Table of Content

1.Problem Statement:

It is a regression problem where we have to predict the selling price of the steel items.

2. Data Cleaning:

In this step we look for missing data and convert the data in an actionable format.

- First Step is to Find Missing Values in the Dataset (Yes ! There are missing values in the dataset).
- "application" and "country" columns missing values are belongs to one customer i.e(2.147484e+09)
- "Material_ref" column has 30% missing values. So, I decided to drop the column and "id" column also.
- Dropped all the missing values from the dataset
- Removed **'e'** value from "quantity tons". Because, it is an irrelevant value compared to the rest of the values.
- Removed “**19950000 & 20191919”** values from the "item_date" column which are not matching with datetime data type
- Removed **“30310101 & 20212222”** values from the "delivery date" column.which are out of datetime value.
- Remove **"selling_price==0"** rows.
- In "selling_price” & “quantity tons” columns have negative values. So, converted into positive values.
- Changed data types of few columns:

    1. changed "customer" column data type into "int64"
    2. changed "quantity tons" data type into "float64"
    3. changed both "item_date" & "delivery date" columns into datetime

3. Creating New Features:

- created **"amount_spent"** column with quantity tons & selling_price
- created a **"delivery_days"** column. Which took to deliver the steel item.
- created the columns - “month & weekday” for the item_date column. For better understanding the dataset pattern.

4. Data Visualization & Mining :

1. Top 10 customers with most no.of steel purchase

	customer	country	count
60	30157111	78.0	4987
89	30161088	78.0	3733
252	30201846	25.0	3152
134	30165529	78.0	2728
288	30202938	25.0	2570
312	30205312	32.0	2522
34	30153510	30.0	2510
84	30160378	78.0	2256
322	30205658	32.0	2151
76	30160005	78.0	2132

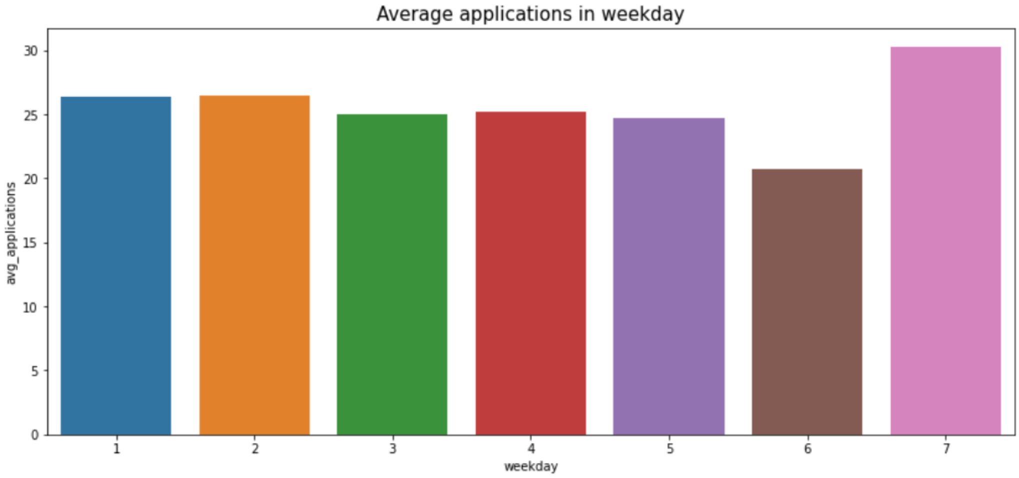
2. Top 10 customers with highest money spent

	customer	country	amount_spent
325	30205728	30.0	5.830132e+11
224	30200964	25.0	4.431428e+10
974	30353306	107.0	4.381604e+09
417	30217607	27.0	9.239216e+08
195	30198507	26.0	3.126882e+08
60	30157111	78.0	2.956766e+08
277	30202645	32.0	2.955646e+08
459	30223403	78.0	2.531759e+08
1058	30394817	78.0	2.507976e+08
697	30287258	27.0	2.437241e+08

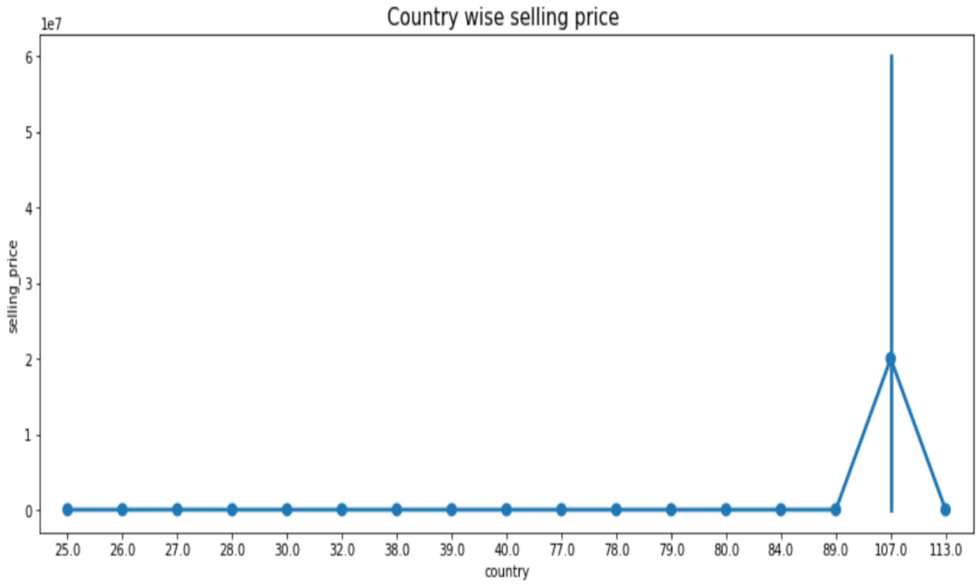
3. Top 5 highest average tons

	customer	country	quantity tons
325	30205728	30.0	4.717074e+06
224	30200964	25.0	3.819233e+05
521	30235913	78.0	4.337218e+03
831	30333845	78.0	4.137256e+03
1061	30395031	78.0	3.607611e+03

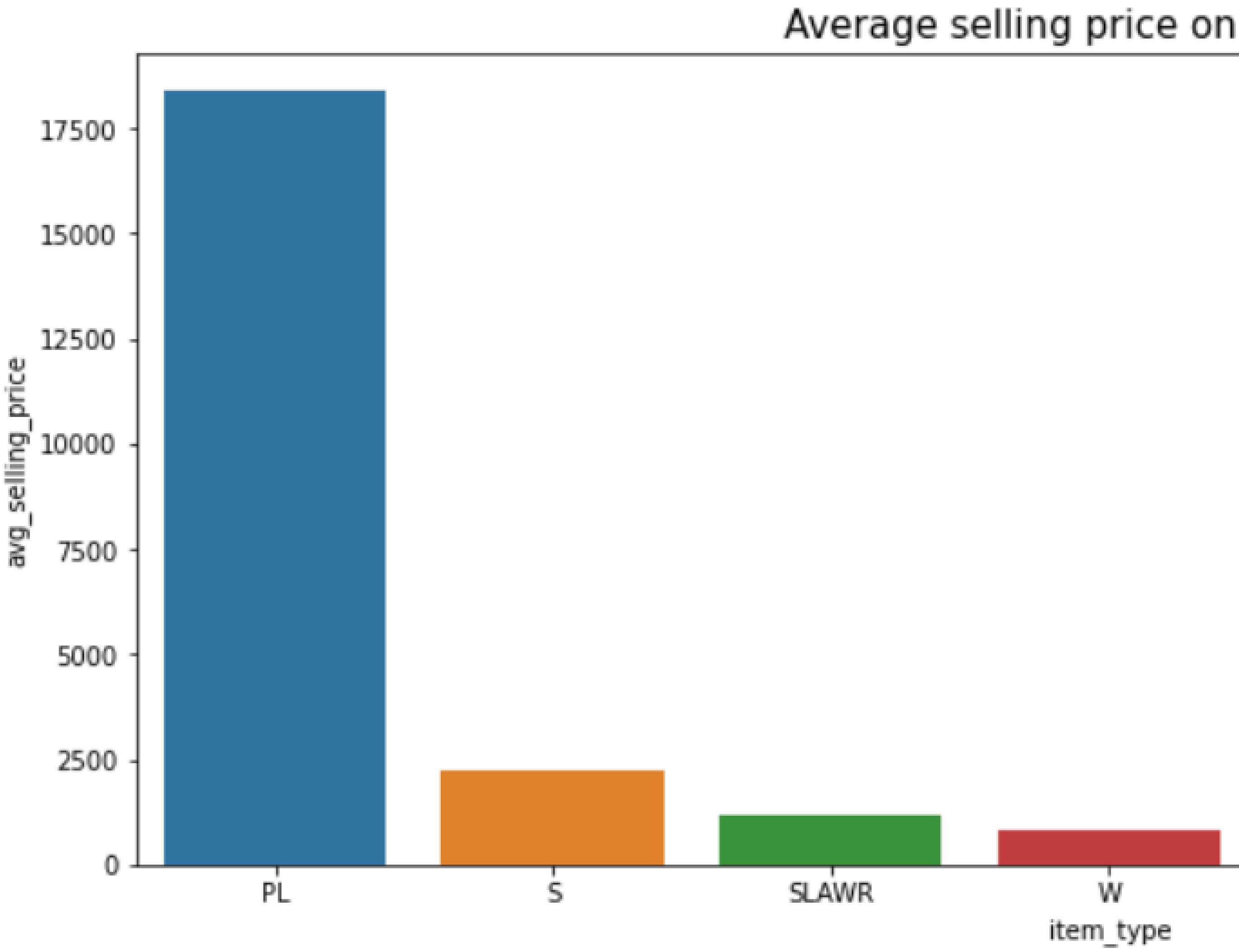
4. Average Applications in weekdays (Monday=1, ……sunday =7)



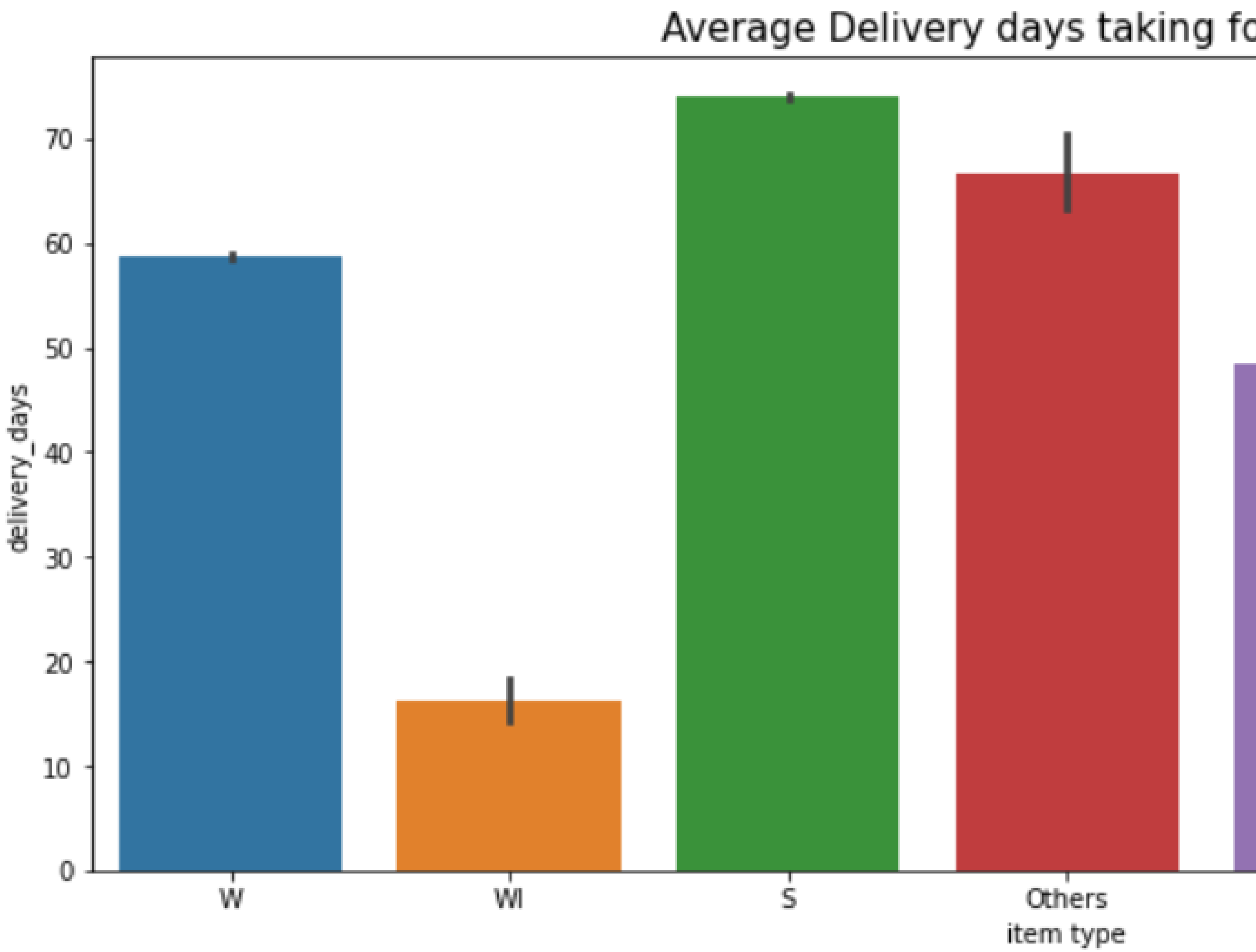
5. Country wise selling price pattern



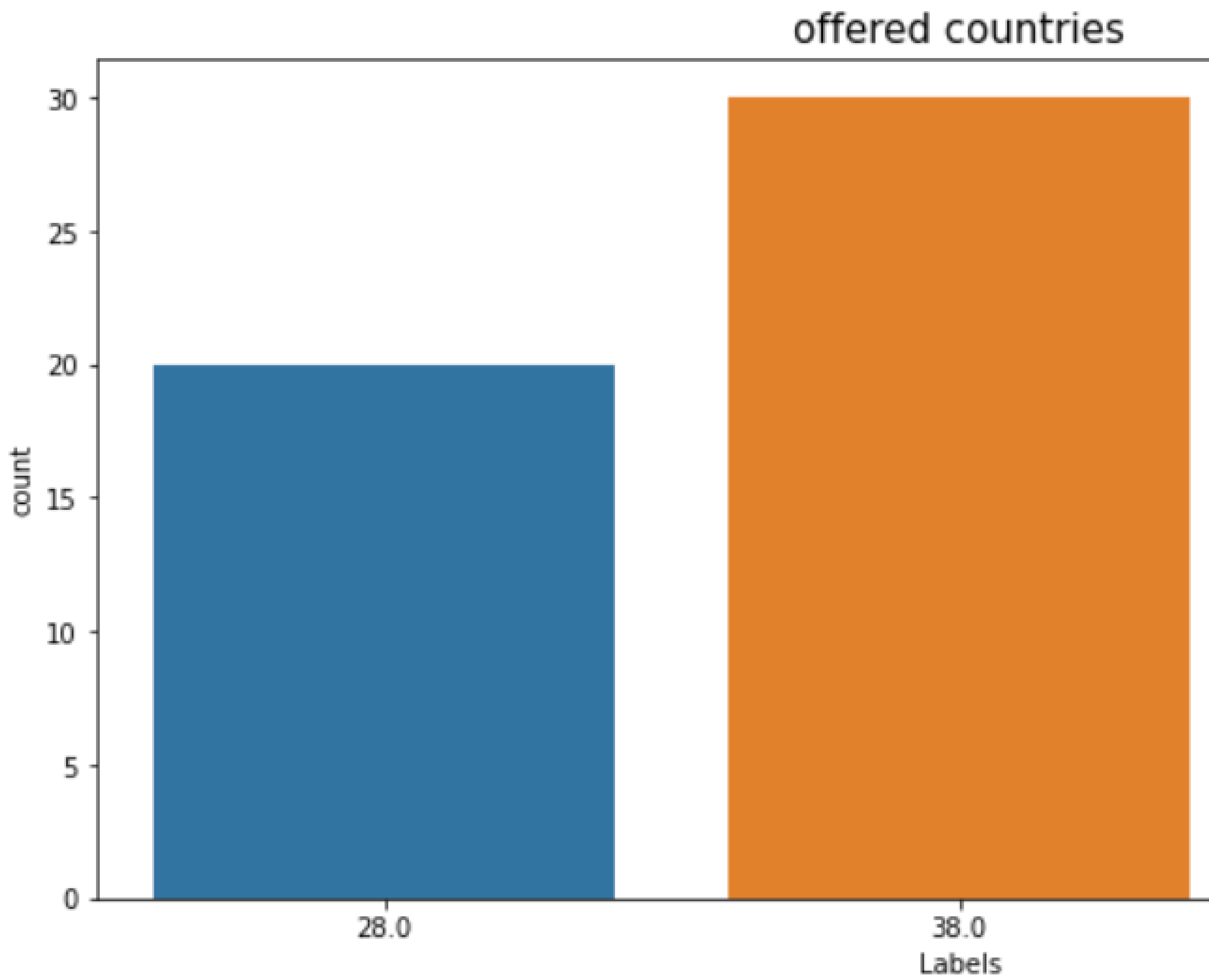
6. Average Selling price on item type



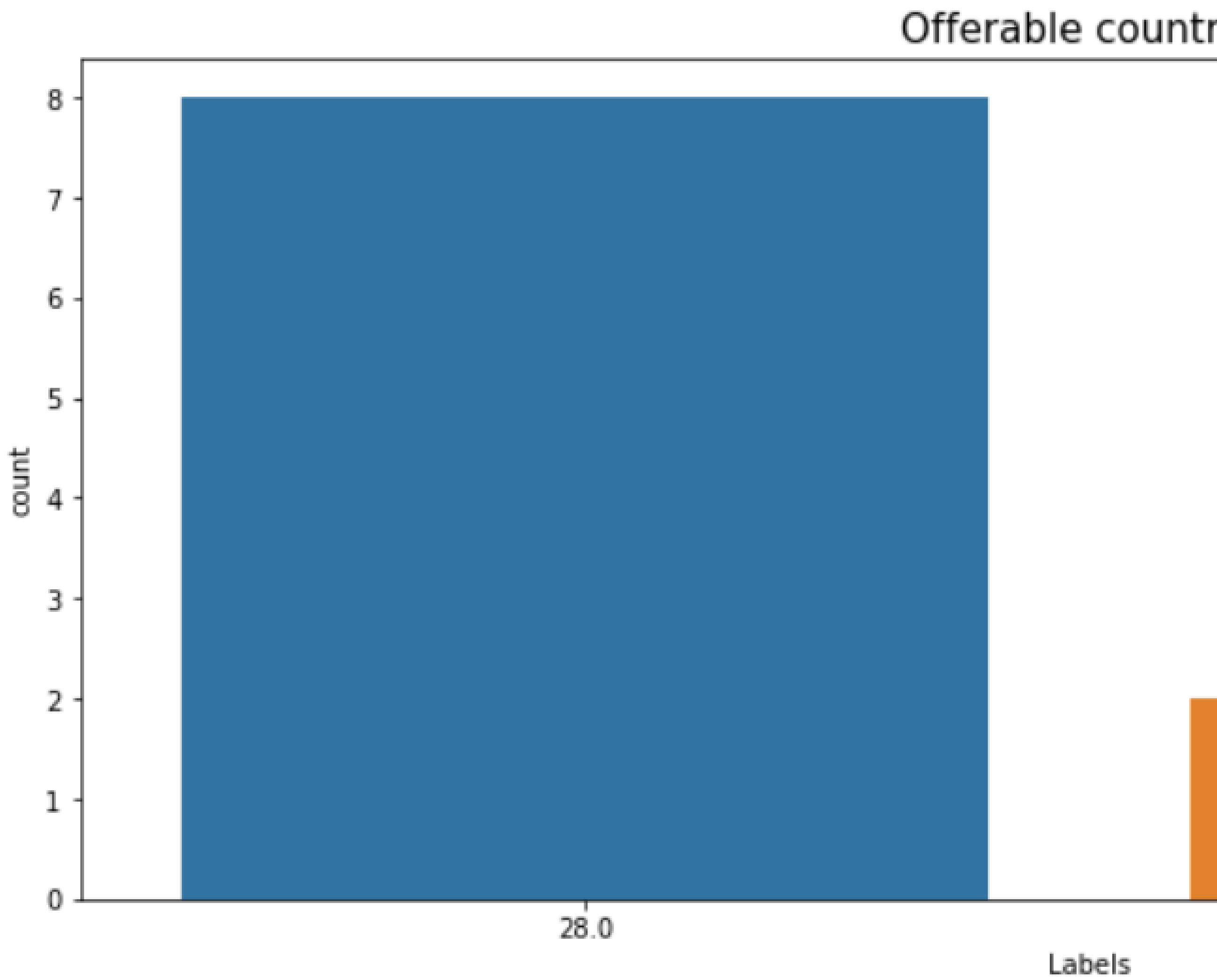
7. Average Delivery days taking for item type



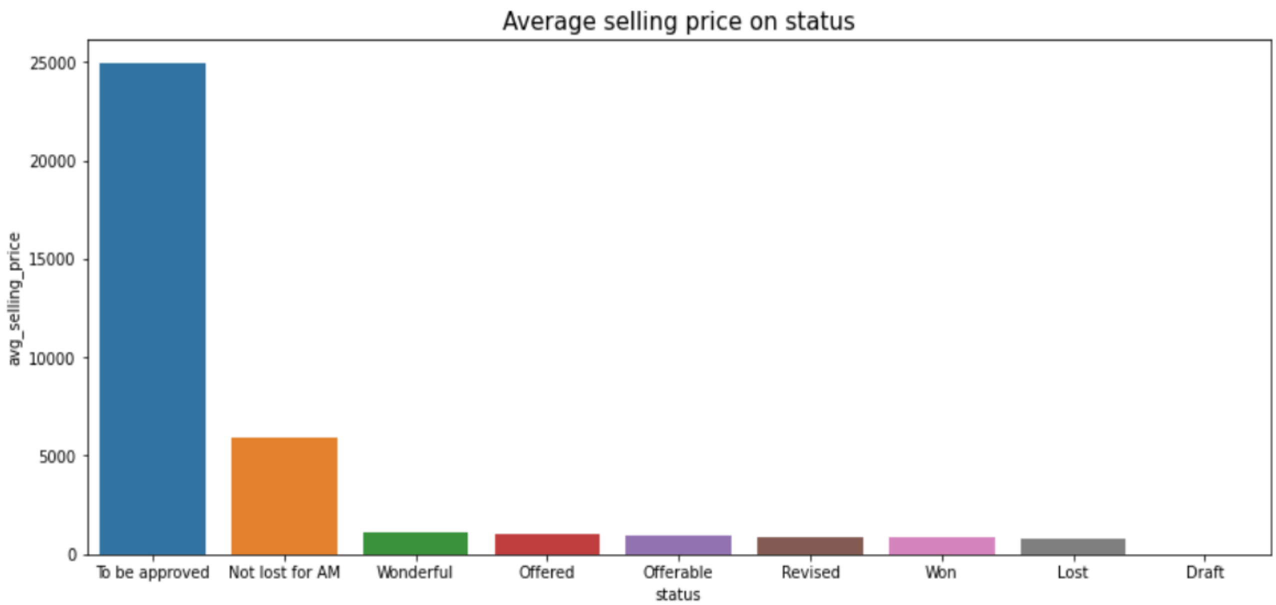
8. Status “offered” countries



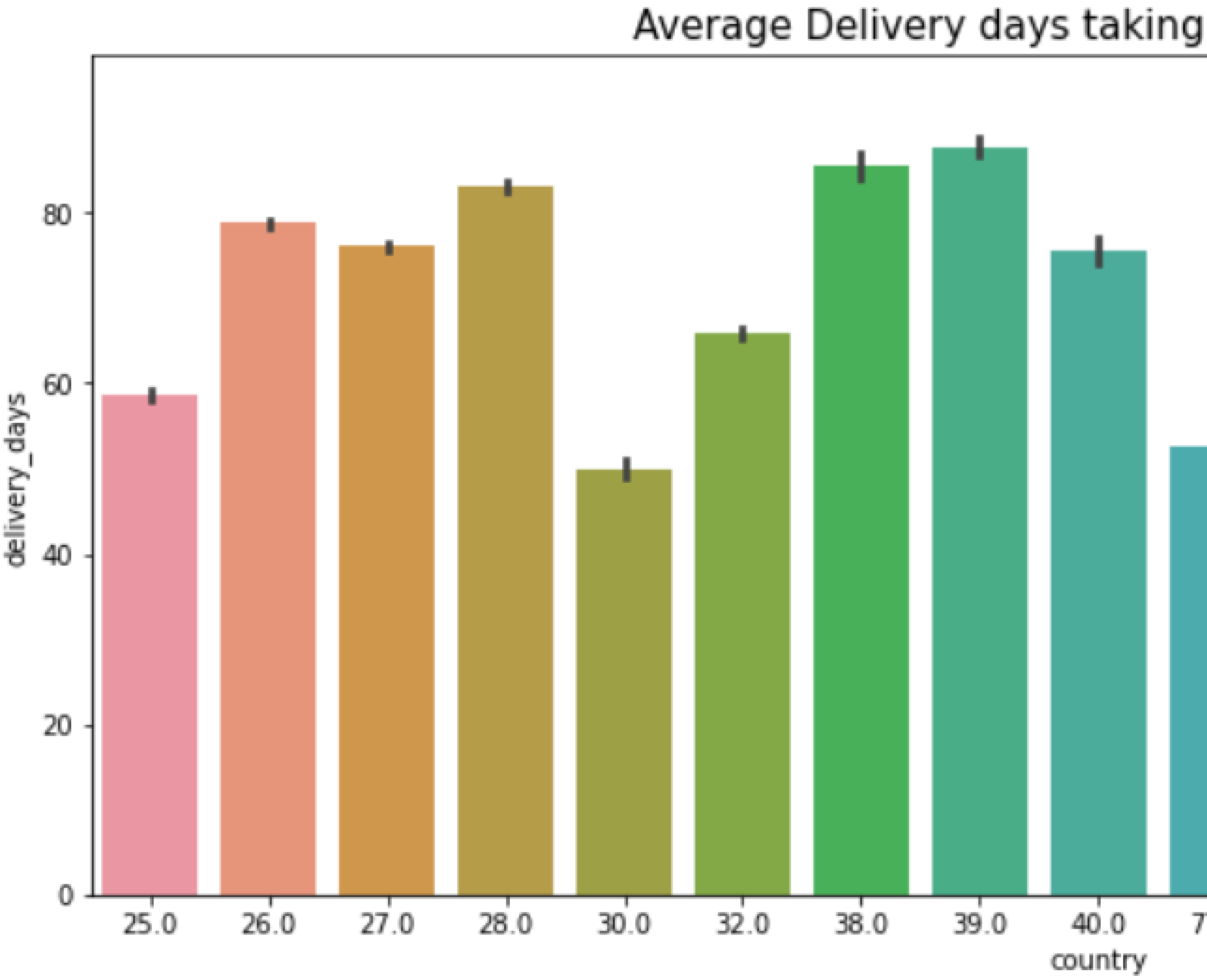
9. Status “offerable” countries



10. Average Selling Price on Status column



11.Average delivery days taking by each country



You can see more visual patterns in notebook

Mining:

Item date column Insight:

- The month of **April** has the lowest purchase rate. it might be because of our dataset has only one day value i.e(2021-04-01)
- **Saturday** has the lowest purchase rate. and followed by **Sunday**

Application column Insight :

- The month of **September(9)** is getting the highest applications. and there is no big difference between all the months they are getting decent applications.
- **Sunday** is getting the highest applications. And least applications on **saturday**
- Item Type **SLAWR** is getting the highest application. And PL is getting lowest application
- country **107** is getting the highest application. And 113 is getting lowest application

Country column Insight:

- All countries have the same selling price, except country 107 its selling price is double compared to the rest of the countries.
- Country **30** customers are purchasing the highest quantity(tons) of steel items which is 10 times higher than other country customers.
- Country **107** spent more money compared to other countries. And 89 spending least amount.
- All countries Item type buying priorities are W < S < PL < others < WI < IPL < SLAWR.

Item Type column Insight:

- Most buyed item type is **W** and followed by **S**
- Item type **PL** is the highest average selling price.
- Item type **S** taking the longest delivery days. and **Wi** taking least days

Status column Insight :

Item Type:

- In each category of status, Item type "w" has upper hand and followed by "S"
- But, Status ("offerable", "Offered" & "lost") offers only **W and S** item types.
- And rest of the status categories shares all item_types

Week day:

- In each category of status, Weekday **Saturday(6)" and "Sunday(7)** have their share. It could be because of Holiday.
- Mostly on **Monday** customers have high chances of getting offers.
- Mostly on **Monday and Friday** customers get offers.
- And the rest of the status categories share weekdays almost equally.

Country:

- Status("Offered") countries are only **28, 38, 113**. and order is **38 < 28 < 113**
- status("Offerable") countries are only **28 and 38** and order is **28 < 38**
- country **78** has the highest "Draft" , which is 5 times more than others. and countries **89 and 107** has very least draft
- country **26** has the highest "Lost" , and least are **89 and 80**
- country **26** has highest "revise" and least are **77 and 79**
- Except countries **89 and 107** each country has status "won" and highest status "won" country is **78**

avg_selling price:

- status **To be approved** has the highest selling price. And this could be of highest demand.
- And status **Draft** has the lowest selling price which is even less than 20.

5. Exploratory Data Analysis (EDA) & Preprocessing:

- Categorised the variable into Categorical and Numeric columns. This Categorization helps to understand the nature of the data
- Remove unwanted columns “item_date, Item_month, Item_weekday and delivery date”
- Check the Target variable for Normality (Gaussian Distribution). The target variable y is skewed
- checked skewness of the numerical distribution. But, none of the features are in normal distribution
- Check the Correlation with numerical variables and with target feature
- Outlier detection using Z-score and IQr methods.
- Based on the result of calculation using Z-score as well as the IQR, it can be seen that the number of deleted rows based on the IQR is far more than z-score. Therefore, I decided to choose the Z-score method to remove the outliers.
- Scaling technique used to scale the numeric data based on the type of distribution. So, our numeric features do not have Normal Distribution. We will be using the Normalisation technique.
- Target Relationship with predictor

	numerical_predictor	correlation_w_target
0	amount_spent	0.006337
1	country	0.002993
2	product_ref	0.002117
3	customer	0.001747
4	application	0.001462
5	delivery_days	0.000930
6	width	0.000584
7	quantity tons	-0.000010
8	thickness	-0.002467

6. Model Evaluation:

- Tried Linear Regression, RandomForestRegression, LGBM and Cat Boost regression models. but, didn't give the best result here.
- So XGBoost Regression gave the best result among all the models.
- I have used RandomSearchCV to perform Hyper Parameter Tuning

- Before tuning R2_score of XGB

```
]: # 3. Xgb regressor
   from xgboost import XGBRegressor
   from sklearn.metrics import r2_score

   xgb = XGBRegressor(random_state=101,n_estimators=100)
   xgb.fit(X_train,y_train)

   preds_valid = xgb.predict(X_test)

   print('Training Score(r2_score)',r2_score(y_train,xgb.predi
   print()
   print('Test Score(r2_score)',r2_score(y_test,xgb.predict(X_

   Training Score(r2_score) 0.9999999825556192

   Test Score(r2_score) 0.6443887977709057
```

- Train the model with hyper parameters

```
: from sklearn.model_selection import RandomizedSearchCV, Gri
import numpy as np

hyperparameters = {
                    'max_depth' : [7],
                    'gamma' : [11.11],
                    'tree_method' : ['hist'],
                    'min_child_weight' : [1],

                    'colsample_bytree' : [float(x) for x in
                    'eta' : [float(x) for x in np.linspace(

                    'lambda' : [float(x) for x in np.linspa
                    'alpha' : [float(x) for x in np.linspac
                    }

from xgboost import XGBClassifier
xg = XGBRegressor(random_state=101)
xg_tuned = RandomizedSearchCV(xg, hyperparameters, cv= 5, r
xg_tuned.fit(X_train,y_train)
```

- After Hyperparamter tune R2_Score is

```
9]: preds_valid = xg_tuned.predict(X_test)

print('Training Score(r2_score)',r2_score(y_train,xg_tune
print()
print('Test Score(r2_score)',r2_score(y_test,xg_tuned.pre
```

```
Training Score(r2_score) 0.9999999874613504

Test Score(r2_score) 0.9783415138938402
```