Heat Exchanger Price Prediction Model

Import Libraries

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

Import dataset

In [2]:
```
data = pd.read_excel('D:/Data Science/Capstone Project/ML/TTR/dataset1.xlsx')
```

In [3]:
```
data.shape
```

Out[3]: (1364, 13)

In [4]:
```
data.head()
```

Out[4]:

	TTR part num	Part Level	Customer	Segment	End Customer	Category	Description	Core Thickness	Co Wi
0	B1DQA5001M00	ECS	Cummins	Gen Set	OEM	Domestic	QSL-9	102.0	56
1	A5SMA5001M00	ECS	Ashok Leyland	Commercial Vehicle	OEM	Domestic	Chassis Mounted Radiator	NaN	N
2	B1DQA5050M00	ECS	Cummins	Gen Set	OEM	Domestic	QSL-9	102.0	56
3	A5RZA5001M00	ECS	Ashok Leyland	Commercial Vehicle	OEM	Domestic	Roof Mounted radiator	48.0	55
4	A0VDA5001M00	ECS	Tata Motors	Commercial Vehicle	OEM	Domestic	LPO 1515-62-WB	36.0	53

Check missing values and Data types

In [427]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1364 entries, 0 to 1363
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   TTR part num    1364 non-null   object 
 1   Part Level      1364 non-null   object 
 2   Customer        1364 non-null   object 
 3   Segment         1364 non-null   object 
 4   End Customer    1364 non-null   object 
 5   Category        1364 non-null   object 
 6   Description     1364 non-null   object 
 7   Core Thickness  1104 non-null   float64
 8   Core Width      1104 non-null   float64
 9   Core Height     1104 non-null   float64
 10  Tube per core   1104 non-null   float64
 11  Tank            1364 non-null   object 
 12  Price           1364 non-null   float64
dtypes: float64(5), object(8)
memory usage: 138.7+ KB
```

Data types are okay so, no conversion required

In [428]:
```
data.isnull().sum()
```

Out[428]:
```
TTR part num        0
Part Level          0
Customer            0
Segment             0
End Customer        0
Category            0
Description         0
Core Thickness    260
Core Width        260
Core Height       260
Tube per core     260
Tank                0
Price               0
dtype: int64
```

In [429]:
```
data.shape
```

Out[429]:
```
(1364, 13)
```

Lets treat the null values

In [430]:
```
data.isnull().sum()
```

Out[430]:
```
TTR part num        0
Part Level          0
Customer            0
Segment             0
End Customer        0
Category            0
Description         0
Core Thickness    260
Core Width        260
Core Height       260
Tube per core     260
Tank                0
Price               0
dtype: int64
```

We need to drop the null values since it invloves the technical features and replacing it with any metric

would hamper the accuracy of the data.

In [431]:
```
data_new = data.dropna()
```

In [432]:
```
data_new.isnull().sum()
```

Out[432]:
```
TTR part num      0
Part Level        0
Customer          0
Segment           0
End Customer      0
Category          0
Description       0
Core Thickness    0
Core Width        0
Core Height       0
Tube per core     0
Tank              0
Price             0
dtype: int64
```

In [433]:
```
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1104 entries, 0 to 1363
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   TTR part num    1104 non-null   object 
 1   Part Level      1104 non-null   object 
 2   Customer        1104 non-null   object 
 3   Segment         1104 non-null   object 
 4   End Customer    1104 non-null   object 
 5   Category        1104 non-null   object 
 6   Description     1104 non-null   object 
 7   Core Thickness  1104 non-null   float64
 8   Core Width      1104 non-null   float64
 9   Core Height     1104 non-null   float64
 10  Tube per core   1104 non-null   float64
 11  Tank            1104 non-null   object 
 12  Price           1104 non-null   float64
dtypes: float64(5), object(8)
memory usage: 120.8+ KB
```

In [434]:
```
data_new.describe()
```

Out[434]:

	Core Thickness	Core Width	Core Height	Tube per core	Price
count	1104.000000	1104.000000	1104.000000	1104.000000	1104.000000
mean	42.706341	509.052717	611.947011	99.005435	6676.901701
std	11.742686	112.817412	154.468325	36.569733	5801.950447
min	12.000000	74.800000	198.000000	5.000000	0.000000
25%	36.000000	486.700000	500.000000	80.000000	2841.520000
50%	48.000000	537.200000	590.000000	108.000000	5062.691154
75%	48.000000	552.600000	740.000000	124.000000	9755.285000
max	126.000000	834.900000	1100.000000	305.000000	50867.000000

Lets check what product range we are dealing with

In [435]:
```
pd.unique(data_new['Part Level'].values)
```

Out[435]:
```
array(['ECS', 'Radiator', 'Core', 'CRFMS', 'RFMS', 'Intercooler'],
      dtype=object)
```

The data consists total 6 products ECS (Engine Cooling System), Raditor, Core (Sub-assembly), CRFMS (Condenser Fan Motor Shroud), RFMS (Radiator Fan Motor Shroud) & Intercooler (Charged Air Cooler)

Lets check the customer portfolio of the organisation

In [436]:
```
pd.unique(data_new['Customer'].values)
```

Out[436]:
```
array(['Cummins', 'Ashok Leyland', 'Tata Motors', 'Kirloskar', 'MWM',
       'Mahindra Powerol', 'Greaves', 'JBM', 'TMTL', 'Volvo Eicher',
       'Mahindra Tractors', 'Volvo', 'Kubota', 'Kamaz Vectra',
       'Force Motors', 'MTBD', 'Daimler', 'PSA', 'General Motors',
       'John Deere', 'Swaraj', 'M&M', 'New Holland', 'Sonalika', 'Honda',
       'Isuzu', 'CNH', 'Tafe', 'TAFE', 'Piaggio', 'Renault'], dtype=object)
```

Lets know check the segment and category coloumn

In [437]:
```
pd.unique(data_new['Segment'].values)
```

Out[437]:
```
array(['Gen Set', 'Commercial Vehicle', 'Tractor', 'Passenger Car',
       '3 Wheeler'], dtype=object)
```

The data consists of 5 segments

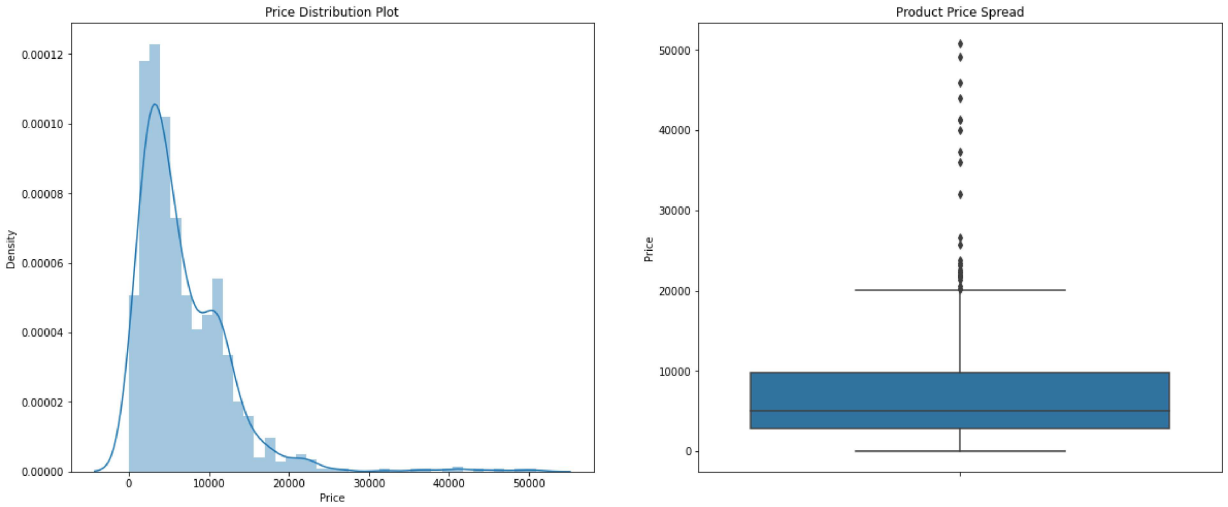
In [438]:
```
pd.unique(data_new['Category'].values)
```

Out[438]:
```
array(['Domestic', 'Export'], dtype=object)
```

Check for skewness and outliers in data.

In [439]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Price Distribution Plot')
sns.distplot(data_new.Price)

plt.subplot(1,2,2)
plt.title('Product Price Spread')
sns.boxplot(y=data_new.Price)

plt.show()
```

```
D:\Data Science\Anaconda\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [440]:
```
data_new.Price.describe()
```

Out[440]:
```
count     1104.000000
mean      6676.901701
std       5801.950447
min          0.000000
25%       2841.520000
50%       5062.691154
75%       9755.285000
max      50867.000000
Name: Price, dtype: float64
```

The plot seemed to be right-skewed, meaning that the most prices in the dataset are low.

There is a significant difference between the mean and the median of the price distribution.

The data points are far spread out from the mean, which indicates a high variance in the products prices.

75% of the prices are below Rs.9895.54 remaining 25% are between Rs.9,895.54 and Rs.50,867.

There are values observed in the price distribution spread box/whiskers plot, in which there are value above Rs. 20,000. As per marketing team these values are not outliers since, the price of higher cooling capacity products will be high.

Need to convert the columns name for ease of syntax

In [441]:
```
# change the column names
data_new.rename(index=str, columns={'TTR part num': 'partcode',
                                    'Part Level' : 'part_level',
                                    'Customer' : 'customer',
                                    'Segment' : 'segment',
                                    'End Customer' : 'end_customer',
                                    'Category' : 'category',
                                    'Description' : 'description',
                                    'Core Thickness' : 'core_thickness',
                                      'Core Width' : 'core_width',
                                    'Core Height' : 'core_height',
                                    'Fin Type' : 'fin_type',
                                    'Tube per core' : 'tube_per_core',
                                      'Tank' : 'tank',
                                    'Price' : 'price'}, inplace=True)
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1104 entries, 0 to 1363
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   partcode        1104 non-null   object 
 1   part_level      1104 non-null   object 
 2   customer        1104 non-null   object 
 3   segment         1104 non-null   object 
 4   end_customer    1104 non-null   object 
 5   category        1104 non-null   object 
 6   description     1104 non-null   object 
 7   core_thickness  1104 non-null   float64
 8   core_width      1104 non-null   float64
 9   core_height     1104 non-null   float64
 10  tube_per_core   1104 non-null   float64
 11  tank            1104 non-null   object 
 12  price           1104 non-null   float64
dtypes: float64(5), object(8)
memory usage: 120.8+ KB

D:\Data Science\Anaconda\lib\site-packages\pandas\core\frame.py:4296: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  return super().rename(
```

Since partcode and description are unique values lets drop them.

In [442]:
```
del data_new['partcode']
```

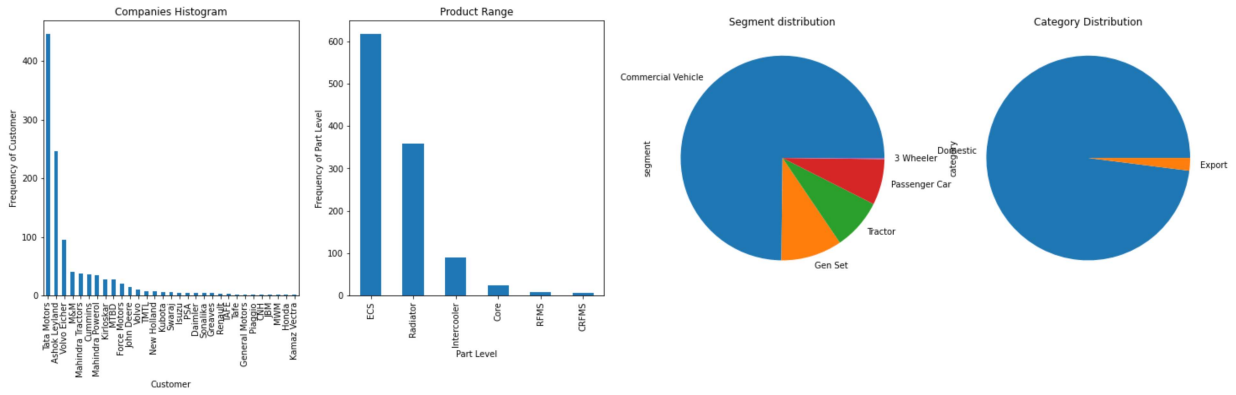
In [443]:
```
del data_new['description']
```

In [444]:
```
data_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1104 entries, 0 to 1363
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   part_level      1104 non-null   object 
 1   customer        1104 non-null   object 
 2   segment         1104 non-null   object 
 3   end_customer    1104 non-null   object 
 4   category        1104 non-null   object 
 5   core_thickness  1104 non-null   float64
 6   core_width      1104 non-null   float64
 7   core_height     1104 non-null   float64
 8   tube_per_core   1104 non-null   float64
 9   tank            1104 non-null   object 
 10  price           1104 non-null   float64
dtypes: float64(5), object(6)
memory usage: 103.5+ KB
```

Exploration of the categorical features of the product.

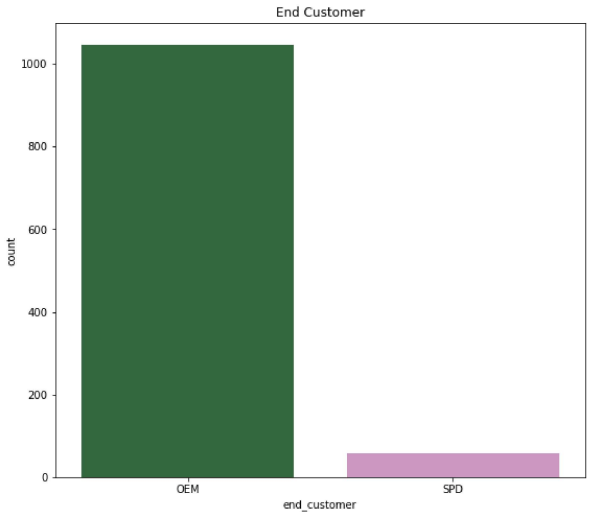
In [445]:
```
plt.figure(figsize=(25, 6))

plt.subplot(1,4,1)
plt1 = data_new.customer.value_counts().plot(kind='bar')
plt.title('Companies Histogram')
plt1.set(xlabel = 'Customer', ylabel='Frequency of Customer')

plt.subplot(1,4,2)
plt1 = data_new.part_level.value_counts().plot(kind='bar')
plt.title('Product Range')
plt1.set(xlabel = 'Part Level', ylabel='Frequency of Part Level')

plt.subplot(1,4,3)
plt1 = data_new.segment.value_counts().plot(kind='pie')
plt.title('Segment distribution')

plt.subplot(1,4,4)
plt1 = data_new.category.value_counts().plot(kind='pie')
plt.title('Category Distribution')

plt.show()
```

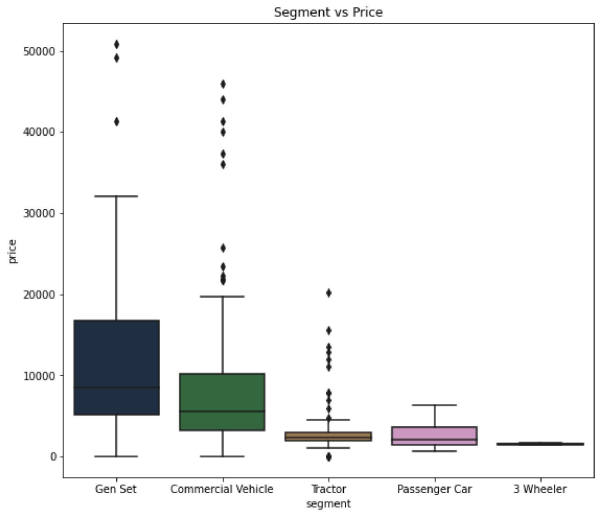


1. Most of the RFQ's recieved are from Tata Motors.
2. Most sold product range is Engine Cooling System whereas least sold product range Condenser Radiator Fan Motor Shroud.
3. Maximum RFQ's belong to Commercial vehicle segment.
4. It seems the organisation has dominance in domestic market along with international presence.

In [446]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('End Customer')
sns.countplot(data_new.end_customer, palette=("cubehelix"))

plt.subplot(1,2,2)
plt.title('Segment vs Price')
sns.boxplot(x=data_new.segment, y=data_new.price, palette=("cubehelix"))

plt.show()
```

```
D:\Data Science\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

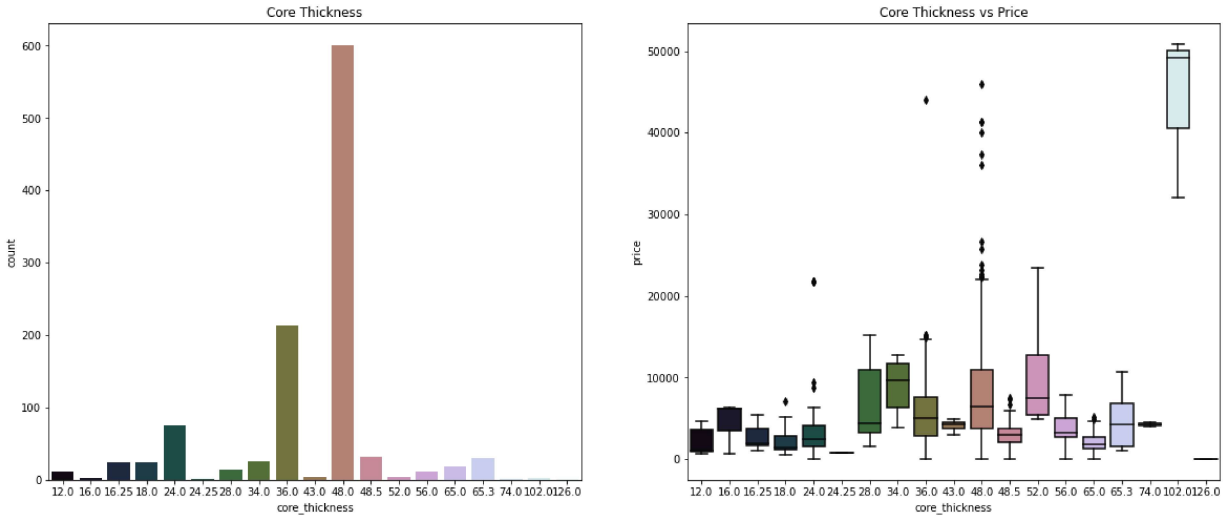




1. Share of business in OEM (Orignal Equipment Manufacturer) is maximum.
2. It seems OEM are managing their SPD (Spare Parts Division) by buying with extra schedule.
3. Gen set segment is highly priced segment with second highest market share in product range.
4. Second expensive segment is commercial vehicle with highest market share within the organisations product range.

In [447]:
```
plt.figure(figsize=(20,8))

plt.subplot(1,2,1)
plt.title('Core Thickness')
sns.countplot(data_new.core_thickness, palette=("cubehelix"))

plt.subplot(1,2,2)
plt.title('Core Thickness vs Price')
sns.boxplot(x=data_new.core_thickness, y=data_new.price, palette=("cubehelix"))

plt.show()
```

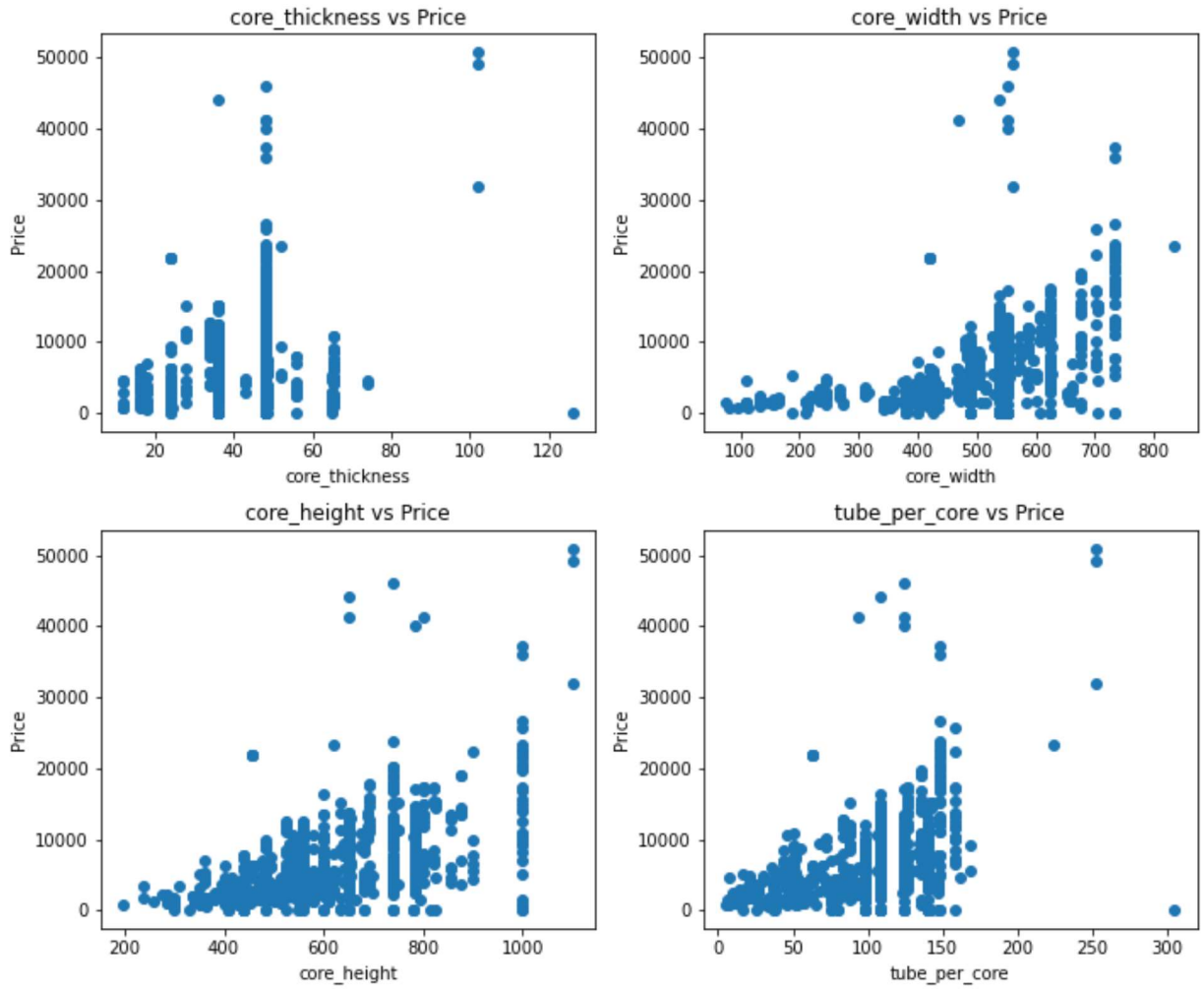
```
D:\Data Science\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



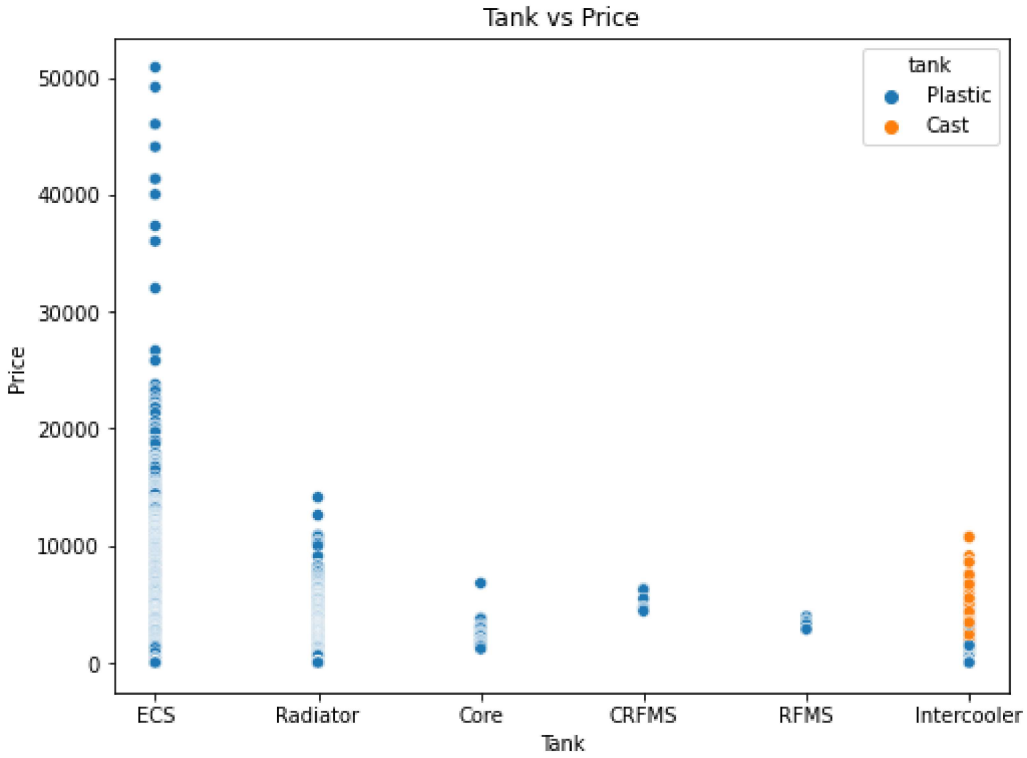
1. Count of products with core_thickness is high for 48mm and second highest is 36mm.
2. Price range of the product for 48mm thickness core is also wide with higher count of products.
3. Most expensive core_thickness is 102mm but count of the products is quite less.

Lets check the relation of technical features of the product versus the price.

In [448]:
```
def scatter(x,fig):
    plt.subplot(5,2,fig)
    plt.scatter(data_new[x],data_new['price'])
    plt.title(x+' vs Price')
    plt.ylabel('Price')
    plt.xlabel(x)

plt.figure(figsize=(10,20))

scatter('core_thickness', 1)
scatter('core_width', 2)
scatter('core_height', 3)
scatter('tube_per_core', 4)

plt.tight_layout()
```



In [449]:
```
plt.figure(figsize=(8,6))

plt.title('Tank vs Price')
sns.scatterplot(x=data_new['part_level'],y=data_new['price'],hue=data_new['tank']
plt.xlabel('Tank')
plt.ylabel('Price')

plt.show()
plt.tight_layout()
```

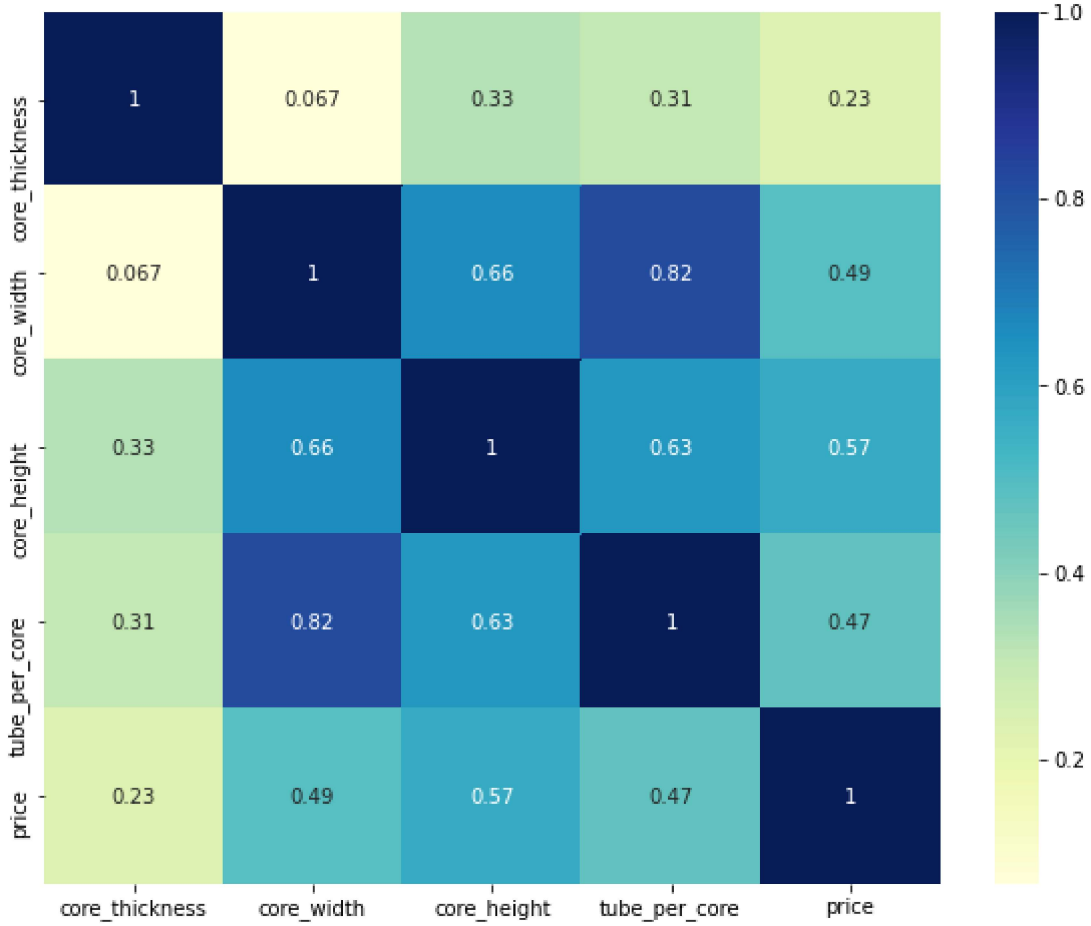



```
<Figure size 432x288 with 0 Axes>
```

It seems Casting tanks are only used in a particular product and that is intercooler.

Also, we cannot relate the tank type and price of the product. Going ahead we shall drop the column tank.

In [450]:
```
plt.figure(figsize = (10, 8))
sns.heatmap(data_new.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



There is high correlation between Core_width and tubes_per_core so, we will drop one of them to reduce the dimensions.

This would help in tackling the highly corelated features.

Going further categorical data needs to be converted into numeric.

But before that, we need to check the feature importance.

In [451]:
```
data1 = data_new[['part_level', 'segment','end_customer', 'category','core_thickn
                  'core_width', 'core_height','tube_per_core', 'tank','price']]
data1.head()
```

Out[451]:

	part_level	segment	end_customer	category	core_thickness	core_width	core_height	tube_
0	ECS	Gen Set	OEM	Domestic	102.0	561.4	1100.0	
2	ECS	Gen Set	OEM	Domestic	102.0	561.4	1100.0	
3	ECS	Commercial Vehicle	OEM	Domestic	48.0	552.6	740.0	
4	ECS	Commercial Vehicle	OEM	Domestic	36.0	537.2	650.0	
5	ECS	Commercial Vehicle	OEM	Domestic	48.0	552.6	650.0	

In [452]:
```
data1.columns
```

Out[452]:
```
Index(['part_level', 'segment', 'end_customer', 'category', 'core_thickness',
       'core_width', 'core_height', 'tube_per_core', 'tank', 'price'],
      dtype='object')
```

Now before passing the data through our selected, lets create the dummies and convert all the categorical values into numerical ones

In [453]:
```
def dummies(x,df):
    temp = pd.get_dummies(df[x], drop_first = True)
    df = pd.concat([df, temp], axis = 1)
    df.drop([x], axis = 1, inplace = True)
    return df

data1 = dummies('part_level',data1)
data1 = dummies('segment',data1)
data1 = dummies('end_customer',data1)
data1 = dummies('category',data1)
```

In [454]:
```
data1.head()
```

Out[454]:

	core_thickness	core_width	core_height	tube_per_core	tank	price	Core	ECS	Intercoole
0	102.0	561.4	1100.0	252.0	Plastic	50867.00	0	1	
2	102.0	561.4	1100.0	252.0	Plastic	49170.00	0	1	
3	48.0	552.6	740.0	124.0	Plastic	46000.00	0	1	
4	36.0	537.2	650.0	108.0	Plastic	44066.00	0	1	
5	48.0	552.6	650.0	124.0	Plastic	41342.77	0	1	

In [455]:
```
data1.shape
```

Out[455]: (1104, 17)

Now lets drop the target variable and other feature which had no significance with price variation

In [456]:
```
X = data1.drop('price', axis=1)
X = data1.drop('tank', axis=1)
X.head()
```

Out[456]:

	core_thickness	core_width	core_height	tube_per_core	price	Core	ECS	Intercooler	RFM
0	102.0	561.4	1100.0	252.0	50867.00	0	1	0	
2	102.0	561.4	1100.0	252.0	49170.00	0	1	0	
3	48.0	552.6	740.0	124.0	46000.00	0	1	0	
4	36.0	537.2	650.0	108.0	44066.00	0	1	0	
5	48.0	552.6	650.0	124.0	41342.77	0	1	0	

In [457]:
```
y= data1[['price']]
y.head()
```

Out[457]:

	price
0	50867.00
2	49170.00
3	46000.00
4	44066.00
5	41342.77

Spliting the data before passing it through the algorithm.

In [458]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

Now lets apply Random forest model

In [459]:
```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=1000, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

```
<ipython-input-459-e6157cc909fb>:4: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_sampl
es,), for example using ravel().
  regressor.fit(X_train, y_train)
```

Lets evaluate the model using MAE and R2 score.

In [460]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('R2 Score:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 62.313689639383625
R2 Score: 0.9962964095953447
```

As per the evaluation metric the R2 score is close to 1 which is good

Mean Absolute error is 62.13

Lets check whether this error can be reudeced further

Applying standardization technique on the target variable

In [461]:
```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

Splitting stadardized data once again

In [462]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

In [463]:
```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=1000, random_state=0)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

```
<ipython-input-463-2ddd8a3b7ca5>:4: DataConversionWarning: A column-vector y wa
s passed when a 1d array was expected. Please change the shape of y to (n_sampl
es,), for example using ravel().
  rf.fit(X_train, y_train)
```

In [464]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('R2 Score:', metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error: 0.010766650934735467
R2 Score: 0.9962599276175893
```

As per the evaluation the R2 score is good and the error obtained has also reduced which we have achieved using standardization.