Retail Analytics: Price Optimization Model

Project Mentors

Mr. Sharath Manikonda
Director, Innodatatics
https://www.linkedin.com/in/sharat-chandra

C. Gomathi
Mentor
https://www.linkedin.com/in/gomathi-chakravarthy-20a103241/

Team Members:

Name: Srikanth Dasari
LinkedIn URL – https://www.linkedin.com/in/srikanth-dasari-001a8b1ab

Name: Prajay Urkude
LinkedIn URL – https://www.linkedin.com/in/prajay-urkude-393a46110/

Name: Darshana Mahindrakar
LinkedIn URL – http://www.linkedin.com/in/darshana-mahindrakar-458773217

Name: Arish Naqvi
LinkedIn URL- https://www.linkedin.com/in/arish-naqvi-7bb789141/

Name: Kapil Khot
LinkedIn URL – https://www.linkedin.com/in/kapil-khot-bb7634238/

Name : Kewal Sarode
LinkedIn URL- https://www.linkedin.com/in/kewal-sarode-8149ab23a/

Name: Dileep Kumar
LinkedIn URL- http://www.linkedin.com/in/dileep-kumar-19919b201

Name: Manpreet Singh
LinkedIn URL – https://www.linkedin.com/in/manpreet-singh-9a5747122

INNODATATICS
Innovation • Data • Analytics

Team Members

Name: Muhammed Amer Hussain
LinkedIn URL –
https://www.linkedin.com/in/muhammed-amer-hussain-91179821a

Name: Uma Mahesh Chinta
LinkedIn URL –
https://www.linkedin.com/in/uma-mahesh-chinta-871a091bb

Name: Shaikh Mohammad Izhar
LinkedIn URL –
https://www.linkedin.com/in/shaikh-mohammad-izhar-793746238/

Name: Dasanagari Anil Kumar
LinkedIn URL –
https://www.linkedin.com/in/dasanagari-anil-kumar-8870551b3

Name: Kiran Inumula
LinkedIn URL –
https://www.linkedin.com/in/kiran-inumula-6545i

Name: Sagar Khiste
LinkedIn URL –
https://www.linkedin.com/in/sagar-khiste-609ba537

Name: Usha Rani Sahoo
LinkedIn URL –
https://www.linkedin.com/in/usha-rani-sahoo-1737b3228

Contents

Price Optimization Definition

- Price optimization is the process of identifying the optimal price point for any given product at any given location that will yield the highest profit

- The right pricing can make or break a business and copying your competitors might mean starting a price war, but making a guess could leave you balking at abysmal sales numbers.

- Hence, successful price optimization is a matter of a balance that can have a major impact on your sales, customer satisfaction, profits, and achievable growth goals.

Project Overview and Scope

- This project focuses on predicting an optimal price for a product which will yield maximum profit by increasing the product sale.
- This prediction system will help the client to get a suggested price based on the selling cost of a product and the number of products sold.
- The project will have a connection between the server where the sales data will be stored and the model will fetch the data to predict the optimized price. The deployment will be using streamlit.
- We are building this tool to minimize time and manpower & provide as accurate and thorough results as possible.

Project Goals

Objectives

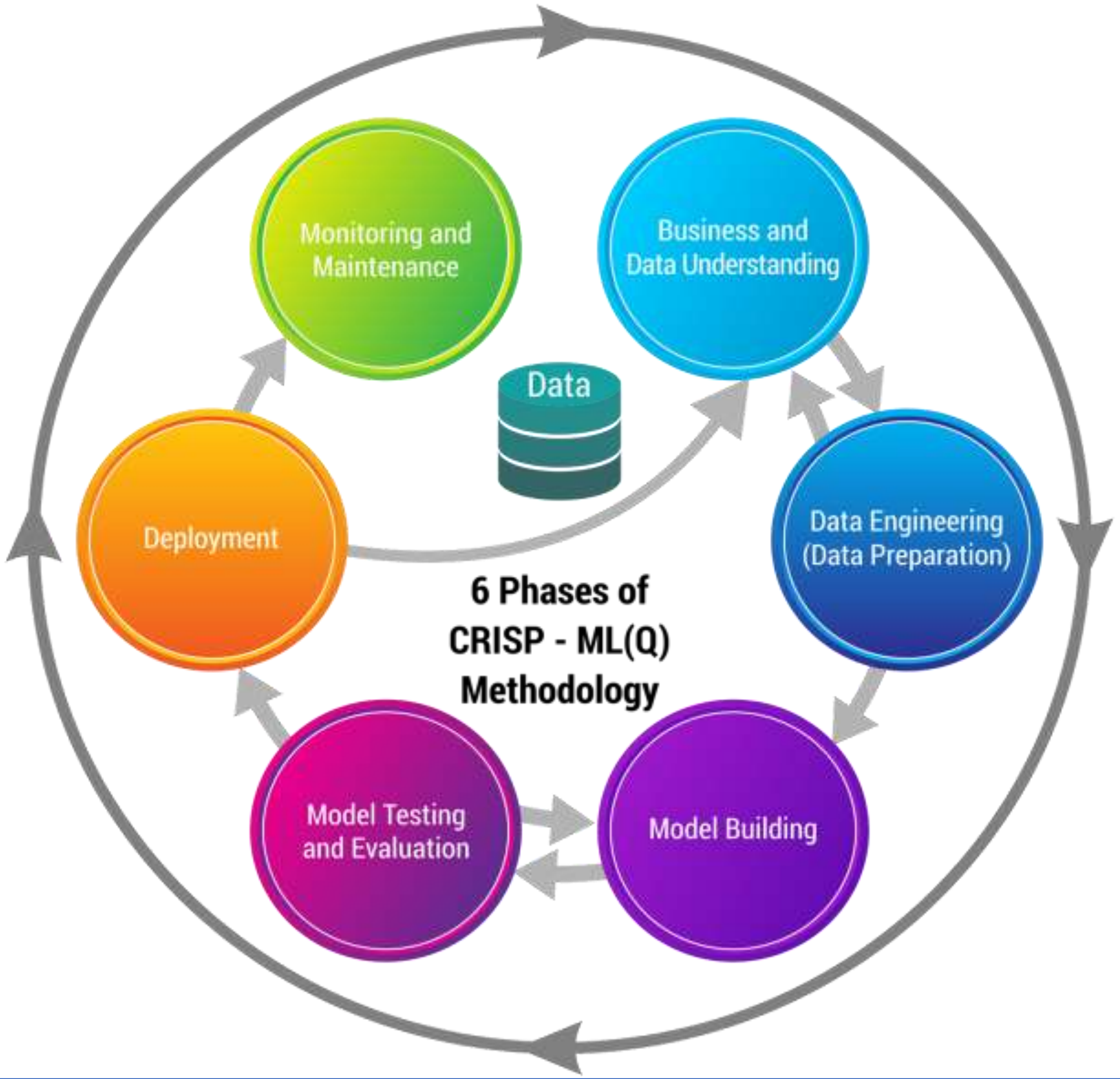
- To identify the best price by understanding pricing policy of each product in the retail market based on price optimization.
- To maximize the profitability and sales by finding the optimal price of a product.
- To minimize churning rate of customers to other vendors.

Constraints

- Changing in market demand of retail products may affect the optimized price.
- Price optimization for a product family –

Any changes in the pricing of one product, may trigger a chain reaction across a product family. Hence, the pricing of product family becomes a daunting task.

CRISP-ML(Q) Methodology


Monitoring and Maintenance
Business and Data Understanding
Data
Data Engineering (Data Preparation)
Deployment
6 Phases of CRISP - ML(Q) Methodology
Model Testing and Evaluation
Model Building

Technical Stacks

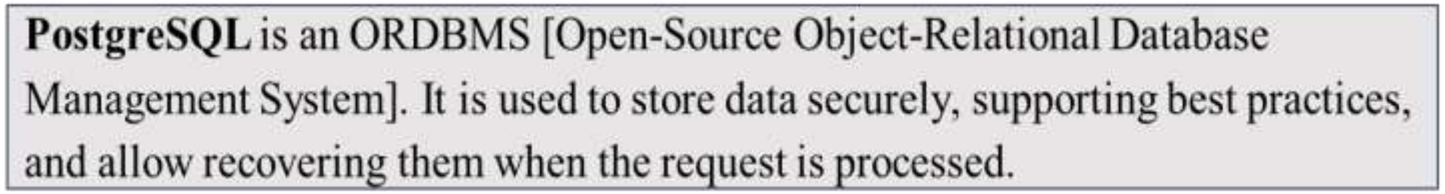
Language

Python is a general-purpose programming language. We used it for Data Cleaning, EDA, Model Building and Visualization.

IDE

Spyder is an integrated development environment (IDE) for scientific programming in the Python language.

Jupyter Notebook is a web-based interactive computing platform. It is an open-source IDE that is used to create Jupyter documents that can be created and shared with live codes.



PostgreSQL is an ORDBMS [Open-Source Object-Relational Database Management System]. It is used to store data securely, supporting best practices, and allow recovering them when the request is processed.

Technical Stacks

Libraries

Numpy can be used to perform a wide variety of mathematical operations on arrays.

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Deployment

Streamlit is an open-source python library for creating and sharing web apps for data science & machine learning projects.

System Requirements

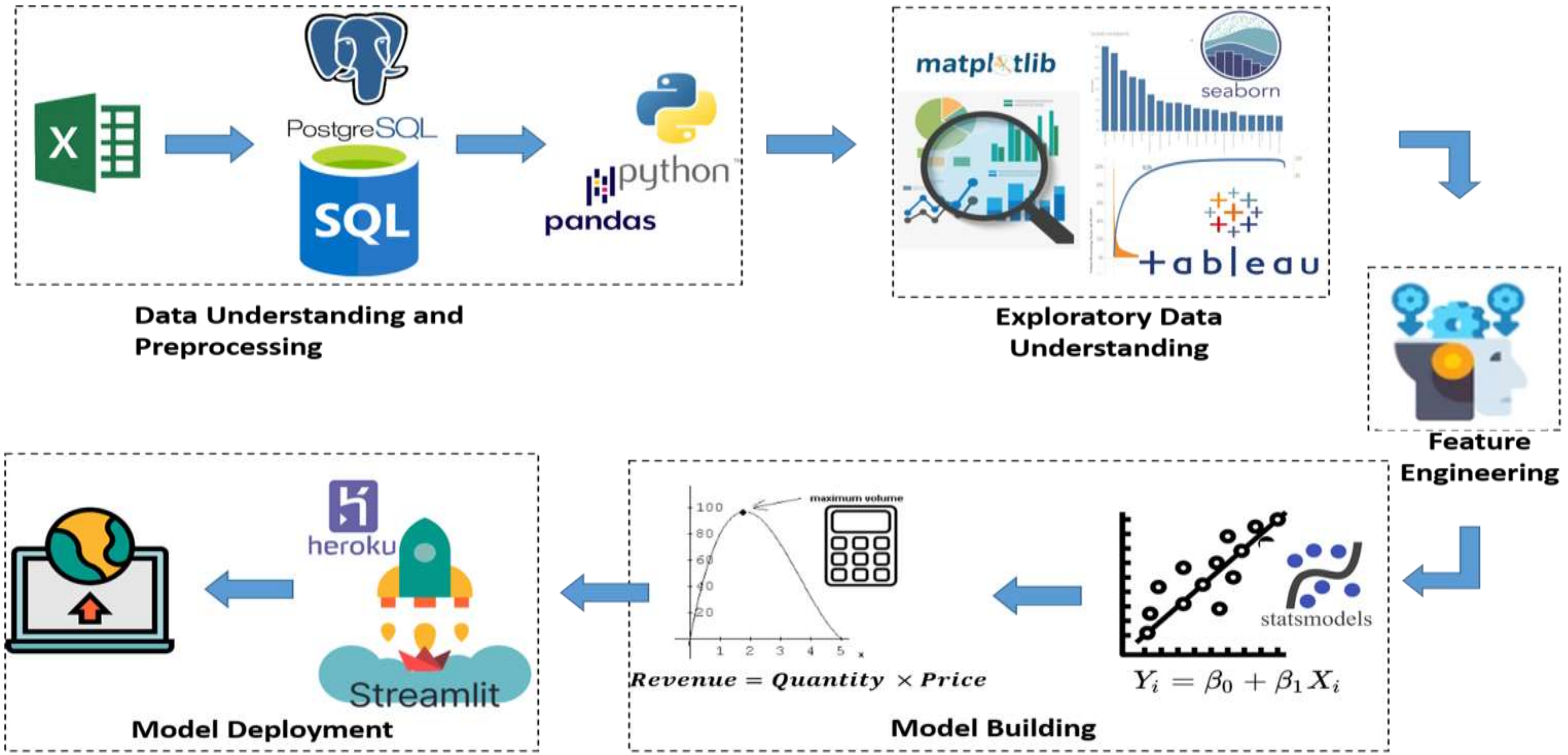
System Requirements

- Memory: 8GB RAM
- CPU: Intel Core i3 and above
- OS: Windows 10 & above

Prepare Environment

- Install python version 3.8
- Install Anaconda software to launch IDE platform like Spyder and Jupyter
- Creating a new Environment
- Install Pandas for preprocessing of data
- Install Numpy for mathematical calculations
- Install Matplotlib for Visualisation
- Install Streamlit for Deployment

Project Architecture



Data Collection

- Data Collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validation techniques.

- The transactional data is recorded from the client on a server. This server is duplicated for study purpose.

- Using PostgreSQL, a virtual local server is created in which a database is created. This database has all the records of transaction.

- The data consists of 37438 records having 4 zones, 128 material categories and 2312 different products.

Data Understanding

Particulars	Description
UID	Unique ID for a Material
NAME	Name of the Material
ZONE	Name of the zone of Business
Brand	Brand of the material
MC	Material Category: Category of the material
Fdate	Month of Sale
NSU	Net Sale Unit: Total units sold in a month
NSV	Net Sale Value: Total sale value in a month
GST Value	GST Value: GST on the NSV
(NSV-GST)	Calculation only
Sales at Cost	Cost to company

Data Understanding

Particulars	Description
SALES AT COST	Calculation only [(NSV-GST) – Sales at Cost]
MARGIN%	Percentage of Margin
Gross Sales	Gross Sales means the grand total of all sales transactions over a given time period.
Gross RGM(P-L)	Gross Margin
Gross Margin % (Q/P*100)	Gross Margin Percentage
MRP	Maximum Retail Price of unit item
SP	Selling price of unit item
DIS	Discount in Rupees.
DIS%	Discount in Percentage

Feature Selection

- Data contains various features and the shape of the data is 20 columns and 37437 rows.
- In this model the data is analyzed considering the below features for the model building.

Data Dictionary

Name of the features	**Description**	**Data Type**
NAME	Name of the Material	Nominal
ZONE	Name of the zone of Business	Nominal
NSU	Net Sale Unit: Total units sold in a month	Ratio
SP	Selling price of unit item	Ratio
Sales at cost	Cost to company	Ratio

Data Pre-Processing

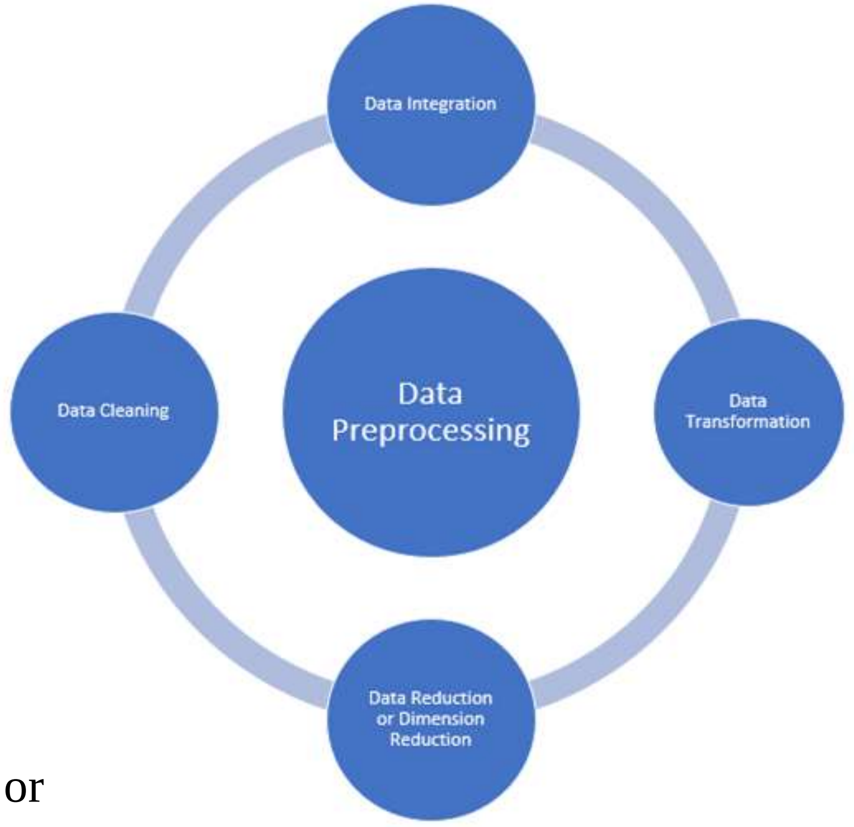
Steps in Data Pre-processing:

- Data Cleaning
  Fill in missing values, smooth noisy data, identify or remove outliers and resolve inconsistencies

- Data Integration
  Integration of multiple databases, data cubes, files, or notes

- Data Transformation
  Normalization (scaling to a specific range)

- Data Reduction
  Obtains reduced representation in volume but produces the same or similar analytical results



EDA Description

Analyzing the Dataset:

- By using these automated EDA’s like “SweetViz” and “D-Tale”, we analyzed datasets to summarize their statistics for numerical data and creating various graphical representations to understand the data better.

Measures of central tendency -

- Mean
- Median
- Mode

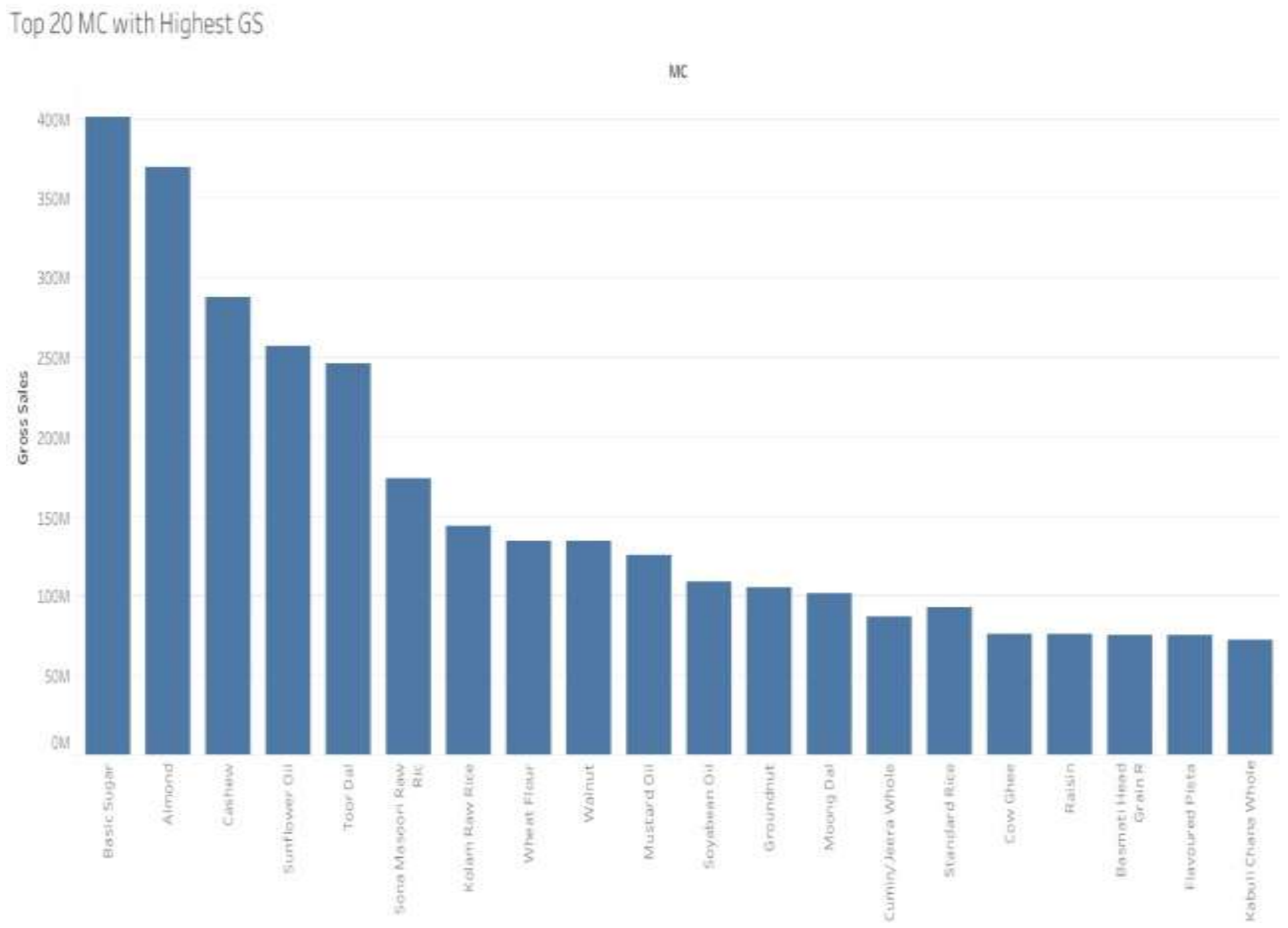
Measures of dispersion -

- Standard Deviation
- Variation

Third and Fourth Business Moment Decisions -

- Skewness
- Kurtosis

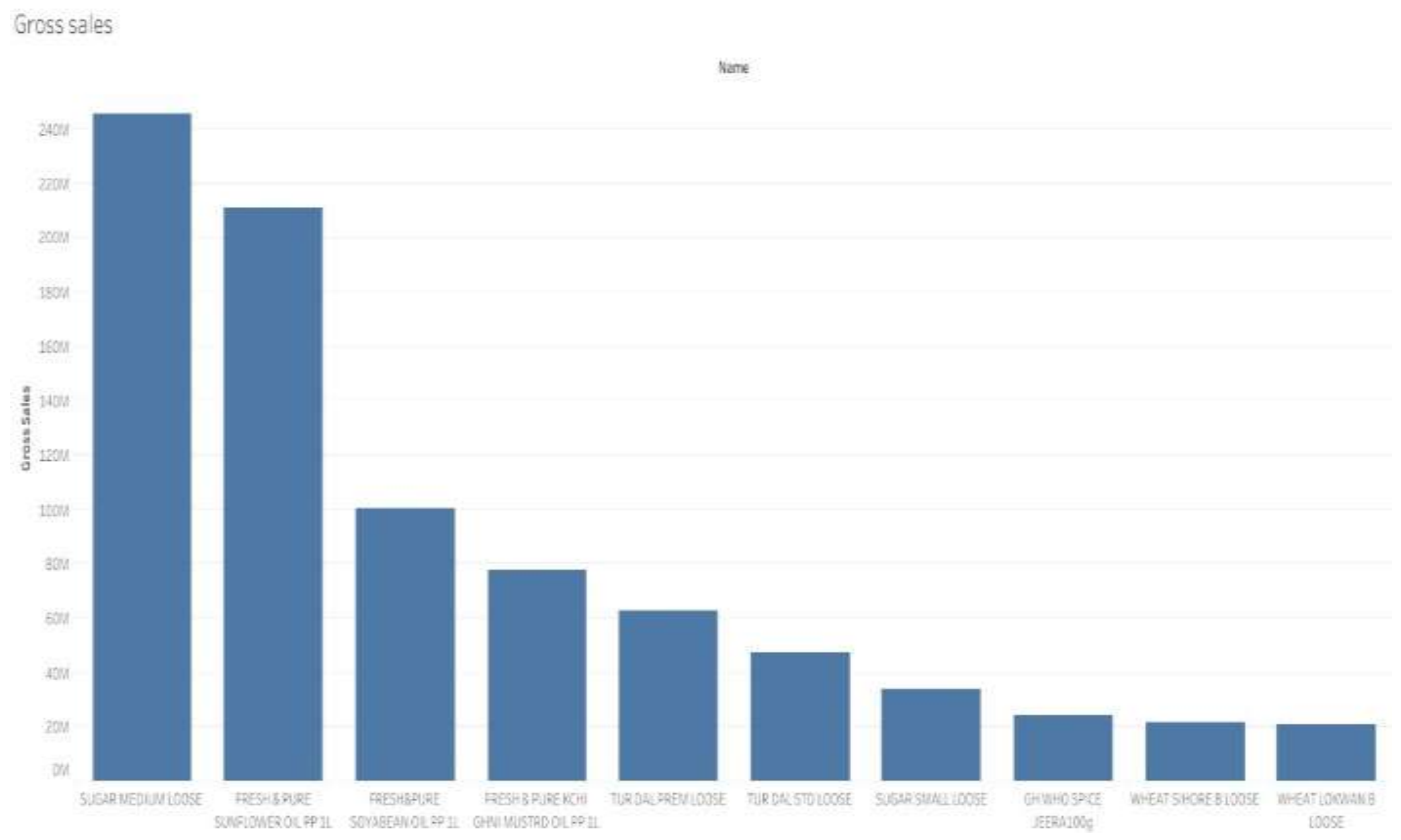
Exploratory Data Analysis



Top 20 Material categories By Gross Sales:

- Material category Brown sugar has the highest gross sale.
- Almond has second highest gross sale.

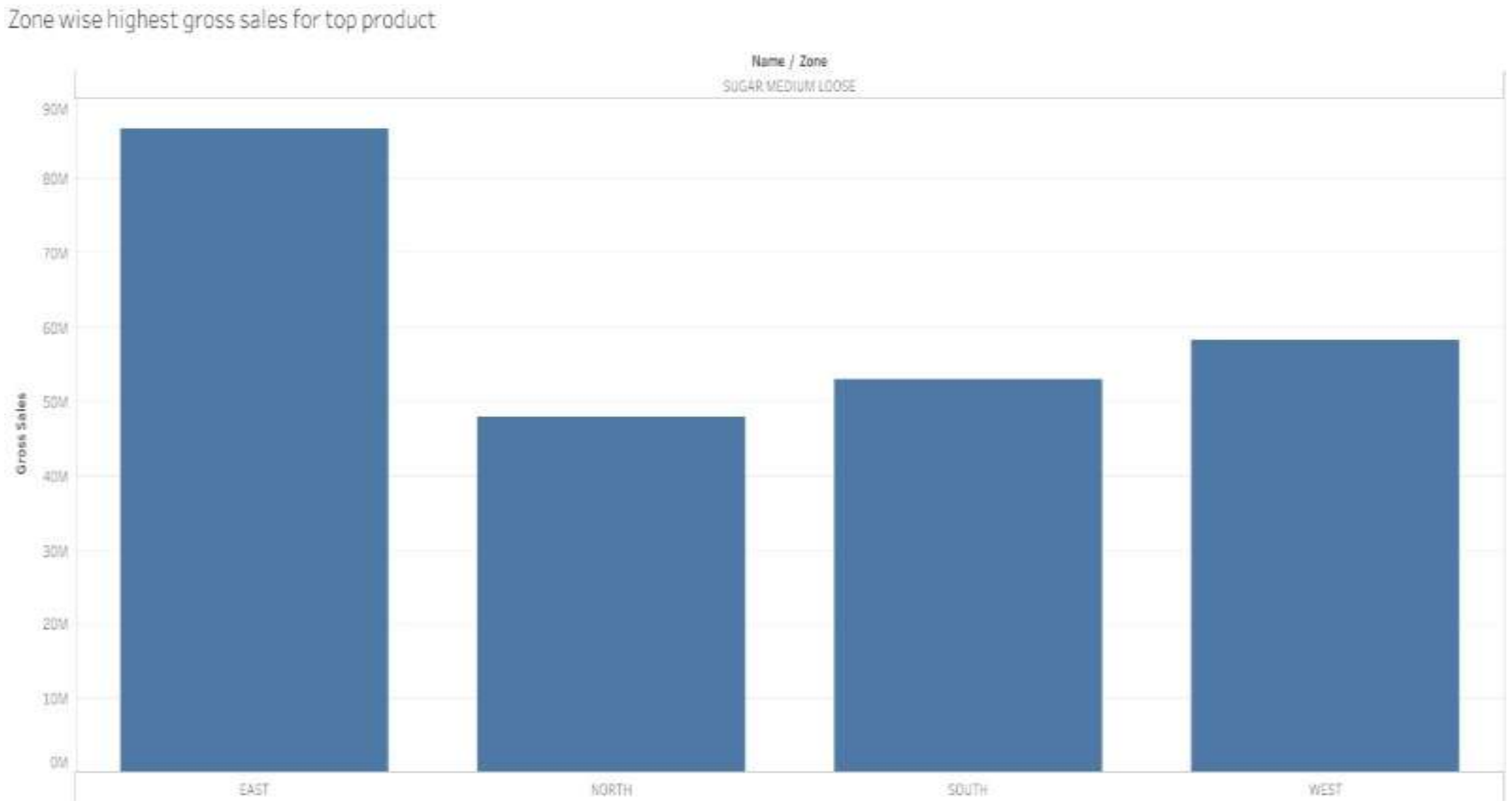
Exploratory Data Analysis



Top 10 Items By Gross Sales:

- Item sugar medium loose has the highest gross sales among all the items.

Exploratory Data Analysis

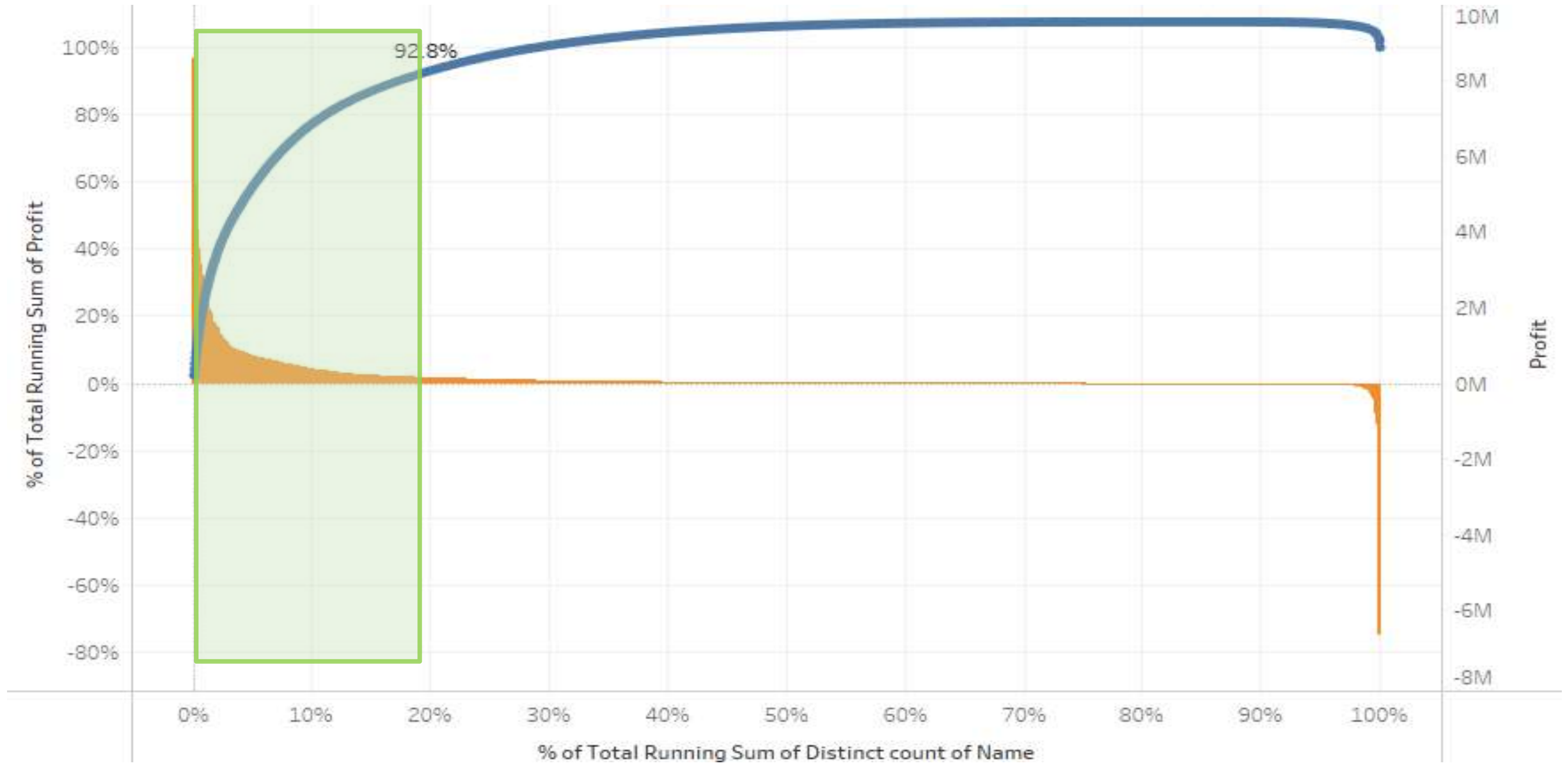


Job Dataset

Zone wise highest gross sales for top products:

- East Zone has highest gross sales
- North Zone has lowest gross sales .

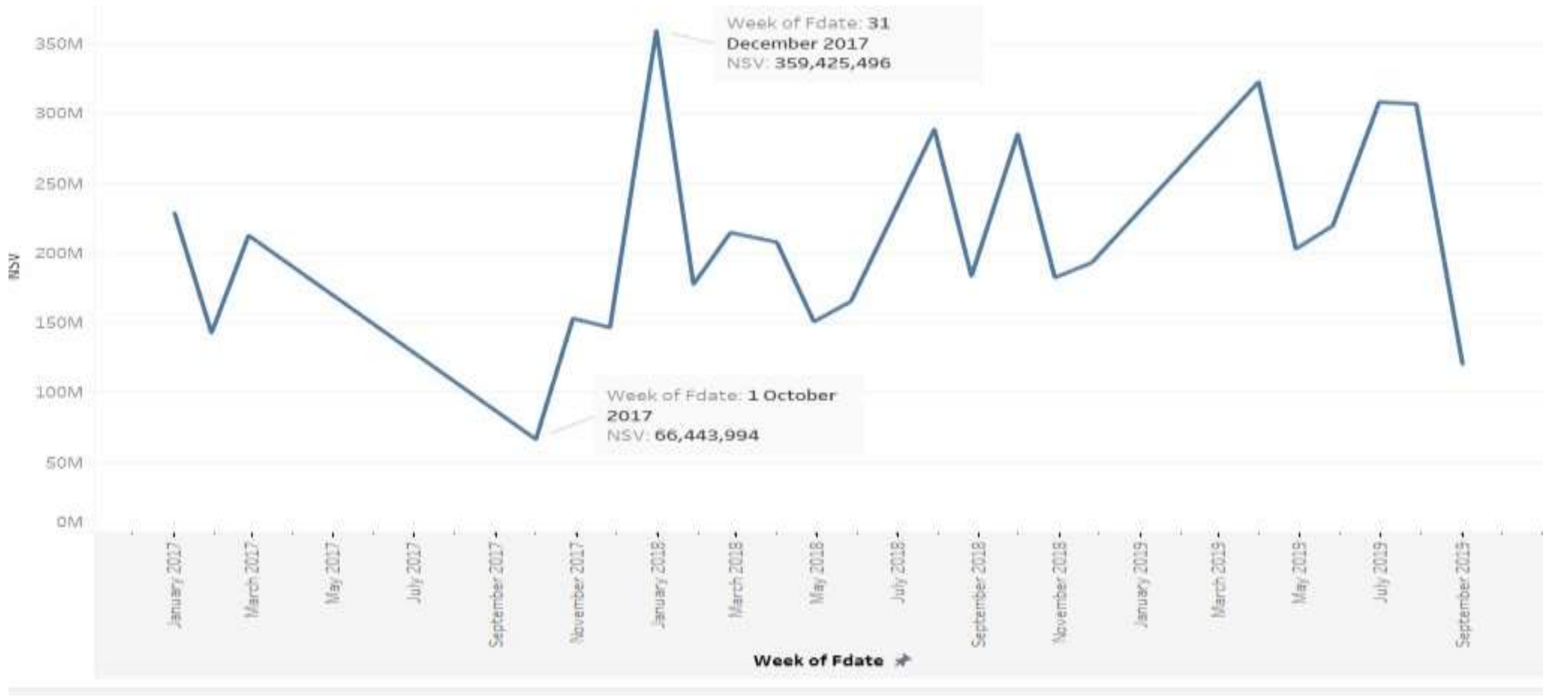
Exploratory Data Analysis:



Pareto Chart:

- This chart 80% of the profit comes from 20% of profit of items. From that we have to keep eye on item under this 20% region.

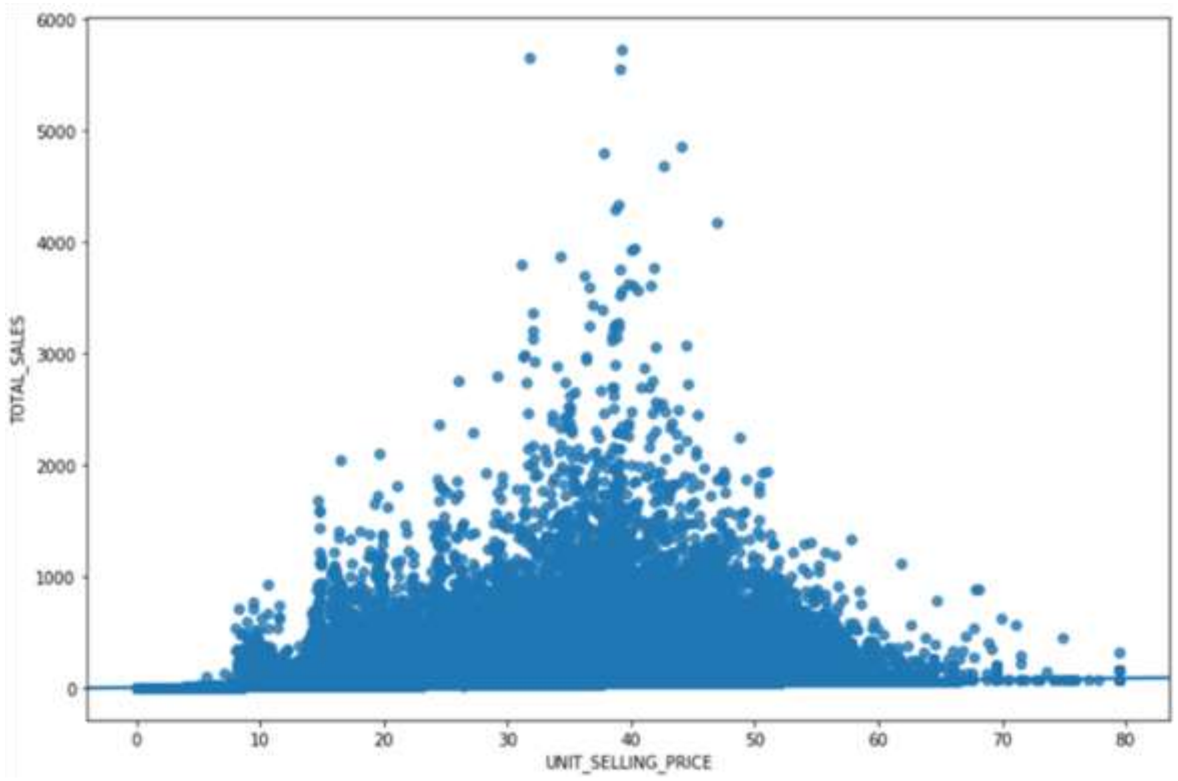
Exploratory Data Analysis



Sales Over time:

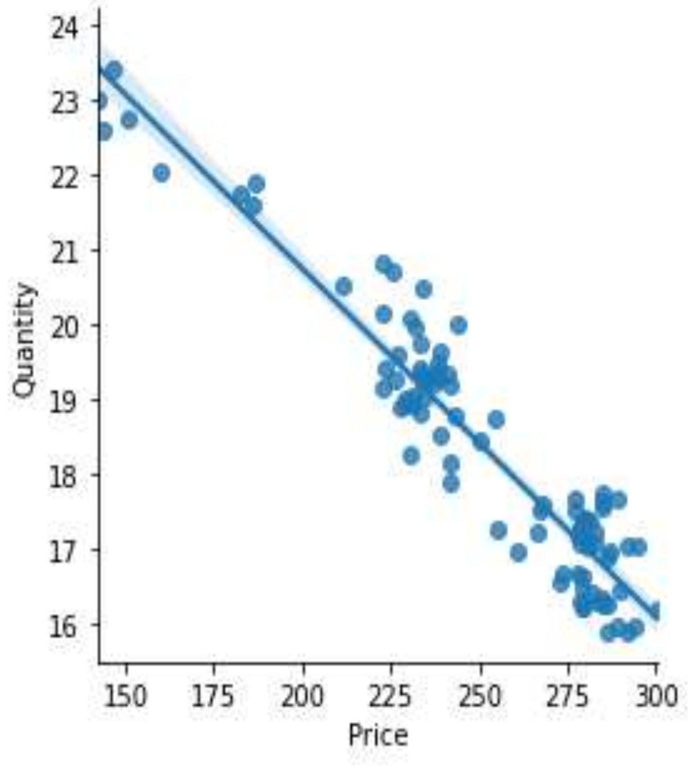
- Consistently sales is getting high in the month of december/ january .
- Consistently sales is getting low in the month of september.

Exploratory Data Analysis



- At Low selling price, sales is less
- After Certain point, higher selling price does not result in higher profit

Model Building



- Optimize selling price of the product to maximize profit and sales.
- Price elasticity ideally needs data on how customers react to price variation.
- A demand curve helps analyze the maximum price at which demand is not impacted.
- We will use existing data to draw demand curve to find range of potential sales price to optimize revenue or profit
- Based on the range, existing sales data will be used to predict or estimate the Selling price.
-

Model Building

Ordinary Least Squares regression (OLS) :

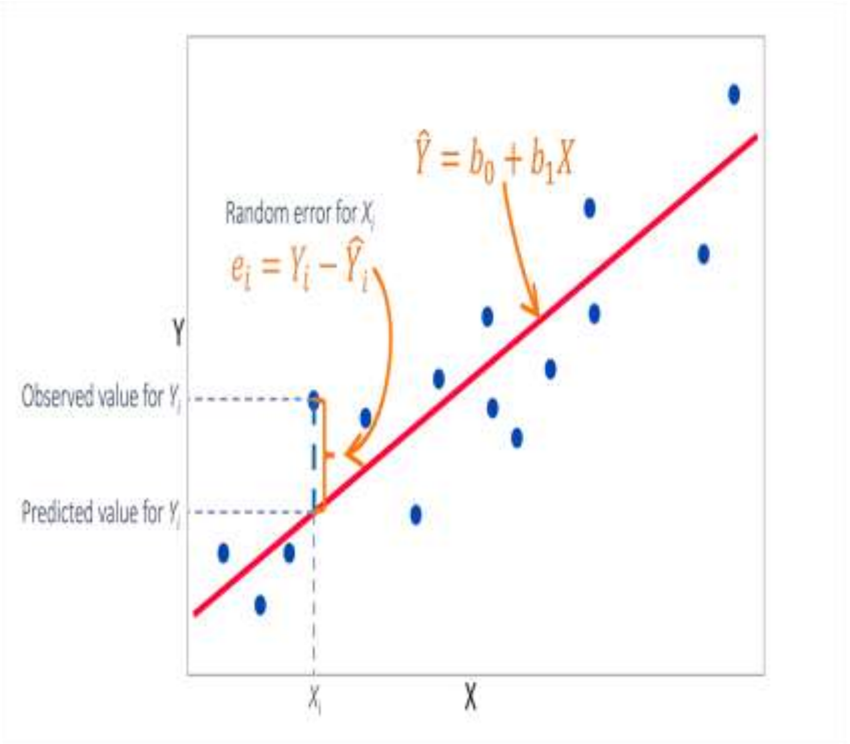
- OLS is a common technique for estimating coefficients of linear regression equations which describe the relationship between one or more independent quantitative variables and a dependent variable (simple or multiple linear regression).
- The simple linear regression is a model with a single regressor (independent variable) x that has a relationship with a response (dependent or target) y that is a
  $y = \beta0 + \beta1\ x + \varepsilon$

  Where

  β0: intercept

  β1: slope (unknown constant)

  ε: random error component



- This is a line where y is the dependent variable we want to predict, x is the independent variable, and β0 and β1 are the coefficients that we need to estimate.

Model Building

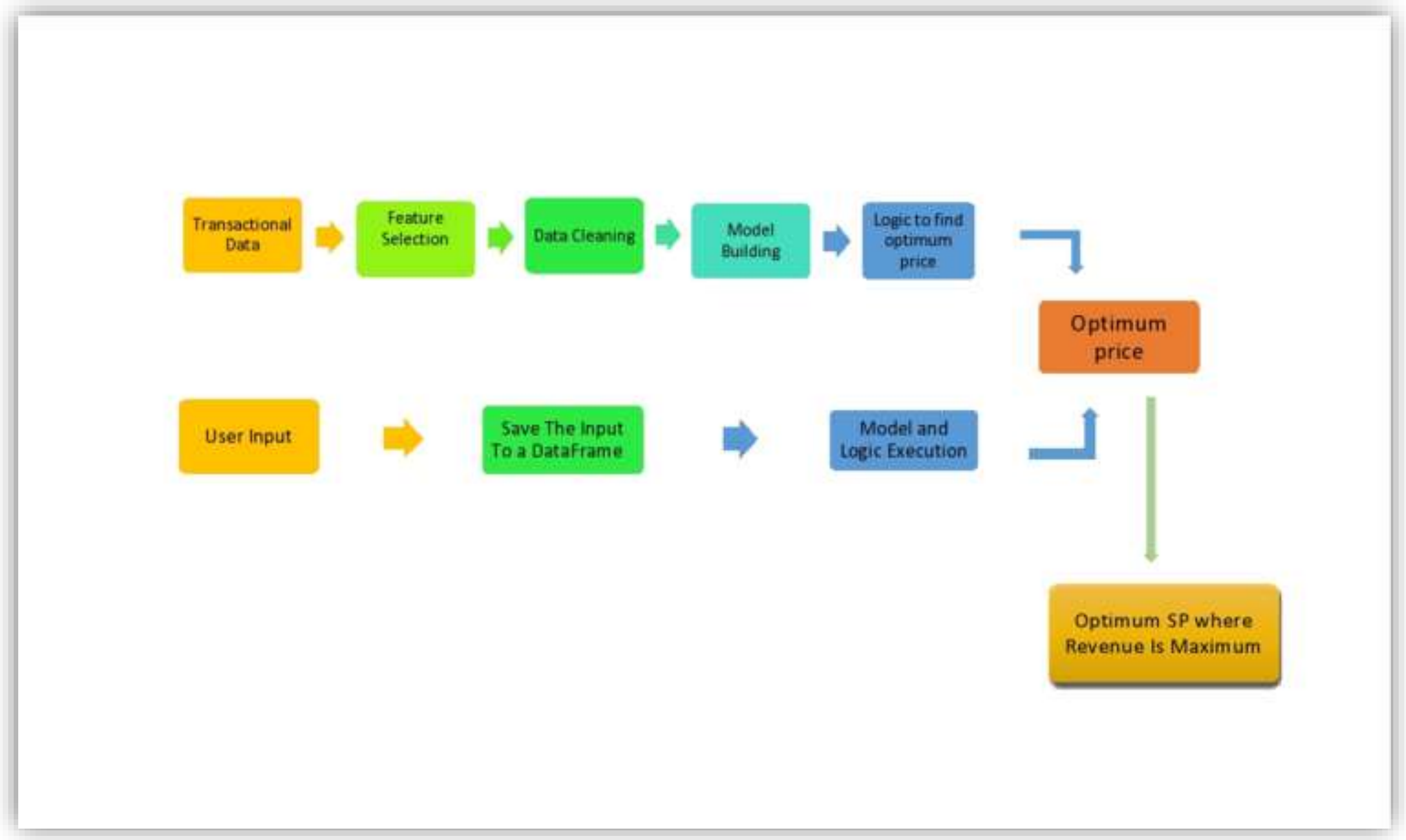

Transactional Data
Feature Selection
Data Cleaning
Model Building
Logic to find optimum price
Optimum price
User Input
Save The Input To a DataFrame
Model and Logic Execution
Optimum SP where Revenue Is Maximum

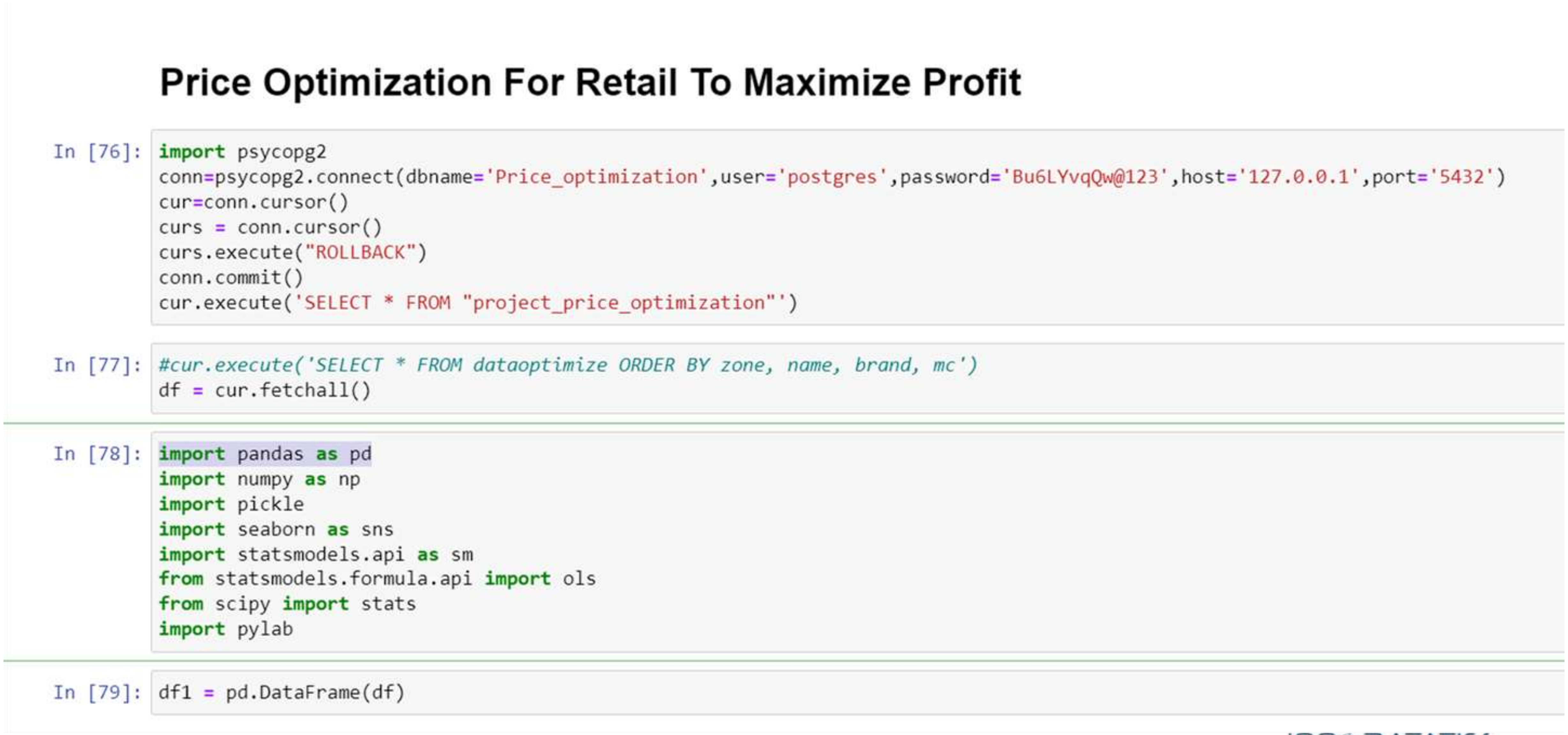
Fig : Model Building Process

Content - Price optimization for maximizing revenue by using Linear regression:
Regression analysis employing the use of historical data is widely used to estimate the effect of changes in price on sales.

Historical data can provide insight as to how sales volume will be affected by changes in price and market variables such as; seasonality, advertising, promotions, competitive product prices and other variables deemed appropriate.

Regression analysis produces a price elasticity measurement that quantify es the price sensitivity of consumers with respect to the observed product.

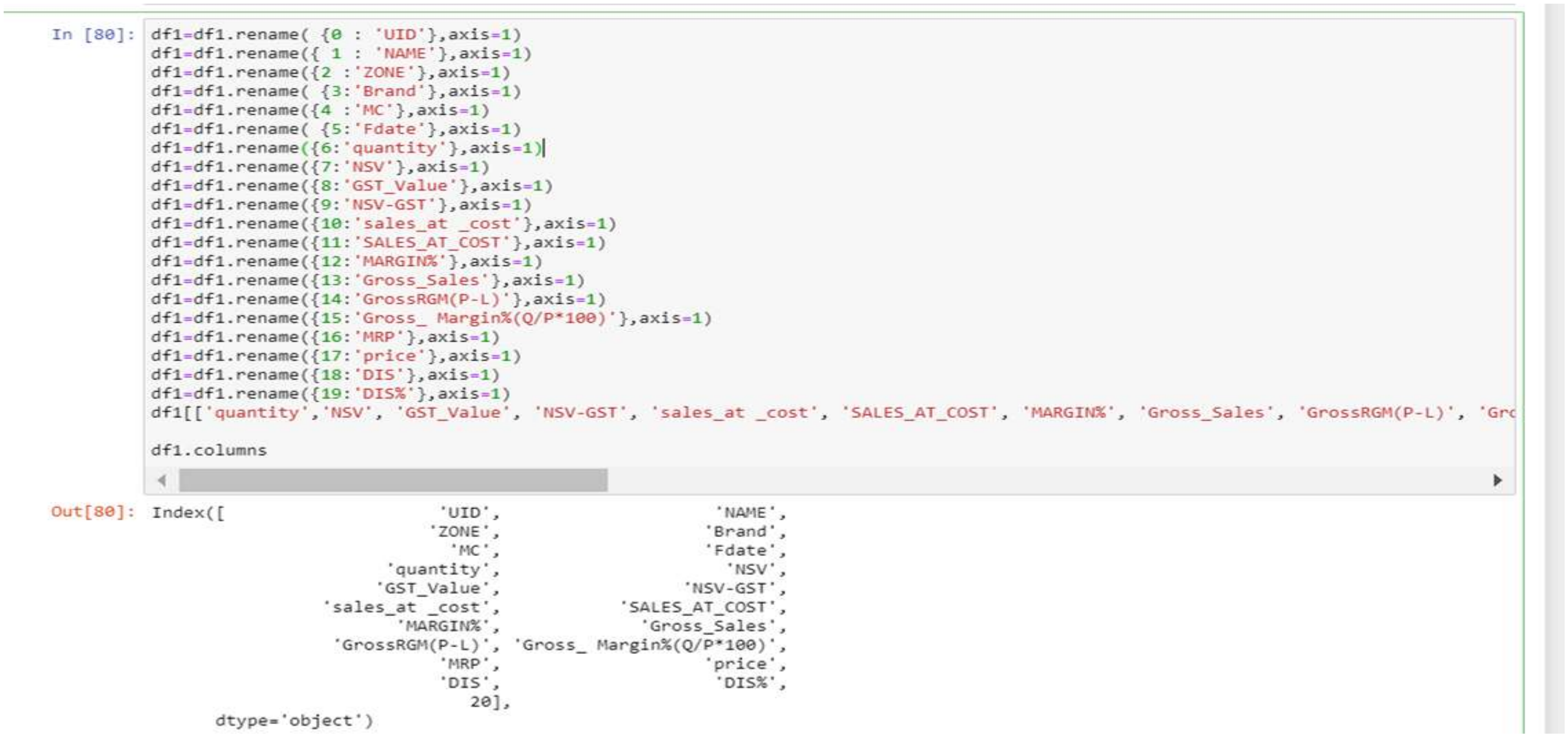
Model Building



Price Optimization For Retail To Maximize Profit

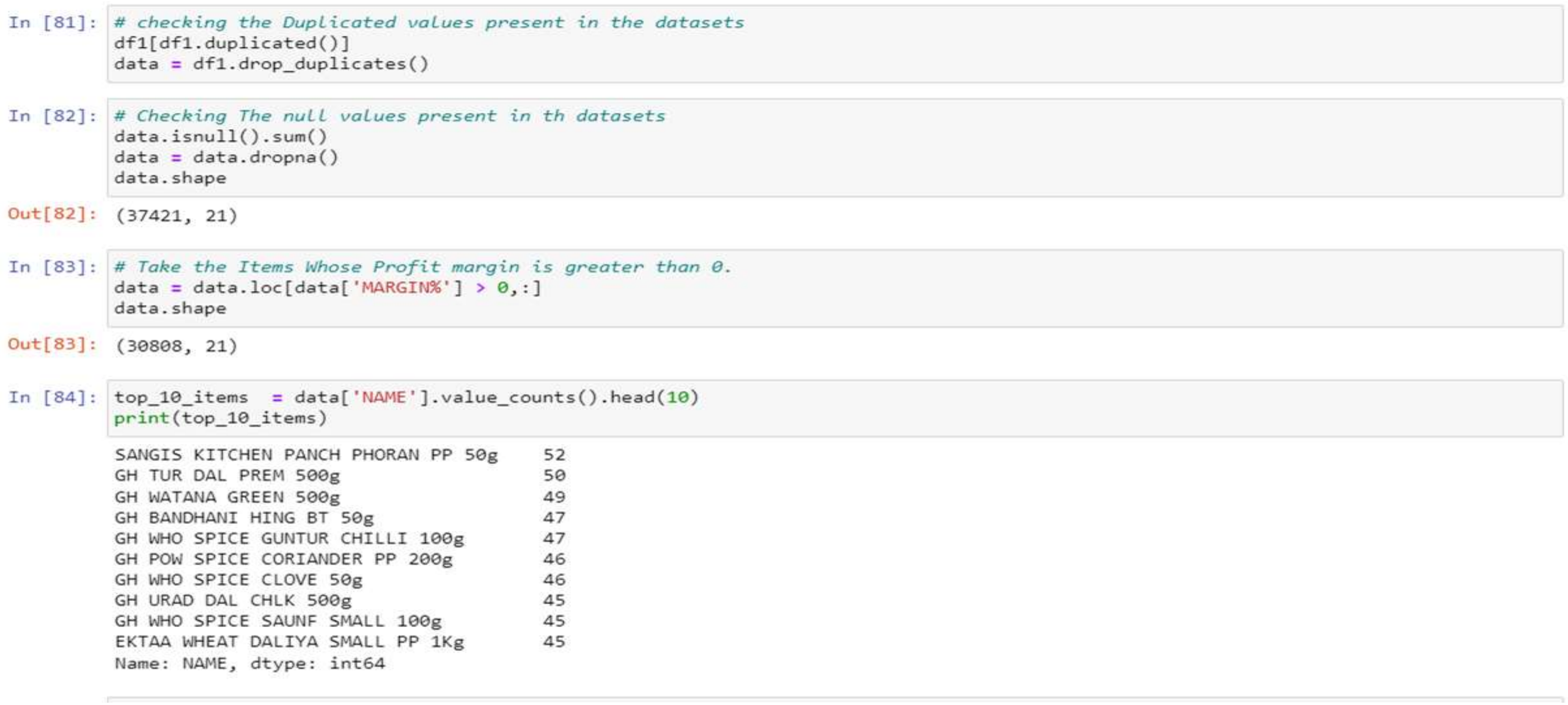
```
In [76]: import psycopg2
         conn=psycopg2.connect(dbname='Price_optimization',user='postgres',password='Bu6LYvqQw@123',host='127.0.0.1',port='5432')
         cur=conn.cursor()
         curs = conn.cursor()
         curs.execute("ROLLBACK")
         conn.commit()
         cur.execute('SELECT * FROM "project_price_optimization"')

In [77]: #cur.execute('SELECT * FROM dataoptimize ORDER BY zone, name, brand, mc')
         df = cur.fetchall()

In [78]: import pandas as pd
         import numpy as np
         import pickle
         import seaborn as sns
         import statsmodels.api as sm
         from statsmodels.formula.api import ols
         from scipy import stats
         import pylab

In [79]: df1 = pd.DataFrame(df)
```

Model Building



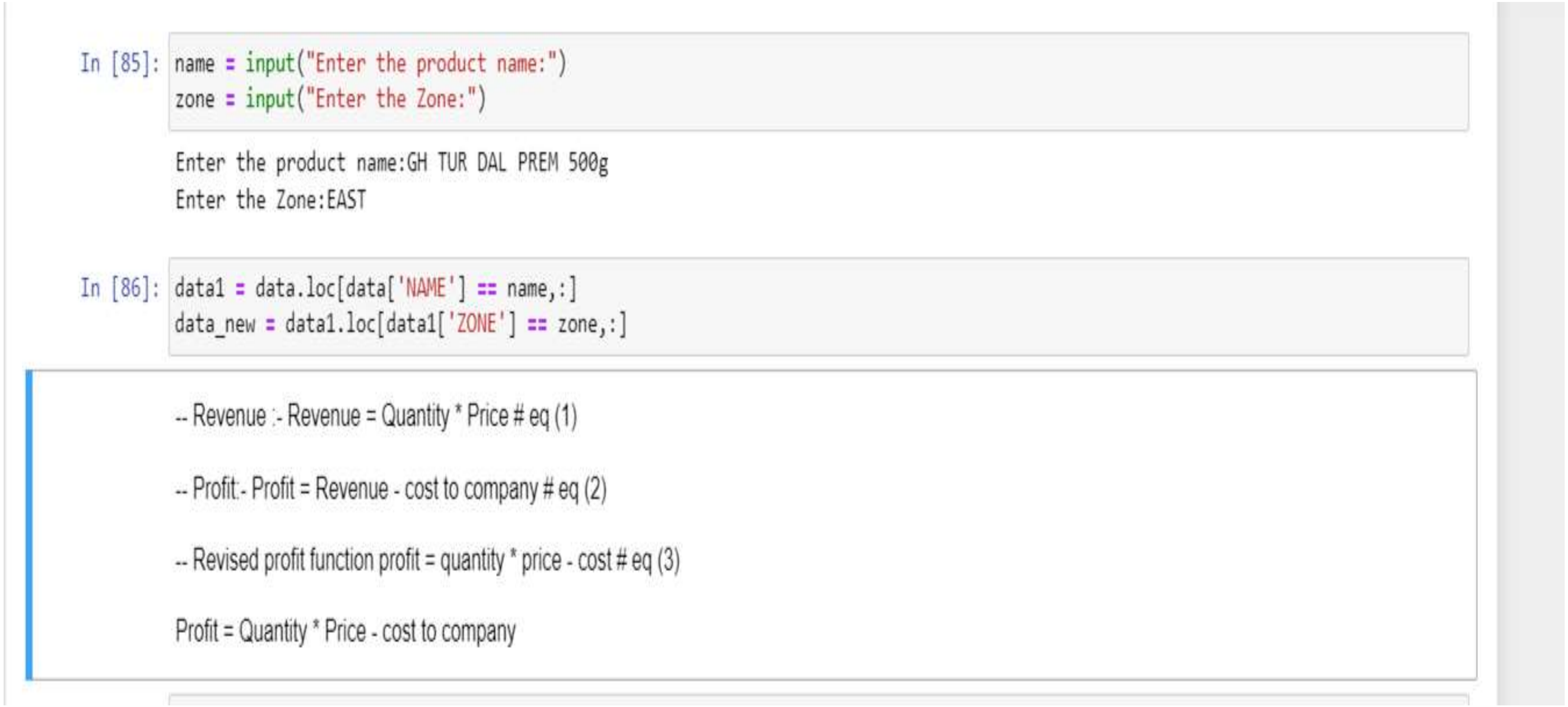
```
In [80]: df1=df1.rename( {0 : 'UID'},axis=1)
         df1=df1.rename({ 1 : 'NAME'},axis=1)
         df1=df1.rename({2 :'ZONE'},axis=1)
         df1=df1.rename( {3:'Brand'},axis=1)
         df1=df1.rename({4 :'MC'},axis=1)
         df1=df1.rename( {5:'Fdate'},axis=1)
         df1=df1.rename({6:'quantity'},axis=1)
         df1=df1.rename({7:'NSV'},axis=1)
         df1=df1.rename({8:'GST_Value'},axis=1)
         df1=df1.rename({9:'NSV-GST'},axis=1)
         df1=df1.rename({10:'sales_at _cost'},axis=1)
         df1=df1.rename({11:'SALES_AT_COST'},axis=1)
         df1=df1.rename({12:'MARGIN%'},axis=1)
         df1=df1.rename({13:'Gross_Sales'},axis=1)
         df1=df1.rename({14:'GrossRGM(P-L)'},axis=1)
         df1=df1.rename({15:'Gross_ Margin%(Q/P*100)'},axis=1)
         df1=df1.rename({16:'MRP'},axis=1)
         df1=df1.rename({17:'price'},axis=1)
         df1=df1.rename({18:'DIS'},axis=1)
         df1=df1.rename({19:'DIS%'},axis=1)
         df1[['quantity','NSV', 'GST_Value', 'NSV-GST', 'sales_at _cost', 'SALES_AT_COST', 'MARGIN%', 'Gross_Sales', 'GrossRGM(P-L)', 'Gro

         df1.columns

Out[80]: Index([                    'UID',                      'NAME',
                                   'ZONE',                     'Brand',
                                     'MC',                     'Fdate',
                               'quantity',                       'NSV',
                              'GST_Value',                   'NSV-GST',
                         'sales_at _cost',             'SALES_AT_COST',
                                'MARGIN%',               'Gross_Sales',
                          'GrossRGM(P-L)', 'Gross_ Margin%(Q/P*100)',
                                    'MRP',                     'price',
                                    'DIS',                      'DIS%',
                                       20],
               dtype='object')
```

Model Building



```
In [81]: # checking the Duplicated values present in the datasets
         df1[df1.duplicated()]
         data = df1.drop_duplicates()

In [82]: # Checking The null values present in th datasets
         data.isnull().sum()
         data = data.dropna()
         data.shape

Out[82]: (37421, 21)

In [83]: # Take the Items Whose Profit margin is greater than 0.
         data = data.loc[data['MARGIN%'] > 0,:]
         data.shape

Out[83]: (30808, 21)

In [84]: top_10_items  = data['NAME'].value_counts().head(10)
         print(top_10_items)

         SANGIS KITCHEN PANCH PHORAN PP 50g    52
         GH TUR DAL PREM 500g                  50
         GH WATANA GREEN 500g                  49
         GH BANDHANI HING BT 50g               47
         GH WHO SPICE GUNTUR CHILLI 100g       47
         GH POW SPICE CORIANDER PP 200g        46
         GH WHO SPICE CLOVE 50g                46
         GH URAD DAL CHLK 500g                 45
         GH WHO SPICE SAUNF SMALL 100g         45
         EKTAA WHEAT DALIYA SMALL PP 1Kg       45
         Name: NAME, dtype: int64
```

Model Building



In [85]:
```
name = input("Enter the product name:")
zone = input("Enter the Zone:")
```

```
Enter the product name:GH TUR DAL PREM 500g
Enter the Zone:EAST
```

In [86]:
```
data1 = data.loc[data['NAME'] == name,:]
data_new = data1.loc[data1['ZONE'] == zone,:]
```

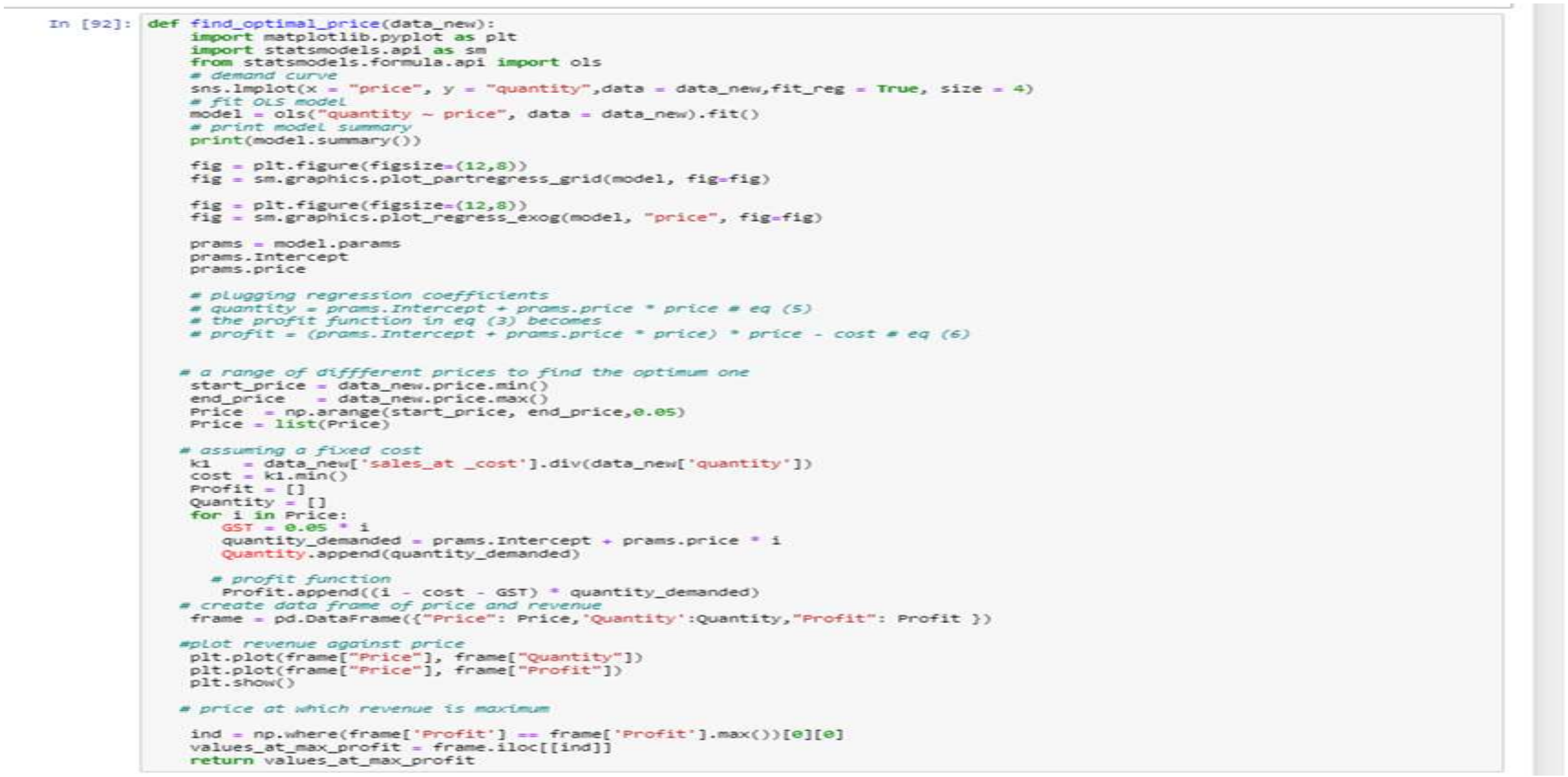
-- Revenue :- Revenue = Quantity * Price # eq (1)

-- Profit:- Profit = Revenue - cost to company # eq (2)

-- Revised profit function profit = quantity * price - cost # eq (3)

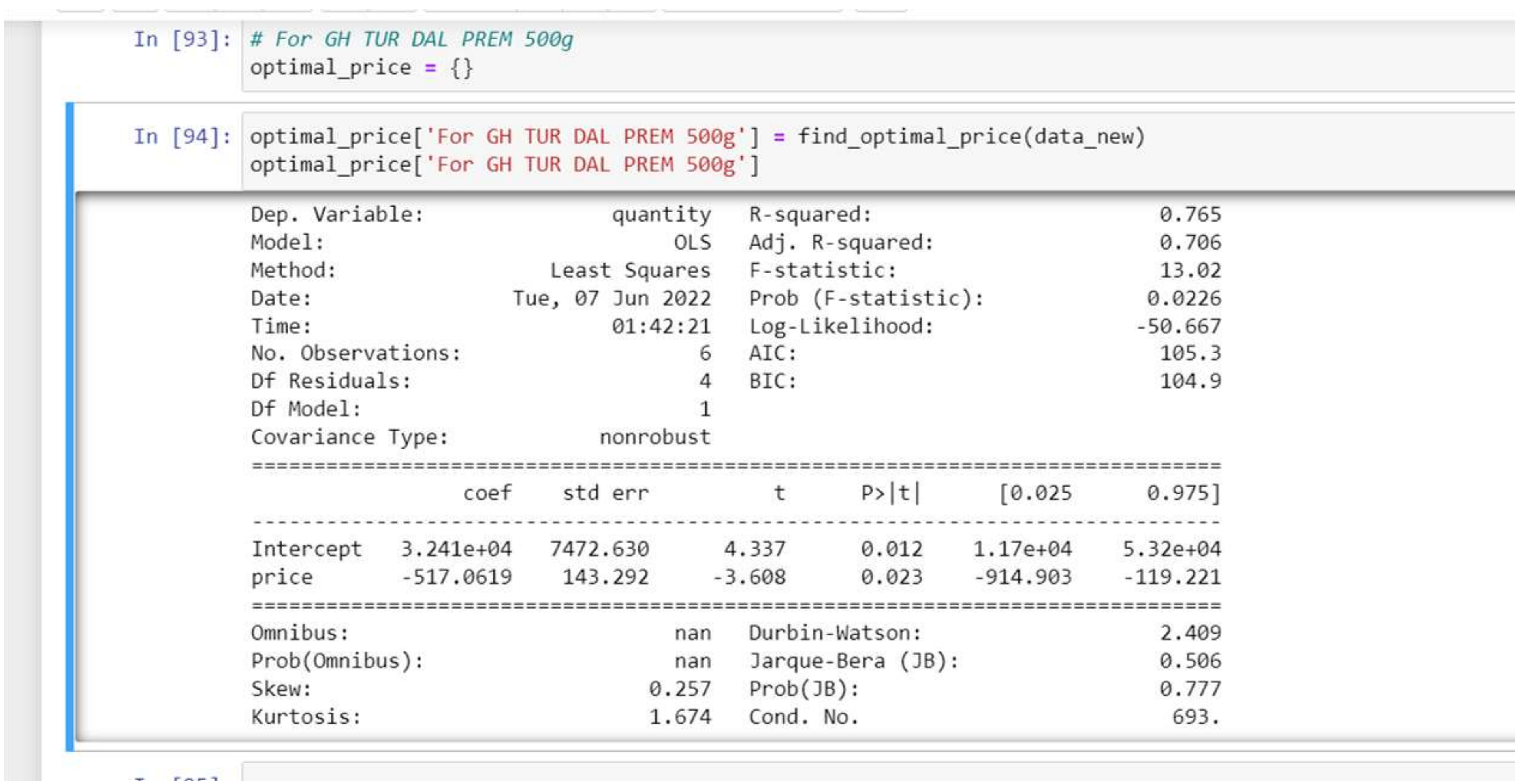
Profit = Quantity * Price - cost to company

Model Building



```
In [92]: def find_optimal_price(data_new):
             import matplotlib.pyplot as plt
             import statsmodels.api as sm
             from statsmodels.formula.api import ols
             # demand curve
             sns.lmplot(x = "price", y = "quantity",data = data_new,fit_reg = True, size = 4)
             # fit OLS model
             model = ols("quantity ~ price", data = data_new).fit()
             # print model summary
             print(model.summary())

             fig = plt.figure(figsize=(12,8))
             fig = sm.graphics.plot_partregress_grid(model, fig=fig)

             fig = plt.figure(figsize=(12,8))
             fig = sm.graphics.plot_regress_exog(model, "price", fig=fig)

             prams = model.params
             prams.Intercept
             prams.price

             # plugging regression coefficients
             # quantity = prams.Intercept + prams.price * price # eq (5)
             # the profit function in eq (3) becomes
             # profit = (prams.Intercept + prams.price * price) * price - cost # eq (6)


            # a range of diffferent prices to find the optimum one
             start_price = data_new.price.min()
             end_price   = data_new.price.max()
             Price  = np.arange(start_price, end_price,0.05)
             Price = list(Price)

            # assuming a fixed cost
             k1    = data_new['sales_at _cost'].div(data_new['quantity'])
             cost = k1.min()
             Profit = []
             Quantity = []
             for i in Price:
                GST = 0.05 * i
                quantity_demanded = prams.Intercept + prams.price * i
                Quantity.append(quantity_demanded)

               # profit function
                Profit.append((i - cost - GST) * quantity_demanded)
            # create data frame of price and revenue
             frame = pd.DataFrame({"Price": Price,'Quantity':Quantity,"Profit": Profit })

            #plot revenue against price
             plt.plot(frame["Price"], frame["Quantity"])
             plt.plot(frame["Price"], frame["Profit"])
             plt.show()

            # price at which revenue is maximum

             ind = np.where(frame['Profit'] == frame['Profit'].max())[0][0]
             values_at_max_profit = frame.iloc[[ind]]
             return values_at_max_profit
```

Model Building

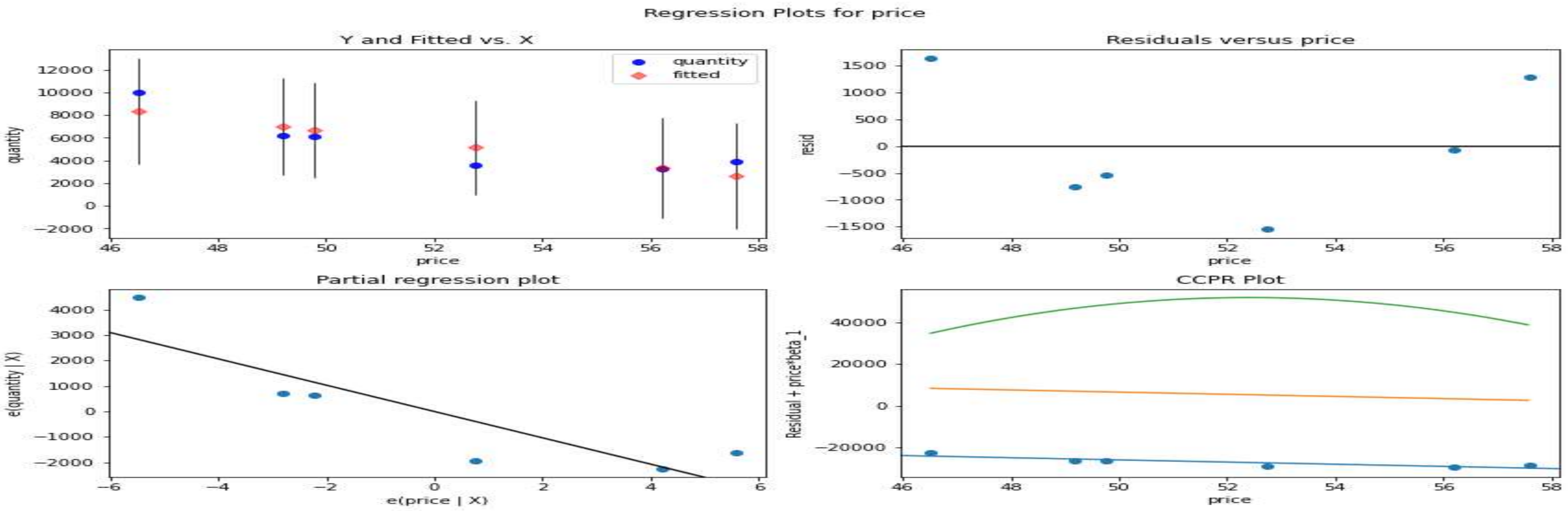


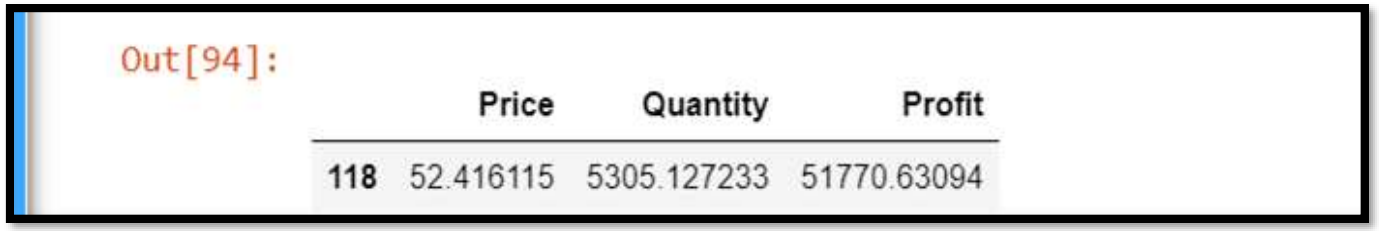
```
In [93]: # For GH TUR DAL PREM 500g
         optimal_price = {}

In [94]: optimal_price['For GH TUR DAL PREM 500g'] = find_optimal_price(data_new)
         optimal_price['For GH TUR DAL PREM 500g']

Dep. Variable:                quantity   R-squared:                       0.765
Model:                             OLS   Adj. R-squared:                  0.706
Method:                  Least Squares   F-statistic:                     13.02
Date:                 Tue, 07 Jun 2022   Prob (F-statistic):             0.0226
Time:                         01:42:21   Log-Likelihood:                -50.667
No. Observations:                    6   AIC:                             105.3
Df Residuals:                        4   BIC:                             104.9
Df Model:                            1
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3.241e+04   7472.630      4.337      0.012    1.17e+04    5.32e+04
price       -517.0619    143.292     -3.608      0.023    -914.903    -119.221
==============================================================================
Omnibus:                           nan   Durbin-Watson:                   2.409
Prob(Omnibus):                     nan   Jarque-Bera (JB):                0.506
Skew:                            0.257   Prob(JB):                        0.777
Kurtosis:                        1.674   Cond. No.                         693.
```

- R – square value is 0.765
- P – Values are less than 0.05 for both the features

Model Building

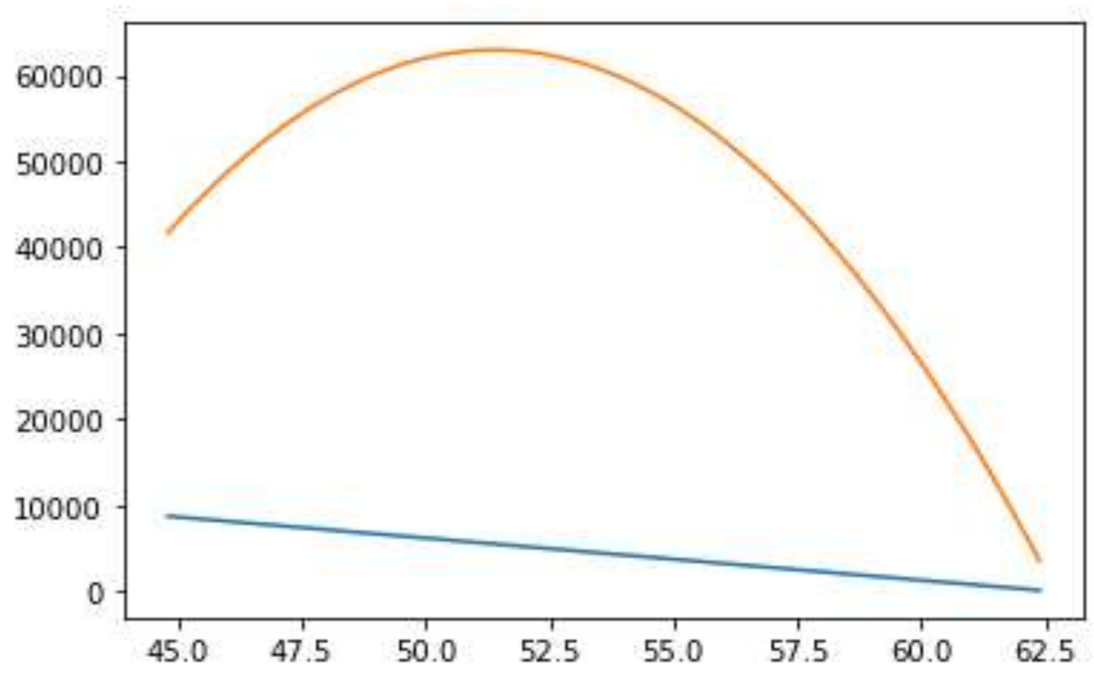
Partial Regression Plot for Popular product GH TUR DAL PREM 500g :

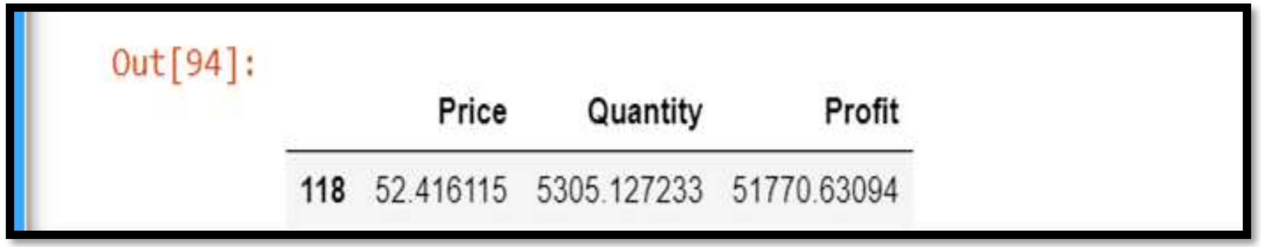



```
Out[94]:
        Price      Quantity       Profit
118  52.416115  5305.127233  51770.63094
```

Model Building

Demand curve for popular product GH TUR DAL PREM 500g :





```
Out[94]:
           Price      Quantity        Profit
118    52.416115   5305.127233   51770.63094
```

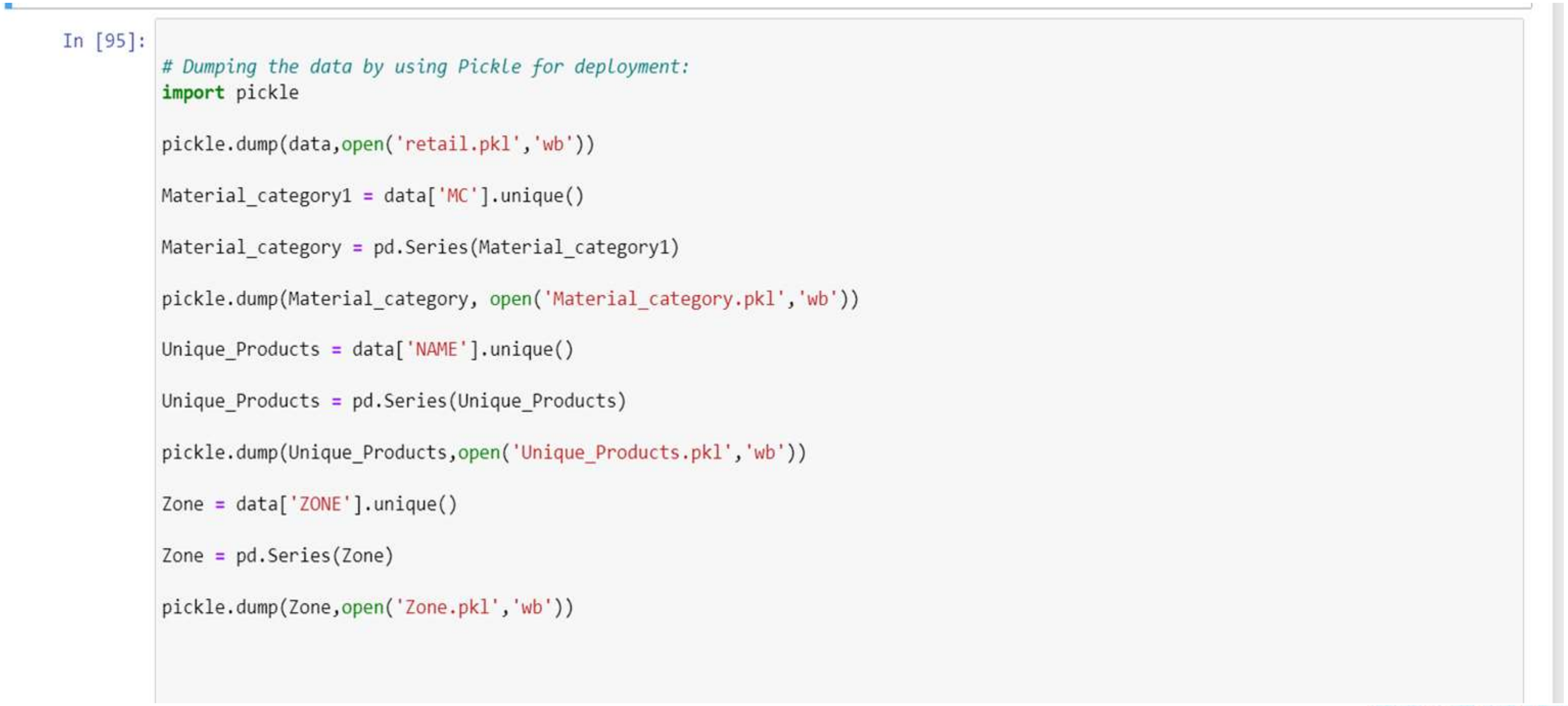
- Demand peak is when selling price is between 49 to 55. above 55 people are resisting to buy it
- If price and revenue are plotted, we can visually identify the peak of the revenue and find the price that makes the revenue at the highest point on the curve.
- So we find that the maximum revenue at different price levels is reached at 550000 when the price is set at 89.

Model Deployment

- Web based deployment was used to deploy code from source control to hosting platform
- Streamlit library was used for app deployment
- Streamlit integrates with GitHub to make it easy to deploy code living on GitHub to apps running on streamlit
- Pickle the required files (unique products,zone)
- Create a GitHub repository containing all the required files (pickled files, installation requirements, text files) etc.
- Create an app in streamlit and connect it to the GitHub repository

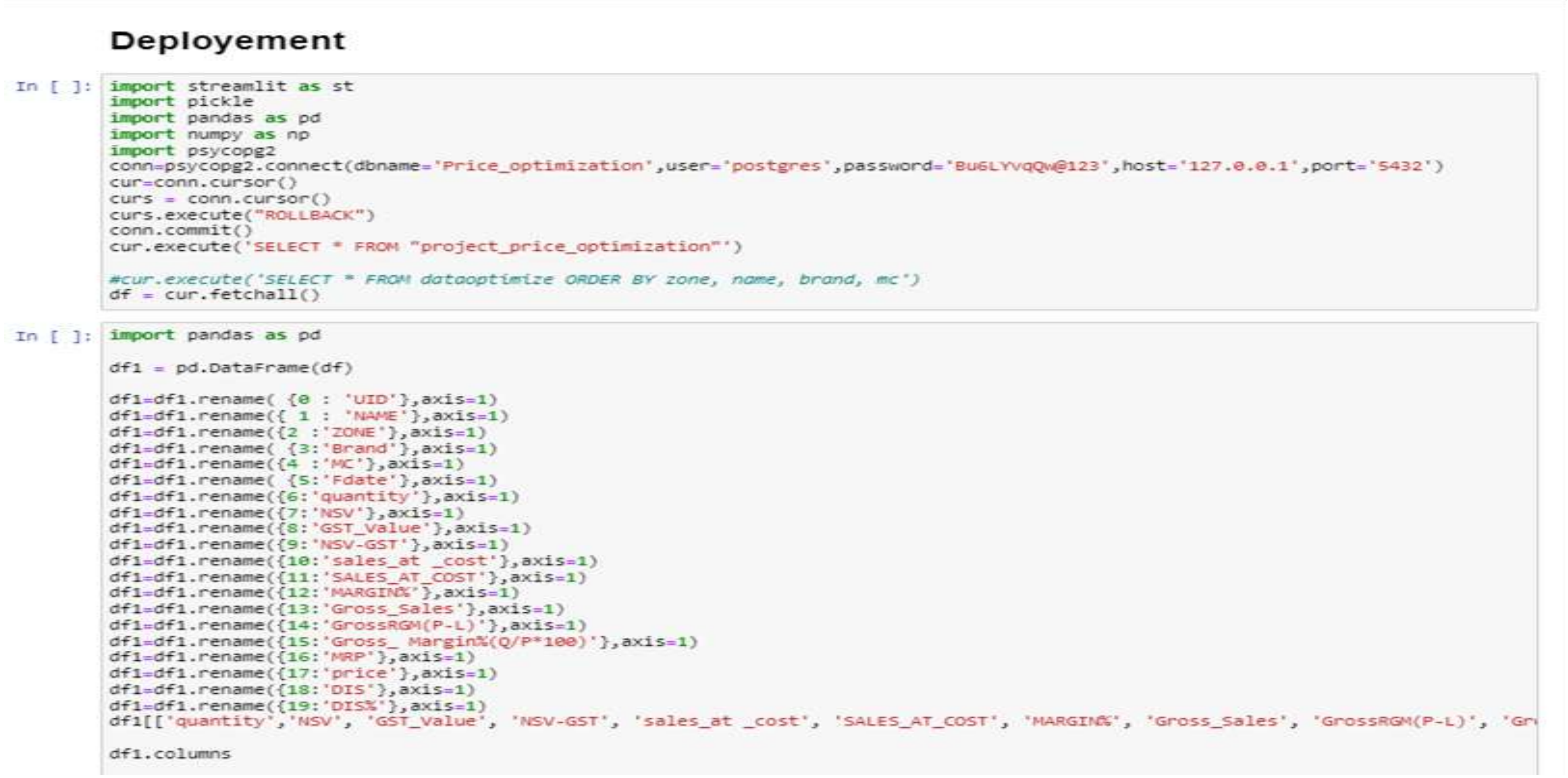
Model Deployment

Creation of Pickle Files:



```
In [95]:
# Dumping the data by using Pickle for deployment:
import pickle

pickle.dump(data,open('retail.pkl','wb'))

Material_category1 = data['MC'].unique()

Material_category = pd.Series(Material_category1)

pickle.dump(Material_category, open('Material_category.pkl','wb'))

Unique_Products = data['NAME'].unique()

Unique_Products = pd.Series(Unique_Products)

pickle.dump(Unique_Products,open('Unique_Products.pkl','wb'))

Zone = data['ZONE'].unique()

Zone = pd.Series(Zone)

pickle.dump(Zone,open('Zone.pkl','wb'))
```

Model Deployment

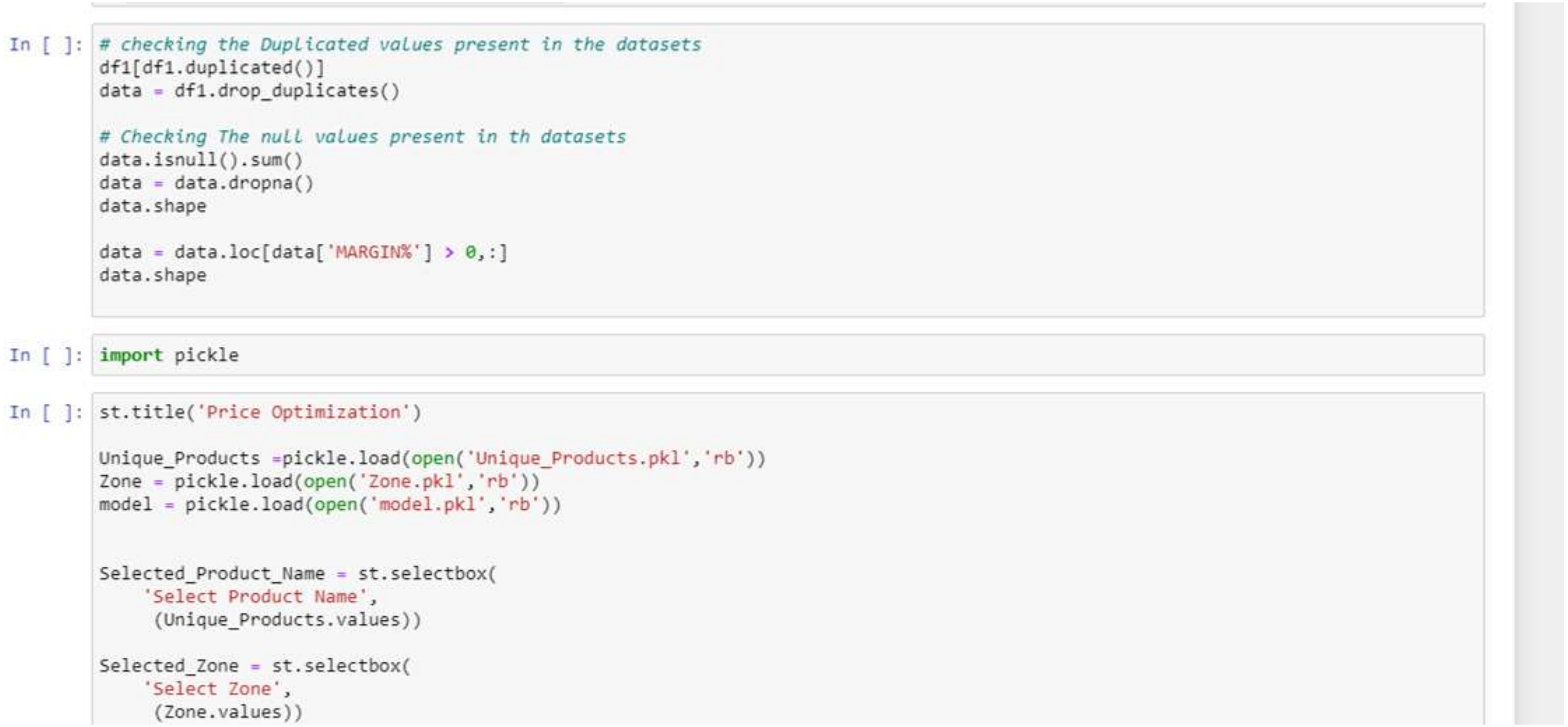


Deployement

```
In [ ]: import streamlit as st
        import pickle
        import pandas as pd
        import numpy as np
        import psycopg2
        conn=psycopg2.connect(dbname='Price_optimization',user='postgres',password='BuGLYvqQw@123',host='127.0.0.1',port='5432')
        cur=conn.cursor()
        curs = conn.cursor()
        curs.execute("ROLLBACK")
        conn.commit()
        cur.execute('SELECT * FROM "project_price_optimization"')

        #cur.execute('SELECT * FROM dataoptimize ORDER BY zone, name, brand, mc')
        df = cur.fetchall()
```

```
In [ ]: import pandas as pd

        df1 = pd.DataFrame(df)

        df1=df1.rename( {0 : 'UID'},axis=1)
        df1=df1.rename({ 1 : 'NAME'},axis=1)
        df1=df1.rename({2 :'ZONE'},axis=1)
        df1=df1.rename( {3:'Brand'},axis=1)
        df1=df1.rename({4 :'MC'},axis=1)
        df1=df1.rename( {5:'Fdate'},axis=1)
        df1=df1.rename({6:'quantity'},axis=1)
        df1=df1.rename({7:'NSV'},axis=1)
        df1=df1.rename({8:'GST_Value'},axis=1)
        df1=df1.rename({9:'NSV-GST'},axis=1)
        df1=df1.rename({10:'sales_at _cost'},axis=1)
        df1=df1.rename({11:'SALES_AT_COST'},axis=1)
        df1=df1.rename({12:'MARGIN%'},axis=1)
        df1=df1.rename({13:'Gross_Sales'},axis=1)
        df1=df1.rename({14:'GrossRGM(P-L)'},axis=1)
        df1=df1.rename({15:'Gross_ Margin%(Q/P*100)'},axis=1)
        df1=df1.rename({16:'MRP'},axis=1)
        df1=df1.rename({17:'price'},axis=1)
        df1=df1.rename({18:'DIS'},axis=1)
        df1=df1.rename({19:'DIS%'},axis=1)
        df1[['quantity','NSV', 'GST_Value', 'NSV-GST', 'sales_at _cost', 'SALES_AT_COST', 'MARGIN%', 'Gross_Sales', 'GrossRGM(P-L)', 'Gr

        df1.columns
```

Model Deployment

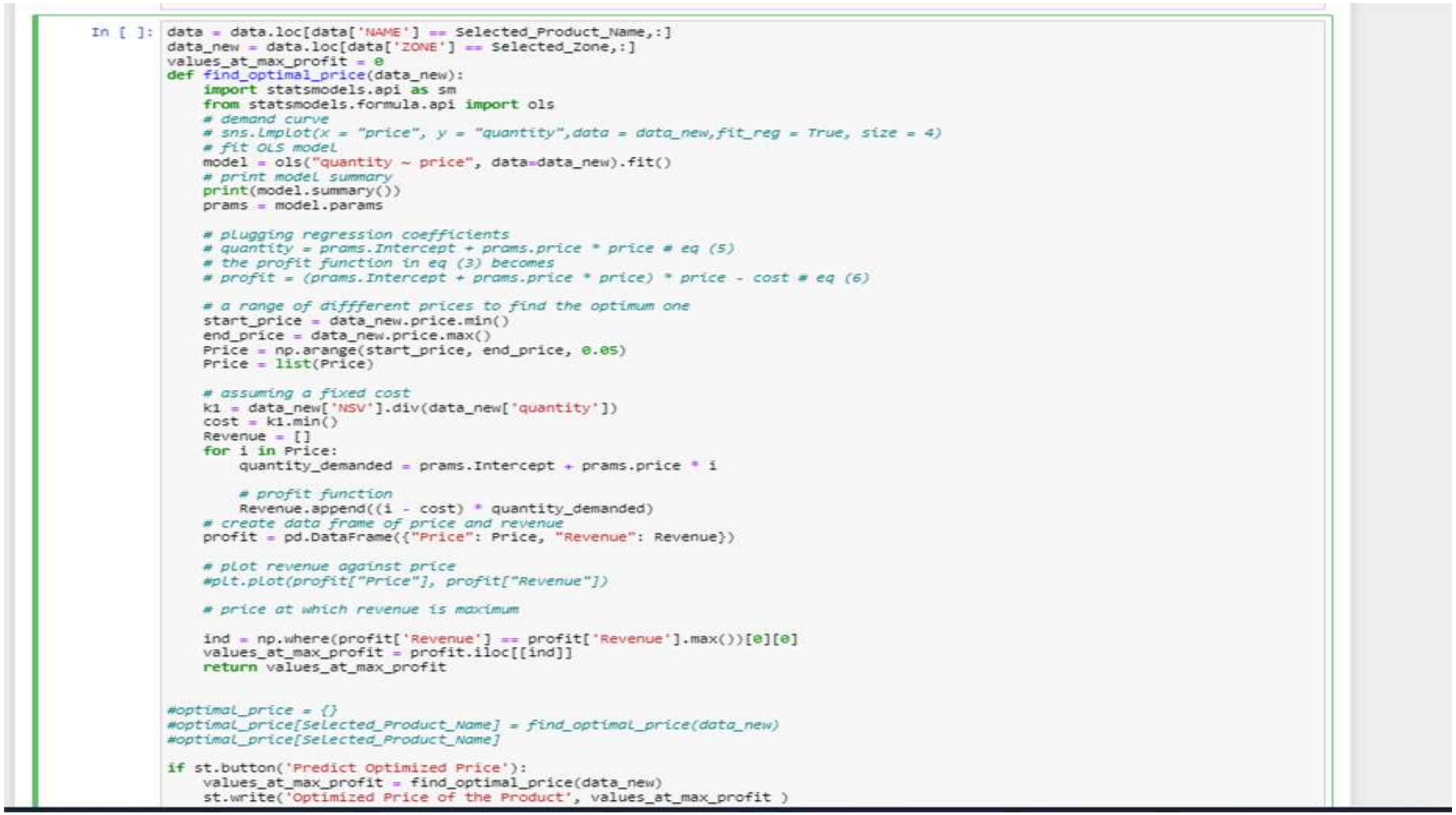


```
In [ ]: # checking the Duplicated values present in the datasets
        df1[df1.duplicated()]
        data = df1.drop_duplicates()

        # Checking The null values present in th datasets
        data.isnull().sum()
        data = data.dropna()
        data.shape

        data = data.loc[data['MARGIN%'] > 0,:]
        data.shape
```

```
In [ ]: import pickle
```

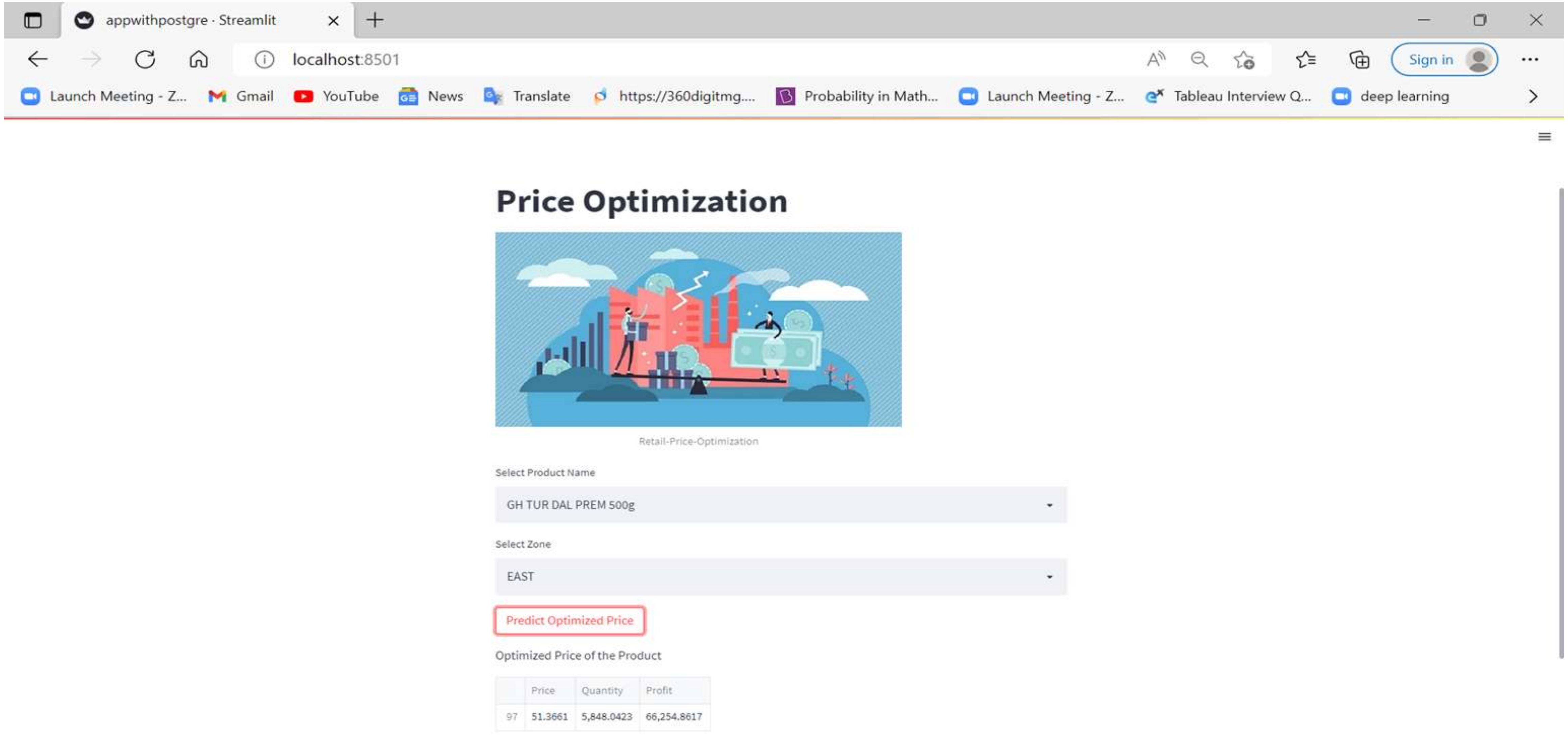
```
In [ ]: st.title('Price Optimization')

        Unique_Products =pickle.load(open('Unique_Products.pkl','rb'))
        Zone = pickle.load(open('Zone.pkl','rb'))
        model = pickle.load(open('model.pkl','rb'))


        Selected_Product_Name = st.selectbox(
            'Select Product Name',
             (Unique_Products.values))

        Selected_Zone = st.selectbox(
            'Select Zone',
             (Zone.values))
```

Model Deployment



```
In [ ]: data = data.loc[data['NAME'] == Selected_Product_Name,:]
        data_new = data.loc[data['ZONE'] == Selected_Zone,:]
        values_at_max_profit = 0
        def find_optimal_price(data_new):
            import statsmodels.api as sm
            from statsmodels.formula.api import ols
            # demand curve
            # sns.lmplot(x = "price", y = "quantity",data = data_new,fit_reg = True, size = 4)
            # fit OLS model
            model = ols("quantity ~ price", data=data_new).fit()
            # print model summary
            print(model.summary())
            prams = model.params

            # plugging regression coefficients
            # quantity = prams.Intercept + prams.price * price # eq (5)
            # the profit function in eq (3) becomes
            # profit = (prams.Intercept + prams.price * price) * price - cost # eq (6)

            # a range of diffferent prices to find the optimum one
            start_price = data_new.price.min()
            end_price = data_new.price.max()
            Price = np.arange(start_price, end_price, 0.05)
            Price = list(Price)

            # assuming a fixed cost
            k1 = data_new['NSV'].div(data_new['quantity'])
            cost = k1.min()
            Revenue = []
            for i in Price:
                quantity_demanded = prams.Intercept + prams.price * i

                # profit function
                Revenue.append((i - cost) * quantity_demanded)
            # create data frame of price and revenue
            profit = pd.DataFrame({"Price": Price, "Revenue": Revenue})

            # plot revenue against price
            #plt.plot(profit["Price"], profit["Revenue"])

            # price at which revenue is maximum

            ind = np.where(profit['Revenue'] == profit['Revenue'].max())[0][0]
            values_at_max_profit = profit.iloc[[ind]]
            return values_at_max_profit


        #optimal_price = {}
        #optimal_price[Selected_Product_Name] = find_optimal_price(data_new)
        #optimal_price[Selected_Product_Name]

        if st.button('Predict Optimized Price'):
            values_at_max_profit = find_optimal_price(data_new)
            st.write('Optimized Price of the Product', values_at_max_profit )
```

Model Deployment Output



Price Optimization

Retail-Price-Optimization

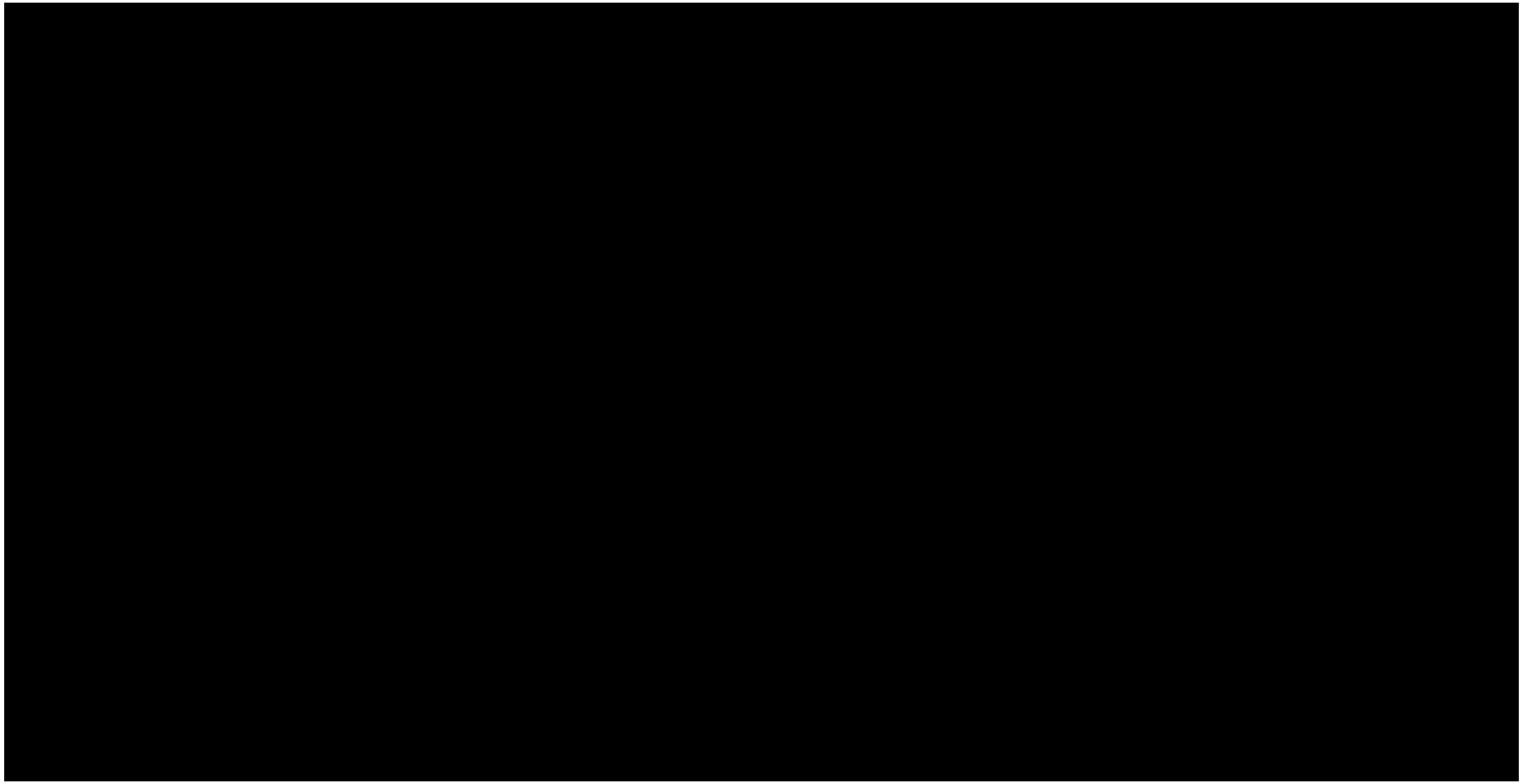
Select Product Name: GH TUR DAL PREM 500g

Select Zone: EAST

Optimized Price of the Product

	Price	Quantity	Profit
97	51.3661	5,848.0423	66,254.8617

Model Deployment Output



Challenges

- Less data for different products
- Dealing with missing values and outlier treatment
- Research on various price optimization techniques

Future Scope

- Customer Behavior Analytics :

Obtain customer data and categorise customers into regular, semi-regular and infrequent. Use analytics and machine learning models to understanding purchase behaviors of this users and adjust price and discounts accordingly.

- Dynamic Pricing :

Pricing can further be optimized by regularly adjusting the selling price (even on daily basis) by inputting the variables.

Queries ?

Thank you!