A REPORT ON
TECHNICAL SEMINAR

Submitted in partial fulfillment of the requirements
For the award of the Degree of

Bachelor of Technology
Of
Poornima University, Jaipur

Session: 2020

Submitted By:
Rahul Nyati
2018PUSCEBDSX06956
IIIrd Year, Computer Science Engineering (Data Science)

Submitted To:
Department of Computer Engineering
School of Engineering & Technology, Poornima University
Ramchandrapura, Sitapura Ext., Jaipur, Rajasthan

ACKNOWLEDGEMENT

I have undergone a technical seminar which was meticulously planned and guided at every stage so that it became a life time experience for me. This could not be realized without the help from numerous sources and people in the Poornima University and Programming Express, Ajmer.

I am thankful to **Dr. Manoj Gupta, Provost, and Poornima University** for providing us a platform to carry out this activity successfully.

I am also very grateful to **Mr. Ravi Godara (HOD, Computer Engineering)** for his kind support and guidance.

I would like to take this opportunity to show our gratitude towards **Mr. Shubham Verma** who helped me in successful completion of my seminar. He has been a guide, motivator & source of inspiration for us to carry out the necessary proceedings for completing this training and related activities successfully and grateful for her/him guidance and support. I am thankful for their kind support and providing us expertise of the domain to develop the project.

I would also like to express our hearts felt appreciation to all of our friends whom direct or indirect suggestions help us to develop this project and to entire team members for their valuable suggestions.

Lastly, thanks to all faculty members of Department of Computer Engineering for their moral support and guidance·

Rahul Nyati

ABSTRACT

Pricing a product is a crucial aspect in any business. Alot of thought process is out into it. There are different strategies to price different kinds of products. There are products whose sales are quite sensitive to their prices and as such a small change in their price can lead to noticeable change in their sales. While there are products whose sales are not much affected by their price - these tend to be either luxury items or necessities (like certain medicines).
Elasticity, is the degree to which the effective desire for something changes as its price changes. In general, people desire things less as those things become more expensive. However, for some products, the customer's desire could drop sharply even with a little price increase, and for other products, it could stay almost the same even with a big price increase. Economists use the term elasticity to denote this sensitivity to price increases. More precisely, price elasticity gives the percentage change in quantity demanded when there is a one percent increase in price, holding everything else constant.

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 Aims and Objective:

1.Finding the best prices of products.

2. Understanding how customers will react to different pricing strategies for products and services, i.e., understanding the elasticity of the product so we will easily find the right price of products to get maximum profits.

1.2 Problem Statement:

Price optimization is nothing more than the process of determining the proper retail value of a **consumer product** or service. While in principle, it may seem that there is not a whole lot to consider, both manufacturers and retail stores dedicate a massive amount of time towards price optimization to ensure that their products will sell quickly while still making a profit. If the item is priced too high, it may not sell at all, while if the cost is reduced too much, the store will unnecessarily limit its buying power. Each of the manufacturers use a price optimization formula based on the overall demand for their product, their level of competition, and the cost of manufacturing their goods.

1.2 Scope:

1. We can take Better and quick decisions easily.

2. This will help to make analysis much easier.

3. User can visualize data through graphs.

4. Will help to make comparison between different products of price.

5. It will provide the maximum discounts to customers with optimal price of products.

6. It will help to company to increases sales and get maximum profits with customers satisfaction.

1.3 Duration and Schedule

Duration of training: 03/12/2018 to 05/02/2019

Total Weeks – 12

Total Days – 61

Table 1.4.1 Schedule of Training

Week No.	Learning
1	Study about the price optimization
2	Data collection/Data gathering
3	Data Pre-processing
4	EDA(Exploratory Data Analysis)
5	Combining the datasets
6	Uncovering Facets of Data With help of visualization
7	Calculating the Price elasticity of any one product using function .
8	Then Calculate the price elasticity of other product using function calling.
9	Create model using OLS and apply on all products.

10	Finding the optimal price for maximum profit.
11	Profit Maximization for all products .
12	Testing of models.

1.4 Data Set Descriptions:-In our Case the datasets are in three different CSV files which contains Product Selling data,Transaction Data ,product data

Product Sell datatypes

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

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 4 columns):
SELL_ID          11 non-null int64
SELL_CATEGORY    11 non-null int64
ITEM_ID          11 non-null int64
ITEM_NAME        11 non-null object
dtypes: int64(3), object(1)
memory usage: 432.0+ bytes
```

Products Selling data like this

```
In [3]: sold.head()
```

Out[3]:

	SELL_ID	SELL_CATEGORY	ITEM_ID	ITEM_NAME
0	1070	0	7821	BURGER
1	3055	0	3052	COFFEE
2	3067	0	5030	COKE
3	3028	0	6249	LEMONADE
4	2051	2	7821	BURGER

Transaction data datatypes

```
In [10]: trans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5404 entries, 0 to 5403
Data columns (total 5 columns):
CALENDAR_DATE    5404 non-null object
PRICE            5404 non-null float64
QUANTITY         5404 non-null int64
SELL_ID          5404 non-null int64
SELL_CATEGORY    5404 non-null int64
dtypes: float64(1), int64(3), object(1)
memory usage: 211.2+ KB
```

Transaction data like this

```
In [7]: trans.head()
```

Out[7]:

	CALENDAR_DATE	PRICE	QUANTITY	SELL_ID	SELL_CATEGORY
0	01/01/12	15.50	46	1070	0
1	01/01/12	12.73	22	2051	2
2	01/01/12	12.75	18	2052	2
3	01/01/12	12.60	30	2053	2
4	01/02/12	15.50	70	1070	0

Transaction detail data datatypes

```
In [14]: dinfo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1349 entries, 0 to 1348
Data columns (total 7 columns):
CALENDAR_DATE          1349 non-null object
YEAR                   1349 non-null int64
HOLIDAY                105 non-null object
IS_WEEKEND             1349 non-null int64
IS_SCHOOLBREAK         1349 non-null int64
AVERAGE_TEMPERATURE    1349 non-null float64
IS_OUTDOOR             1349 non-null int64
dtypes: float64(1), int64(4), object(2)
memory usage: 73.9+ KB
```

Transaction Details data like this

```
In [11]: dinfo.head()
```

Out[11]:

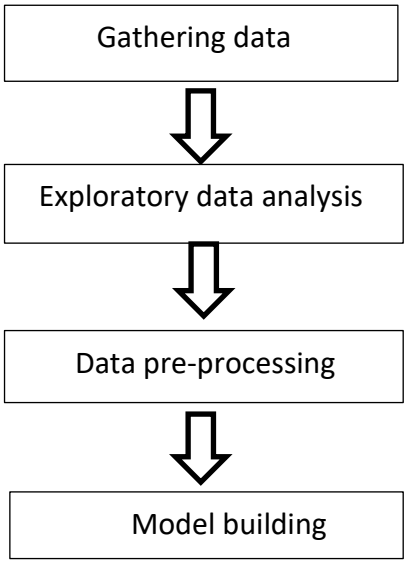
	CALENDAR_DATE	YEAR	HOLIDAY	IS_WEEKEND	IS_SCHOOLBREAK	AVERAGE_TEMPERATUR
0	1/1/12	2012	New Year	1	0	24.
1	1/2/12	2012	New Year	0	0	24.
2	1/3/12	2012	New Year	0	0	32.
3	1/4/12	2012	NaN	0	0	32.
4	1/5/12	2012	NaN	0	0	24.

Datasetlink:

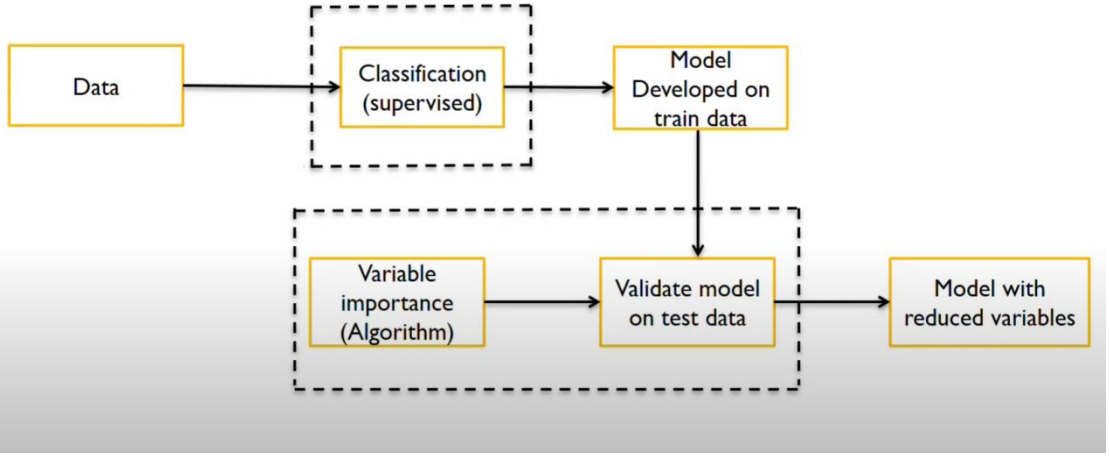
https://onedrive.live.com/?authkey=%21AHDkRFUvt4FI8Wc&cid=7BA8848FE0992BD8&id=7BA8848FE0992BD8%21278810&parId=7BA8848FE0992BD8%21195633&action=locate

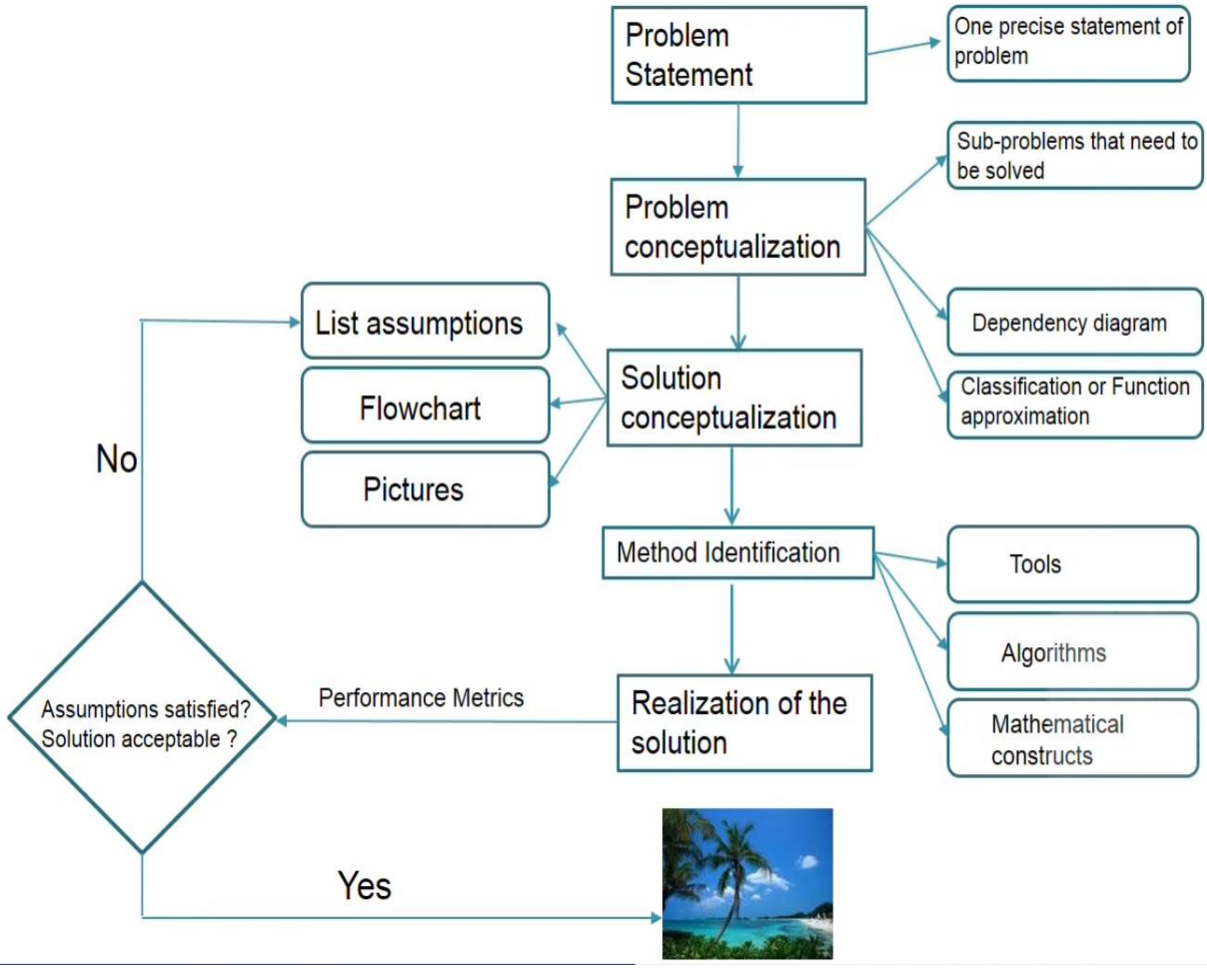
1.5 Work Flow:

The whole approach is depicted by the following flowchart.



- Flow chart:


Data
Classification (supervised)
Model Developed on train data
Variable importance (Algorithm)
Validate model on test data
Model with reduced variables


Problem Statement
One precise statement of problem
Sub-problems that need to be solved
Problem conceptualization
Dependency diagram
Classification or Function approximation
List assumptions
Flowchart
Pictures
Solution conceptualization
No
Method Identification
Tools
Algorithms
Mathematical constructs
Assumptions satisfied? Solution acceptable ?
Performance Metrics
Realization of the solution
Yes

1.6 TOOLS/PLATFORMS USED

1. Python
2. Excel
3. R
4. Jupyter Notebook
5. Google Colab

1.7 PLATFORM USED IN PROGRAM DEVELOPMENT:

SYSTEM CONFIGURATION:

HARDWARE SPECIFICATION:

1. Processor i3
2. RAM =500MB
3. Disk Space=2-3GB

SOFTWARE SPECIFICATION :

1. Windows 7/8/10
2. Linux

Conclusion:

As an undergraduate student I would like to say that training program is an excellent opportunity for us to get to the ground level and experience the things that we would have never gained through going into a job straightly. Industrial training was the first opportunity I got to apply the theories I learnt with the real industry for the real situations. This also gave me the chance to move with different types of people in the industry. Having exposed to such situations I was able to obtain lot of experiences which will be definitely helpful to success my future career as an Engineer.

Chapter 2

Project Development process

1.Used necessary libraries to develeop project in python.

1.1 Pandas:

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables. pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive.

Syntax in python :-**import pandas as pd**

1.2 Numpy:

Numpy is a python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Numpy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. Numpy stands for Numerical Python.

Syntax in python**:-import numpy as np**

1.3 Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on Numpy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.

Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Syntax in python:- **import matplotlib.pyplot as plt**

1.4 Seaborn:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on DataFrames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

Syntax in python:- **import seaborn as sns; sns.set(style="ticks", color_codes=True)**

1.5 Statsmodel:

statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct.

Syntax in python**:- import statsmodels.api as sm**

from statsmodels.formula.api import ols

1.6 Working of Project

I Divide my project into 12 Tasks

TASK 1:-Study about the PRICE OPTIMIZATION

TASK 2:-Data Collection

TASK 3:-Data Pre-processing

TASK 4:-EDA(Exploratory Data Analysis)

TASK 5:-Combining the datasets

TASK 6:-Uncovering Facets of Data With help of visualization

TASK 7:-Calculating the Price elasticity of any one product using function .

TASK 8:-Then Calculate the price elasticity of other product using function calling.

TASK 9:-Create model using OLS and apply on all products.

TASK 10:-Finding the optimal price for maximum profit.

TASK 11:-Profit Maximization for all products .

TASK 12:-Conclusion.

TASK 1:-Study about the PRICE OPTIMIZATION

Price optimization is the use of mathematical analysis by a company to determine how customers will respond to different prices for its products and services through different channels.It is also used to determine the prices that the company determines will best meet its objectives such as maximizing operating profit. The data used in

price optimization can include survey data, operating costs, inventories, and historic prices & sales. Price optimization practice has been implemented in industries including retail, banking, airlines, casinos, hotels, car rental, cruise lines and insurance industries.

Mathematically speaking, the price elasticity of demand is defined to be the percentage change in quantity demanded, q,divided by the percentage change in price, p. The formula for the price elasticity (ϙ) is:

$$e = \%\Delta Q / \%\Delta P$$

TASK 2:-Data Collection

we collected the data from below link in my case the data contains 3 CSV Files which contains Transaction Data,product selling data ,and Transaction Details Data

Data Set Link :-
https://onedrive.live.com/?authkey=%21AHDkRFUvt4FI8Wc&cid=7BA8848FE0992BD8&id=7BA8848FE0992BD8%21278810&parId=7BA8848FE0992BD8%21195633&action=locate

TASK 3:-DATA PRE-PROCESSING

We divide the Data pre-processing in to different segements like

1. Loading the dataset

2.Dealing with missing data

3.Classifying the dependent and Independent Variables

4.Dealing with categorical data

5.Splitting the dataset into training and testing sets

6.Scaling the features

1. LOADING THE DATASET

sold = pd.read_csv('sell.csv')

trans = pd.read_csv('transaction.csv')

dinfo = pd.read_csv('info.csv')

```
sold.head()
```

	SELL_ID	SELL_CATEGORY	ITEM_ID	ITEM_NAME
0	1070	0	7821	BURGER
1	3055	0	3052	COFFEE
2	3067	0	5030	COKE
3	3028	0	6249	LEMONADE
4	2051	2	7821	BURGER

```
trans.head()
```

	CALENDAR_DATE	PRICE	QUANTITY	SELL_ID	SELL_CATEGORY
0	01/01/12	15.50	46	1070	0
1	01/01/12	12.73	22	2051	2
2	01/01/12	12.75	18	2052	2
3	01/01/12	12.60	30	2053	2
4	01/02/12	15.50	70	1070	0

```
dinfo.head()
```

	CALENDAR_DATE	YEAR	HOLIDAY	IS_WEEKEND	IS_SCHOOLBREAK	AVERAGE_TEMPERATURE	IS_OUTDOOR
0	1/1/12	2012	New Year	1	0	24.8	0
1	1/2/12	2012	New Year	0	0	24.8	0
2	1/3/12	2012	New Year	0	0	32.0	1
3	1/4/12	2012	NaN	0	0	32.0	1
4	1/5/12	2012	NaN	0	0	24.8	0

2.DEALING WITH MISSING DATA

We have already noticed the missing fields in the data denoted by "NAN". Machine learning models cannot accommodate missing fields in the data they are provided with.So the missing fields must be filled with values that will not affect the variance of the data or make it more noisy.

Now, we check the missing values in our data sets in python and if any missing values found then filled with appropriate value instead of 'NAN'

```
In [15]: # now we check missing value
         sold.isnull().sum()

Out[15]: SELL_ID            0
         SELL_CATEGORY      0
         ITEM_ID            0
         ITEM_NAME          0
         dtype: int64

In [16]: trans.isnull().sum()

Out[16]: CALENDAR_DATE      0
         PRICE              0
         QUANTITY           0
         SELL_ID            0
         SELL_CATEGORY      0
         dtype: int64

In [17]: dinfo.isnull().sum()

Out[17]: CALENDAR_DATE          0
         YEAR                   0
         HOLIDAY             1244
         IS_WEEKEND             0
         IS_SCHOOLBREAK         0
         AVERAGE_TEMPERATURE    0
         IS_OUTDOOR             0
         dtype: int64
```

HOW TO DEAL WITH MISSING VALUES

In Our case the missing values found in dinfo data and in dinfo data set the missing values are in 'Holiday' column.

Now filled the missing values.

dinfo['HOLIDAY'] =dinfo['HOLIDAY'].fillna("No Holiday")

We filled the missing value by 'No Holiday' because our data in nominal

3.CLASSIFYING THE DEPENDENT AND INDEPENDENT VARIABLES

In our case the quantity is the independent variable and price is the dependent variable

4.Dealing with categorical data

In our case the dependent and independent variable both are numerical so we no need to deal with and categorical data but we required when we combine the data.

```
In [29]: pd.concat([sold.SELL_ID, pd.get_dummies(sold.ITEM_NAME)], axis=1)
```

Out[29]:

	SELL_ID	BURGER	COFFEE	COKE	LEMONADE
0	1070	1	0	0	0
1	3055	0	1	0	0
2	3067	0	0	1	0
3	3028	0	0	0	1
4	2051	1	0	0	0
5	2051	0	0	1	0
6	2052	1	0	0	0
7	2052	0	0	0	1
8	2053	1	0	0	0
9	2053	0	0	1	0
10	2053	0	1	0	0

5.SPLITTING THE DATASET INTO TRAINING AND TESTING SETS

All machine learning models require us to provide a training set for the machine so that the model can train from that data to understand the relations between features and can predict for new observations.When we are provided a single huge dataset with too much of observations ,it is a good idea to split the dataset into to two, a training_set and a test_set, so that we can test our model after its been trained with the training_set.

Scikit-learn comes with a method called train_test_split to help us with this task.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.3, random_state = 0)
```

test_size: the desired size of the test_set. 0.3 denotes 30%.

random_state: This is used to preserve the uniqueness. The split will happen uniquely for a random_state.

Note:-But In Our Case No Need to split the data in to Training and testing sets

6.SCALING THE FEATURES

Since machine learning models rely on numbers to solve relations it is important to have similarly scaled data in a dataset. Scaling ensures that all data in a dataset falls in the same range.Unscaled data can cause inaccurate or false predictions.Some machine learning algorithms can handle feature scaling on its own and doesn't require it explicitly.

The StandardScaler class from the scikit-learn library can help us scale the dataset.

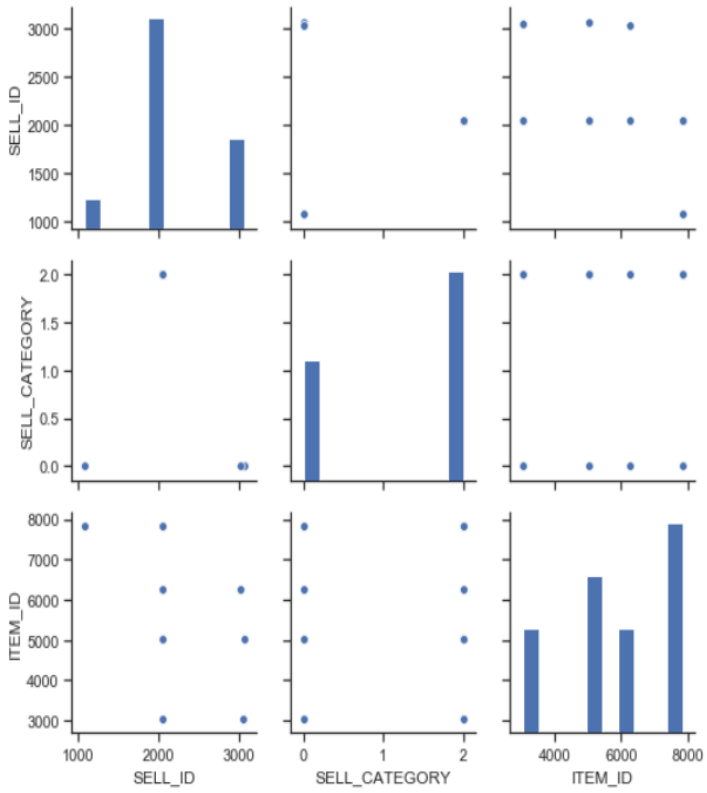
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

sc_y = StandardScaler()
Y_train = Y_train.reshape((len(Y_train), 1))
Y_train = sc_y.fit_transform(Y_train)
Y_train = Y_train.ravel()
```

Note:-**In Our case No need to scale the data because the data is fallen in range that's why we don't need**

TASK 4:-EDA(EXPLORATORY DATA ANALYSIS)

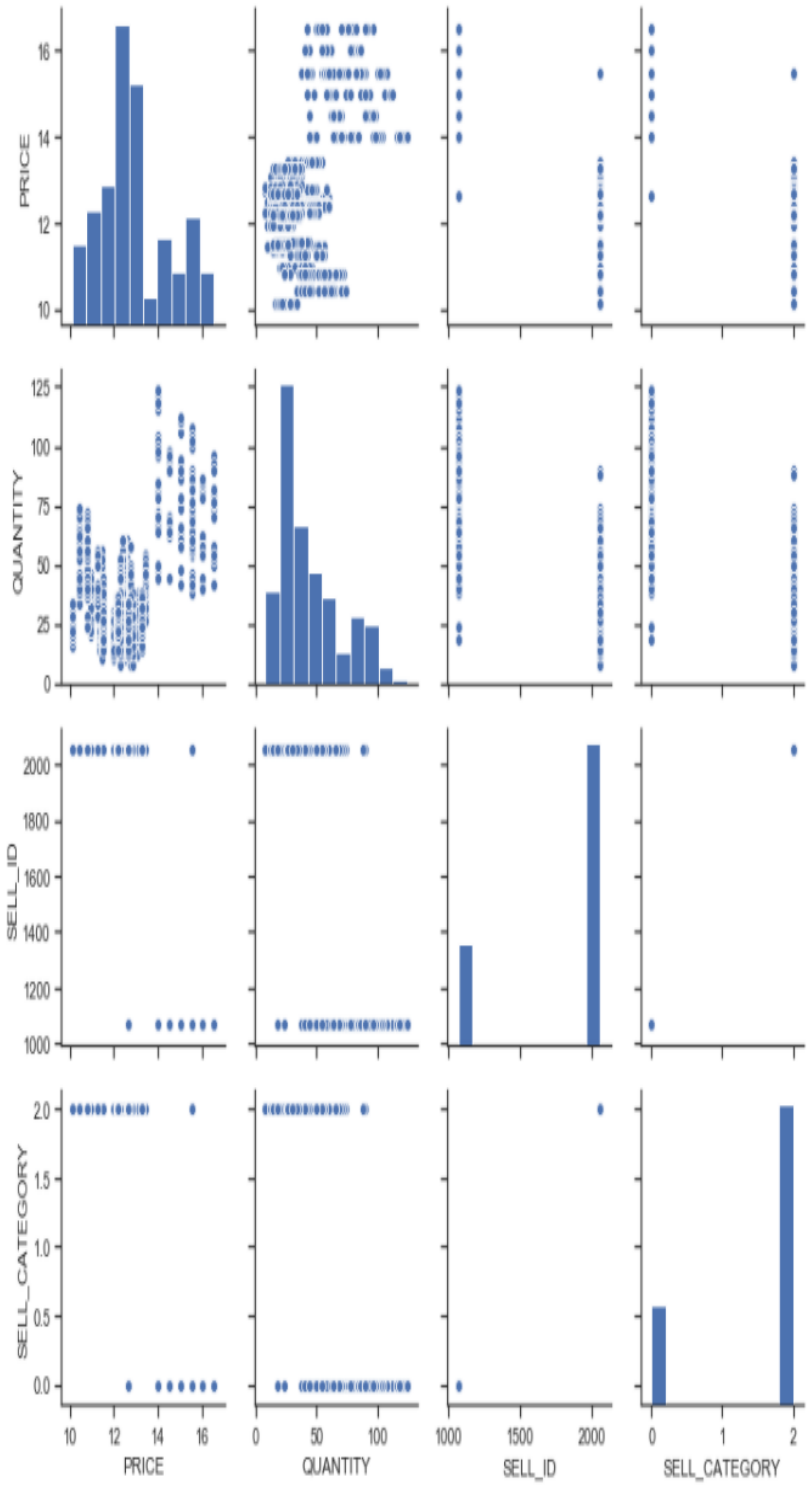
In [21]:
```
# now visualize the data
sns.pairplot(sold)
```

Out[21]: <seaborn.axisgrid.PairGrid at 0x15b47ff3d30>



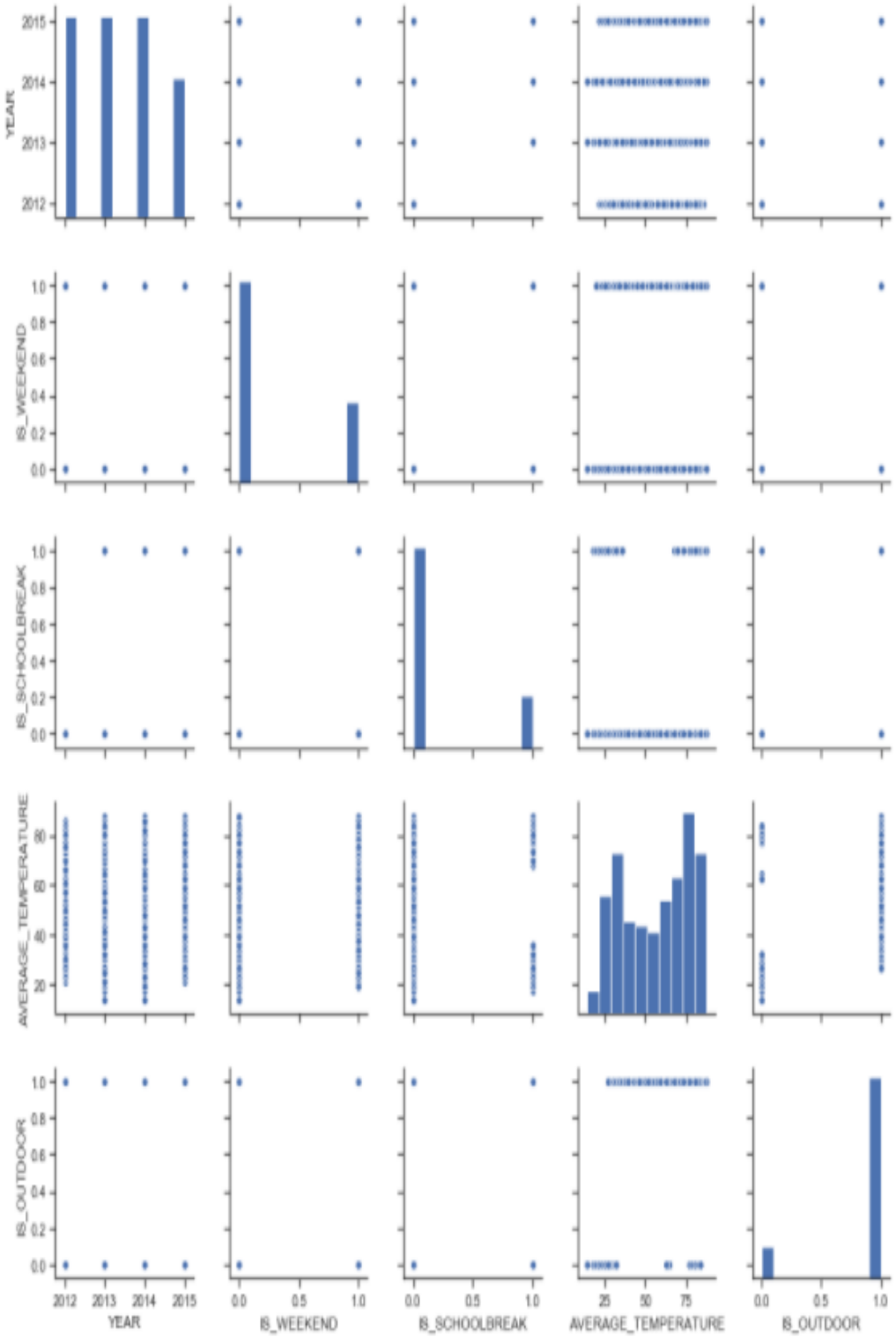
```
In [22]: sns.pairplot(trans)
```

```
Out[22]: <seaborn.axisgrid.PairGrid at 0x15b485b9b38>
```



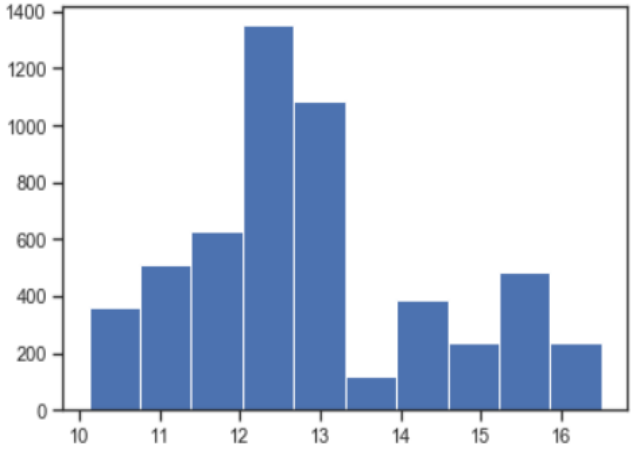
In [23]:
```
sns.pairplot(dinfo)
```

Out[23]: <seaborn.axisgrid.PairGrid at 0x15b48fee0b8>



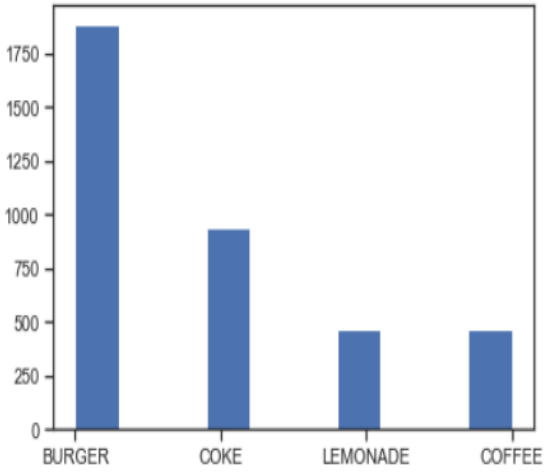
In [12]:
```
plt.hist(transactions.PRICE)
```

Out[12]:
```
(array([ 360.,  509.,  629., 1351., 1082.,  122.,  389.,  240.,  482.,
         240.]),
 array([10.12 , 10.758, 11.396, 12.034, 12.672, 13.31 , 13.948, 14.586,
        15.224, 15.862, 16.5  ]),
 <a list of 10 Patch objects>)
```



In [47]:
```
# Data exploration
plt.hist(bau_data.ITEM_NAME)
```

Out[47]:
```
(array([1884.,    0.,    0.,  942.,    0.,    0.,  471.,    0.,    0.,
         471.]),
 array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
 <a list of 10 Patch objects>)
```



TASK 5:-COMBINING THE DATASETS

```
In [41]: combined_data = pd.merge(intermediate_data, dinfo, on = 'CALENDAR_DATE')
         combined_data.head()
```

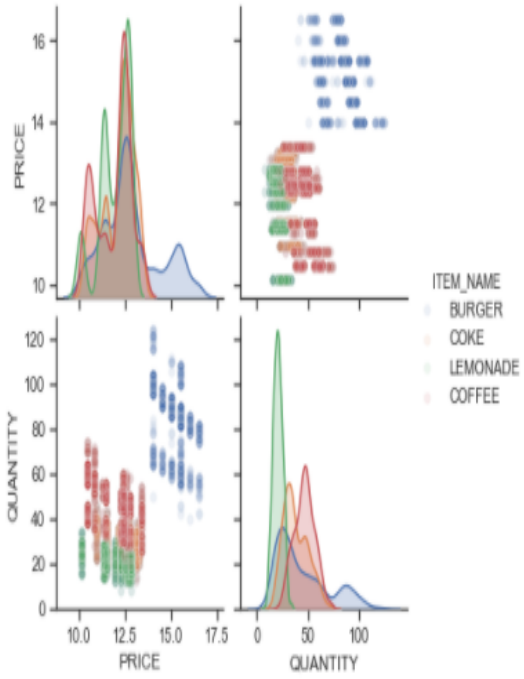
Out[41]:

	SELL_ID	SELL_CATEGORY	ITEM_NAME	CALENDAR_DATE	PRICE	QUANTITY	YEAR	HOLIDAY	IS_WEEKEND	IS_SCHOOLBREAK	AVERAGE_TEMPERATI
0	1070	0	BURGER	1/13/12	15.50	100	2012	No Holiday	0	0	
1	2051	2	BURGER	1/13/12	12.73	40	2012	No Holiday	0	0	
2	2051	2	COKE	1/13/12	12.73	40	2012	No Holiday	0	0	
3	2052	2	BURGER	1/13/12	12.75	26	2012	No Holiday	0	0	
4	2052	2	LEMONADE	1/13/12	12.75	26	2012	No Holiday	0	0	

TASK 6:-UNCOVERING FACETS OF DATA WITH HELP OF VISUALIZATION

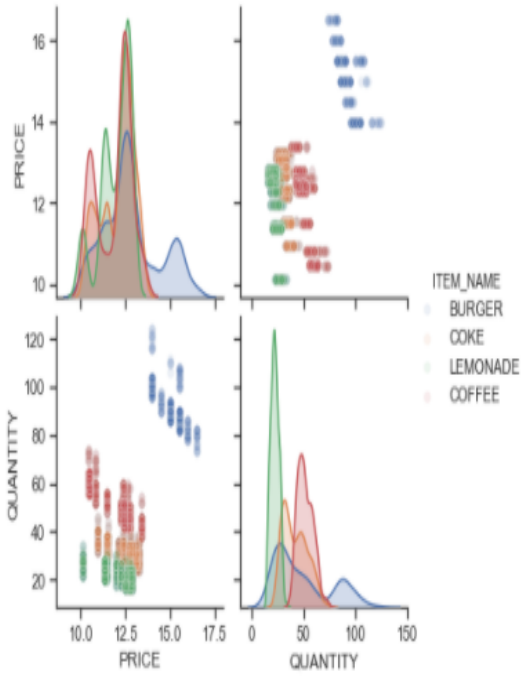
In [51]:
```
sns.pairplot(combined_data[['PRICE','QUANTITY','ITEM_NAME']], hue = 'ITEM_NAME', plot_kws={'alpha':0.1})
```

Out[51]: <seaborn.axisgrid.PairGrid at 0x7fa2b0f0c410>



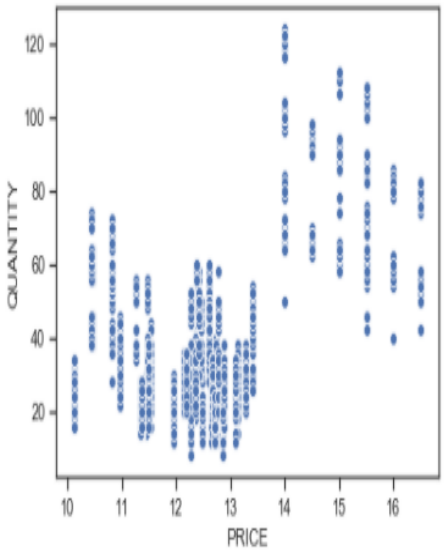
In [52]:
```
sns.pairplot(bau_data[['PRICE','QUANTITY','ITEM_NAME']], hue = 'ITEM_NAME', plot_kws={'alpha':0.1})
```

Out[52]: <seaborn.axisgrid.PairGrid at 0x7fa278e15d90>



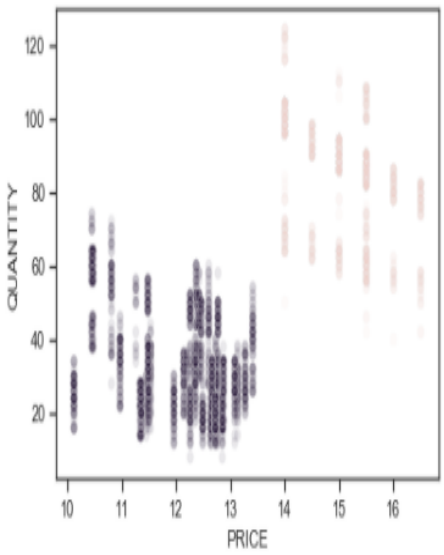
In [60]:
```
burger = combined_data[combined_data['ITEM_NAME'] == 'BURGER']
burger.head()
burger.shape
burger.describe()
sns.scatterplot(x = burger.PRICE, y = burger.QUANTITY )
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x15b68d2f048>



In [61]:
```
burger = combined_data[combined_data['ITEM_NAME'] == 'BURGER']
# print(burger)
# print(burger.describe())
sns.scatterplot(data = burger, x = burger.PRICE, y = burger.QUANTITY , hue = 'SELL_ID', legend=False, alpha = 0.1)
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x15b68b94588>



TASK 7:-CALCULATING THE PRICE ELASTICITY OF ANY ONE PRODUCT USING FUNCTION .

```
elasticities = {}
```

```
def create_model_and_find_elasticity(data):
    model = ols("QUANTITY ~ PRICE", data).fit()
    price_elasticity = model.params[1]
    print("Price elasticity of the product: " + str(price_elasticity))
    print(model.summary())
    fig = plt.figure(figsize=(12,8))
    fig = sm.graphics.plot_partregress_grid(model, fig=fig)
    return price_elasticity, model
```

TASK 8:-THEN CALCULATE THE PRICE ELASTICITY OF OTHER PRODUCT USING FUNCTION CALLING.

1)burger1070_data = bau2_data[(bau2_data['ITEM_NAME'] == "BURGER") & (bau2_data['SELL_ID'] == 1070)]

elasticities['burger_1070'], model_burger_1070 = create_model_and_find_elasticity(burger1070_data)

2) burger2051_data = bau2_data[(bau2_data['ITEM_NAME'] == "BURGER") & (bau2_data['SELL_ID'] == 2051)]

elasticities['burger_2051'], model_burger_2051 = create_model_and_find_elasticity(burger2051_data)

3)burger2053_data = bau2_data[(bau2_data['ITEM_NAME'] == "BURGER") & (bau2_data['SELL_ID'] == 2053)]

elasticities['burger_2053'], model_burger_2053 = create_model_and_find_elasticity(burger2053_data)

4) 1)burger2052_data = bau2_data[(bau2_data['ITEM_NAME'] == "BURGER") & (bau2_data['SELL_ID'] == 2052)]

elasticities['burger_2052'], model_burger_2052 = create_model_and_find_elasticity(burger2052_data)

5) coke_data_2053 = bau2_data[(bau2_data['ITEM_NAME'] == "COKE") & (bau2_data['SELL_ID'] == 2053)]

elasticities['coke_2053'], model_coke_2053 = create_model_and_find_elasticity(coke_data_2053)

6) coke_data_2051 = bau2_data[(bau2_data['ITEM_NAME'] == "COKE") & (bau2_data['SELL_ID'] == 2051)]

elasticities['coke_2051'], model_coke_2051 = create_model_and_find_elasticity(coke_data_2051

7) lemonade_data_2052 = bau2_data[(bau2_data['ITEM_NAME'] == "LEMONADE") & (bau2_data['SELL_ID'] == 2052)]

elasticities['lemonade_2052'], model_lemonade_2052 = create_model_and_find_elasticity(lemonade_data_2052)

8) coffee_data_2053 = bau2_data[(bau2_data['ITEM_NAME'] == "COFFEE") & (bau2_data['SELL_ID'] == 2053)]

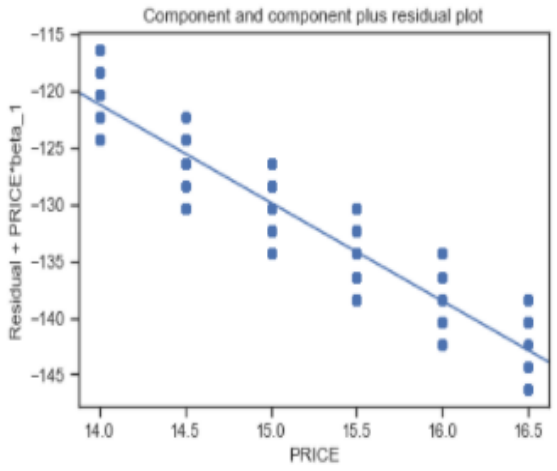
elasticities['coffee_2053'], model_coffee_2053 = create_model_and_find_elasticity(coffee_data_2053)

```
In [105]: elasticities

Out[105]: {'burger_1070': -8.658581488470569,
           'burger_2051': -3.618990615456312,
           'burger_2052': -2.856702984559962,
           'burger_2053': -6.164156666230159,
           'coke_2053': -6.164156666230159,
           'coke_2051': -3.618990615456312,
           'lemonade_2052': -2.856702984559962,
           'coffee_2053': -6.164156666230159}
```

TASK 9:-CREATE MODEL USING OLS AND APPLY ON ALL PRODUCTS.

In [72]:
```
burger_model = ols("QUANTITY ~ PRICE", data=burger_1070).fit()
print(burger_model.summary())
fig = plt.figure(figsize=(12,8))
fig = sm.graphics.plot_ccpr(burger_model, "PRICE")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               QUANTITY   R-squared:                       0.813
Model:                            OLS   Adj. R-squared:                  0.813
Method:                 Least Squares   F-statistic:                     1804.
Date:                Sat, 20 Jun 2020   Prob (F-statistic):          5.51e-153
Time:                        13:27:46   Log-Likelihood:                -1038.8
No. Observations:                 416   AIC:                             2082.
Df Residuals:                     414   BIC:                             2090.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    220.3600      3.090     71.322      0.000     214.287     226.433
PRICE         -8.6586      0.204    -42.474      0.000      -9.059      -8.258
==============================================================================
Omnibus:                      281.738   Durbin-Watson:                   1.989
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               25.687
Skew:                           0.012   Prob(JB):                     2.64e-06
Kurtosis:                       1.783   Cond. No.                         326.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

<Figure size 864x576 with 0 Axes>
```



TASK 10:-FINDING THE OPTIMAL PRICE FOR MAXIMUM PROFIT.

```
def find_optimal_price(data, model, buying_price):
    start_price = data.PRICE.min() - 1
    end_price = data.PRICE.min() + 10
    test = pd.DataFrame(columns = ["PRICE", "QUANTITY"])
    test['PRICE'] = np.arange(start_price, end_price,0.01)
    test['QUANTITY'] = model.predict(test['PRICE'])
    test['PROFIT'] = (test["PRICE"] - buying_price) * test["QUANTITY"]
    plt.plot(test['PRICE'],test['QUANTITY'])
    plt.plot(test['PRICE'],test['PROFIT'])
    plt.show()
    ind = np.where(test['PROFIT'] == test['PROFIT'].max())[0][0]
    values_at_max_profit = test.iloc[[ind]]
    return values_at_max_profit
```

TASK 11:-PROFIT MAXIMIZATION FOR ALL PRODUCTS.

```
optimal_price
```

```
{'burger_1070':       PRICE    QUANTITY       PROFIT
 422   17.22   71.259194   585.750572,
 'burger_2051':       PRICE    QUANTITY       PROFIT
 505   15.02   21.782193   131.128799,
 'burger_2052':       PRICE    QUANTITY      PROFIT
 522   14.34   15.259215   81.484211,
 'burger_2053':       PRICE    QUANTITY       PROFIT
 512   14.57   34.329951   191.217825,
 'coke_2051':        PRICE    QUANTITY       PROFIT
 505   15.02   21.782193   131.128799,
 'coke_2053':        PRICE    QUANTITY       PROFIT
 512   14.57   34.329951   191.217825,
 'lemonade_2052':       PRICE    QUANTITY      PROFIT
 522   14.34   15.259215   81.484211,
 'coffee_2053':       PRICE    QUANTITY       PROFIT
 512   14.57   34.329951   191.217825}
```

TASK 12:-CONCLUSION/ Summary:-

This is the price the cafe should set on it's item to earn maximum profit based on it's previous sales data. It is important to note that this is on a normal day. On 'other' days such as a holiday, or an event taking place have a different impact on customer buying behaviours and pattern. Usually an increase in consumption is seen on such days. These must be treated separately. Similarly, it is important to remove any external effects other than price that will affect the purchase behaviours of customers including the datapoints when the item was on discount.

Once, the new prices are put up, it is important to continuously monitor the sales and profit. If this method of pricing is a part of a rpoduct, a dashboard can be created for the purpose of monitoring these items and calculating the lift in the profit.

Reference:

.

1)https://onedrive.live.com/?authkey=%21AHDkRFUvt4FI8Wc&cid=7BA8848FE0992BD8&id=7BA8848FE0992BD8%21278810&parId=7BA8848FE0992BD8%21195633&action=locate

2) https://towardsdatascience.com/calculating-price-elasticity-of-demand-statistical-modeling-with-python-6adb2fa7824d

3) https://towardsdatascience.com/price-elasticity-data-understanding-and-data-exploration-first-of-all-ae4661da2ecb

4) https://towardsdatascience.com/calculating-price-elasticity-of-demand-statistical-modeling-with-python-6adb2fa7824d

.