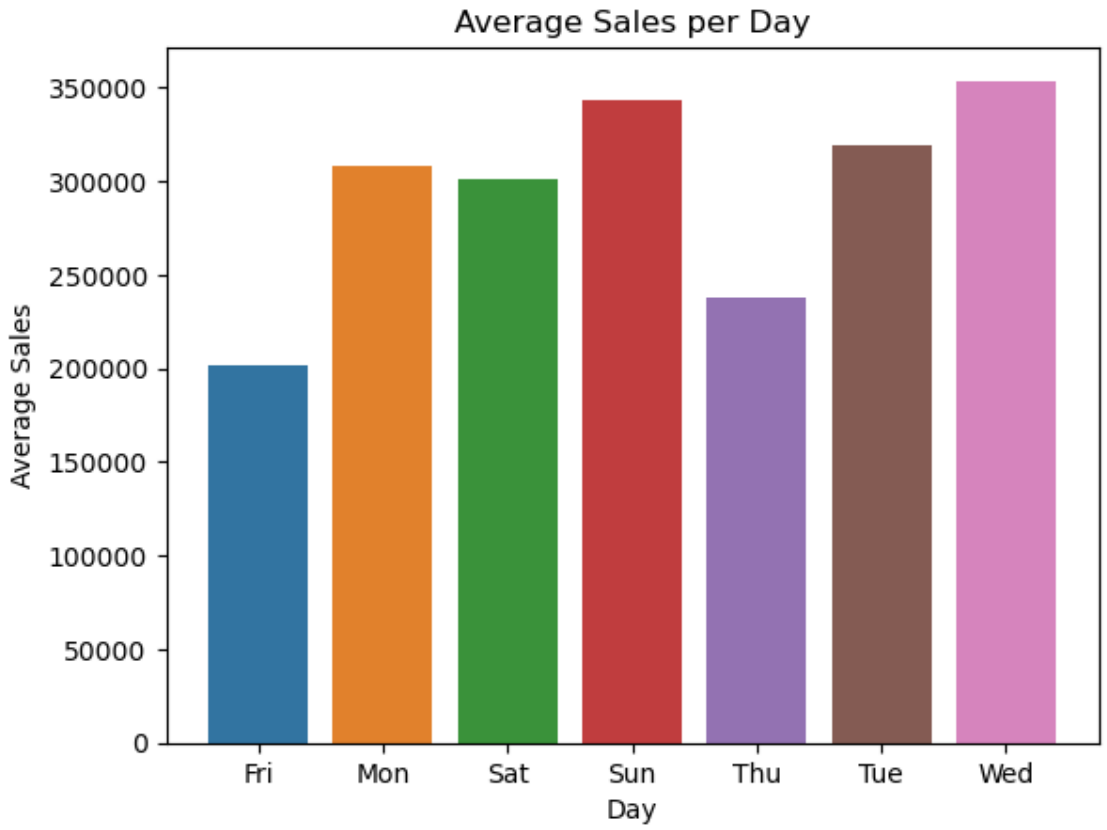

**Calculating and Grouping the Data by The Average Total Sales Per Day and Creating a Visualization that shows Which Day has the Highest Average Total Sales** *Wednesday has the highest average Total Sales*

[67]:
```
average_total_sales_per_day = total_sales_per_day.drop(['DATE', 'WEEK'], axis =␣
↪1).groupby('DAY_NAME').mean().reset_index()
average_total_sales_per_day
```

[67]:

| | DAY_NAME | SALES |
|---|---|---|
| 0 | Fri | 201811.27 |
| 1 | Mon | 308372.71 |
| 2 | Sat | 300780.80 |
| 3 | Sun | 342903.79 |
| 4 | Thu | 238017.98 |
| 5 | Tue | 319302.13 |
| 6 | Wed | 353693.72 |

[68]:
```
sns.barplot(x = 'DAY_NAME', y = 'SALES', data = average_total_sales_per_day)
plt.xlabel('Day')
plt.ylabel('Average Sales')
plt.title('Average Sales per Day')
plt.plot()
plt.show()
```




*Renaming the* `product ID` *column so I can merge the* `cost_of_goods` *and* `sales_per_product` *datasets*

[82]:
```
cost_of_goods
cost_of_goods.rename(columns = {'product_id': 'PRODUCT_ID'}, inplace = True)
cost_of_goods.head()
```

[82]:

| | PRODUCT_ID | purchase_price | selling_price |
|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 |
| 1 | 3495 | 20.01 | 21.75 |
| 2 | 152 | 205.08 | 212.25 |
| 3 | 72 | 508.79 | 540.75 |
| 4 | 974 | 134.01 | 144.25 |

[70]:
```
product_sales_order.head()
```

[70]:

| | PRODUCT_SALES_ORDER_ID | SALES_ORDER_ID | PRODUCT_ID | SALES |
|---|---|---|---|---|
| 0 | 68097611 | 6460123 | 2438 | 390.00 |
| 1 | 68009163 | 6448262 | 1087 | 122.25 |
| 2 | 68097089 | 6460054 | 142 | 166.32 |
| 3 | 68060918 | 6455556 | 415 | 226.50 |
| 4 | 68144928 | 6466586 | 361 | 336.00 |

*Calculating the Total Sales per Product and adding it to a separate dataframe*

[71]:
```
sales_per_product = product_sales_order.drop(['PRODUCT_SALES_ORDER_ID',␣
↪'SALES_ORDER_ID'], axis = 1)\
                    .groupby('PRODUCT_ID').sum().reset_index()
sales_per_product
```

[71]:

| | PRODUCT_ID | SALES |
|---|---|---|
| 0 | 72 | 154686.00 |
| 1 | 142 | 173423.15 |
| 2 | 152 | 175879.50 |
| 3 | 168 | 135221.25 |
| 4 | 361 | 210914.40 |
| 5 | 415 | 193916.36 |
| 6 | 417 | 191528.15 |
| 7 | 583 | 1113452.75 |
| 8 | 974 | 129715.15 |
| 9 | 1070 | 244516.10 |
| 10 | 1080 | 137755.25 |
| 11 | 1087 | 91228.00 |
| 12 | 1222 | 17892.75 |
| 13 | 1765 | 58835.00 |
| 14 | 2438 | 912763.75 |
| 15 | 2778 | 690420.75 |
| 16 | 3495 | 6560.00 |
| 17 | 3575 | 88374.00 |
| 18 | 3900 | 148996.75 |
| 19 | 6952 | 102443.25 |

```
20         7558  240661.25
21         8120  106944.25
22         9046  116856.25
23         9081  861589.25
24         9289  177972.86
25         9468  331632.75
```

*Merging the* `cost_of_goods` *and* `sales_per_product` *datasets*

[72]:
```
cost_sales_merged = pd.merge(cost_of_goods, sales_per_product, on =␣
↪'PRODUCT_ID')
cost_sales_merged
```

[72]:

| | PRODUCT_ID | purchase_price | selling_price | SALES |
|---|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 | 193916.36 |
| 1 | 3495 | 20.01 | 21.75 | 6560.00 |
| 2 | 152 | 205.08 | 212.25 | 175879.50 |
| 3 | 72 | 508.79 | 540.75 | 154686.00 |
| 4 | 974 | 134.01 | 144.25 | 129715.15 |
| 5 | 2438 | 40.89 | 43.00 | 912763.75 |
| 6 | 3900 | 501.99 | 557.75 | 148996.75 |
| 7 | 361 | -200.00 | 189.75 | 210914.40 |
| 8 | 3575 | 268.08 | 282.25 | 88374.00 |
| 9 | 6952 | 144.83 | 155.50 | 102443.25 |
| 10 | 1765 | 157.19 | 163.00 | 58835.00 |
| 11 | 2778 | 156.36 | 166.50 | 690420.75 |
| 12 | 9468 | 92.06 | 96.50 | 331632.75 |
| 13 | 1070 | 45.26 | 47.00 | 244516.10 |
| 14 | 9046 | 204.78 | 215.50 | 116856.25 |
| 15 | 168 | 100000000.00 | 68.50 | 135221.25 |
| 16 | 583 | 263.05 | 278.75 | 1113452.75 |
| 17 | 1080 | 143.07 | 153.25 | 137755.25 |
| 18 | 142 | 182.22 | 189.75 | 173423.15 |
| 19 | 7558 | 394.02 | 412.50 | 240661.25 |
| 20 | 1222 | 345.67 | 350.00 | 17892.75 |
| 21 | 9081 | 67.21 | 72.00 | 861589.25 |
| 22 | 417 | 182.44 | 189.75 | 191528.15 |
| 23 | 1087 | 187.53 | 197.50 | 91228.00 |
| 24 | 9289 | 388.80 | 408.25 | 177972.86 |
| 25 | 8120 | 116.21 | 125.75 | 106944.25 |

**Calculating the Margin for each Product, and the Average Margin, and Filling the Outliers for Purchase Price with the Average Margin Subtracted from the Selling Price**

[73]:
```
cost_sales_merged['margin'] = cost_sales_merged.selling_price -␣
↪cost_sales_merged.purchase_price
cost_sales_merged
```

[73]:

| | PRODUCT_ID | purchase_price | selling_price | SALES | margin |
|---|---|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 | 193916.36 | 11.19 |
| 1 | 3495 | 20.01 | 21.75 | 6560.00 | 1.74 |
| 2 | 152 | 205.08 | 212.25 | 175879.50 | 7.17 |
| 3 | 72 | 508.79 | 540.75 | 154686.00 | 31.96 |
| 4 | 974 | 134.01 | 144.25 | 129715.15 | 10.24 |
| 5 | 2438 | 40.89 | 43.00 | 912763.75 | 2.11 |
| 6 | 3900 | 501.99 | 557.75 | 148996.75 | 55.76 |
| 7 | 361 | -200.00 | 189.75 | 210914.40 | 389.75 |
| 8 | 3575 | 268.08 | 282.25 | 88374.00 | 14.17 |
| 9 | 6952 | 144.83 | 155.50 | 102443.25 | 10.67 |
| 10 | 1765 | 157.19 | 163.00 | 58835.00 | 5.81 |
| 11 | 2778 | 156.36 | 166.50 | 690420.75 | 10.14 |
| 12 | 9468 | 92.06 | 96.50 | 331632.75 | 4.44 |
| 13 | 1070 | 45.26 | 47.00 | 244516.10 | 1.74 |
| 14 | 9046 | 204.78 | 215.50 | 116856.25 | 10.72 |
| 15 | 168 | 100000000.00 | 68.50 | 135221.25 | -99999931.50 |
| 16 | 583 | 263.05 | 278.75 | 1113452.75 | 15.70 |
| 17 | 1080 | 143.07 | 153.25 | 137755.25 | 10.18 |
| 18 | 142 | 182.22 | 189.75 | 173423.15 | 7.53 |
| 19 | 7558 | 394.02 | 412.50 | 240661.25 | 18.48 |
| 20 | 1222 | 345.67 | 350.00 | 17892.75 | 4.33 |
| 21 | 9081 | 67.21 | 72.00 | 861589.25 | 4.79 |
| 22 | 417 | 182.44 | 189.75 | 191528.15 | 7.31 |
| 23 | 1087 | 187.53 | 197.50 | 91228.00 | 9.97 |
| 24 | 9289 | 388.80 | 408.25 | 177972.86 | 19.45 |
| 25 | 8120 | 116.21 | 125.75 | 106944.25 | 9.54 |

[74]:
```
average_margin = np.round(np.mean(cost_sales_merged.margin[:7])\
                          + np.mean(cost_sales_merged.margin[8:15]) + np.
 ↪mean(cost_sales_merged.margin[16:]), 2)
average_margin
```

[74]:
```
36.14
```

[75]:
```
cost_sales_merged.replace(cost_sales_merged.purchase_price[7],␣
 ↪cost_sales_merged.selling_price[7] - average_margin, inplace = True)
cost_sales_merged.replace(cost_sales_merged.purchase_price[15],␣
 ↪cost_sales_merged.selling_price[15] - average_margin, inplace = True)
cost_sales_merged['margin'] = cost_sales_merged.selling_price -␣
 ↪cost_sales_merged.purchase_price
cost_sales_merged
```

[75]:

| | PRODUCT_ID | purchase_price | selling_price | SALES | margin |
|---|---|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 | 193916.36 | 11.19 |
| 1 | 3495 | 20.01 | 21.75 | 6560.00 | 1.74 |
| 2 | 152 | 205.08 | 212.25 | 175879.50 | 7.17 |

| | | | | | |
|---|---|---|---|---|---|
| 3 | 72 | 508.79 | 540.75 | 154686.00 | 31.96 |
| 4 | 974 | 134.01 | 144.25 | 129715.15 | 10.24 |
| 5 | 2438 | 40.89 | 43.00 | 912763.75 | 2.11 |
| 6 | 3900 | 501.99 | 557.75 | 148996.75 | 55.76 |
| 7 | 361 | 153.61 | 189.75 | 210914.40 | 36.14 |
| 8 | 3575 | 268.08 | 282.25 | 88374.00 | 14.17 |
| 9 | 6952 | 144.83 | 155.50 | 102443.25 | 10.67 |
| 10 | 1765 | 157.19 | 163.00 | 58835.00 | 5.81 |
| 11 | 2778 | 156.36 | 166.50 | 690420.75 | 10.14 |
| 12 | 9468 | 92.06 | 96.50 | 331632.75 | 4.44 |
| 13 | 1070 | 45.26 | 47.00 | 244516.10 | 1.74 |
| 14 | 9046 | 204.78 | 215.50 | 116856.25 | 10.72 |
| 15 | 168 | 32.36 | 68.50 | 135221.25 | 36.14 |
| 16 | 583 | 263.05 | 278.75 | 1113452.75 | 15.70 |
| 17 | 1080 | 143.07 | 153.25 | 137755.25 | 10.18 |
| 18 | 142 | 182.22 | 189.75 | 173423.15 | 7.53 |
| 19 | 7558 | 394.02 | 412.50 | 240661.25 | 18.48 |
| 20 | 1222 | 345.67 | 350.00 | 17892.75 | 4.33 |
| 21 | 9081 | 67.21 | 72.00 | 861589.25 | 4.79 |
| 22 | 417 | 182.44 | 189.75 | 191528.15 | 7.31 |
| 23 | 1087 | 187.53 | 197.50 | 91228.00 | 9.97 |
| 24 | 9289 | 388.80 | 408.25 | 177972.86 | 19.45 |
| 25 | 8120 | 116.21 | 125.75 | 106944.25 | 9.54 |

**Calculating the Margin Percentage for each Product**

[76]:
```
cost_sales_merged['margin_percentage'] = (cost_sales_merged.margin/
 ↪cost_sales_merged.selling_price)*100
cost_sales_merged
```

[76]:

| | PRODUCT_ID | purchase_price | selling_price | SALES | margin | \ |
|---|---|---|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 | 193916.36 | 11.19 | |
| 1 | 3495 | 20.01 | 21.75 | 6560.00 | 1.74 | |
| 2 | 152 | 205.08 | 212.25 | 175879.50 | 7.17 | |
| 3 | 72 | 508.79 | 540.75 | 154686.00 | 31.96 | |
| 4 | 974 | 134.01 | 144.25 | 129715.15 | 10.24 | |
| 5 | 2438 | 40.89 | 43.00 | 912763.75 | 2.11 | |
| 6 | 3900 | 501.99 | 557.75 | 148996.75 | 55.76 | |
| 7 | 361 | 153.61 | 189.75 | 210914.40 | 36.14 | |
| 8 | 3575 | 268.08 | 282.25 | 88374.00 | 14.17 | |
| 9 | 6952 | 144.83 | 155.50 | 102443.25 | 10.67 | |
| 10 | 1765 | 157.19 | 163.00 | 58835.00 | 5.81 | |
| 11 | 2778 | 156.36 | 166.50 | 690420.75 | 10.14 | |
| 12 | 9468 | 92.06 | 96.50 | 331632.75 | 4.44 | |
| 13 | 1070 | 45.26 | 47.00 | 244516.10 | 1.74 | |
| 14 | 9046 | 204.78 | 215.50 | 116856.25 | 10.72 | |
| 15 | 168 | 32.36 | 68.50 | 135221.25 | 36.14 | |
| 16 | 583 | 263.05 | 278.75 | 1113452.75 | 15.70 | |

| | | | | | |
|---|---|---|---|---|---|
| 17 | 1080 | 143.07 | 153.25 | 137755.25 | 10.18 |
| 18 | 142 | 182.22 | 189.75 | 173423.15 | 7.53 |
| 19 | 7558 | 394.02 | 412.50 | 240661.25 | 18.48 |
| 20 | 1222 | 345.67 | 350.00 | 17892.75 | 4.33 |
| 21 | 9081 | 67.21 | 72.00 | 861589.25 | 4.79 |
| 22 | 417 | 182.44 | 189.75 | 191528.15 | 7.31 |
| 23 | 1087 | 187.53 | 197.50 | 91228.00 | 9.97 |
| 24 | 9289 | 388.80 | 408.25 | 177972.86 | 19.45 |
| 25 | 8120 | 116.21 | 125.75 | 106944.25 | 9.54 |

| | margin_percentage |
|---|---|
| 0 | 3.97 |
| 1 | 8.00 |
| 2 | 3.38 |
| 3 | 5.91 |
| 4 | 7.10 |
| 5 | 4.91 |
| 6 | 10.00 |
| 7 | 19.05 |
| 8 | 5.02 |
| 9 | 6.86 |
| 10 | 3.56 |
| 11 | 6.09 |
| 12 | 4.60 |
| 13 | 3.70 |
| 14 | 4.97 |
| 15 | 52.76 |
| 16 | 5.63 |
| 17 | 6.64 |
| 18 | 3.97 |
| 19 | 4.48 |
| 20 | 1.24 |
| 21 | 6.65 |
| 22 | 3.85 |
| 23 | 5.05 |
| 24 | 4.76 |
| 25 | 7.59 |

### Calculating the Gross Profit for each Product

[77]:
```
cost_sales_merged['units_sold'] = np.round(cost_sales_merged.SALES/
 ↪cost_sales_merged.selling_price, 0)
cost_sales_merged['gross_profit'] = cost_sales_merged.margin *␣
 ↪cost_sales_merged.units_sold
cost_sales_merged
```

[77]:

| | PRODUCT_ID | purchase_price | selling_price | SALES | margin | margin_percentage | units_sold | gross_profit |
|---|---|---|---|---|---|---|---|---|
| 0 | 415 | 270.56 | 281.75 | 193916.36 | 11.19 | 3.97 | 688.00 | 7698.72 |
| 1 | 3495 | 20.01 | 21.75 | 6560.00 | 1.74 | 8.00 | 302.00 | 525.48 |
| 2 | 152 | 205.08 | 212.25 | 175879.50 | 7.17 | 3.38 | 829.00 | 5943.93 |
| 3 | 72 | 508.79 | 540.75 | 154686.00 | 31.96 | 5.91 | 286.00 | 9140.56 |
| 4 | 974 | 134.01 | 144.25 | 129715.15 | 10.24 | 7.10 | 899.00 | 9205.76 |
| 5 | 2438 | 40.89 | 43.00 | 912763.75 | 2.11 | 4.91 | 21227.00 | 44788.97 |
| 6 | 3900 | 501.99 | 557.75 | 148996.75 | 55.76 | 10.00 | 267.00 | 14887.92 |
| 7 | 361 | 153.61 | 189.75 | 210914.40 | 36.14 | 19.05 | 1112.00 | 40187.68 |
| 8 | 3575 | 268.08 | 282.25 | 88374.00 | 14.17 | 5.02 | 313.00 | 4435.21 |
| 9 | 6952 | 144.83 | 155.50 | 102443.25 | 10.67 | 6.86 | 659.00 | 7031.53 |
| 10 | 1765 | 157.19 | 163.00 | 58835.00 | 5.81 | 3.56 | 361.00 | 2097.41 |
| 11 | 2778 | 156.36 | 166.50 | 690420.75 | 10.14 | 6.09 | 4147.00 | 42050.58 |
| 12 | 9468 | 92.06 | 96.50 | 331632.75 | 4.44 | 4.60 | 3437.00 | 15260.28 |
| 13 | 1070 | 45.26 | 47.00 | 244516.10 | 1.74 | 3.70 | 5202.00 | 9051.48 |
| 14 | 9046 | 204.78 | 215.50 | 116856.25 | 10.72 | 4.97 | 542.00 | 5810.24 |
| 15 | 168 | 32.36 | 68.50 | 135221.25 | 36.14 | 52.76 | 1974.00 | 71340.36 |
| 16 | 583 | 263.05 | 278.75 | 1113452.75 | 15.70 | 5.63 | 3994.00 | 62705.80 |
| 17 | 1080 | 143.07 | 153.25 | 137755.25 | 10.18 | 6.64 | 899.00 | 9151.82 |
| 18 | 142 | 182.22 | 189.75 | 173423.15 | 7.53 | | | |
| 19 | 7558 | 394.02 | 412.50 | 240661.25 | 18.48 | | | |
| 20 | 1222 | 345.67 | 350.00 | 17892.75 | 4.33 | | | |
| 21 | 9081 | 67.21 | 72.00 | 861589.25 | 4.79 | | | |
| 22 | 417 | 182.44 | 189.75 | 191528.15 | 7.31 | | | |
| 23 | 1087 | 187.53 | 197.50 | 91228.00 | 9.97 | | | |
| 24 | 9289 | 388.80 | 408.25 | 177972.86 | 19.45 | | | |
| 25 | 8120 | 116.21 | 125.75 | 106944.25 | 9.54 | | | |

```
18              3.97      914.00       6882.42
19              4.48      583.00      10773.84
20              1.24       51.00        220.83
21              6.65    11967.00      57321.93
22              3.85     1009.00       7375.79
23              5.05      462.00       4606.14
24              4.76      436.00       8480.20
25              7.59      850.00       8109.00
```

**Top 3 Products in terms of Gross profit**

[78]:
```
cost_sales_merged.nlargest(3, 'gross_profit')
```

[78]:
```
    PRODUCT_ID  purchase_price  selling_price      SALES  margin  \
15         168           32.36          68.50  135221.25   36.14
16         583          263.05         278.75 1113452.75   15.70
21        9081           67.21          72.00  861589.25    4.79

    margin_percentage  units_sold  gross_profit
15              52.76     1974.00      71340.36
16               5.63     3994.00      62705.80
21               6.65    11967.00      57321.93
```

**Bottom 3 Products in terms of Gross profit**

[79]:
```
cost_sales_merged.nsmallest(3, 'gross_profit')
```

[79]:
```
    PRODUCT_ID  purchase_price  selling_price    SALES  margin  \
20        1222          345.67         350.00 17892.75    4.33
1         3495           20.01          21.75  6560.00    1.74
10        1765          157.19         163.00 58835.00    5.81

    margin_percentage  units_sold  gross_profit
20               1.24       51.00        220.83
1                8.00      302.00        525.48
10               3.56      361.00       2097.41
```

**My Recommendations to Further Increase the Gross Profit** *To increase the gross profit I suggest the following:* * Increase the selling price for the products with low margin. * Using data analytics to target high value customers more effectively.

**The SQL Query** WITH unique_products_per_day as
(
SELECT DATE, COUNT(DISTINCT CASE WHEN Order_status = 'Delivered' THEN ORDER_ID END) as 'No. of Unique Delivered Products', COUNT(DISTINCT CASE WHEN Order_status = 'Canceled' THEN ORDER_ID END) as 'No. of Unique Canceled Products', CASE WHEN COUNT(DISTINCT CASE WHEN Order_status = 'Delivered' THEN ORDER_ID END) > 5 THEN 1 ELSE 0 END AS 'more_than_5_delivered_flag'
FROM Sales_orders

```
WHERE (
DATE BETWEEN '2023-10-01' AND '2023-10-07'
)
GROUP BY 1
)
SELECT *
FROM unique_products_per_day;
```

**Query Results**

| ORDER_ID | DATE | DAY_NAME | Order_status | Sales |
|---|---|---|---|---|
| 6385549 | 2023-10-01 | Mon | Delivered | 1512.25 |
| 6385833 | 2023-10-01 | Sun | Delivered | 285.5 |
| 6386866 | 2023-10-07 | Sun | Canceled | 114.25 |
| 6387133 | 2023-10-01 | Sun | Delivered | 285.5 |
| 6387333 | 2023-10-01 | Sun | Delivered | 285.5 |
| 6387475 | 2023-10-06 | Tue | Delivered | 197.5 |
| 6387633 | 2023-10-01 | Sun | Delivered | 285.5 |
| 6389833 | 2023-10-01 | Sun | Delivered | 285.5 |

| PRODUCT_ID | purchase_price | selling_price |
|---|---|---|
| 72 | 508.79 | 540.75 |
| 152 | 205.08 | 212.25 |
| 415 | 270.56 | 281.75 |
| 3495 | 20.01 | 21.75 |

| PRODUCT_ID | PRODUCT_SALES_ORDER_ID | SALES_ORDER_ID | SALES |
|---|---|---|---|
| 68009163 | 6448262 | 1087 | 122.25 |
| 68060918 | 6455556 | 415 | 226.5 |
| 68097089 | 6460054 | 142 | 166.32 |
| 68097611 | 6460123 | 2438 | 390 |

| DATE | No. of Unique Delivered Products | No. of Unique Canceled Products | more_than_5_delivered_flag |
|---|---|---|---|
| 2023-10-01 | 6 | 0 | 1 |
| 2023-10-06 | 1 | 0 | 0 |
| 2023-10-07 | 0 | 1 | 0 |