Importing the libraries

In [ ]:
```
import numpy as np
import pandas as pd
```

Read the dataset

In [ ]:
```
df=pd.read_csv('/content/train_0irEZ2H.csv')
```

Checking the dataset

In [ ]:
```
df.head()
```

Out[ ]:

	record_ID	week	store_id	sku_id	total_price	base_price	is_featured_sku	is_display_sku	units_sold
0	1	17/01/11	8091	216418	99.0375	111.8625	0	0	20
1	2	17/01/11	8091	216419	99.0375	99.0375	0	0	28
2	3	17/01/11	8091	216425	133.9500	133.9500	0	0	19
3	4	17/01/11	8091	216233	133.9500	133.9500	0	0	44
4	5	17/01/11	8091	217390	141.0750	141.0750	0	0	52

In [ ]:
```
df.tail()
```

Out[ ]:

	record_ID	week	store_id	sku_id	total_price	base_price	is_featured_sku	is_display_sku	units_sold
150145	212638	09/07/13	9984	223245	235.8375	235.8375	0	0	38
150146	212639	09/07/13	9984	223153	235.8375	235.8375	0	0	30
150147	212642	09/07/13	9984	245338	357.6750	483.7875	1	1	31
150148	212643	09/07/13	9984	547934	141.7875	191.6625	0	1	12
150149	212644	09/07/13	9984	679023	234.4125	234.4125	0	0	15

In [ ]:
```
df.shape
```

Out[ ]:
```
(150150, 9)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150150 entries, 0 to 150149
Data columns (total 9 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   record_ID        150150 non-null  int64  
 1   week             150150 non-null  object 
 2   store_id         150150 non-null  int64  
 3   sku_id           150150 non-null  int64  
 4   total_price      150149 non-null  float64
 5   base_price       150150 non-null  float64
 6   is_featured_sku  150150 non-null  int64  
 7   is_display_sku   150150 non-null  int64  
 8   units_sold       150150 non-null  int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 10.3+ MB
```

In [ ]:
```
df.describe()
```

Out[ ]:

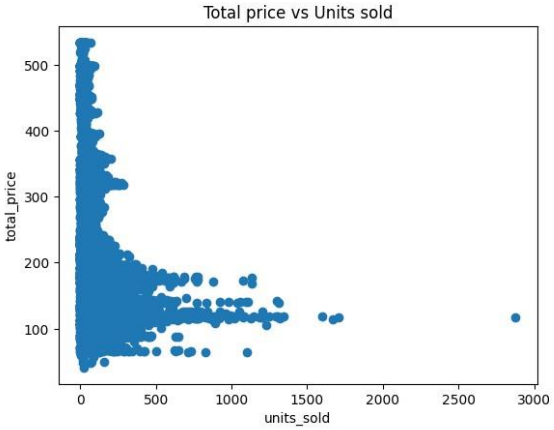
	record_ID	store_id	sku_id	total_price	base_price	is_featured_sku	is_display_sku	units_sold
count	150150.000000	150150.000000	150150.000000	150149.000000	150150.000000	150150.000000	150150.000000	150150.000000
mean	106271.555504	9199.422511	254761.132468	206.626751	219.425927	0.095611	0.133200	51.674206
std	61386.037861	615.591445	85547.306447	103.308516	110.961712	0.294058	0.339792	60.207904
min	1.000000	8023.000000	216233.000000	41.325000	61.275000	0.000000	0.000000	1.000000
25%	53111.250000	8562.000000	217217.000000	130.387500	133.237500	0.000000	0.000000	20.000000
50%	106226.500000	9371.000000	222087.000000	198.075000	205.912500	0.000000	0.000000	35.000000
75%	159452.750000	9731.000000	245338.000000	233.700000	234.412500	0.000000	0.000000	62.000000
max	212644.000000	9984.000000	679023.000000	562.162500	562.162500	1.000000	1.000000	2876.000000

Visualizing the data

***Scatter* plot**

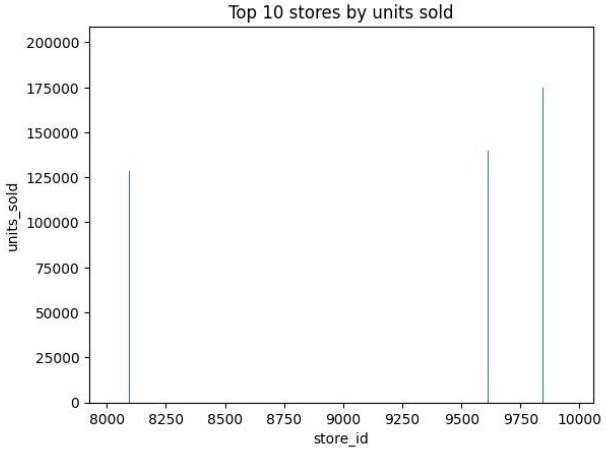
In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
plt.scatter(df['units_sold'],df['total_price'])
plt.xlabel('units_sold')
plt.ylabel('total_price')
plt.title('Total price vs Units sold')
plt.show()
```



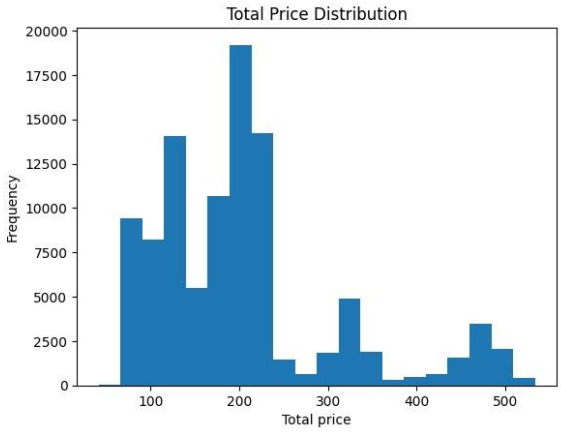
Bar plot

In [ ]:
```
top_stores=df.groupby('store_id')['units_sold'].sum().sort_values(ascending=False).head(10)
plt.bar(top_stores.index,top_stores.values)
plt.xlabel('store_id')
plt.ylabel('units_sold')
plt.title('Top 10 stores by units sold')
plt.show()
```



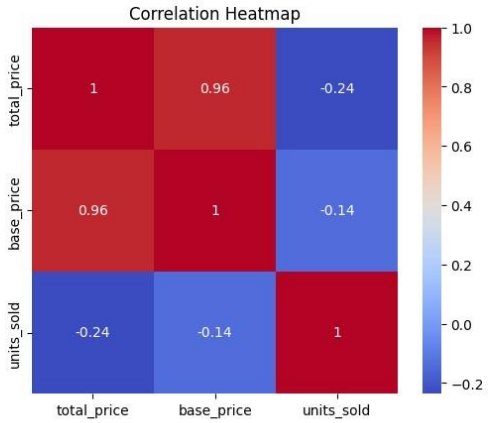
Histograms

In [ ]:
```
plt.hist(df['total_price'],bins=20)
plt.xlabel('Total price')
plt.ylabel('Frequency')
plt.title('Total Price Distribution')
plt.show()
```



Heatmaps

In [ ]:
```
corr_matrix=df[['total_price','base_price','units_sold']].corr()
sns.heatmap(corr_matrix,annot=True,cmap='coolwarm',square=True)
plt.title('Correlation Heatmap')
plt.show()
```

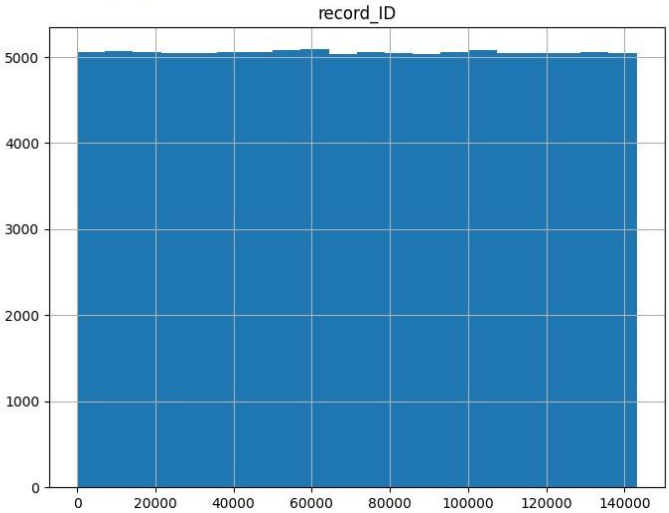


Analysing the data

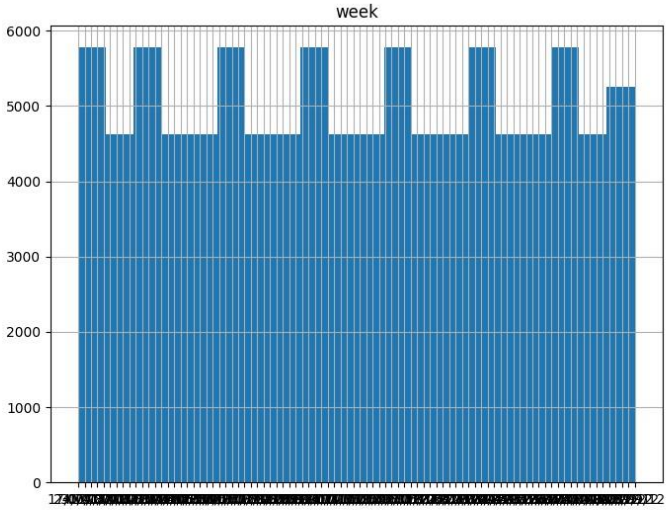
Univariate analysis

In [ ]:
```
for col in df.columns:
  print(df[col].describe())
  plt.figure(figsize=(8,6))
  df[col].hist(bins=20)
  plt.title(col)
  plt.show()
```

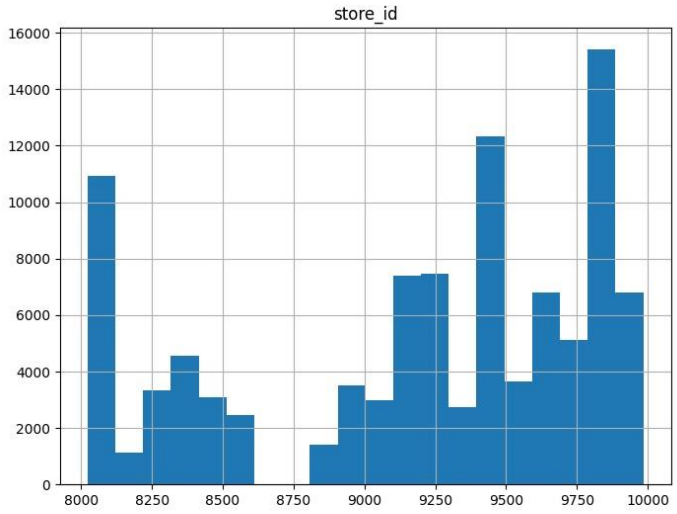
```
count    101116.000000
mean      71540.014132
std       41313.032569
min           1.000000
25%       35784.750000
50%       71508.500000
75%      107294.000000
max      143133.000000
Name: record_ID, dtype: float64
```



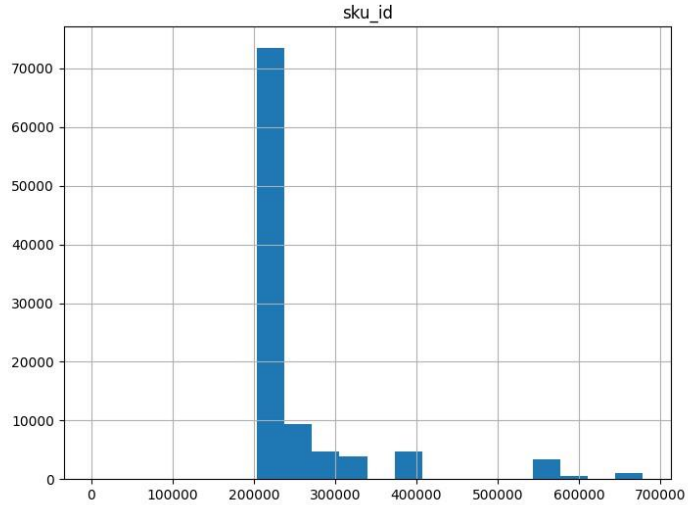

```
count       101116
unique          88
top       17/01/11
freq          1155
Name: week, dtype: object
```



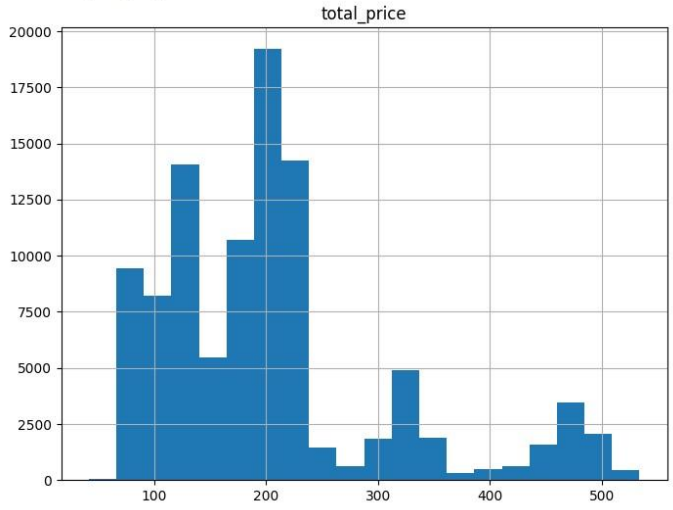

```
count    101116.000000
mean       9196.694282
std         615.893275
min        8023.000000
25%        8562.000000
50%        9371.000000
75%        9731.000000
max        9984.000000
Name: store_id, dtype: float64
```



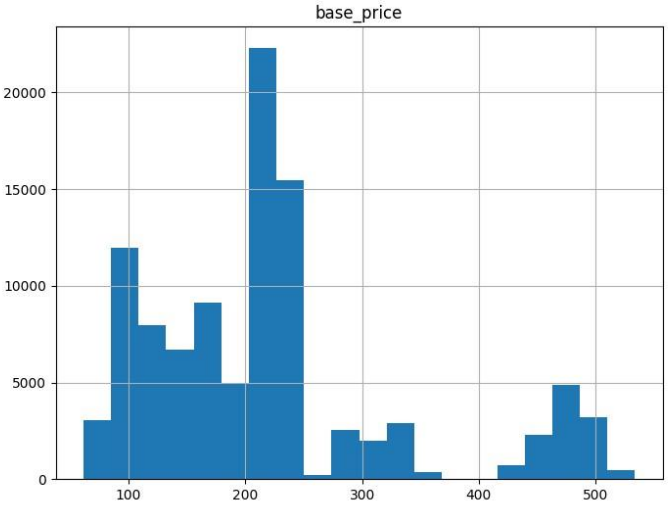

```
count    101116.000000
mean     254797.426243
std       85615.850212
min          21.000000
25%      217217.000000
50%      222087.000000
75%      245338.000000
max      679023.000000
Name: sku_id, dtype: float64
```



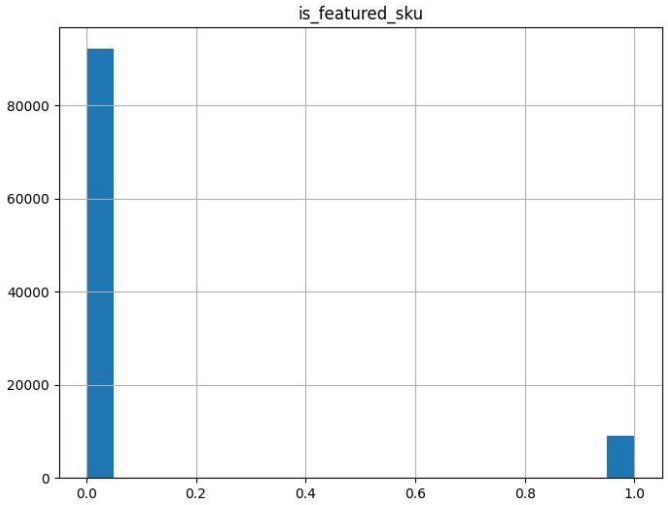

```
count    101115.000000
mean        206.218195
std         103.968070
min          41.325000
25%         131.100000
50%         196.650000
75%         231.562500
max         533.662500
Name: total_price, dtype: float64
```



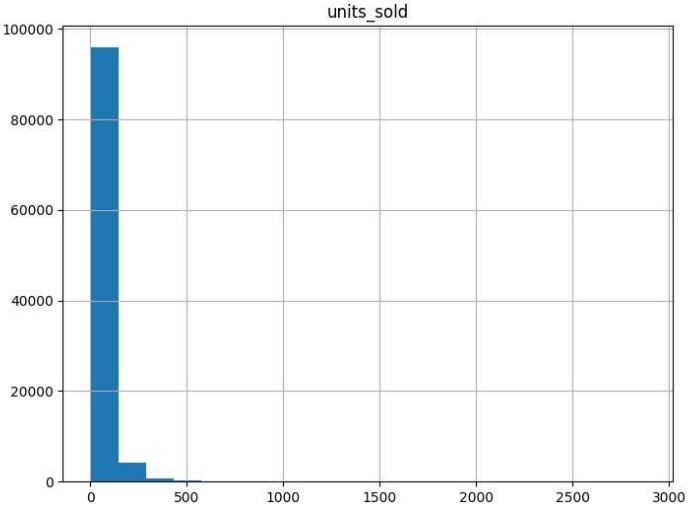

```
count    101115.000000
mean        218.512803
std         109.716634
min          61.275000
25%         133.237500
50%         205.912500
75%         234.412500
max         533.662500
Name: base_price, dtype: float64
```




```
count    101115.000000
mean          0.088800
std           0.284456
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
Name: is_featured_sku, dtype: float64
```




```
count    101115.000000
mean          0.127320
std           0.333333
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max           1.000000
Name: is_display_sku, dtype: float64
```



Multivariate Analysis

In [ ]:

```
from pandas.plotting import scatter_matrix
scatter_matrix(df[['total_price','base_price','units_sold']],figsize=(10,8))
plt.show()
```

Data preprocessing

Handling the missing values

In [ ]:
```
categorical_cols=df.select_dtypes(include=['object']).columns
print(categorical_cols)
```

```
Index(['week'], dtype='object')
```

In [ ]:
```
numerical_cols=df.select_dtypes(include=['int64','float64']).columns
print(numerical_cols)
```

```
Index(['record_ID', 'store_id', 'sku_id', 'total_price', 'base_price',
       'is_featured_sku', 'is_display_sku', 'units_sold'],
      dtype='object')
```

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
record_ID          0
week               0
store_id           0
sku_id             0
total_price        1
base_price         0
is_featured_sku    0
is_display_sku     0
units_sold         0
dtype: int64
```

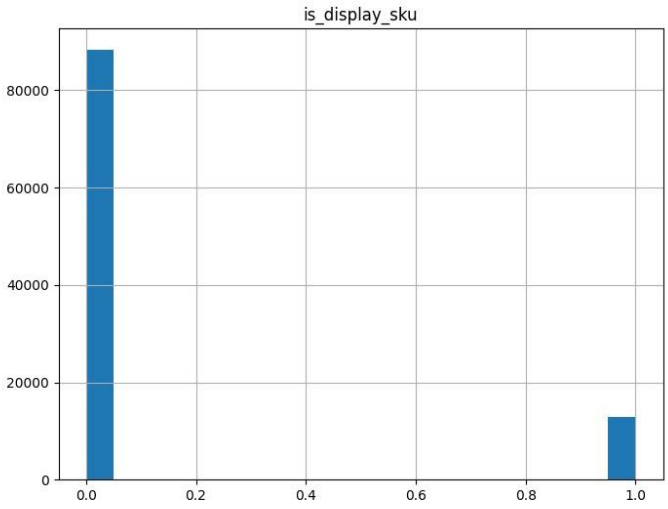
In [ ]:
```
data=df.drop(['week','total_price','is_featured_sku','is_display_sku'],axis=1)
```

In [ ]:
```
data
```

Out[ ]:

	record_ID	store_id	sku_id	base_price	units_sold
0	1	8091	216418	111.8625	20
1	2	8091	216419	99.0375	28
2	3	8091	216425	133.9500	19
3	4	8091	216233	133.9500	44
4	5	8091	217390	141.0750	52
...	...	...	...	...	...
150145	212638	9984	223245	235.8375	38
150146	212639	9984	223153	235.8375	30
150147	212642	9984	245338	483.7875	31
150148	212643	9984	547934	191.6625	12
150149	212644	9984	679023	234.4125	15

150150 rows × 5 columns




```
count    101115.000000
mean         51.917045
std          61.924794
min           1.000000
25%          20.000000
50%          35.000000
75%          62.000000
max        2876.000000
Name: units_sold, dtype: float64
```

Splitting data into train and test

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x=data[['record_ID','store_id','sku_id','base_price']]
y=data['units_sold']
```

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

Model building

Linear Regression Model

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [ ]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
pred=lr.predict(x_test)
print("Mean Squared Error:",mean_squared_error(y_test,pred))
print("R2 Score:",r2_score(y_test,pred))
```

```
Mean Squared Error: 3103.1409991414653
R2 Score: 0.05424514437673167
```

In [ ]:
```
pred=lr.predict([[1,8091,216418,111.625]])
pred
```

Random Forest Regressor

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,r2_score
```

In [ ]:
```
model=RandomForestRegressor()
```

In [ ]:
```
model.fit(x_train,y_train)
pred=model.predict(x_test)
```

In [ ]:
```
print("Mean Squared Error:",mean_squared_error(y_test,pred))
print("R2 Score:",r2_score(y_test,pred))
```

```
Mean Squared Error: 892.5601685747586
R2 Score: 0.7279713962082139
```

In [ ]:
```
pred=model.predict([[1,8091,216418,111.8625]])
pred
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomFore
stRegressor was fitted with feature names
  warnings.warn(
```

Out[ ]:
```
array([77.96])
```

Decision Tree Regressor

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
pred=dt.predict(x_test)
print("Mean Squared Error:",mean_squared_error(y_test,pred))
print("R2 Score:",r2_score(y_test,pred))
print(accuracy_score(pred,y_test))
```

```
Mean Squared Error: 1610.1784215784216
R2 Score: 0.5092604361036386
0.031302031302031304
```

In [ ]:
```
y_p=dt.predict([[1,8091,216418,111.8625]])
y_p
```

In [ ]:
```
#save the model
import pickle
filename='stock.pkl'
pickle.dump(model,open(filename,'wb'))
```

In [ ]:
```
#rf_regressor
model.fit(x_train,y_train)
features=np.array([[1,8091,216418,111.8625]])
print(model.predict(features))
```

save the model with joblib

In [ ]:
```
import joblib
joblib.dump(model,'stock.pkl')
```

Load the model with joblib

In [ ]:
```
import joblib
model=joblib.load('stock.pkl')
```

Inspect the pickle file

In [ ]:
```
import sklearn
print(sklearn.__version__)
```

Rebuild the Model

In [ ]:
```
import joblib
joblib.dump(model,'stock.pkl')
```

Load the retrained model

In [ ]:
```
import joblib
rf_regressor=joblib.load('stock.pkl')
```