Question?

Can you estiamate price by its commodity, category or province?

Data Acquisition

https://www.kaggle.com/datasets/amaanfaheem/pakistan-food-prices-2022

About Dataset

This dataset contains Food Prices data for Pakistan, sourced from the World Food Programmed Price Database. The World Food Programmed Price Database covers foods such as maize, rice, beans, fish, and sugar for 98 countries and some 3000 markets. It is updated weekly but contains to a large extent monthly data. The data goes back as far.

Let us import the necessary liabraries and read our DataSet

In [820…
```
# just get rid of errors
import warnings
warnings.simplefilter("ignore")
```

In [821…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn import linear_model, svm
```

Extract, Transform, Load and Data wrangling

Extraction

In [822…
```
df = pd.read_csv("Pakistan_Food_Prices.csv") # reading from CSV
```

In [823…
```
df
```

Out[823…

	date	Provinces name	City Name	City market	latitude	longitude	category
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers
...	...	...	...	...	...	...	...
9718	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9719	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9720	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9721	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9722	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts

9723 rows × 14 columns

In [824…

```
df.columns # some names are capital letters
```

Out[824…

```
Index(['date', 'Provinces name', 'City Name', 'City market', 'latitude',
       'longitude', 'category', 'commodity', 'unit', 'price flag',
       'price type', 'currency', 'price', 'usd price'],
      dtype='object')
```

In [825…

```
df.head(5) # printing first 5 rows
```

Out[825…

	date	Provinces name	City Name	City market	latitude	longitude	category	c
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers	
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers	
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers	
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers	
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers	

In [826…

```
df.tail(5) # printing last 5 rows
```

Out[826…

	date	Provinces name	City Name	City market	latitude	longitude	category	commodi
9718	9/15/2022	SINDH	Karachi	Karachi	24.9056	67.0822	oil and fats	Gh (artifici
9719	9/15/2022	SINDH	Karachi	Karachi	24.9056	67.0822	oil and fats	Oil (cookin
9720	9/15/2022	SINDH	Karachi	Karachi	24.9056	67.0822	pulses and nuts	Beans(mas
9721	9/15/2022	SINDH	Karachi	Karachi	24.9056	67.0822	pulses and nuts	Lent (masu
9722	9/15/2022	SINDH	Karachi	Karachi	24.9056	67.0822	pulses and nuts	Lent (moon

In [827…

```
df.sample(5) # printing random samples
```

Out[827…

	date	Provinces name	City Name	City market	latitude	longitude	cat
8578	7/15/2021	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	miscella
1150	1/15/2011	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cerea
6267	12/15/2018	PUNJAB	Multan	Multan	30.195556	71.475278	nor
6096	10/15/2018	PUNJAB	Lahore	Lahore	31.549722	74.343611	miscella
1174	2/15/2011	PUNJAB	Multan	Multan	30.195556	71.475278	cerea

Data wrangling

In [828…

```
df.shape # data set shape
```

Out[828…

```
(9723, 14)
```

In [829…

```
df.info() # information about data set columns, null and non-null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9723 entries, 0 to 9722
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            9723 non-null   object 
 1   Provinces name  9723 non-null   object 
 2   City Name       9723 non-null   object 
 3   City market     9723 non-null   object 
 4   latitude        9723 non-null   float64
 5   longitude       9723 non-null   float64
 6   category        9723 non-null   object 
 7   commodity       9723 non-null   object 
 8   unit            9723 non-null   object 
 9   price flag      9723 non-null   object 
 10  price type      9723 non-null   object 
 11  currency        9723 non-null   object 
 12  price           9723 non-null   float64
 13  usd price       9723 non-null   float64
dtypes: float64(4), object(10)
memory usage: 1.0+ MB
```

In [830…

```
df.describe()
```

Out[830…

	latitude	longitude	price	usd price
count	9723.000000	9723.000000	9723.000000	9723.000000
mean	30.165392	70.333449	142.025046	0.650650
std	3.000443	2.843966	164.049134	0.751547
min	24.905600	67.012500	9.000000	0.041200
25%	30.187222	67.082200	49.000000	0.224500
50%	30.195556	71.475278	94.790000	0.434300
75%	31.549722	71.580182	176.600000	0.809000
max	34.008366	74.343611	1343.000000	6.152600

In [831…

```
df.columns
```

Out[831…

```
Index(['date', 'Provinces name', 'City Name', 'City market', 'latitude',
       'longitude', 'category', 'commodity', 'unit', 'price flag',
       'price type', 'currency', 'price', 'usd price'],
      dtype='object')
```

In [832…

```
df["Provinces name"].describe()
```

Out[832…

```
count       9723
unique         4
top       PUNJAB
freq        3923
Name: Provinces name, dtype: object
```

In [833…

```
df["commodity"].describe()
```

Out[833…

```
count                         9723
unique                          17
top       Rice (basmati, broken)
freq                          1125
Name: commodity, dtype: object
```

In [834…

```
df["City market"].describe()
```

Out[834…

```
count        9723
unique          5
top       Karachi
freq         1976
Name: City market, dtype: object
```

In [835…

```
df.isnull().sum() # no null values
```

Out[835…

```
date              0
Provinces name    0
City Name         0
City market       0
latitude          0
longitude         0
category          0
commodity         0
unit              0
price flag        0
price type        0
currency          0
price             0
usd price         0
dtype: int64
```

In [836…

```
df["City market"].value_counts()
```

Out[836…

```
City market
Karachi     1976
Lahore      1970
Peshawar    1969
Multan      1953
Quetta      1855
Name: count, dtype: int64
```

In [837…

```
df["commodity"].value_counts()
```

Out[837…

```
commodity
Rice (basmati, broken)                                 1125
Wheat flour                                            1023
Wheat                                                   902
Poultry                                                 567
Sugar                                                   567
Oil (cooking)                                           567
Ghee (artificial)                                       565
Fuel (diesel)                                           525
Beans(mash)                                             500
Wage (non-qualified labour, non-agricultural)           488
Lentils (masur)                                         488
Eggs                                                    473
Fuel (petrol-gasoline)                                  470
Lentils (moong)                                         450
Milk                                                    440
Rice (coarse)                                           368
Salt                                                    205
Name: count, dtype: int64
```

In [838…

```
df.duplicated().sum() # no duplicated values
```

Out[838…

```
0
```

In [839…

```
df.drop(['usd price','price flag','price type','currency'],axis=1,inplace
# if you run more than one time it will give you error
```

In [840…

```
df
```

Out[840…

	date	Provinces name	City Name	City market	latitude	longitude	category
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers
...	...	...	...	...	...	...	...
9718	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9719	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9720	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9721	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9722	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts

9723 rows × 10 columns

In [841…

```
df['year'] = pd.DatetimeIndex(df['date']).year
```

In [842…

```
df
```

Out[842…

	date	Provinces name	City Name	City market	latitude	longitude	category
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers
...	...	...	...	...	...	...	...
9718	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9719	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9720	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9721	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9722	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts

9723 rows × 11 columns

In [843…

```
time = df["date"].str.split("/", expand=True)
df[["month", "day", "year"]] = time.astype(int)
```

In [844…

```
cols = df[['Provinces name', 'City Name', 'City market', 'category', 'com

for i in cols.columns:
    print("\n", df[i].unique())
```

```
['BALOCHISTAN' 'KHYBER PAKHTUNKHWA' 'PUNJAB' 'SINDH']

['Quetta' 'Peshawar' 'Lahore' 'Multan' 'Karachi']

['Quetta' 'Peshawar' 'Lahore' 'Multan' 'Karachi']

['cereals and tubers' 'meat, fish and eggs' 'miscellaneous food'
'oil and fats' 'non-food' 'pulses and nuts' 'milk and dairy']

['Rice (basmati, broken)' 'Wheat flour' 'Wheat' 'Rice (coarse)' 'Poultry'
'Sugar' 'Ghee (artificial)' 'Oil (cooking)' 'Eggs'
'Wage (non-qualified labour, non-agricultural)' 'Lentils (masur)'
'Fuel (diesel)' 'Beans(mash)' 'Milk' 'Salt' 'Fuel (petrol-gasoline)'
'Lentils (moong)']
```

Data visualization

In [845…

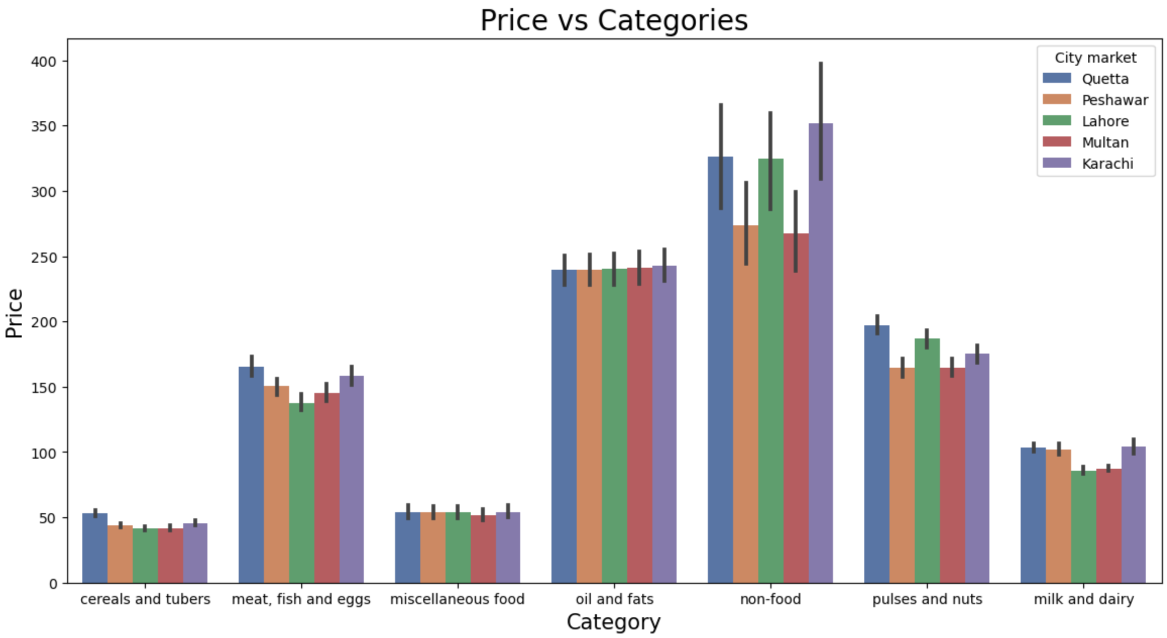
```
map = px.scatter_mapbox(df, lat="latitude", lon="longitude",size='price',
```

In [846…

```
map.update_layout(mapbox_style="open-street-map")
```

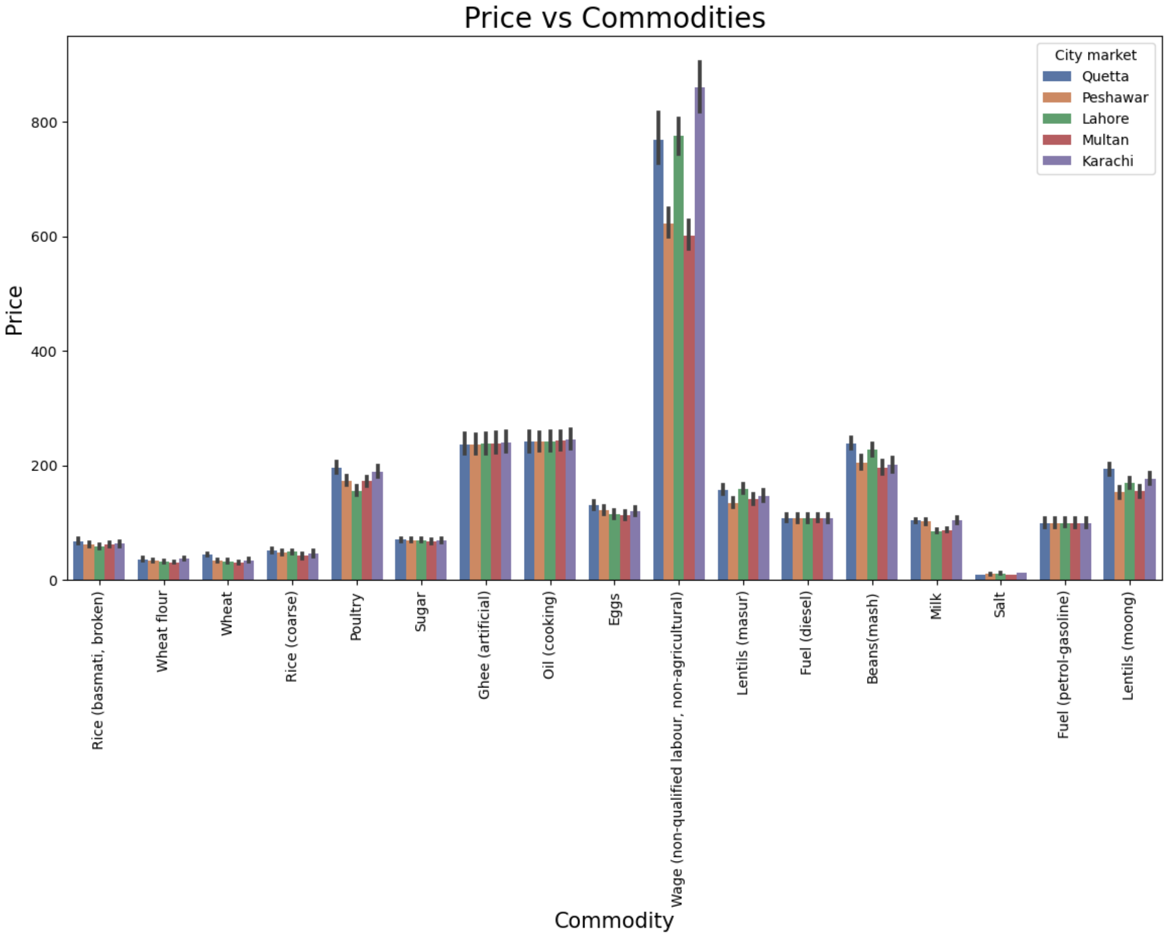
In [847…

```
fig = plt.subplots(figsize=(14, 7))
sns.barplot(x = df["category"], y = df["price"], hue = df["City market"],
plt.xticks(rotation=0);
plt.xlabel("Category", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price vs Categories", fontsize=20)
plt.savefig('sample_plot.png')
```



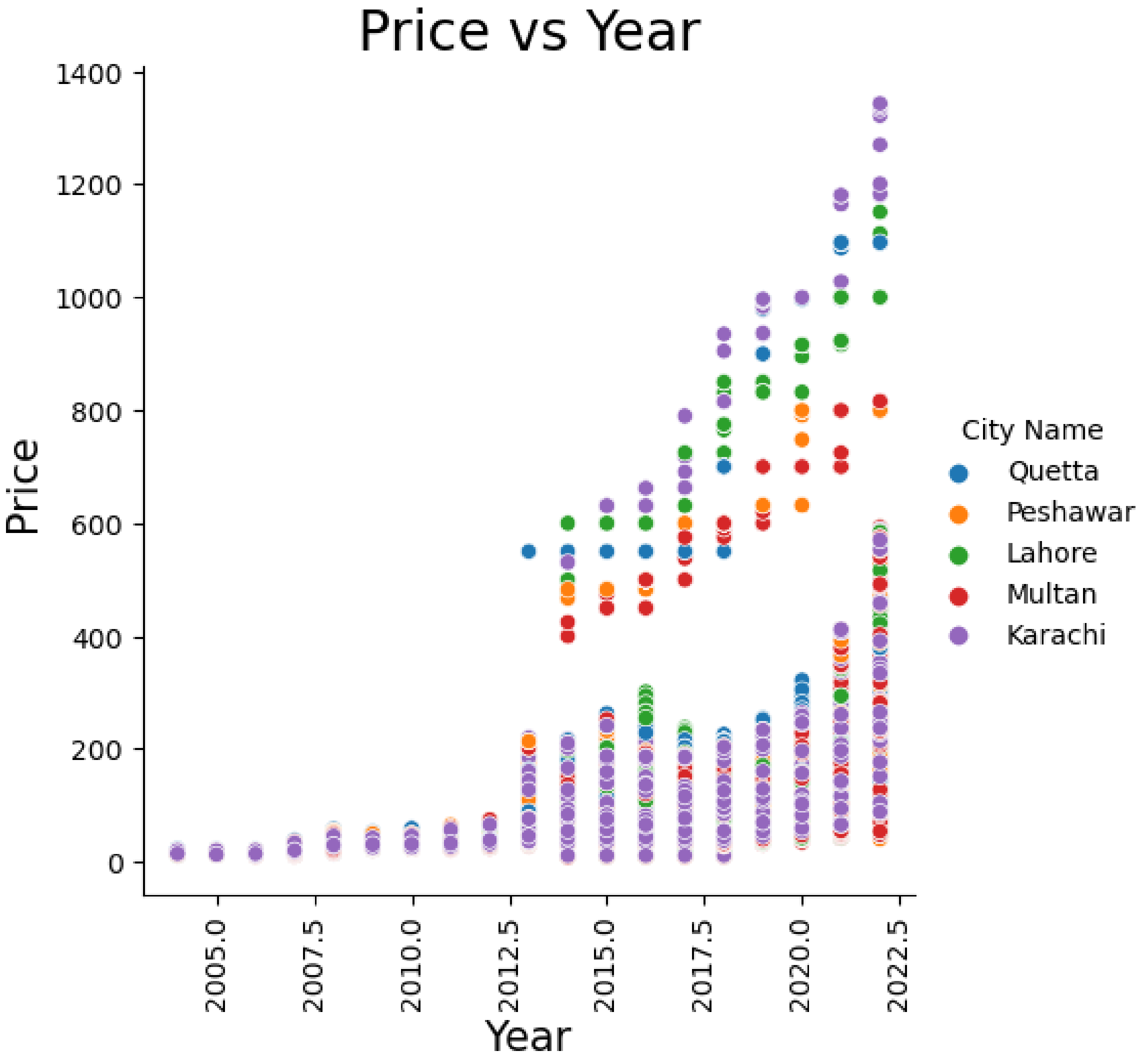
In [848…

```
fig = plt.subplots(figsize=(14, 7))
sns.barplot(x=df["commodity"], y=df["price"], hue=df["City market"],palet
plt.xticks(rotation=90);
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price vs Commodities", fontsize=20)
plt.savefig('Price vs Commodities.png')
```



In [849…

```
df["year"].astype("int")
sns.relplot(x=df["year"], y=df["price"], hue=df["City Name"])
plt.xticks(rotation=90);
plt.xlabel("Year", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price vs Year", fontsize=20)
plt.savefig('Price vs Year.png')
```

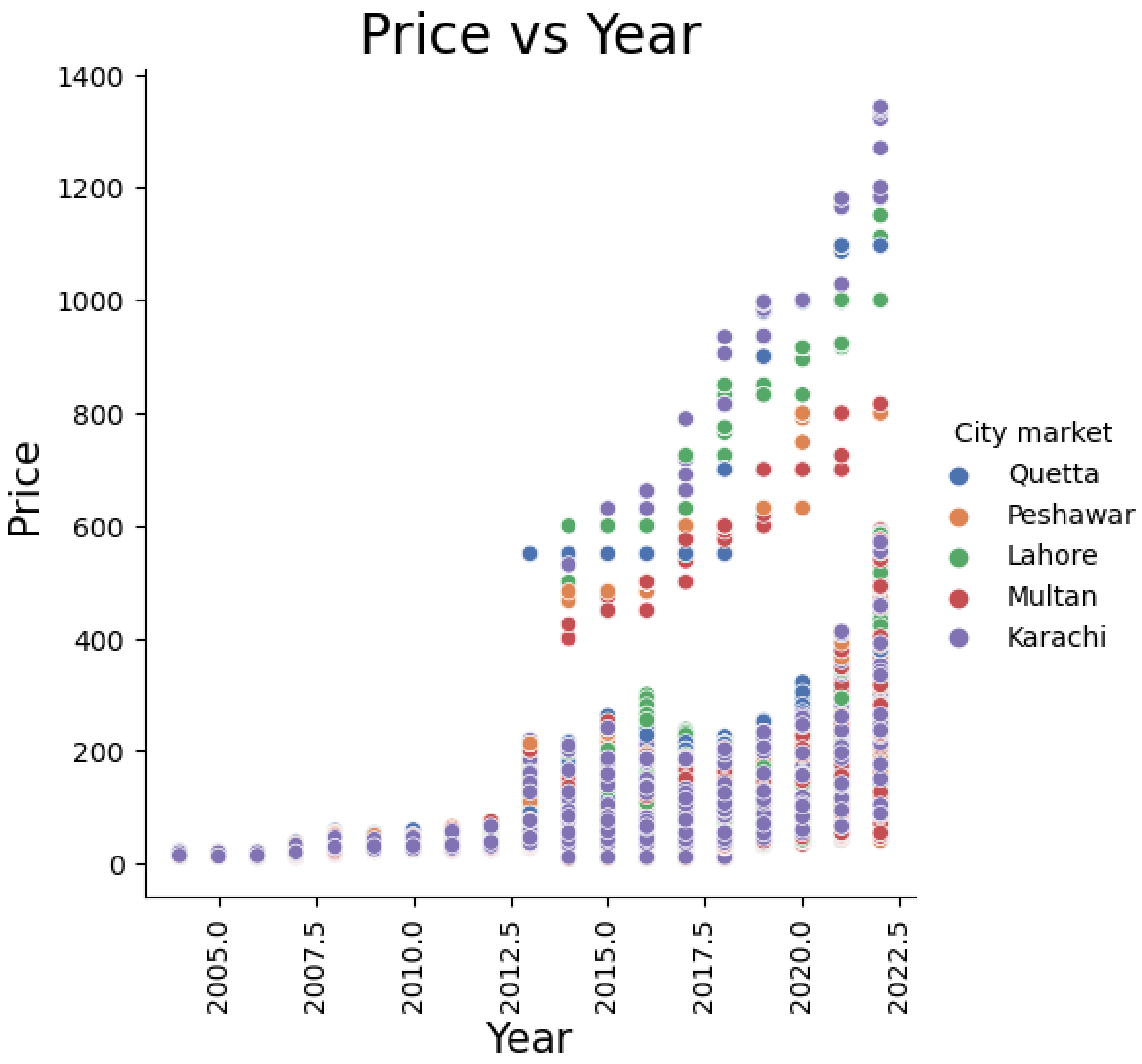


In [850…

```
# Create 3D scatter plot
fig = px.scatter_3d(df, x="commodity", y="City market", z="price",
    #labels={"lon": "longitude", "lat": "latitude", "price": "price"},
    width=1000,
    height=1000,
)

# Refine formatting
fig.update_traces(
    marker={"size": 4, "line": {"width": 3, "color": "DarkSlateGrey"}},
    selector={"mode": "markers"},
)

# Display figure
fig.show()
```

In [851…

```
#fig = plt.subplots(figsize=(16, 9))
sns.relplot(x=df["year"], y=df["price"], hue=df["City market"],palette="d
plt.xticks(rotation=90);
plt.xlabel("Year", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price vs Year", fontsize=20)
plt.savefig('Price vs Year.png')
plt.show()
```

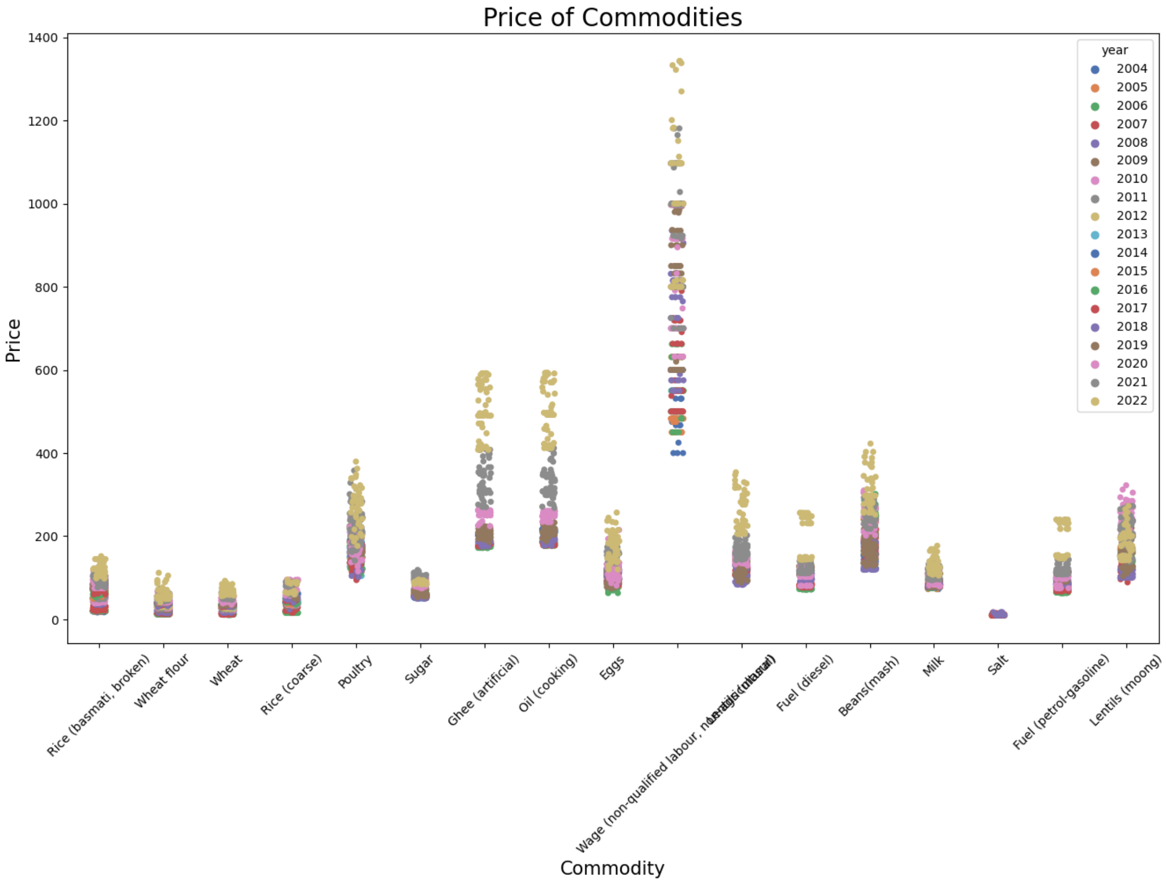


In [ ]:

```

```

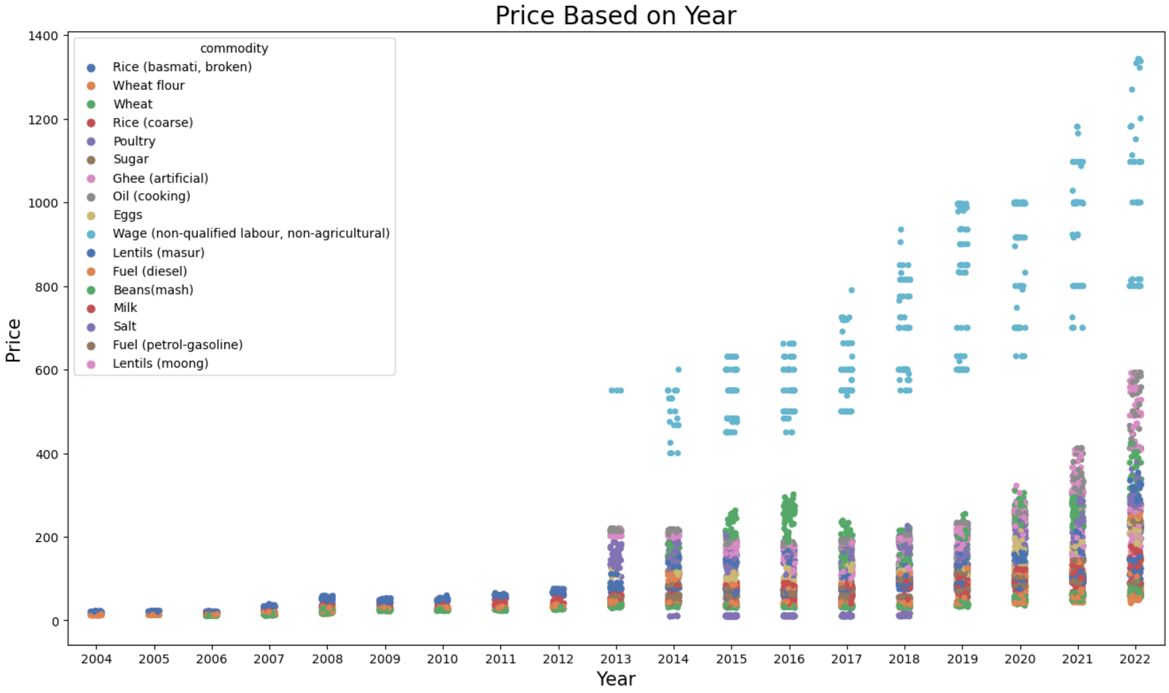
In [852…

```
fig = plt.subplots(figsize=(16, 9))
sns.stripplot(x=df["commodity"], y=df["price"], hue=df["year"],palette="d
plt.xticks(rotation=45);
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Commodities", fontsize=20)
plt.savefig('Price vs Commodities.png')
plt.show()
```



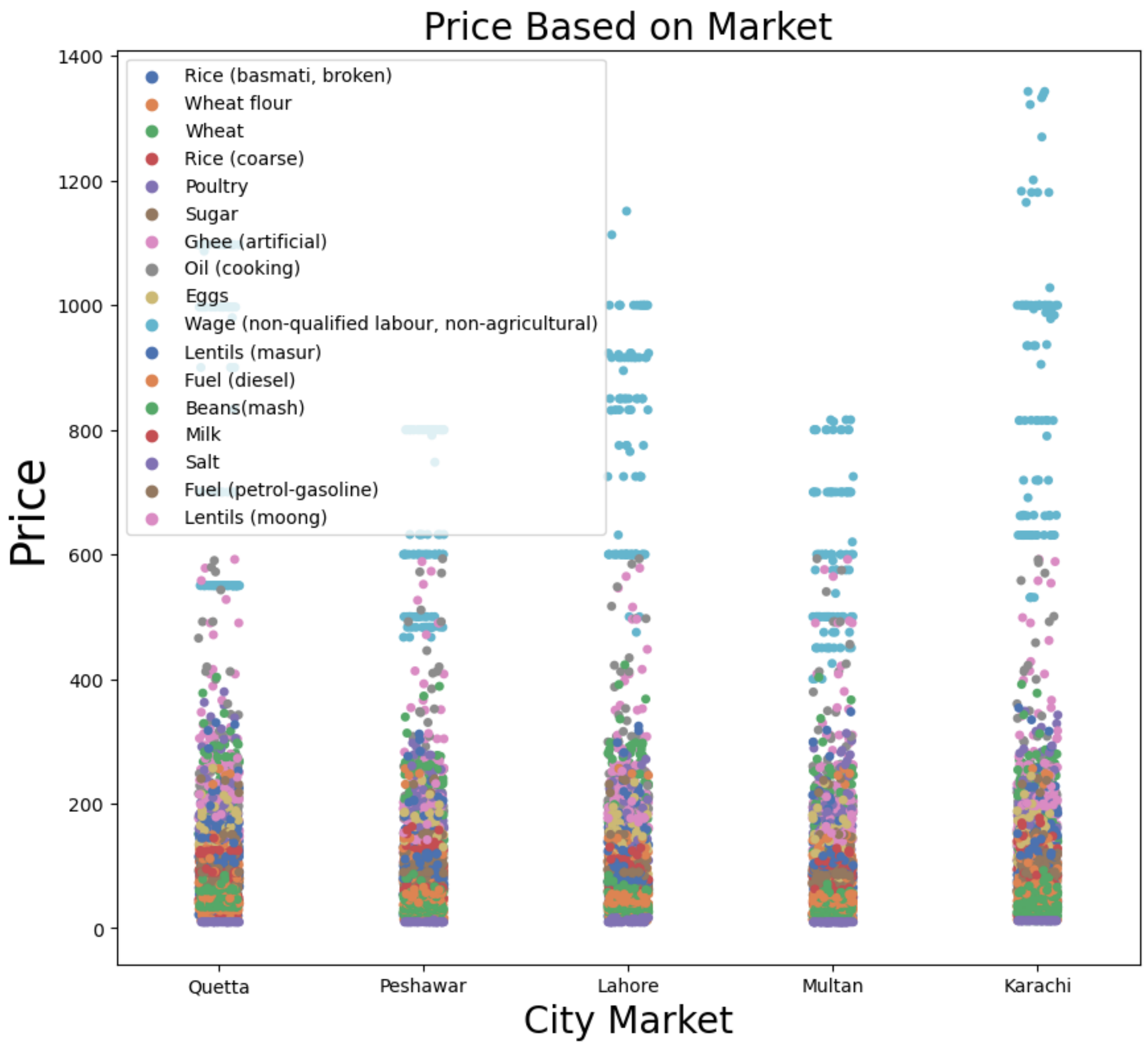
In [853…

```
fig = plt.subplots(figsize=(16, 9))
sns.stripplot(x=df["year"], y=df["price"], hue=df["commodity"],palette="d
plt.xlabel("Year", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price Based on Year", fontsize=20)
plt.savefig('Price Based on Year.png')
plt.show()
```



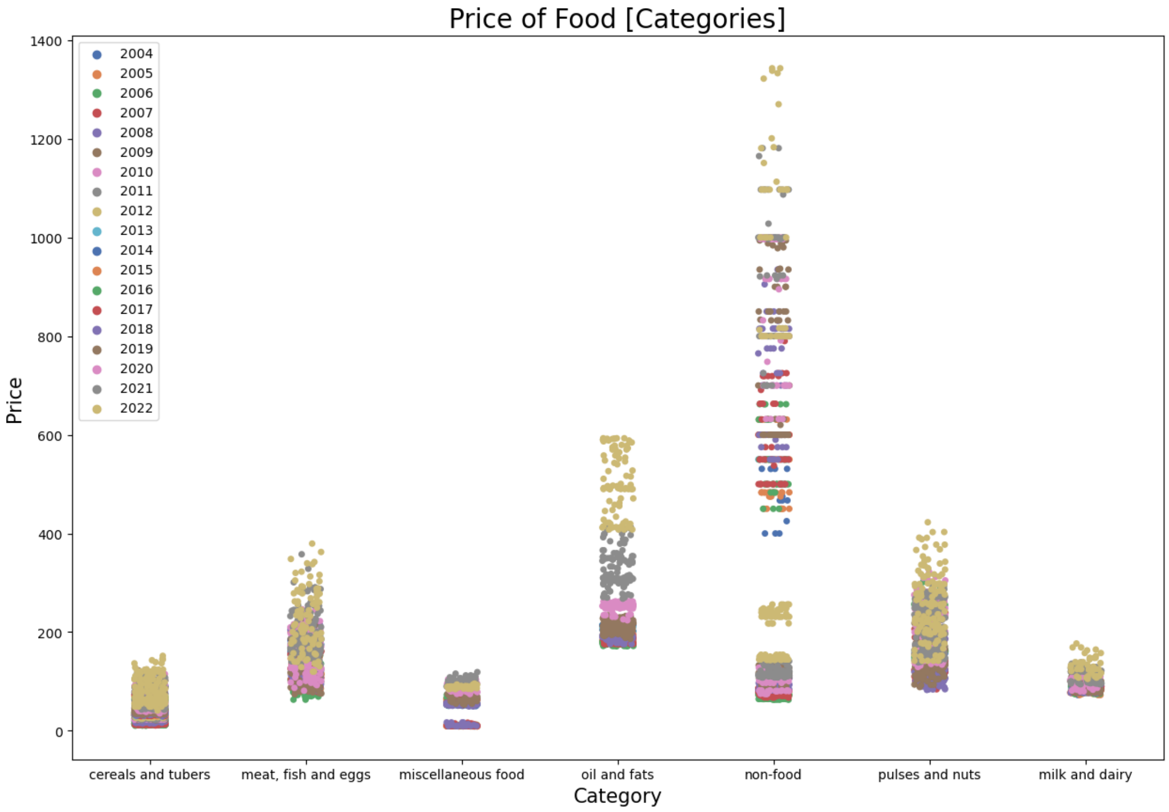
In [854…

```
fig = plt.subplots(figsize=(10, 9))
sns.stripplot(x=df["City market"], y=df["price"], hue=df["commodity"],pal
plt.xlabel("City Market", fontsize=20)
plt.ylabel("Price", fontsize=25)
plt.title("Price Based on Market", fontsize=20)
plt.legend(loc ="upper left")
plt.savefig('Price Based on Market.png')
plt.show()
```



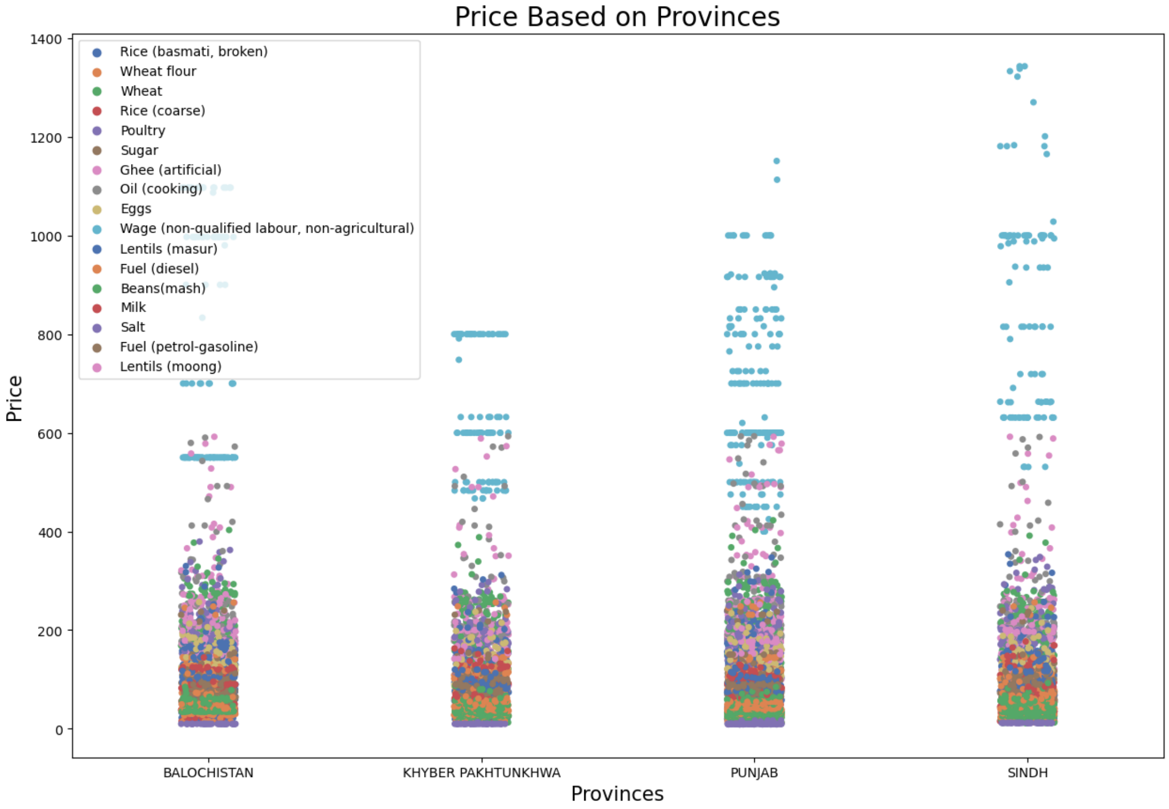
In [855…

```
fig = plt.subplots(figsize=(15, 10))
sns.stripplot(x=df["category"], y=df["price"], hue=df["year"],palette="de
plt.xlabel("Category", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Food [Categories]", fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



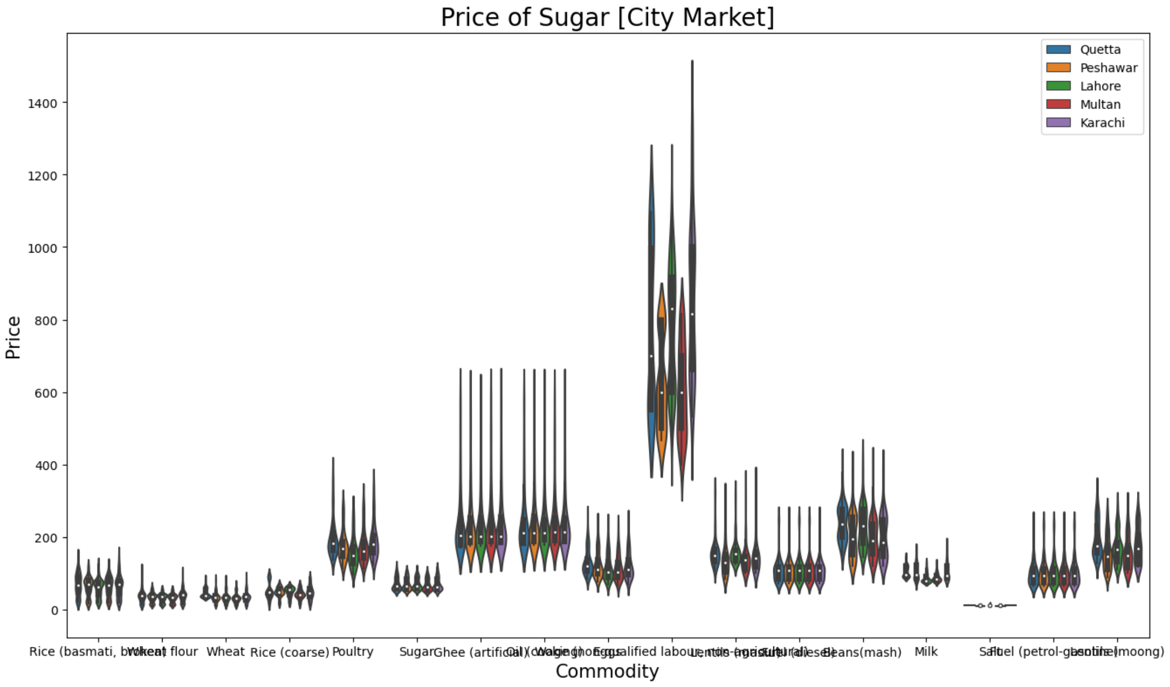
In [856…

```
fig = plt.subplots(figsize=(15, 10))
sns.stripplot(x=df["Provinces name"], y=df["price"], hue=df["commodity"],
plt.xlabel("Provinces", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price Based on Provinces", fontsize=20)
plt.legend(loc ="upper left")
plt.savefig('Price based on Provinces.png')
plt.show()
```



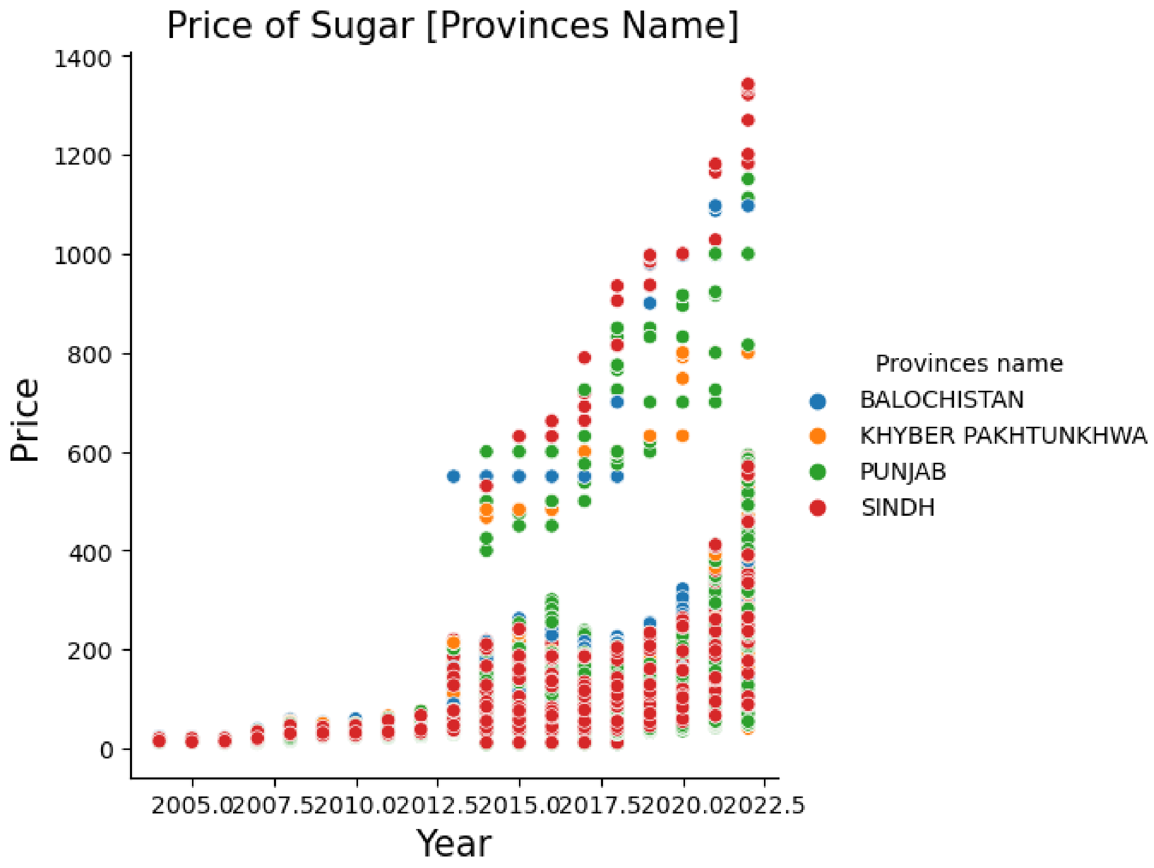
In [857…

```
fig = plt.subplots(figsize=(16, 9))

sns.violinplot(x =df["commodity"], y=df["price"], hue=df["City market"])
plt.xticks(rotation=0);
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Sugar [City Market]", fontsize=20)
plt.legend(loc ="upper right")
plt.savefig('Price of Sugar based on City Market.png')
```



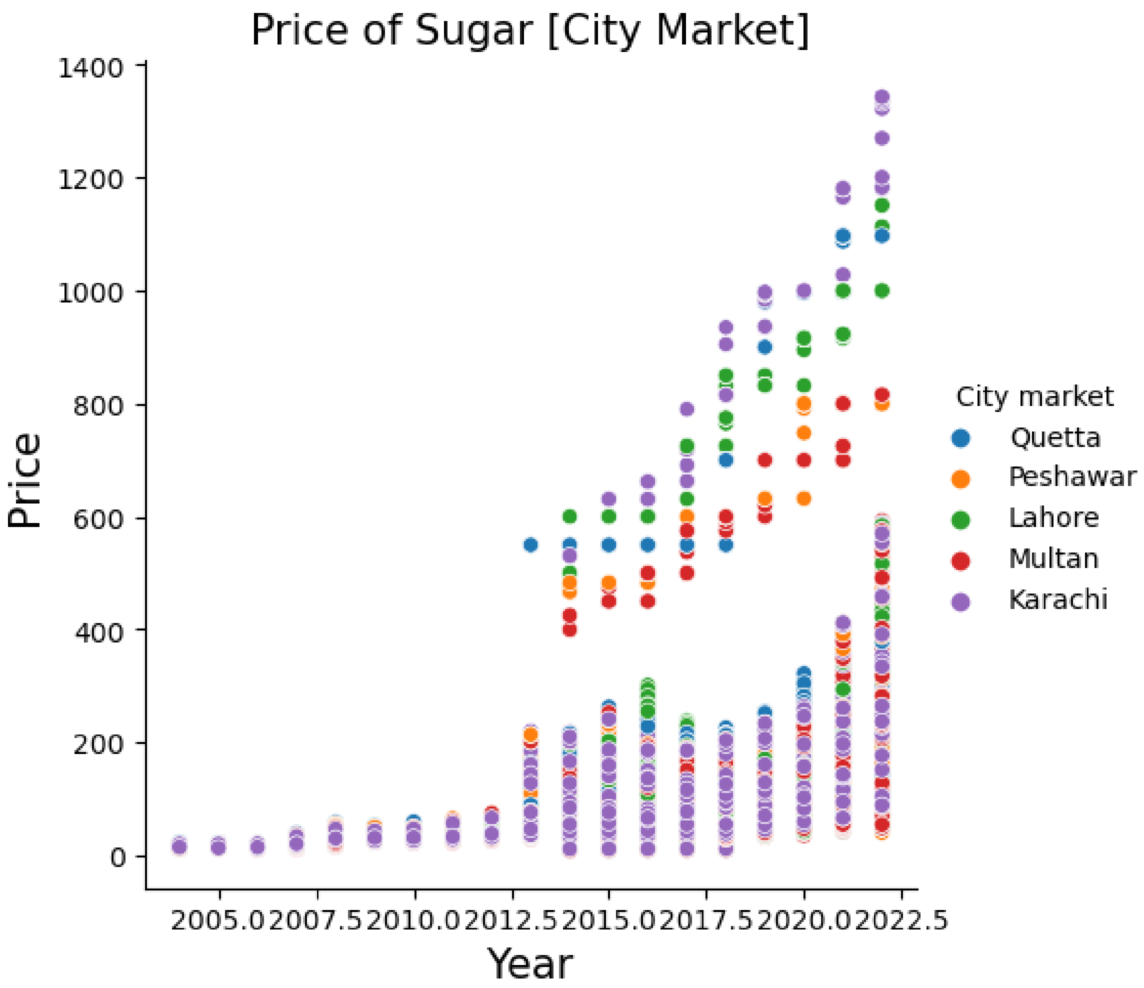
In [858…

```
sns.relplot(x = df["year"], y = df["price"], hue=df["Provinces name"])
plt.xlabel("Year", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Sugar [Provinces Name]", fontsize=15)
plt.savefig('Price of Sugar by Provinces Name.png')
plt.show()
```



In [859…

```
sns.relplot(x = df["year"], y = df["price"], hue = df["City market"])
plt.xlabel("Year", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Sugar [City Market]", fontsize=15)
plt.savefig('Price of Sugar by City Market.png')
plt.show()
```



```
In [860…  wage_food_df = df[df["commodity"].str.contains("Wage (non-qualified labou
```

```
In [861…  wage_food_df
```

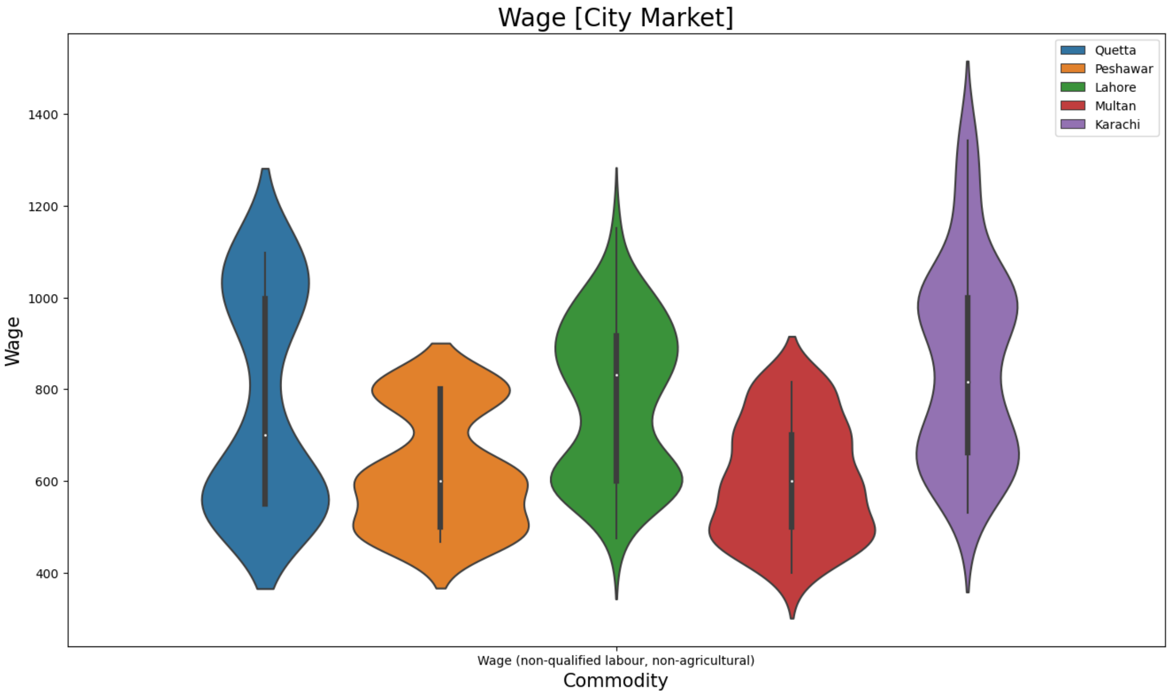
Out[861…

	date	Provinces name	City Name	City market	latitude	longitude	categor
1785	9/15/2013	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	non-foo
1821	10/15/2013	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	non-foo
1842	11/15/2013	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	non-foo
1882	1/15/2014	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	non-foo
1893	1/15/2014	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	non-foo
...	...	...	...	...	...	...	.
9656	9/15/2022	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	non-foo
9671	9/15/2022	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	non-foo
9686	9/15/2022	PUNJAB	Lahore	Lahore	31.549722	74.343611	non-foo
9701	9/15/2022	PUNJAB	Multan	Multan	30.195556	71.475278	non-foo
9717	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	non-foo

488 rows × 13 columns

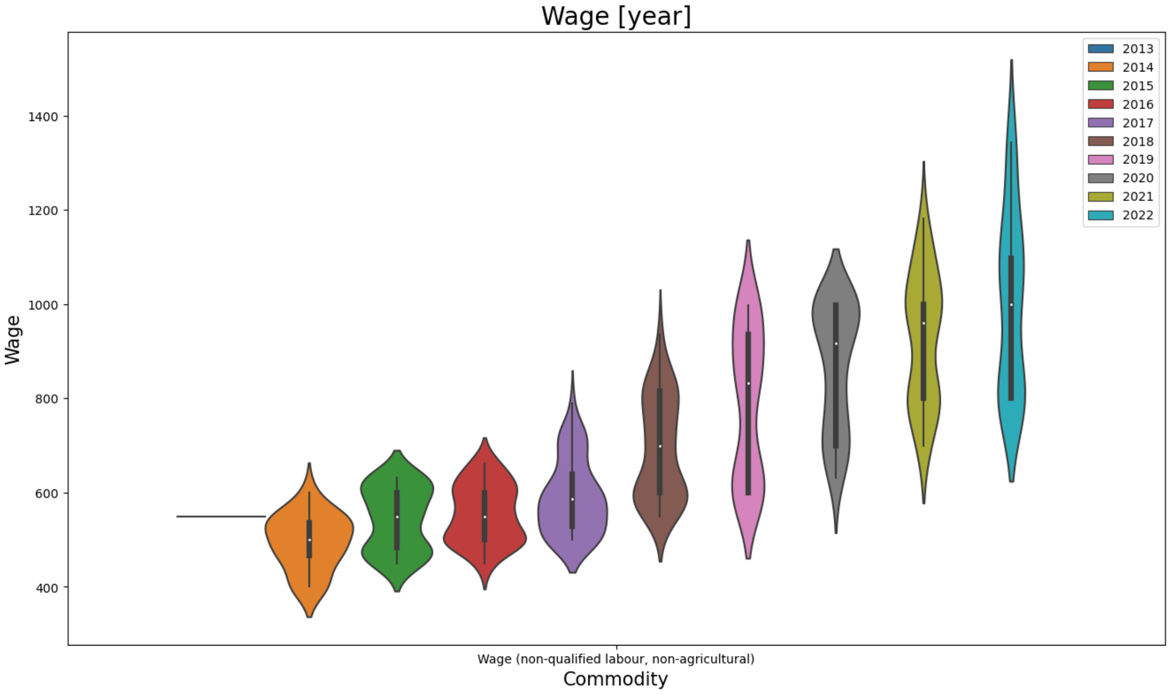
In [862…

```
fig = plt.subplots(figsize=(16, 9))
sns.violinplot(x = wage_food_df["commodity"], y = wage_food_df["price"],
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Wage", fontsize=15)
plt.title("Wage [City Market]", fontsize=20)
plt.legend(loc ="upper right")
plt.savefig('Wages by City Market.png')
```



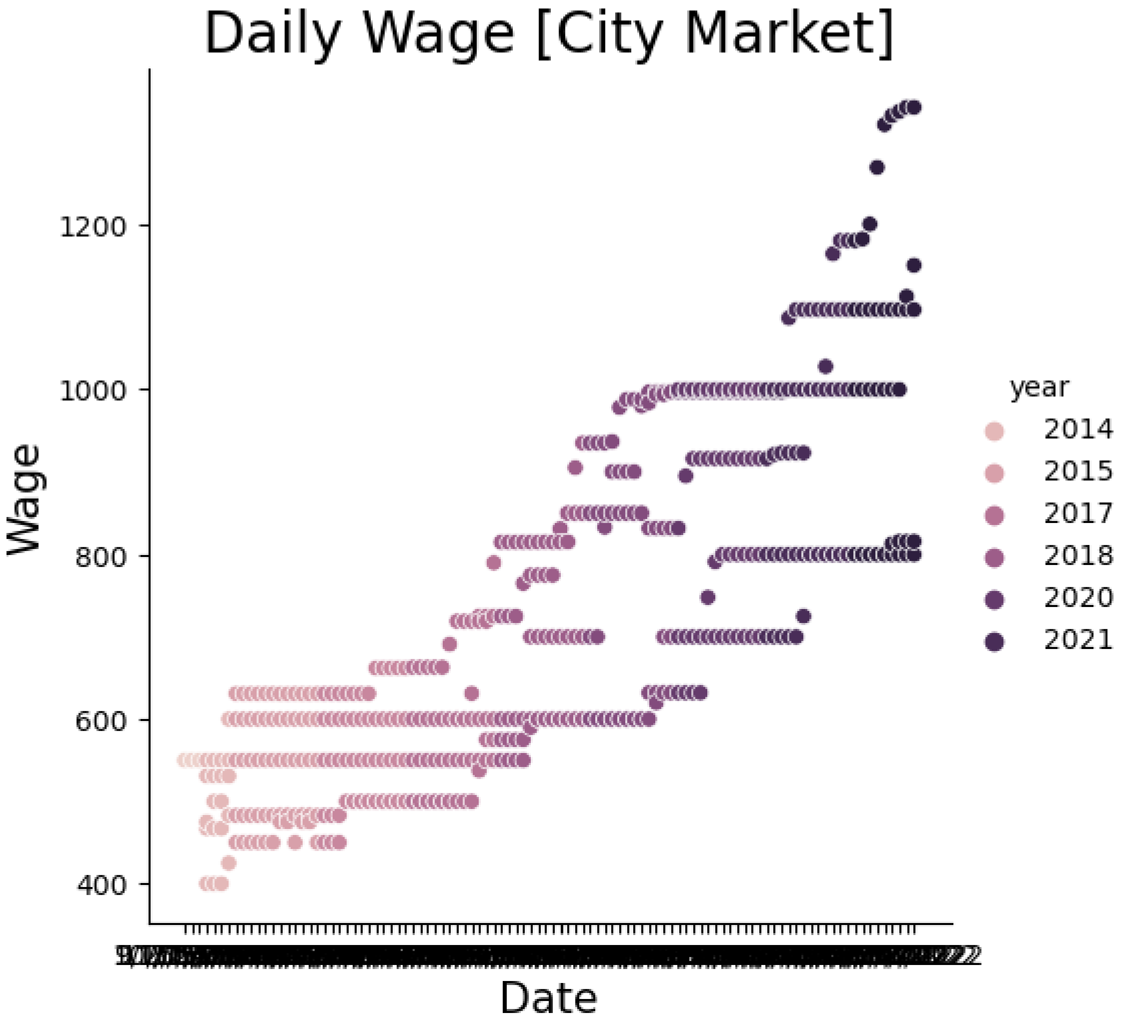
In [863…

```
fig = plt.subplots(figsize=(16, 9))
sns.violinplot(x = wage_food_df["commodity"], y = wage_food_df["price"],
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Wage", fontsize=15)
plt.title("Wage [year]", fontsize=20)
plt.legend(loc ="upper right");
plt.savefig('Wage per year.png')
```



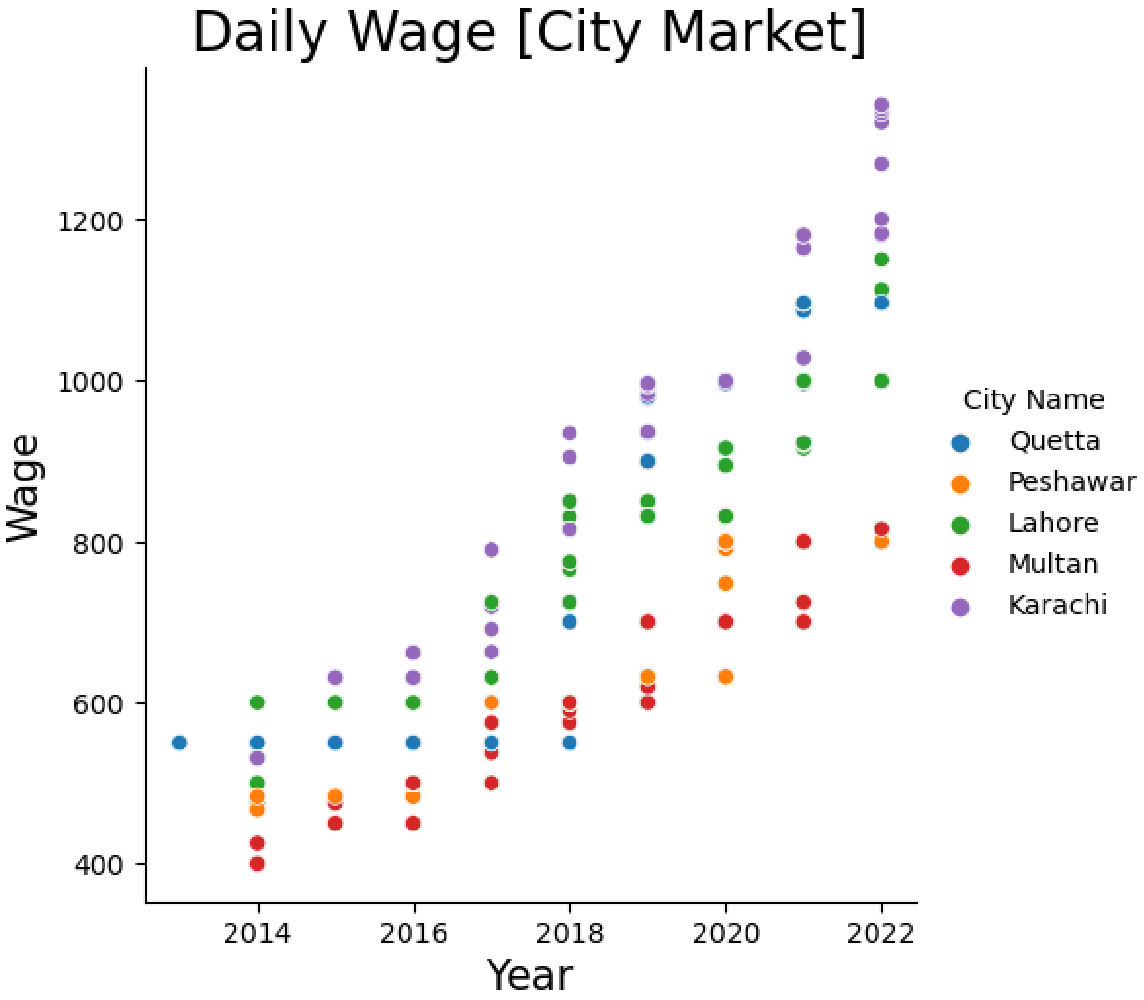
In [864…

```
sns.relplot(x =wage_food_df["date"], y=wage_food_df["price"], hue=wage_fo
plt.xlabel("Date", fontsize=15)
plt.ylabel("Wage", fontsize=15)
plt.title("Daily Wage [City Market]", fontsize=20)
plt.savefig('Dail wage by city market.png')
plt.show()
```



In [865…

```
sns.relplot(x =wage_food_df["year"], y=wage_food_df["price"], hue=wage_fo
plt.xlabel("Year", fontsize=15)
plt.ylabel("Wage", fontsize=15)
plt.title("Daily Wage [City Market]", fontsize=20)
plt.savefig('Daily Wage by City Market.png')
plt.show()
```

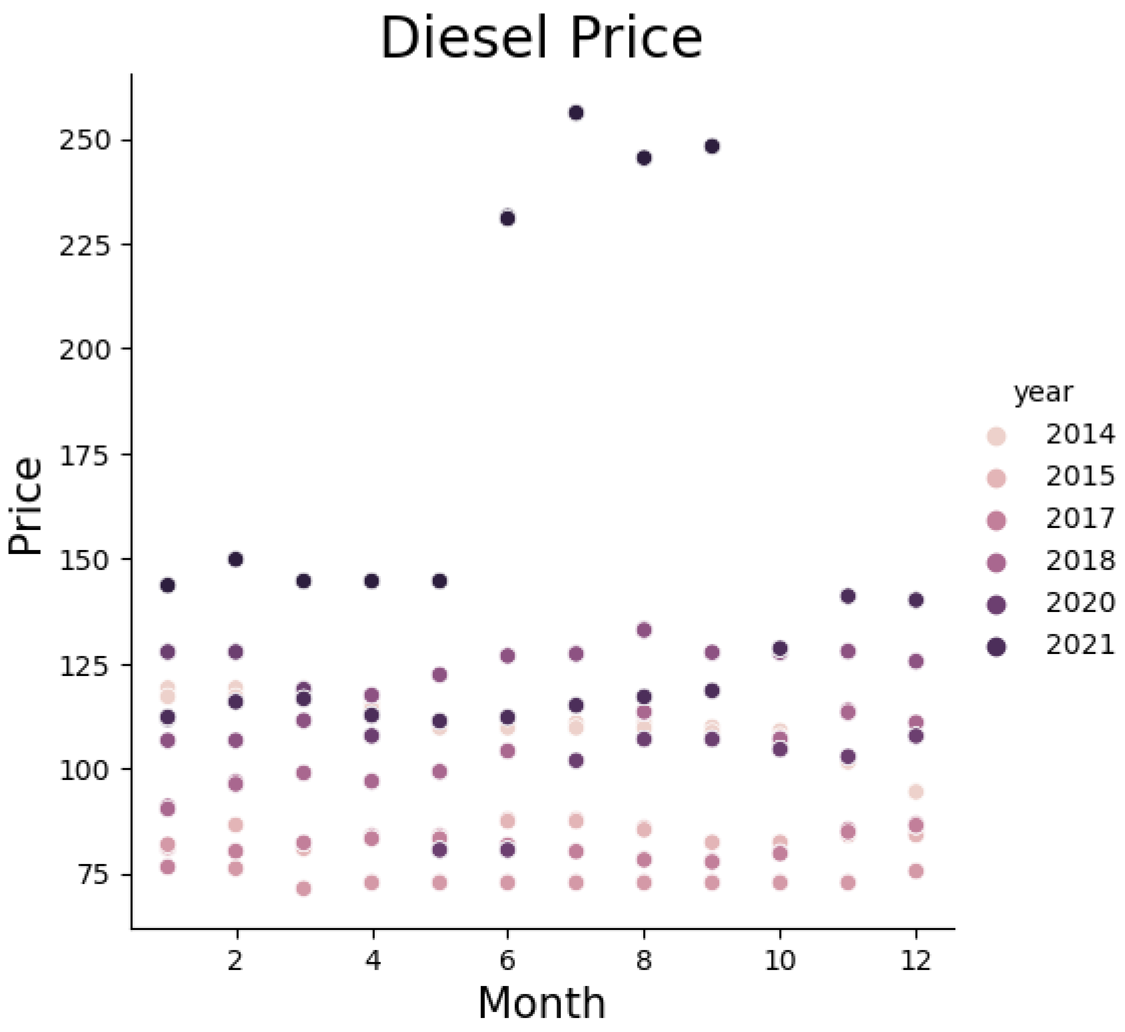


In [866…

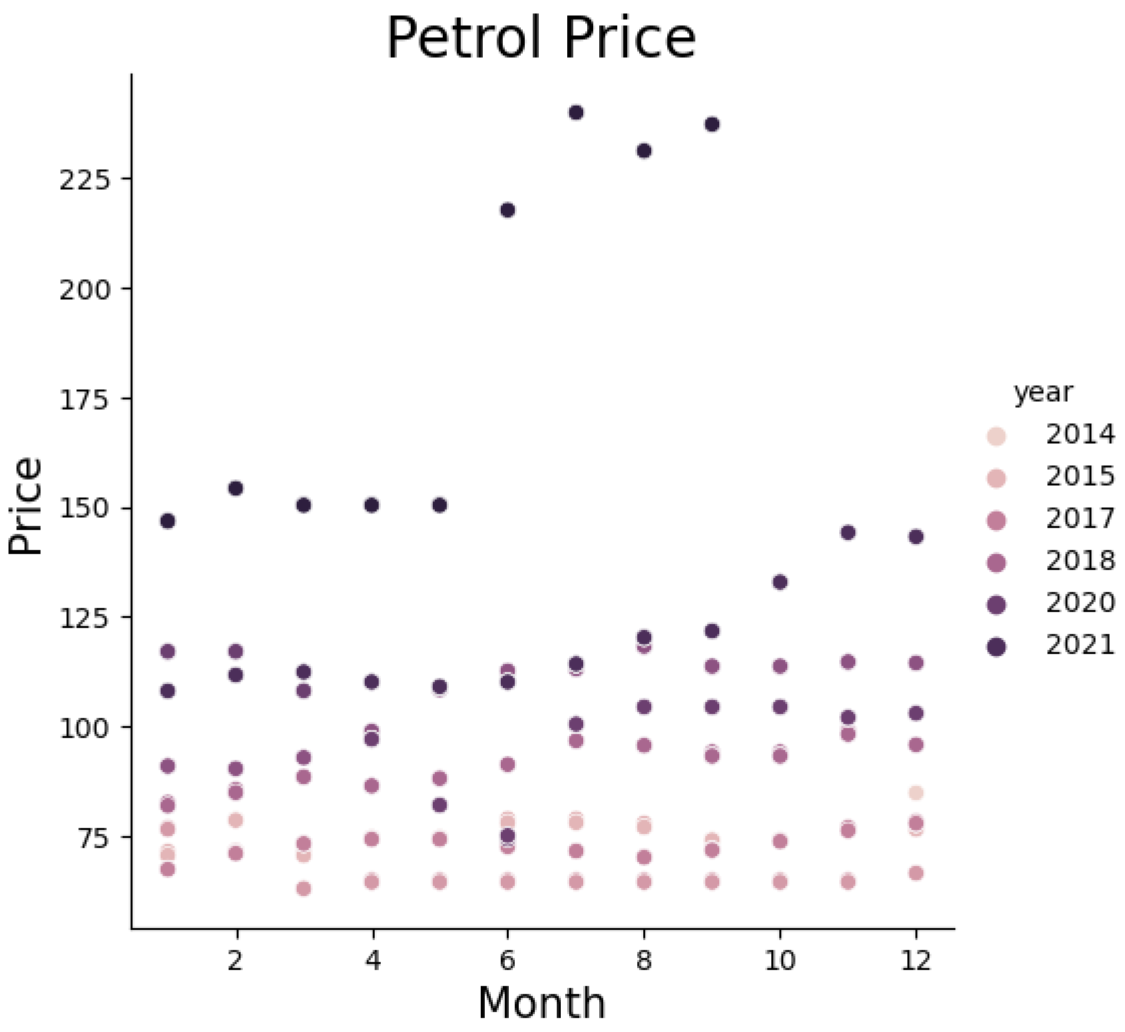
```
# df.drop(["latitude", "longitude"], axis=1, inplace=True)
diesel_food_df = df[df["commodity"].str.contains("Fuel (diesel)", regex=F
petrol_food_df = df[df["commodity"].str.contains("Fuel (petrol-gasoline)"
print(diesel_food_df["commodity"].unique())
print(petrol_food_df["commodity"].unique())
print("\n", "Diesel Shape", diesel_food_df.shape, "\n", "Gasoline", petro
print("\n", "Diesel Info", diesel_food_df.info(), "\n", "Gasoline", petro
['Fuel (diesel)']
['Fuel (petrol-gasoline)']

 Diesel Shape (525, 13)
 Gasoline (470, 13)

<class 'pandas.core.frame.DataFrame'>
Index: 525 entries, 1881 to 9715
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            525 non-null    object 
 1   Provinces name  525 non-null    object 
 2   City Name       525 non-null    object 
 3   City market     525 non-null    object 
 4   latitude        525 non-null    float64
 5   longitude       525 non-null    float64
 6   category        525 non-null    object 
 7   commodity       525 non-null    object 
 8   unit            525 non-null    object 
 9   price           525 non-null    float64
 10  year            525 non-null    int64  
 11  month           525 non-null    int64  
 12  day             525 non-null    int64  
dtypes: float64(3), int64(3), object(7)
memory usage: 57.4+ KB
<class 'pandas.core.frame.DataFrame'>
Index: 470 entries, 2419 to 9716
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            470 non-null    object 
 1   Provinces name  470 non-null    object 
 2   City Name       470 non-null    object 
 3   City market     470 non-null    object 
 4   latitude        470 non-null    float64
 5   longitude       470 non-null    float64
 6   category        470 non-null    object 
 7   commodity       470 non-null    object 
 8   unit            470 non-null    object 
 9   price           470 non-null    float64
 10  year            470 non-null    int64  
 11  month           470 non-null    int64  
 12  day             470 non-null    int64  
dtypes: float64(3), int64(3), object(7)
memory usage: 51.4+ KB

 Diesel Info None
 Gasoline None
```

In [867…

```
sns.relplot(x = diesel_food_df["month"], y = diesel_food_df["price"], hue
plt.xlabel("Month", fontsize = 15)
plt.ylabel("Price", fontsize = 15)
plt.title("Diesel Price", fontsize = 20)
plt.savefig('Diesel Price.png')
```



```
In [868…  sns.relplot(x = petrol_food_df["month"], y = petrol_food_df["price"], hue
          plt.xlabel("Month", fontsize = 15)
          plt.ylabel("Price", fontsize = 15)
          plt.title("Petrol Price", fontsize = 20)
          plt.savefig('Price of Petrol.png')
```

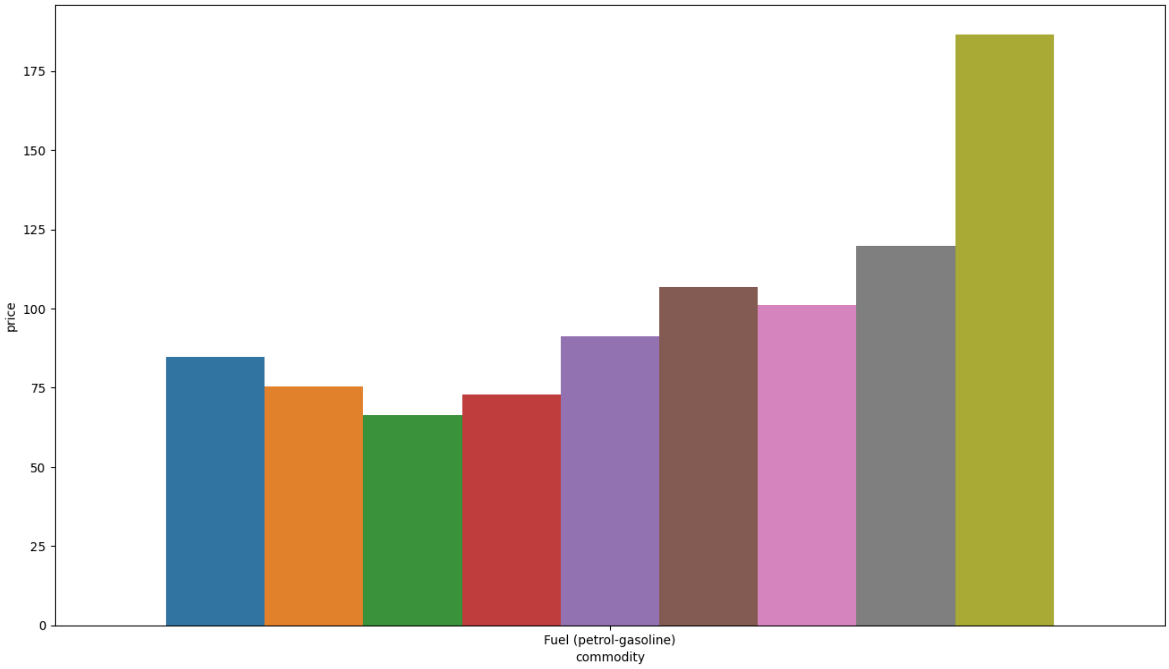


In [869…

```
fig = plt.subplots(figsize=(16, 9))
sns.barplot(x=petrol_food_df["commodity"], y = petrol_food_df["price"], h
plt.xlabel("Commodity [Petrol]", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Petrol Price", fontsize=20)
plt.legend(loc ="upper right")
plt.savefig('Petrol Price.png')
```

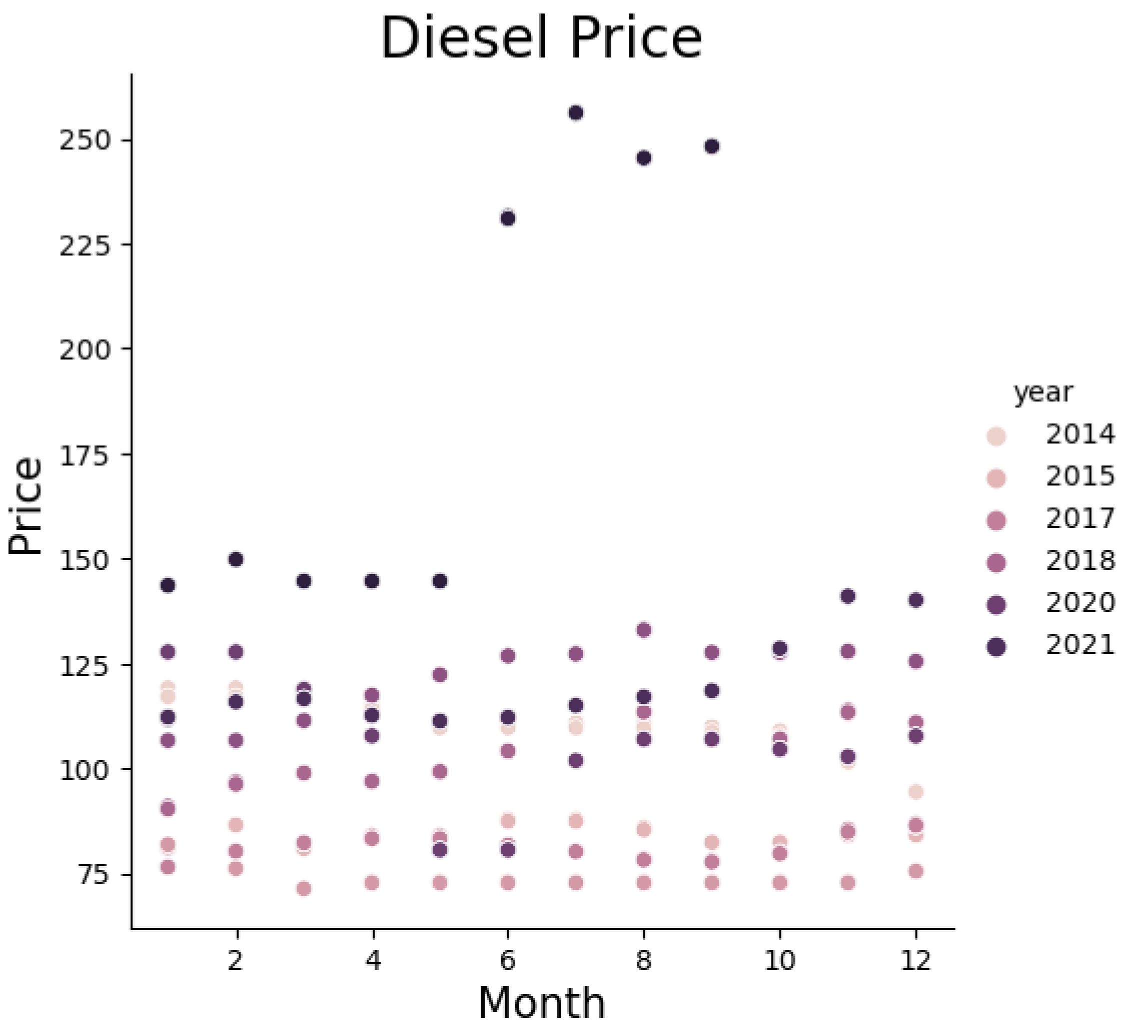
```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
Cell In[869], line 2
      1 fig = plt.subplots(figsize=(16, 9))
----> 2 sns.barplot(x=petrol_food_df["commodity"], y = petrol_food_df["pri
ce"], hue = petrol_food_df["year"], ci=0 )
      3 plt.xlabel("Commodity [Petrol]", fontsize=15)
      4 plt.ylabel("Price", fontsize=15)

File ~/anaconda3/lib/python3.11/site-packages/seaborn/categorical.py:2763,
in barplot(data, x, y, hue, order, hue_order, estimator, errorbar, n_boot,
units, seed, orient, color, palette, saturation, width, errcolor, errwidt
h, capsize, dodge, ci, ax, **kwargs)
   2760 if ax is None:
   2761     ax = plt.gca()
-> 2763 plotter.plot(ax, kwargs)
   2764 return ax

File ~/anaconda3/lib/python3.11/site-packages/seaborn/categorical.py:1587,
in _BarPlotter.plot(self, ax, bar_kws)
   1585 """Make the plot."""
   1586 self.draw_bars(ax, bar_kws)
-> 1587 self.annotate_axes(ax)
   1588 if self.orient == "h":
   1589     ax.invert_yaxis()

File ~/anaconda3/lib/python3.11/site-packages/seaborn/categorical.py:767,
in _CategoricalPlotter.annotate_axes(self, ax)
    764     ax.set_ylim(-.5, len(self.plot_data) - .5, auto=None)
    766 if self.hue_names is not None:
--> 767     ax.legend(loc="best", title=self.hue_title)

File ~/anaconda3/lib/python3.11/site-packages/matplotlib/axes/_axes.py:32
2, in Axes.legend(self, *args, **kwargs)
    204 @_docstring.dedent_interpd
    205 def legend(self, *args, **kwargs):
    206     """
    207     Place a legend on the Axes.
    208
   (...)
    320     .. plot:: gallery/text_labels_and_annotations/legend.py
    321     """
--> 322     handles, labels, kwargs = mlegend._parse_legend_args([self], *
args, **kwargs)
    323     self.legend_ = mlegend.Legend(self, handles, labels, **kwargs)
    324     self.legend_._remove_method = self._remove_legend

File ~/anaconda3/lib/python3.11/site-packages/matplotlib/legend.py:1361, i
n _parse_legend_args(axs, handles, labels, *args, **kwargs)
   1357     handles = [handle for handle, label
   1358                in zip(_get_legend_handles(axs, handlers), labels)]
   1360 elif len(args) == 0:  # 0 args: automatically detect labels and ha
ndles.
-> 1361     handles, labels = _get_legend_handles_labels(axs, handlers)
   1362     if not handles:
   1363         log.warning(
   1364             "No artists with labels found to put in legend.  Note
that "
```

```
   1365             "artists whose label start with an underscore are igno
red "
   1366             "when legend() is called with no argument.")

File ~/anaconda3/lib/python3.11/site-packages/matplotlib/legend.py:1291, i
n _get_legend_handles_labels(axs, legend_handler_map)
   1289 for handle in _get_legend_handles(axs, legend_handler_map):
   1290     label = handle.get_label()
-> 1291     if label and not label.startswith('_'):
   1292         handles.append(handle)
   1293         labels.append(label)

AttributeError: 'numpy.int64' object has no attribute 'startswith'
```



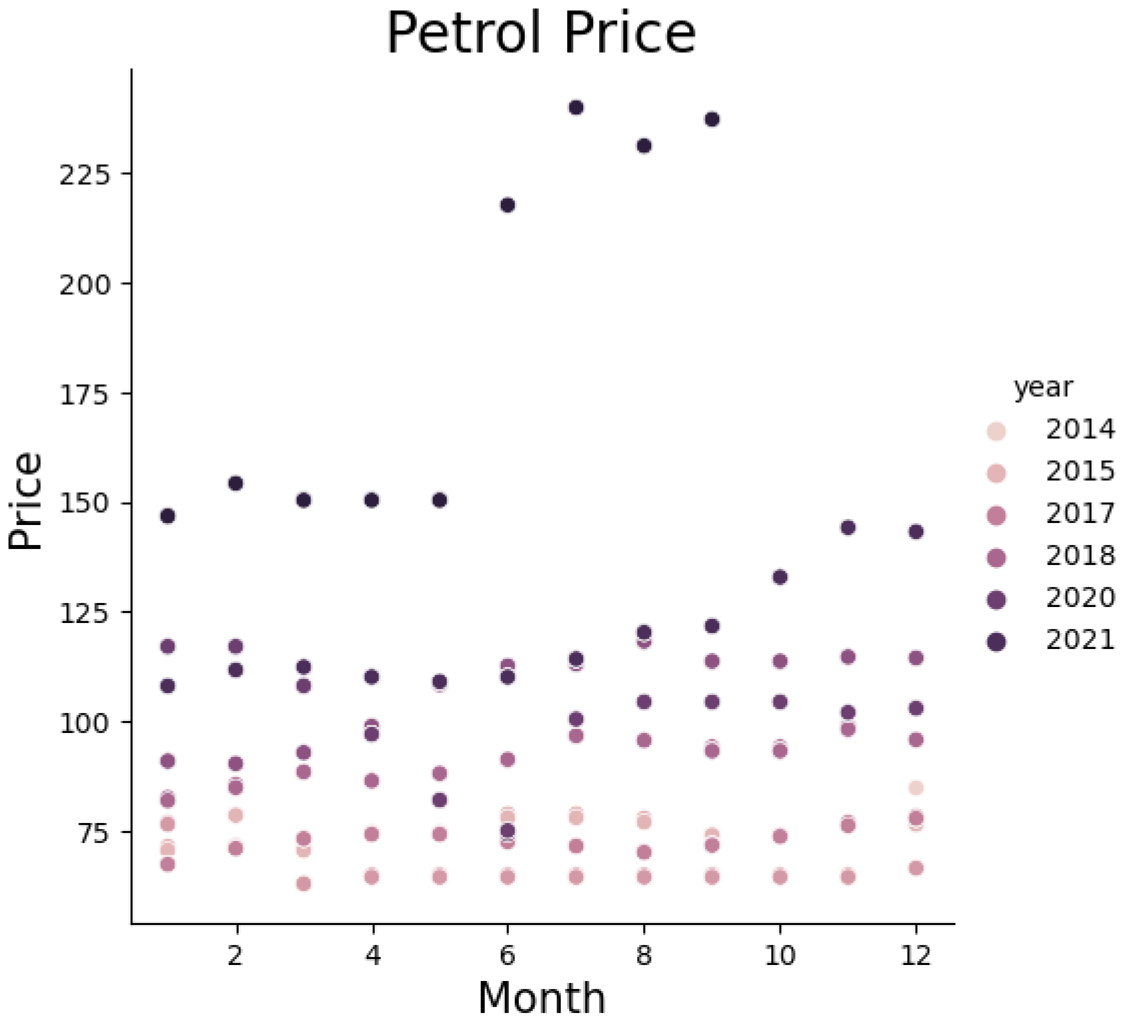
In [870…

```
sns.relplot(x = diesel_food_df["month"], y = diesel_food_df["price"], hue
plt.xlabel("Month", fontsize = 15)
plt.ylabel("Price", fontsize = 15)
plt.title("Diesel Price", fontsize = 20)
plt.savefig('Price of Diesel.png')
```



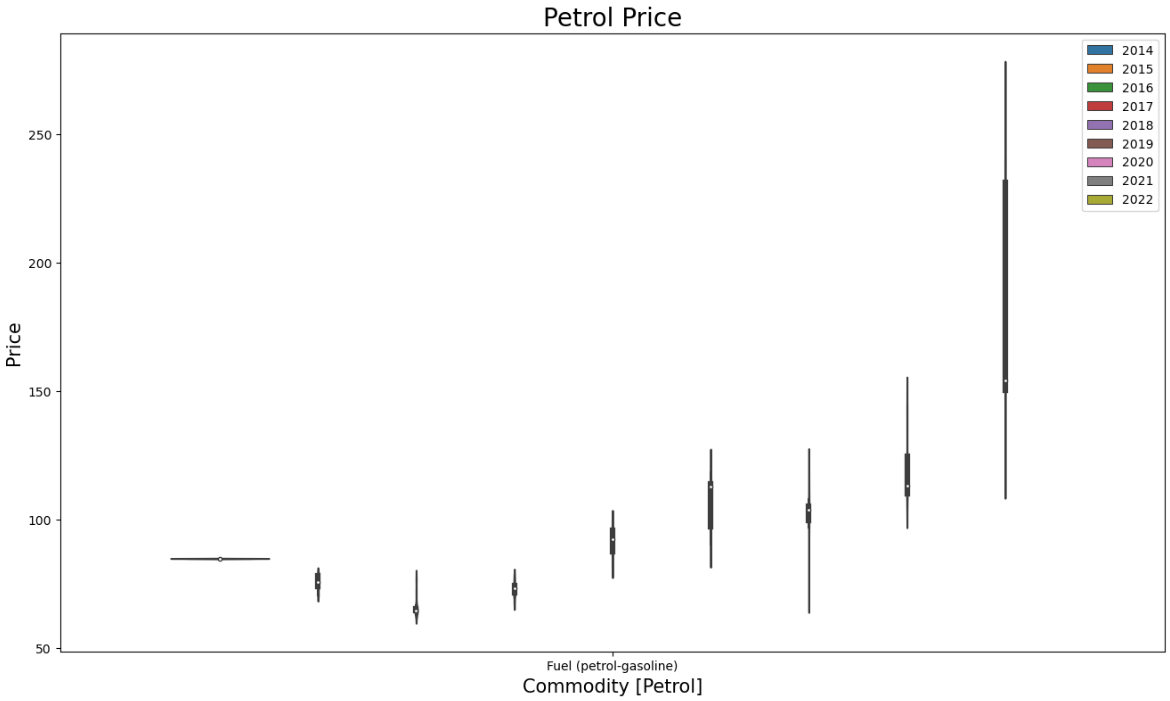
In [871…

```
sns.relplot(x = petrol_food_df["month"], y = petrol_food_df["price"], hue
plt.xlabel("Month", fontsize = 15)
plt.ylabel("Price", fontsize = 15)
plt.title("Petrol Price", fontsize = 20)
plt.savefig('Price of Petrol.png')
```



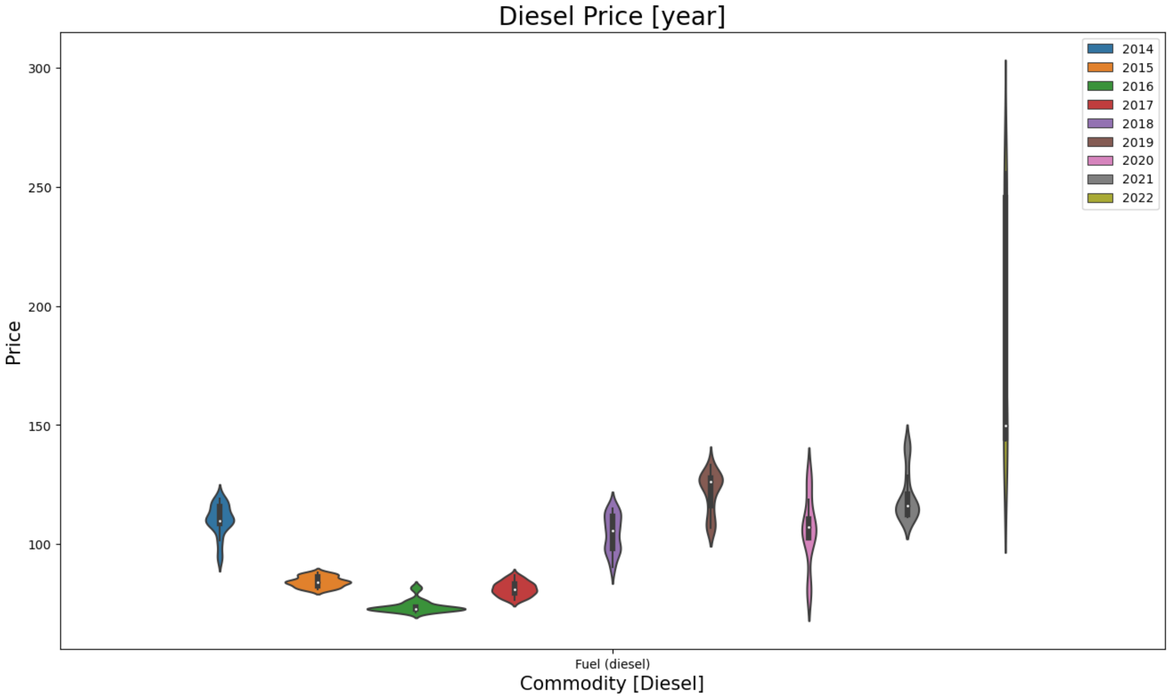
In [872…

```
fig = plt.subplots(figsize=(16, 9))
sns.violinplot(x=petrol_food_df["commodity"], y=petrol_food_df["price"],
plt.xlabel("Commodity [Petrol]", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Petrol Price", fontsize=20)
plt.legend(loc ="upper right")
plt.savefig('Price of petrol.png')
plt.show()
```



In [873…

```
fig = plt.subplots(figsize=(16, 9))
sns.violinplot(x=diesel_food_df["commodity"], y=diesel_food_df["price"],
plt.xlabel("Commodity [Diesel]", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Diesel Price [year]", fontsize=20)
plt.legend(loc ="upper right")
plt.savefig('Price of Diesel.png')
plt.show()
```



In [874…

```
df
```

Out[874…

	date	Provinces name	City Name	City market	latitude	longitude	category
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers
...	...	...	...	...	...	...	...
9718	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9719	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9720	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9721	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9722	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts

9723 rows × 13 columns

In [875…

```
non_food_df = df[df["category"].str.contains("non-food", regex=True)]
non_food_df.drop(["latitude", "longitude"], axis=1, inplace=True)
non_food_df.head()
```

Out[875…

	date	Provinces name	City Name	City market	category	commodity	unit	price
1785	9/15/2013	BALOCHISTAN	Quetta	Quetta	non-food	Wage (non-qualified labour, non-agricultural)	Day	550.0
1821	10/15/2013	BALOCHISTAN	Quetta	Quetta	non-food	Wage (non-qualified labour, non-agricultural)	Day	550.0
1842	11/15/2013	BALOCHISTAN	Quetta	Quetta	non-food	Wage (non-qualified labour, non-agricultural)	Day	550.0
1881	1/15/2014	BALOCHISTAN	Quetta	Quetta	non-food	Fuel (diesel)	L	117.1
1882	1/15/2014	BALOCHISTAN	Quetta	Quetta	non-food	Wage (non-qualified labour, non-agricultural)	Day	550.0

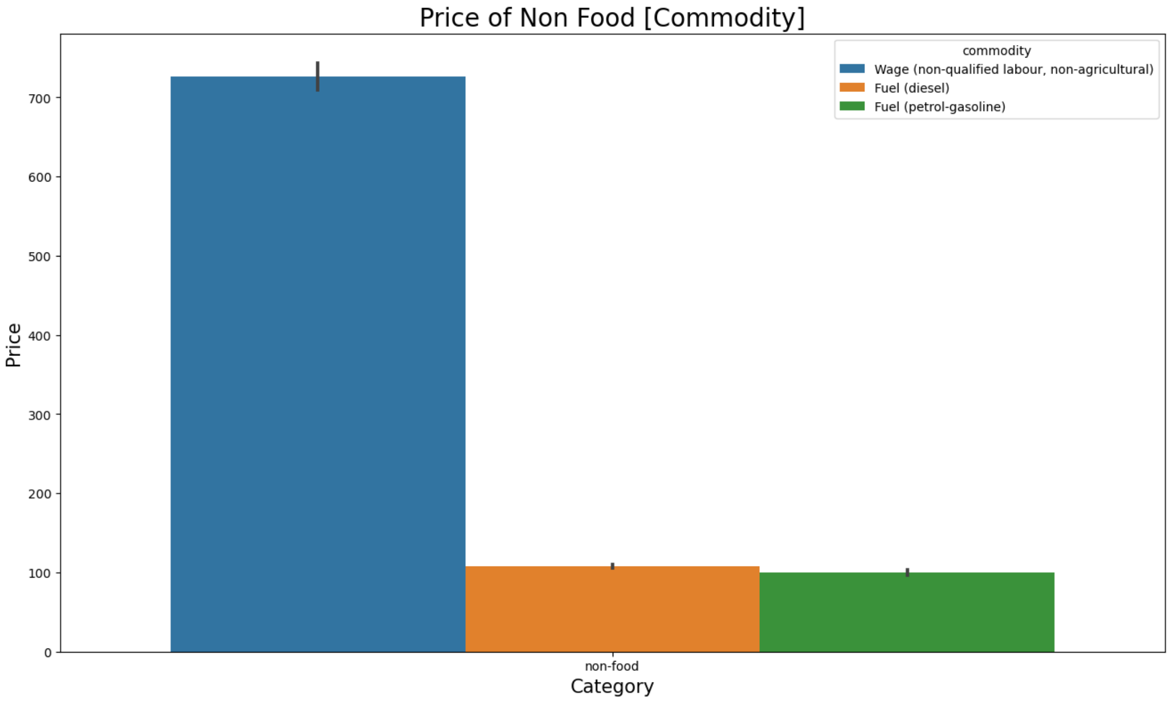
In [876…

```
non_food = non_food_df[["Provinces name", "City Name", "City market","cat
                        "commodity", "unit"]]

for i in non_food.columns:
    print("\n", non_food[i].unique())
```

```
['BALOCHISTAN' 'KHYBER PAKHTUNKHWA' 'PUNJAB' 'SINDH']

['Quetta' 'Peshawar' 'Lahore' 'Multan' 'Karachi']

['Quetta' 'Peshawar' 'Lahore' 'Multan' 'Karachi']

['non-food']

['Wage (non-qualified labour, non-agricultural)' 'Fuel (diesel)'
'Fuel (petrol-gasoline)']

['Day' 'L']
```

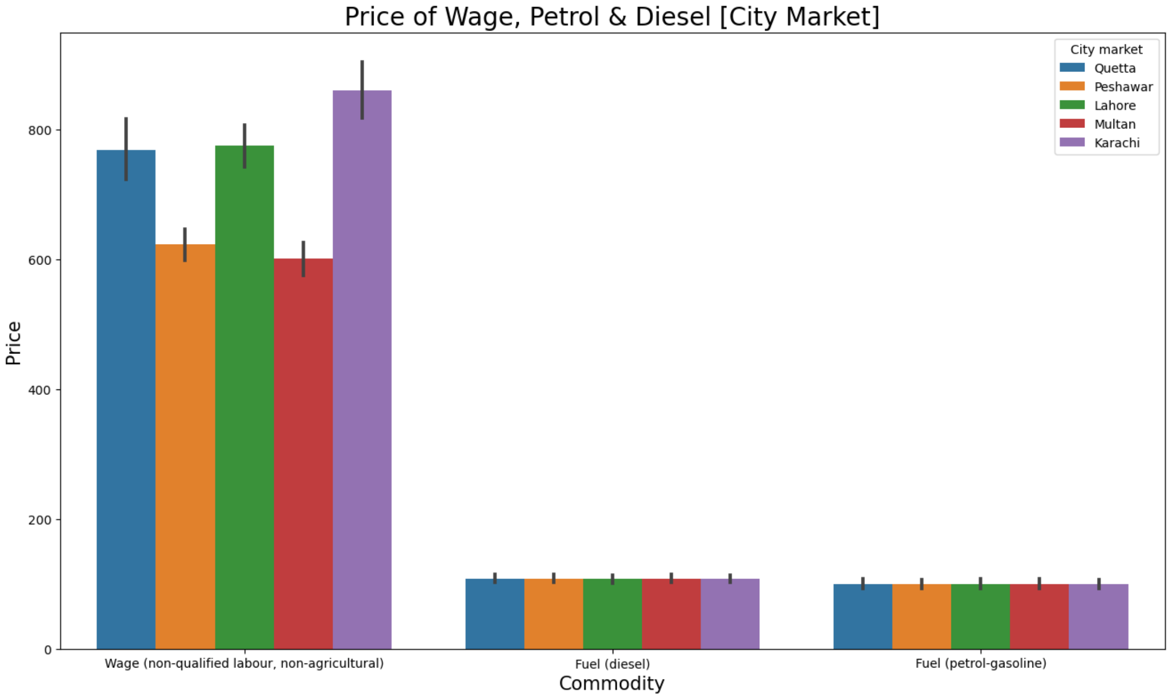
In [877…

```
fig = plt.subplots(figsize=(16, 9))
sns.barplot(x=non_food_df["category"], y=non_food_df["price"], hue=non_fo
plt.xlabel("Category", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Non Food [Commodity]", fontsize=20)
plt.savefig('Price of Non Food Commodity.png')
plt.show()
```



In [878…

```
fig = plt.subplots(figsize=(16, 9))
sns.barplot(x=non_food_df["commodity"], y=non_food_df["price"], hue=non_f
plt.xlabel("Commodity", fontsize=15)
plt.ylabel("Price", fontsize=15)
plt.title("Price of Wage, Petrol & Diesel [City Market]", fontsize=20)
plt.savefig('Price of Wage, Petrol & Diesel [City Market].png')
plt.show()
```



In [879…

```
df
```

Out[879…

	date	Provinces name	City Name	City market	latitude	longitude	category
0	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
1	1/15/2004	BALOCHISTAN	Quetta	Quetta	30.187222	67.012500	cereals and tubers
2	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
3	1/15/2004	KHYBER PAKHTUNKHWA	Peshawar	Peshawar	34.008366	71.580182	cereals and tubers
4	1/15/2004	PUNJAB	Lahore	Lahore	31.549722	74.343611	cereals and tubers
...	...	...	...	...	...	...	...
9718	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9719	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	oil and fats
9720	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9721	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts
9722	9/15/2022	SINDH	Karachi	Karachi	24.905600	67.082200	pulses and nuts

9723 rows × 13 columns

In [880…

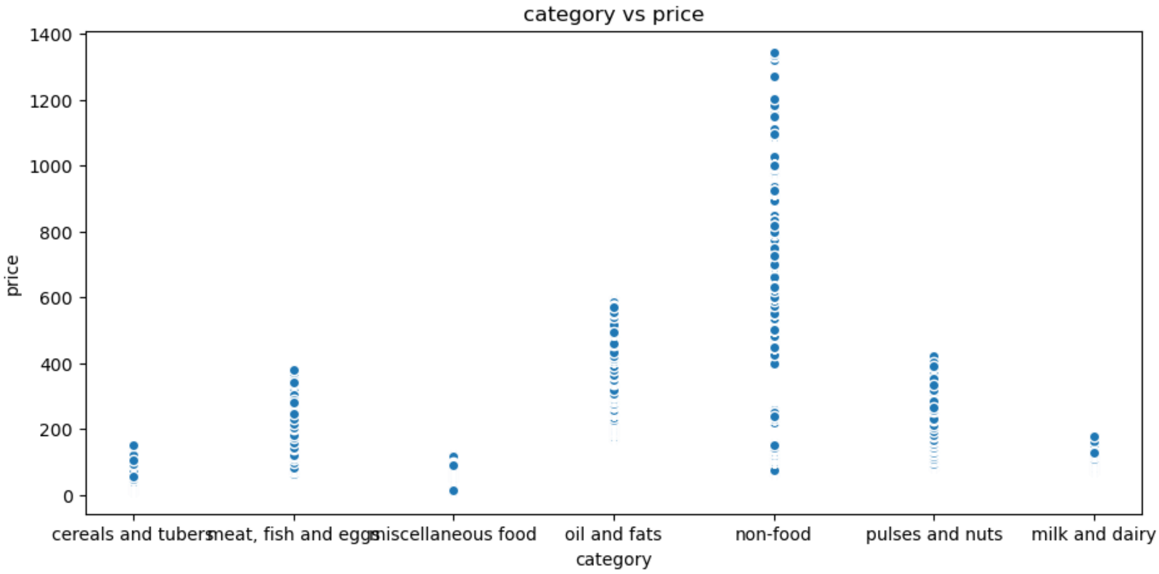
```
columns_to_drop = ['date','latitude','longitude','City market','City Name
```

In [881…

```
df = df.drop(columns=columns_to_drop)
```

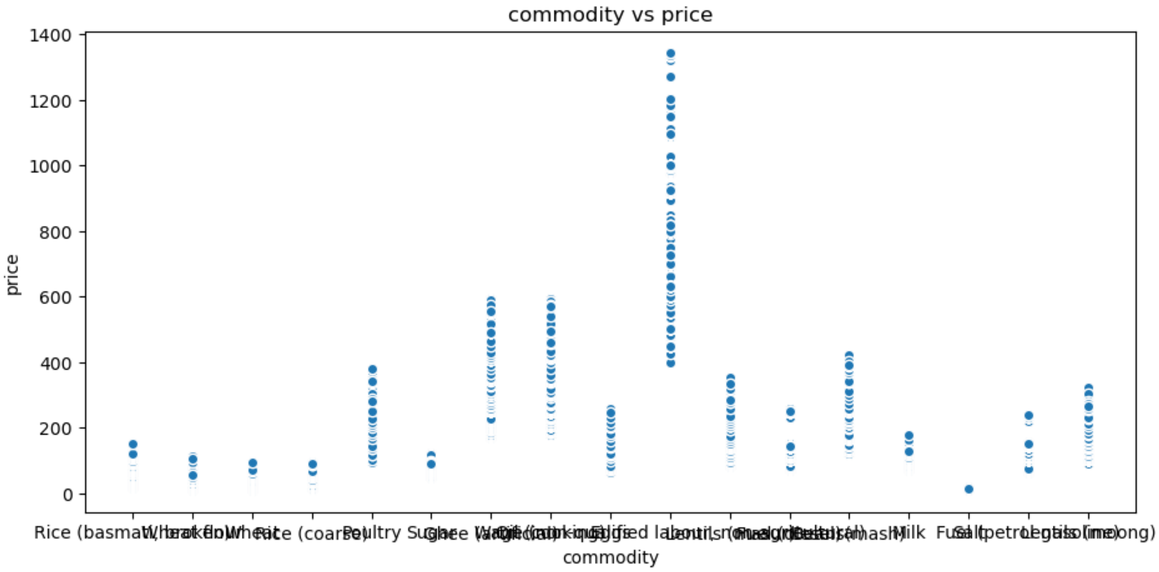
In [882…

```
plt.figure(figsize=(50,5))
plt.subplot(1,4,1)
plt.scatter(x=df['category'],y=df['price'],edgecolors='white')
plt.title('category vs price')
plt.xlabel('category')
plt.ylabel('price')
plt.savefig("category vs price.png")
```



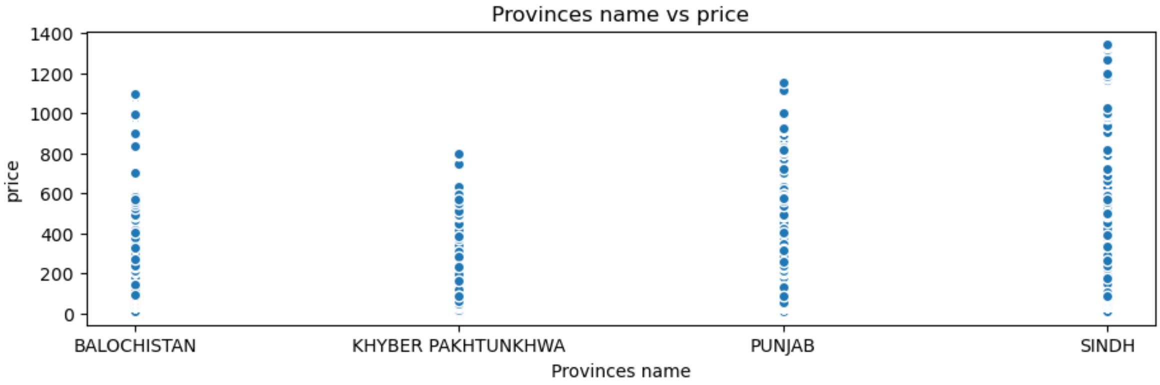
In [883…

```
plt.figure(figsize=(50,5))
plt.subplot(1,4,1)
plt.scatter(x=df['commodity'],y=df['price'],edgecolors='white')
plt.title('commodity vs price')
plt.xlabel('commodity')
plt.ylabel('price')
plt.savefig("commodity vs price")
```



In [884…

```
plt.figure(figsize=(50,3))
plt.subplot(1,4,1)
plt.scatter(x=df['Provinces name'],y=df['price'],edgecolors='white')
plt.title('Provinces name vs price')
plt.xlabel('Provinces name')
plt.ylabel('price')
plt.savefig("provinces names vs price")
```



Choosing Machine Learning Model

Choose

As you can see that one realtion is not enough to predict the Price. So, we have to use Multivarible Model

In [907…

```
from sklearn import svm
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.metrics import explained_variance_score
```

In [908…

```
le = LabelEncoder()
```

In [909…

```
y=df['price']
x=df.drop("price",axis=1)
```

In [910…

```
le.fit(np.unique(x))
x
```

Out[910…

	Provinces name	category	commodity
0	BALOCHISTAN	cereals and tubers	Rice (basmati, broken)
1	BALOCHISTAN	cereals and tubers	Wheat flour
2	KHYBER PAKHTUNKHWA	cereals and tubers	Rice (basmati, broken)
3	KHYBER PAKHTUNKHWA	cereals and tubers	Wheat flour
4	PUNJAB	cereals and tubers	Rice (basmati, broken)
...	...	...	...
9718	SINDH	oil and fats	Ghee (artificial)
9719	SINDH	oil and fats	Oil (cooking)
9720	SINDH	pulses and nuts	Beans(mash)
9721	SINDH	pulses and nuts	Lentils (masur)
9722	SINDH	pulses and nuts	Lentils (moong)

9723 rows × 3 columns

In [911…

```
x= pd.DataFrame(le.transform(samp) for samp in x.values)
```

In [912…

```
x
```

Out[912…

	0	1	2
0	0	21	13
1	0	21	20
2	6	21	13
3	6	21	20
4	11	21	13
...	...	...	...
9718	15	26	5
9719	15	26	10
9720	15	27	1
9721	15	27	7
9722	15	27	8

9723 rows × 3 columns

In [913…

```
X_train, X_test, y_train, y_test = train_test_split(x, y, random_state=0,
```

In [914…

```
X_train.shape
```

Out[914…

```
(7292, 3)
```

In [915…

```
X_test.shape
```

Out[915…

```
(2431, 3)
```

In [916…

```
y_test.values.reshape
```

Out[916…

```
<function ndarray.reshape>
```

In [917…

```
y_test
```

Out[917…

```
1651     51.54
8360     90.00
7643    259.29
2774     41.00
1332     33.40
         ...  
378      20.25
412      12.80
737      32.50
3932    166.43
9315    107.69
Name: price, Length: 2431, dtype: float64
```

In [918…

```
y_train.values.reshape
```

Out[918…

```
<function ndarray.reshape>
```

In [919…

```
y_train
```

Out[919…

```
2000     42.00
6069    186.00
1590     66.69
971      27.38
4558     60.27
         ...  
7891     56.90
9225    144.63
4859     85.00
3264     54.18
2732     57.84
Name: price, Length: 7292, dtype: float64
```

Support Vector Machine

In [950…

```
# Create a Random Forest Regressor
support_vector = svm.SVR()

# Train the model
support_vector.fit(X_test, y_test)

# Make predictions on the test set
y_pred = support_vector.predict(X_test)

# Evaluate Model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)
rmse = np.sqrt(mse)
print('Root Mean Squared Error (RMSE)',rmse)
print("Mean Square Error", mse)
print("Mean Absolute Error", mae)
print("R 2 Score", r2*100)
print("Explained Variance Score", evs*100)
```

```
Root Mean Squared Error (RMSE) 168.27522045869281
Mean Square Error 28316.549820421667
Mean Absolute Error 73.80155126261897
R 2 Score -3.293660240778906
Explained Variance Score 5.044238002384304
```

Linear Regression

In [949…

```
# Create a Linear Regressor
linear_regression = LinearRegression()

# Train the model
linear_regression.fit(X_test, y_test)

# Make predictions on the test set
y_pred = linear_regression.predict(X_test)

# Make predictions on the test set
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)
rmse = np.sqrt(mse)
print('Root Mean Squared Error (RMSE)',rmse)
print("Mean Square Error", mse)
print("Mean Absolute Error", mae)
print("R2 Score", r2*100)
print("Explained Variance Score", evs*100)
```

```
Root Mean Squared Error (RMSE) 142.53273347325083
Mean Square Error 20315.580111356754
Mean Absolute Error 90.61382136081875
R2 Score 25.892432414085633
Explained Variance Score 25.89243241408562
```

MLPRegressor

In [948…

```
# Create an MLPRegressor
nn = MLPRegressor(hidden_layer_sizes=(100,), max_iter=1000, random_state=

# Train the model
nn.fit(X_test, y_test)

# Make predictions on the test set
y_pred = nn.predict(X_test)

# Evaluate model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)

rmse = np.sqrt(mse)
```

```
print('Root Mean Squared Error (RMSE)',rmse)
print("Mean Square Error", mse)
print("Mean Absolute Error", mae)
print("R2 Score", r2*100)
print("Explained Variance Score", evs*100)
```

```
Root Mean Squared Error (RMSE) 141.88188520159466
Mean Square Error 20130.469348358485
Mean Absolute Error 84.17868674757146
R2 Score 26.56768304953787
Explained Variance Score 26.626812180771132
```

AdaBoostRegressor

In [947…

```
# Create an AdaBoostRegressor with a base estimator (e.g., DecisionTreeRe
base_estimator = DecisionTreeRegressor(max_depth=1)
adaboost_regressor = AdaBoostRegressor(base_estimator=base_estimator, n_e

# Train the model
adaboost_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = adaboost_regressor.predict(X_test)

# Evaluate model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)
rmse = np.sqrt(mse)

print('Root Mean Squared Error (RMSE)',rmse)
print("Mean Square Error", mse)
print("Mean Absolute Error", mae)
print("R2 Score", r2*100)
print("Explained Variance Score", evs*100)
```

```
Root Mean Squared Error (RMSE) 120.62228310308092
Mean Square Error 14549.7351809998
Mean Absolute Error 73.7212194207669
R2 Score 46.925193503072
Explained Variance Score 46.96202967637013
```

Random Forest Regressor

In [946…

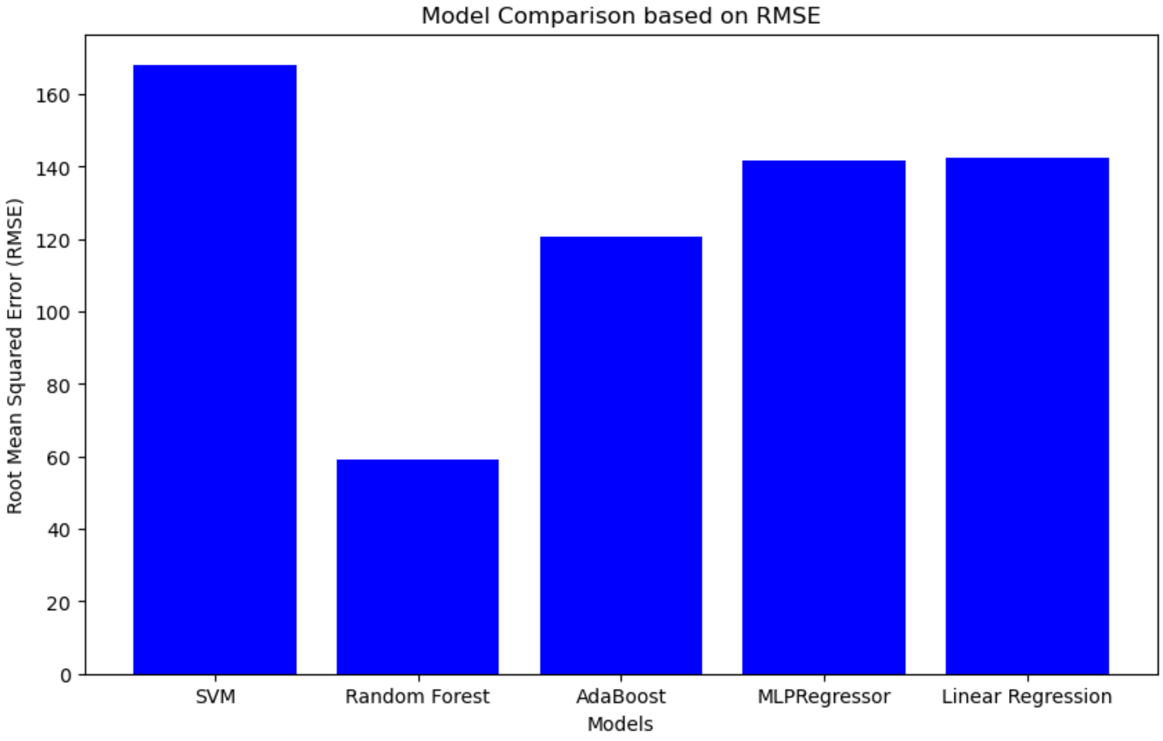
```
# Create a Random Forest Regressor
random_forest_regressor = RandomForestRegressor(n_estimators=100, random_

# Train the model
random_forest_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = random_forest_regressor.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
evs = explained_variance_score(y_test, y_pred)
rmse = np.sqrt(mse)
print('Root Mean Squared Error (RMSE)',rmse)
print("Mean Square Error", mse)
print("Mean Absolute Error", mae)
print("R2 Score", r2*100)
print("Explained Variance Score", evs*100)
```

```
Root Mean Squared Error (RMSE) 59.30947709463605
Mean Square Error 3517.614073239158
Mean Absolute Error 34.92137966193879
R2 Score 87.16837908418825
Explained Variance Score 87.20058331486858
```

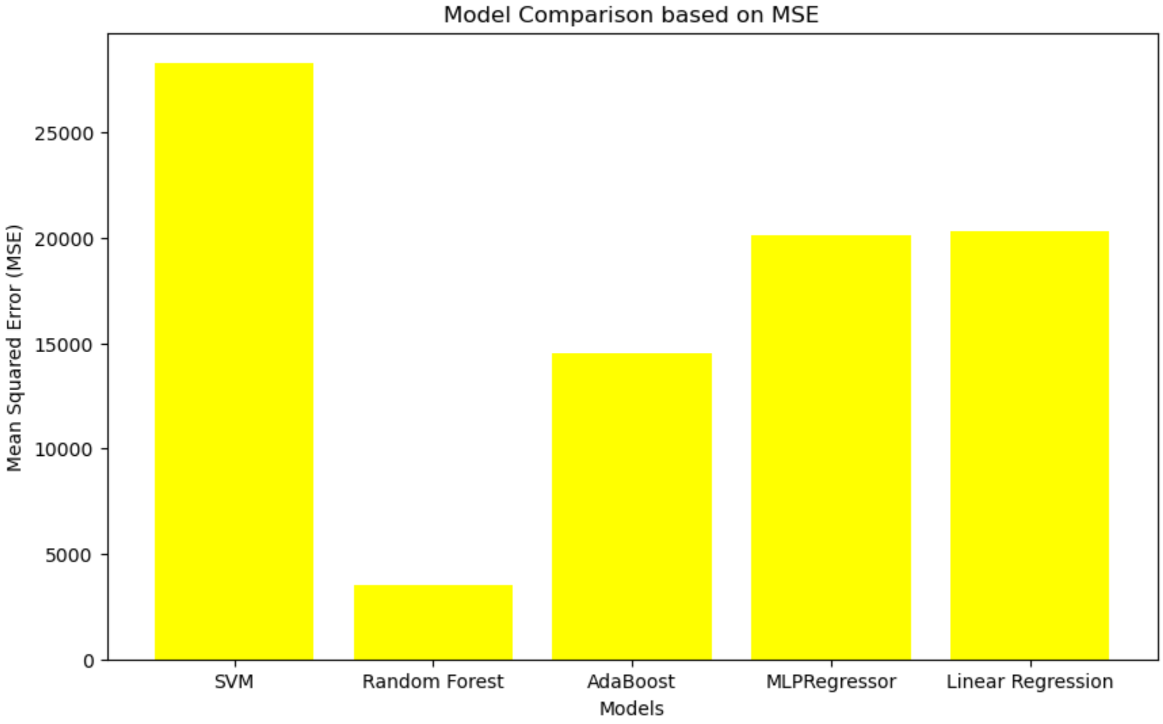
In [962…

```
import matplotlib.pyplot as plt

# Assuming you have a list of model names and their corresponding RMSE va
model_names = ['SVM', 'Random Forest', 'AdaBoost', 'MLPRegressor', 'Linea
rmse_values = [168.2, 59.3, 120.6, 141.8, 142.5]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, rmse_values, color='blue')
plt.xlabel('Models')
plt.ylabel('Root Mean Squared Error (RMSE)')
plt.title('Model Comparison based on RMSE')
plt.show()
```



In [963…

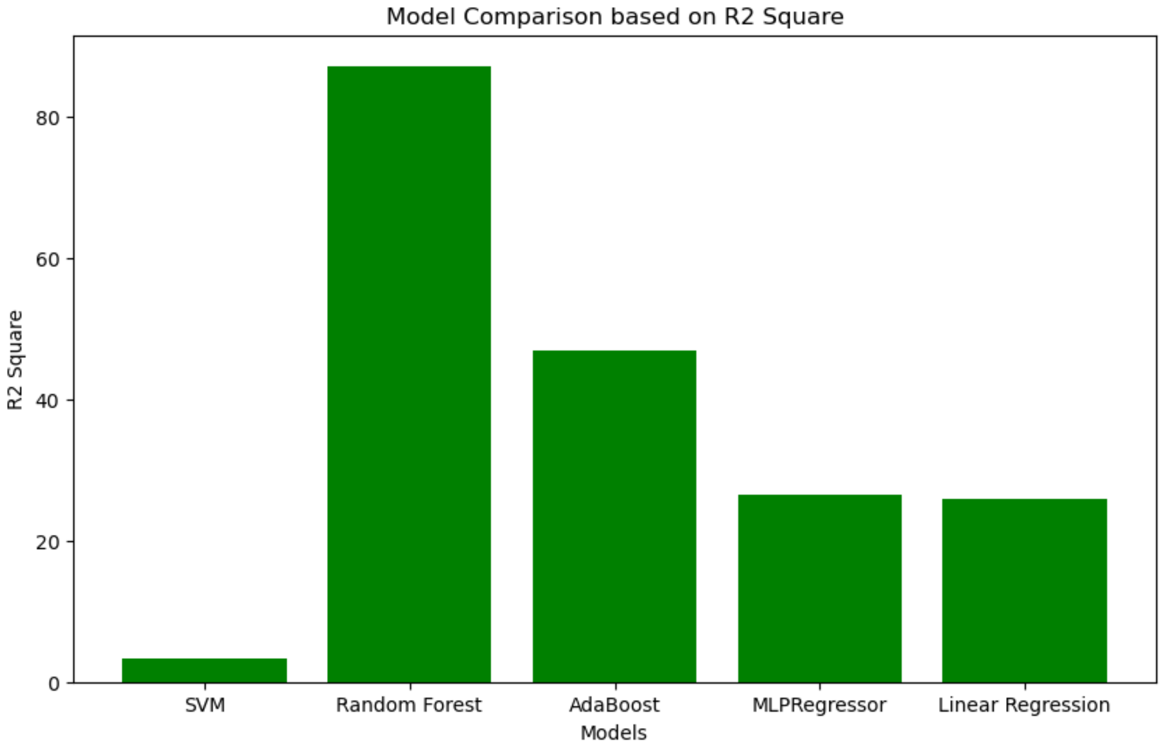
```
import matplotlib.pyplot as plt

# Assuming you have a list of model names and their corresponding RMSE va
model_names = ['SVM', 'Random Forest', 'AdaBoost', 'MLPRegressor', 'Linea
rmse_values = [28316.5, 3517.6, 14549.7, 20130.5, 20315.5]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, rmse_values, color='yellow')
plt.xlabel('Models')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Model Comparison based on MSE')
plt.show()
plt.savefig("MSE.png")
```



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

In [964…

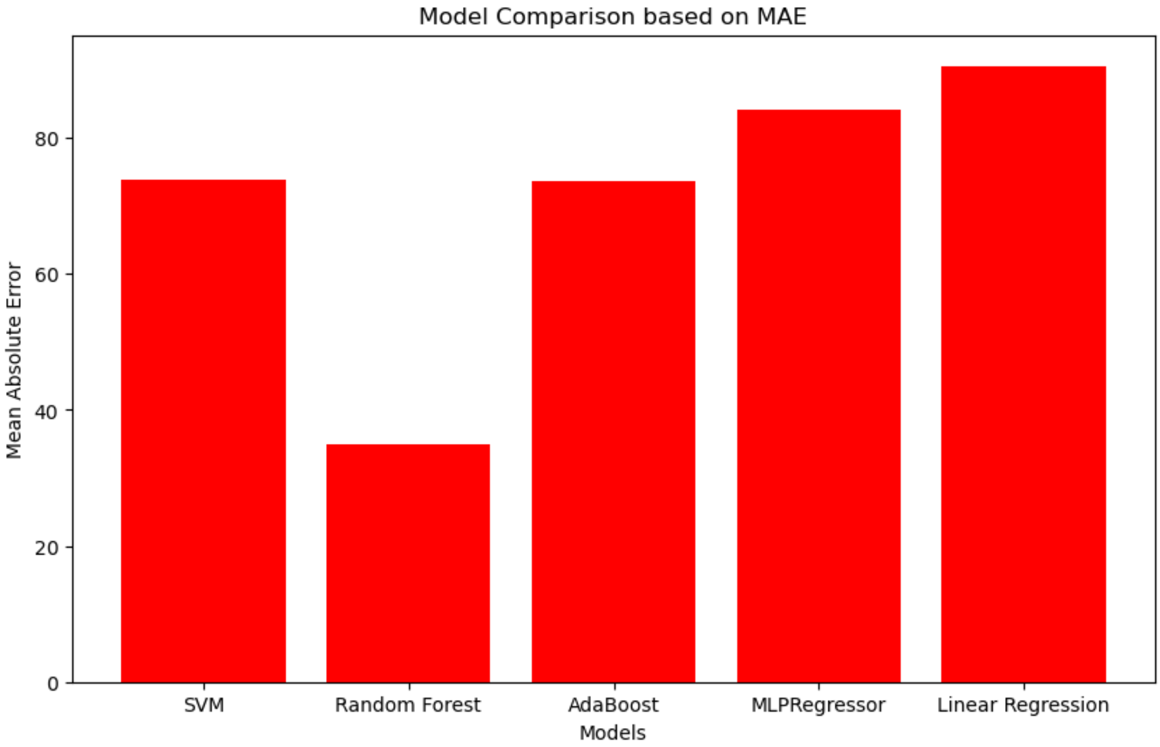
```
import matplotlib.pyplot as plt

# Assuming you have a list of model names and their corresponding RMSE va
model_names = ['SVM', 'Random Forest', 'AdaBoost', 'MLPRegressor', 'Linea
r2_values = [3.29, 87.1, 46.92, 26.5, 25.8]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, r2_values, color='green')
plt.xlabel('Models')
plt.ylabel('R2 Square')
plt.title('Model Comparison based on R2 Square')
plt.show()
plt.savefig("R2 Sqauare.png")
```



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

In [965…

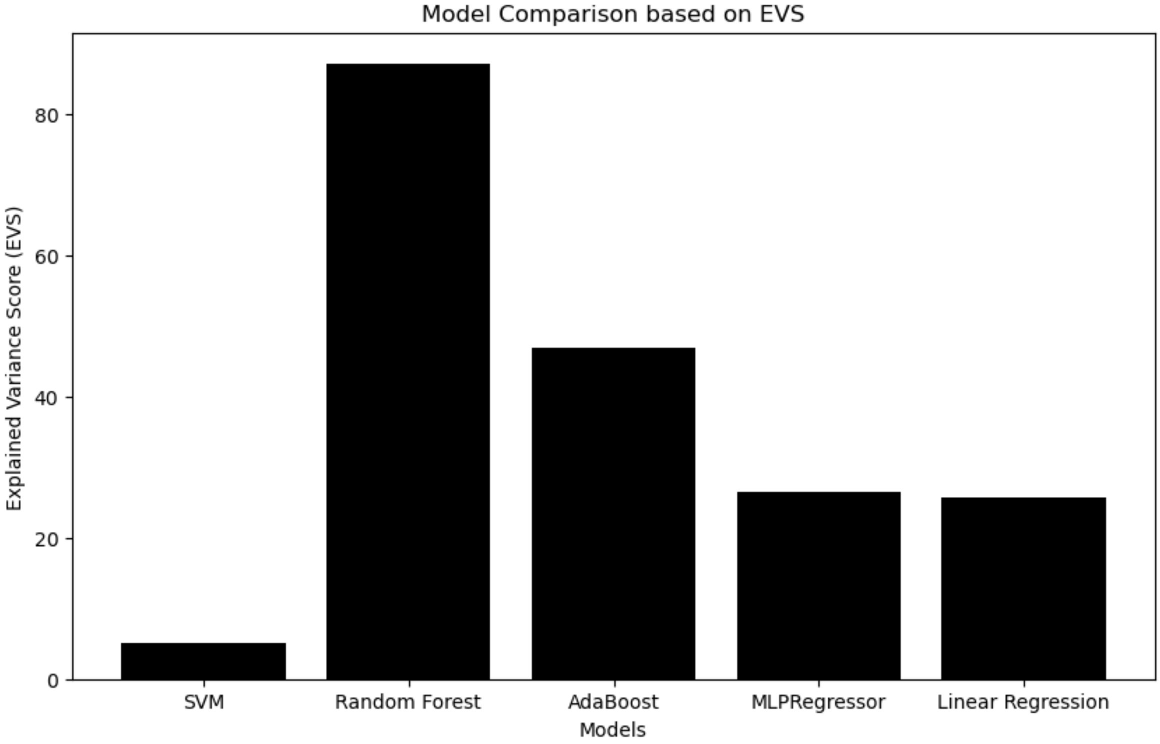
```
import matplotlib.pyplot as plt

# Assuming you have a list of model names and their corresponding RMSE va
model_names = ['SVM', 'Random Forest', 'AdaBoost', 'MLPRegressor', 'Linea
mae_values = [73.8, 34.9, 73.7, 84.1, 90.6]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, mae_values, color='Red')
plt.xlabel('Models')
plt.ylabel('Mean Absolute Error')
plt.title('Model Comparison based on MAE')
plt.show()
plt.savefig("MAE.png")
```



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

In [966…

```
import matplotlib.pyplot as plt

# Assuming you have a list of model names and their corresponding RMSE va
model_names = ['SVM', 'Random Forest', 'AdaBoost', 'MLPRegressor', 'Linea
rmse_values = [5.04, 87.20, 46.96, 26.6, 25.8]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(model_names, rmse_values, color='black')
plt.xlabel('Models')
plt.ylabel('Explained Variance Score (EVS)')
plt.title('Model Comparison based on EVS')
plt.show()
plt.savefig("EVS.png")
```


Model Comparison based on EVS
Explained Variance Score (EVS)
0
20
40
60
80
SVM
Random Forest
AdaBoost
MLPRegressor
Linear Regression
Models

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