Importing Libraries and dataset

In [1]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
products = pd.read_csv("data.csv", encoding='unicode_escape')
```

In [3]:
```
products
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	Ur King
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	Ur King
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	Ur King
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	Ur King
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	Ur King
...	...	...	...	...	...	...	...	
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	Fra
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	Fra
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	Fra
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	Fra
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	Fra

541909 rows × 8 columns

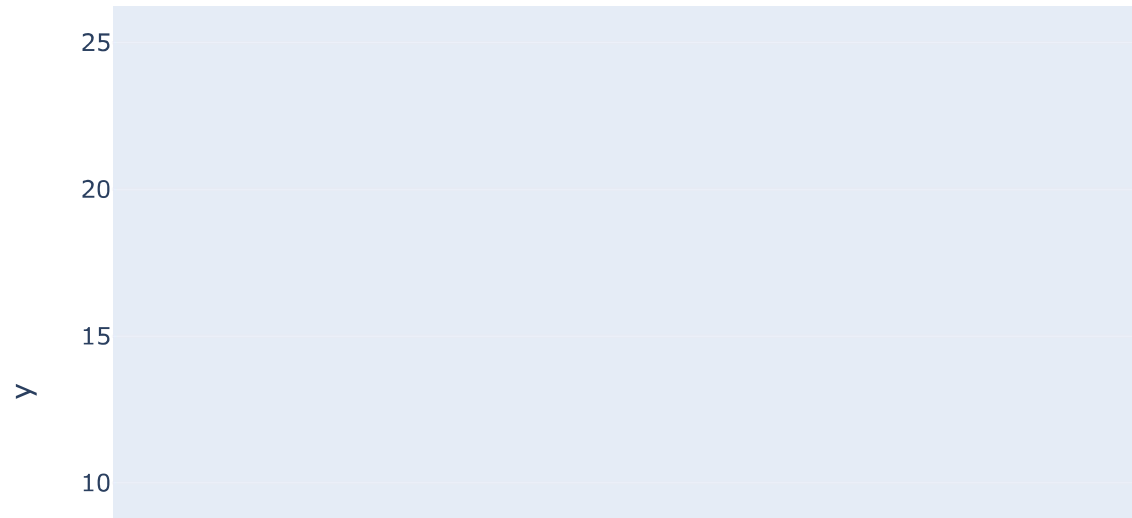
In [4]:
```
#Statistical analysis of each features of the dataset
products.describe(include="all")
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custo
count	541909	541909	540455	541909.000000	541909	541909.000000	406829.0
unique	25900	4070	4223	NaN	23260	NaN	
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	NaN	10/31/2011 14:41	NaN	
freq	1114	2313	2369	NaN	1114	NaN	
mean	NaN	NaN	NaN	9.552250	NaN	4.611114	15287.6
std	NaN	NaN	NaN	218.081158	NaN	96.759853	1713.6
min	NaN	NaN	NaN	-80995.000000	NaN	-11062.060000	12346.0
25%	NaN	NaN	NaN	1.000000	NaN	1.250000	13953.0
50%	NaN	NaN	NaN	3.000000	NaN	2.080000	15152.0
75%	NaN	NaN	NaN	10.000000	NaN	4.130000	16791.0
max	NaN	NaN	NaN	80995.000000	NaN	38970.000000	18287.0

Data Preprocessing

In [5]:
```
#Checking the missing values in the dataset since it can introduce bias and ca
fig = px.bar(x=products.columns , y = (products.isnull().sum()/products.shape[
fig.show()
```



In [6]:
```
#Dropped this column since it was non informative and unique to each customer
products.drop("CustomerID" , axis = 1 , inplace = True)
```

In [7]:
```
#Null values in this colum was less than 5 % so it was better to drop those
products.dropna(subset=['Description'] , inplace=True)
```

In [8]:
```
#After imputation work , no null values
products.isnull().sum()
```

Out[8]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
Country        0
dtype: int64
```

In [9]:
```
print("Shape of the dataset:" ,products.shape)
```

```
Shape of the dataset: (540455, 7)
```

In [10]:
```
#Checking the datatypes to ensure proper work flow
products.dtypes
```

Out[10]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
Country         object
dtype: object
```

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

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	United Kingdom

In [12]:
```
#Formatting the raw date
products['Date'] = pd.to_datetime(products['InvoiceDate'])
products['Month-Year'] = products['Date'].dt.strftime('%b-%Y')
products.drop(['InvoiceDate','Date'],axis=1,inplace=True)
```

In [13]:
```
products
```

Out[13]:

	InvoiceNo	StockCode	Description	Quantity	UnitPrice	Country	Month-Year
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	United Kingdom	Dec-2010
1	536365	71053	WHITE METAL LANTERN	6	3.39	United Kingdom	Dec-2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2.75	United Kingdom	Dec-2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	3.39	United Kingdom	Dec-2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	3.39	United Kingdom	Dec-2010
...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	0.85	France	Dec-2011
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2.10	France	Dec-2011
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	4.15	France	Dec-2011
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	4.15	France	Dec-2011
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	4.95	France	Dec-2011

540455 rows × 7 columns

EDA

In [14]:
```
#Analysing products with high sales
products["Description"].value_counts()
```

Out[14]:
```
WHITE HANGING HEART T-LIGHT HOLDER     2369
REGENCY CAKESTAND 3 TIER               2200
JUMBO BAG RED RETROSPOT                2159
PARTY BUNTING                          1727
LUNCH BAG RED RETROSPOT                1638
                                       ...
Missing                                   1
historic computer difference?....se       1
DUSTY PINK CHRISTMAS TREE 30CM            1
WRAP BLUE RUSSIAN FOLKART                 1
PINK BERTIE MOBILE PHONE CHARM            1
Name: Description, Length: 4223, dtype: int64
```

In [15]:
```
#Checking the entities of the max sale product
products_maximum = products[products["Description"] == "WHITE HANGING HEART T-
products_maximum
```

Out[15]:

	InvoiceNo	StockCode	Description	Quantity	UnitPrice	Country	Month-Year
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	United Kingdom	Dec-2010
49	536373	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	United Kingdom	Dec-2010
66	536375	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.55	United Kingdom	Dec-2010
220	536390	85123A	WHITE HANGING HEART T-LIGHT HOLDER	64	2.55	United Kingdom	Dec-2010
262	536394	85123A	WHITE HANGING HEART T-LIGHT HOLDER	32	2.55	United Kingdom	Dec-2010
...	...	...	...	...	...	...	...
537291	581246	85123A	WHITE HANGING HEART T-LIGHT HOLDER	1	2.95	United Kingdom	Dec-2011
537326	581253	85123A	WHITE HANGING HEART T-LIGHT HOLDER	2	2.95	United Kingdom	Dec-2011
537852	581356	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.95	United Kingdom	Dec-2011
539979	581452	85123A	WHITE HANGING HEART T-LIGHT HOLDER	32	2.55	United Kingdom	Dec-2011
540217	581472	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2.95	United Kingdom	Dec-2011

2369 rows × 7 columns

In [16]:
```
#Now lets even check the entities of 2nd highest sale product
products_sec_maximum = products[products["Description"] == "REGENCY CAKESTAND
products_sec_maximum
```

Out[16]:

	InvoiceNo	StockCode	Description	Quantity	UnitPrice	Country	Month-Year
880	536477	22423	REGENCY CAKESTAND 3 TIER	16	10.95	United Kingdom	Dec-2010
936	536502	22423	REGENCY CAKESTAND 3 TIER	2	12.75	United Kingdom	Dec-2010
1092	536525	22423	REGENCY CAKESTAND 3 TIER	2	12.75	United Kingdom	Dec-2010
1155	536528	22423	REGENCY CAKESTAND 3 TIER	1	12.75	United Kingdom	Dec-2010
1197	536530	22423	REGENCY CAKESTAND 3 TIER	1	12.75	United Kingdom	Dec-2010
...	...	...	...	...	...	...	...
539891	581449	22423	REGENCY CAKESTAND 3 TIER	1	12.75	United Kingdom	Dec-2011
539892	581449	22423	REGENCY CAKESTAND 3 TIER	1	12.75	United Kingdom	Dec-2011
540216	581472	22423	REGENCY CAKESTAND 3 TIER	2	12.75	United Kingdom	Dec-2011
541231	581495	22423	REGENCY CAKESTAND 3 TIER	10	12.75	United Kingdom	Dec-2011
541290	581497	22423	REGENCY CAKESTAND 3 TIER	8	24.96	United Kingdom	Dec-2011

2200 rows × 7 columns

In [17]:
```
#We can se that even the quantities are varying , so lets check according to t
products_q = products.groupby('Description')['Quantity'].sum().reset_index()
products_q.columns = ['Description', 'Total Quantity']
products_q
```

Out[17]:

	Description	Total Quantity
0	4 PURPLE FLOCK DINNER CANDLES	144
1	50'S CHRISTMAS GIFT BAG LARGE	1913
2	DOLLY GIRL BEAKER	2448
3	I LOVE LONDON MINI BACKPACK	389
4	I LOVE LONDON MINI RUCKSACK	1
...	...	...
4218	wrongly marked carton 22804	-256
4219	wrongly marked. 23343 in box	-3100
4220	wrongly sold (22719) barcode	170
4221	wrongly sold as sets	-600
4222	wrongly sold sets	-975

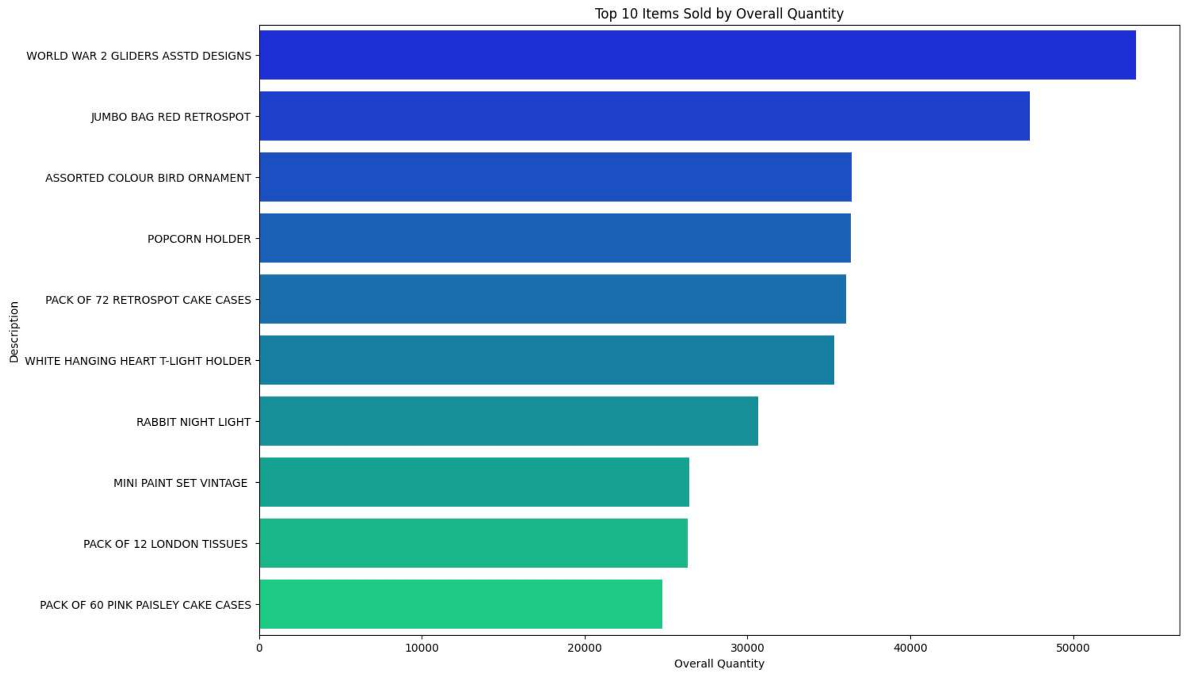
4223 rows × 2 columns

In [18]:
```
Top_10 = products_q.sort_values(by='Total Quantity', ascending=False).head(10)
Top_10
```

Out[18]:

	Description	Total Quantity
4009	WORLD WAR 2 GLIDERS ASSTD DESIGNS	53847
1866	JUMBO BAG RED RETROSPOT	47363
244	ASSORTED COLOUR BIRD ORNAMENT	36381
2740	POPCORN HOLDER	36334
2395	PACK OF 72 RETROSPOT CAKE CASES	36039
3918	WHITE HANGING HEART T-LIGHT HOLDER	35317
2803	RABBIT NIGHT LIGHT	30680
2161	MINI PAINT SET VINTAGE	26437
2361	PACK OF 12 LONDON TISSUES	26315
2393	PACK OF 60 PINK PAISLEY CAKE CASES	24753

In [19]:
```
#Top 10 items
plt.figure(figsize=(15, 10))
sns.barplot(data=Top_10, x="Total Quantity", y="Description", capsize=3, palet
plt.title("Top 10 Items Sold by Overall Quantity")
plt.xlabel("Overall Quantity")
plt.ylabel("Description")
plt.show()
```



In [20]:
```
#Lets extract top 15 products based on the unit price
products_expensive = products.sort_values(by = "UnitPrice" , ascending=False).
products_expensive
```

Out[20]:

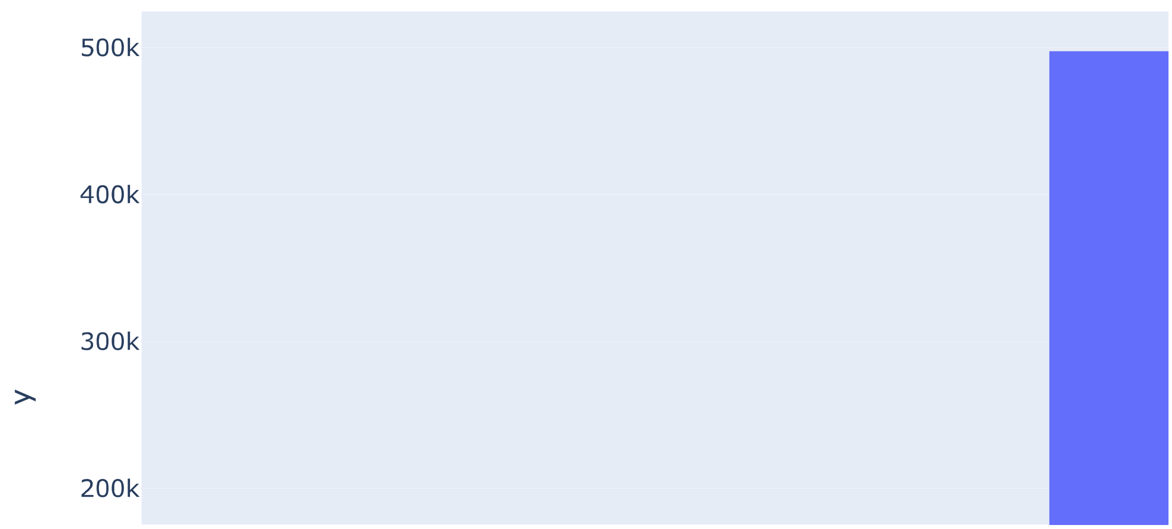
	InvoiceNo	StockCode	Description	Quantity	UnitPrice	Country	Month-Year
222681	C556445	M	Manual	-1	38970.00	United Kingdom	Jun-2011
524602	C580605	AMAZONFEE	AMAZON FEE	-1	17836.46	United Kingdom	Dec-2011
43702	C540117	AMAZONFEE	AMAZON FEE	-1	16888.02	United Kingdom	Jan-2011
43703	C540118	AMAZONFEE	AMAZON FEE	-1	16453.71	United Kingdom	Jan-2011
15016	C537630	AMAZONFEE	AMAZON FEE	-1	13541.33	United Kingdom	Dec-2010
15017	537632	AMAZONFEE	AMAZON FEE	1	13541.33	United Kingdom	Dec-2010
16356	C537651	AMAZONFEE	AMAZON FEE	-1	13541.33	United Kingdom	Dec-2010
16232	C537644	AMAZONFEE	AMAZON FEE	-1	13474.79	United Kingdom	Dec-2010
524601	C580604	AMAZONFEE	AMAZON FEE	-1	11586.50	United Kingdom	Dec-2011
299982	A563185	B	Adjust bad debt	1	11062.06	United Kingdom	Aug-2011
446533	C574902	AMAZONFEE	AMAZON FEE	-1	8286.22	United Kingdom	Nov-2011
173382	551697	POST	POSTAGE	1	8142.75	United Kingdom	May-2011
173277	C551685	POST	POSTAGE	-1	8142.75	United Kingdom	May-2011
342635	C566899	AMAZONFEE	AMAZON FEE	-1	7427.97	United Kingdom	Sep-2011
191386	C553355	AMAZONFEE	AMAZON FEE	-1	7006.83	United Kingdom	May-2011

In [21]:
```
# We can see that all the expensive products quantity is -1 , also means they
```

In [22]:
```
#Lets look to the top 5 countries
country_counts = products['Country'].value_counts()

# Select the top 5 countries
top_5_countries = country_counts.head()

# Convert the top countries data into a DataFrame
top_5_df = top_5_countries.reset_index()
top_5_df.columns = ['country', 'count']

# Create a bar plot using Plotly Express
fig = px.bar(top_5_df, x='country', y='count', title='Top 5 Countries')

# Show the plot
fig.show()
```

In [23]:
```
#Visualizing number of sales based on months
products['month'] = products["Month-Year"].str[:3]
month_count_value = products["month"].value_counts()
month_count_value=month_count_value.sort_values()
fig = px.bar(x=month_count_value.index , y = month_count_value.values)
fig.show()
```

In [24]:
```
#Visualizing number of sales based on years
products['year'] = products["Month-Year"].str[4:]
year_count_value = products["year"].value_counts()
year_count_value=year_count_value.sort_values()


fig = px.bar(x=year_count_value.index , y = year_count_value.values)
fig.show()
```



In [25]:
```
#Removing non informative features
products.drop("Description" , axis=1, inplace=True)
products.drop("StockCode" , axis=1, inplace=True)
products.drop("InvoiceNo" , axis=1, inplace=True)
products.drop("Month-Year" , axis=1, inplace=True)
products.drop("year" , axis=1, inplace=True)
products.drop("month" , axis=1, inplace=True)
```

In [26]:
```
products
```

Out[26]:

	Quantity	UnitPrice	Country
0	6	2.55	United Kingdom
1	6	3.39	United Kingdom
2	8	2.75	United Kingdom
3	6	3.39	United Kingdom
4	6	3.39	United Kingdom
...	...	...	...
541904	12	0.85	France
541905	6	2.10	France
541906	4	4.15	France
541907	4	4.15	France
541908	3	4.95	France

540455 rows × 3 columns

In [27]:
```
#Encoding Categorical column to Numerical column
categorical_cols = ["Country"]
```

In [28]:
```
from sklearn.preprocessing import LabelEncoder
Lc = LabelEncoder()
products["Country"]= Lc.fit_transform(products["Country"])
```

In [29]:
```
products
```

Out[29]:

	Quantity	UnitPrice	Country
0	6	2.55	36
1	6	3.39	36
2	8	2.75	36
3	6	3.39	36
4	6	3.39	36
...	...	...	...
541904	12	0.85	13
541905	6	2.10	13
541906	4	4.15	13
541907	4	4.15	13
541908	3	4.95	13

540455 rows × 3 columns

In [30]:
```
#Adding total price column
products['Total Price'] = products['UnitPrice'] * products['Quantity']
products.head()
```

Out[30]:

	Quantity	UnitPrice	Country	Total Price
0	6	2.55	36	15.30
1	6	3.39	36	20.34
2	8	2.75	36	22.00
3	6	3.39	36	20.34
4	6	3.39	36	20.34

In [31]:
```
#Analysing Correlation
sns.heatmap(products.corr(), cmap='Purples', annot=True, fmt=".2f")
```

Out[31]: <AxesSubplot: >

Data spliting and Modelling

In [32]:
```
#Splitting the dataset
X = products.drop("Total Price" , axis =1)

#Since we need to predict total price only so let make it our target dependent

y = products["Total Price"]
```

In [33]:
```
X
```

Out[33]:

	Quantity	UnitPrice	Country
0	6	2.55	36
1	6	3.39	36
2	8	2.75	36
3	6	3.39	36
4	6	3.39	36
...	...	...	...
541904	12	0.85	13
541905	6	2.10	13
541906	4	4.15	13
541907	4	4.15	13
541908	3	4.95	13

540455 rows × 3 columns

In [34]:
```
y
```

Out[34]:
```
0         15.30
1         20.34
2         22.00
3         20.34
4         20.34
          ...
541904    10.20
541905    12.60
541906    16.60
541907    16.60
541908    14.85
Name: Total Price, Length: 540455, dtype: float64
```

In [35]:
```
#Train test split
from sklearn.model_selection import train_test_split
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_s
```

In [37]:
```
#Modelling Part
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

In [38]:
```
#Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [39]:
```
models = [
    LinearRegression(),
    RandomForestRegressor(n_estimators=104, random_state=42),
    Lasso(alpha = 10),
    Ridge(alpha = 10)
]
```

Model Selection

In [40]:
```
for mo in models:
    mo.fit(X_train, y_train)
    y_pred = mo.predict(X_test)

    r2_value = r2_score(y_test,y_pred)
    print(f"{mo.__class__.__name__} R2 Score: {r2_value}")
```

```
LinearRegression R2 Score: 0.751005325438475
RandomForestRegressor R2 Score: 0.23141765659557267
Lasso R2 Score: 0.7708431661652508
Ridge R2 Score: 0.7510249006566289
```

Insights:
Now our model is ready after being trained we have used 4 regressor out of which r2 lasso is giving the best result currently.So we will go with that now

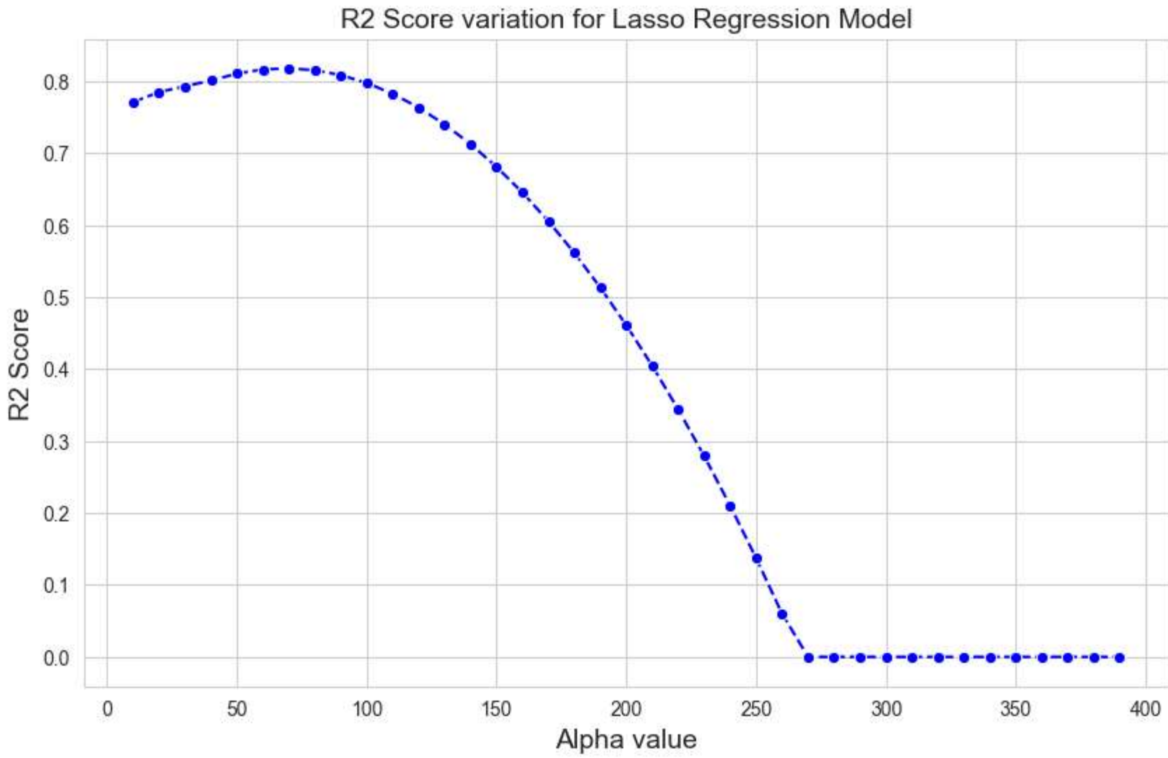
In [41]:
```
r2_lasso = []
```

In [42]:
```
for x in range(10,500,10):
    model_l = Lasso(alpha = x)
    model_l.fit(X_train, y_train)
    y_pred = model_l.predict(X_test)

    r2_value = r2_score(y_test,y_pred)
    r2_lasso.append(r2_value)
```

In [43]:
```
#Varios results based on changing the hyperparameter
r2_lasso
```

Out[43]:
```
[0.7708431661652508,
 0.7848088566806755,
 0.7929348664595779,
 0.8007956732676886,
 0.8106115280398072,
 0.8162851121013125,
 0.8178164254522047,
 0.8152054680924836,
 0.8084522400221493,
 0.797556741241202,
 0.7825189717496415,
 0.7633389315474678,
 0.7400166206346808,
 0.7125520390112807,
 0.6809451866772673,
 0.6451960636326408,
 0.6053046698774012,
 0.5612710054115484,
 0.5130950702350823,
 0.46077686434800325,
 0.40431638775031054,
 0.34371364044200525,
 0.2789686224230863,
 0.21008133369355475,
 0.1370517742534093,
 0.059879944102651295,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06,
 -4.538539260190433e-06]
```

In [44]:
```
sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))
sns.lineplot(x=list(range(10,400,10)), y=r2_lasso[:39], marker='o', color='blu
plt.title('R2 Score variation for Lasso Regression Model',fontsize=14)
plt.xlabel('Alpha value',fontsize=14)
plt.ylabel('R2 Score',fontsize=14)
plt.show()
```



In [45]:
```
max(r2_lasso)
```

Out[45]:
```
0.8178164254522047
```

Our model is completed and ready to deploy , the regressor selection can be done based on user reequirement and type of the dataset

In [ ]:
```

```