OPTIMUM PRODUCT WEIGHT SHIPPMENT TO THE WAREHOUSE

PROBLEM STATEMENT

- Miss match in the demand and supply.
- An inventory cost loss to the company

OBJECTIVES

Goal & Objective: This exercise aims to build a model, using historical data that will determine the optimum weight of the product to be shipped each time to the warehouse.

Also, try to analyze the demand pattern in different pockets of the country so management can drive the advertisement campaign, particularly in those pockets.

This is the first phase of the agreement; hence, the company has shared very limited information. Once you are able to showcase a tangible impact with this much information then the company will open the

360-degree data lake for your consulting company to build a more robust model.

DATA SOURCE INFORMATION

S_no	Field's Name	Description
1	Warehouse_id	Product warehouse_id
2	WH_Manager_ID	Employee ID of warehouse manager
3	Location_type	Location of warehouse-like in city or village
4	WH_capacity_size	Storage capacity size of the warehouse
5	Zone	Zone of the warehouse
6	WH_regional_zone	Regional zone of the warehouse under each zone
7	num_refill_req_l3m	Number of times refilling has been done in last 3 months
8	transport_issue_l1y	Any transport issue like accident or goods stolen reported in last one

DATA SOURCE INFORMATION

S_no	Field's Name	Description
9	Competitor_in_mkt	Number of instant noodles competitors in the market
10	retail_shop_num	Number of the retail shop that sell the product under the warehouse area
11	wh_owner_type	Company is owning the warehouse or they have got the warehouse on rent
12	distributor_num	The number of distributer works in between warehouse and retail shops.
13	flood_impacted	Warehouse is in the Flood impacted area indicator.
14	Flood_proof	Warehouse is a flood-proof indicator. Like storage is at some height not flood_proof
15	electric_supply	Warehouse have electric back up like generator, so they can run the warehouse in load shedding
16	dist_from_hub	Distance between warehouse to the production hub in Kms

DATA SOURCE INFORMATION

S_no	Field's Name	Description
17	workers_num	Number of workers working in the warehouse
18	wh_est_year	Warehouse established year
19	storage_issue_reported_l3m	Warehouse reported storage issue to corporate office in last 3 months. Like rat, fungus because of moisture etc.
20	temp_reg_mach	Warehouse have temperature regulating machine indicator
21	approved_wh_govt_certificate	What kind of standard certificate has been issued to the warehouse from government regulatory body
22	wh_breakdown_l3m	Number of time warehouse face a breakdown in last 3 months. Like strike from worker, flood, or electrical failure
23	govt_check_l3m	Number of time government Officers have been visited the warehouse to check the quality and expire of stored food in last 3 months
24	**product_wg_ton(Target_column)**	Product has been shipped in last 3 months. Weight is in tons

EXPLORATORY DATA ANALYSIS – DATA CLEANING

Step 1 identify number of missing values from each column and identify columns with a large number of na's/missing columns

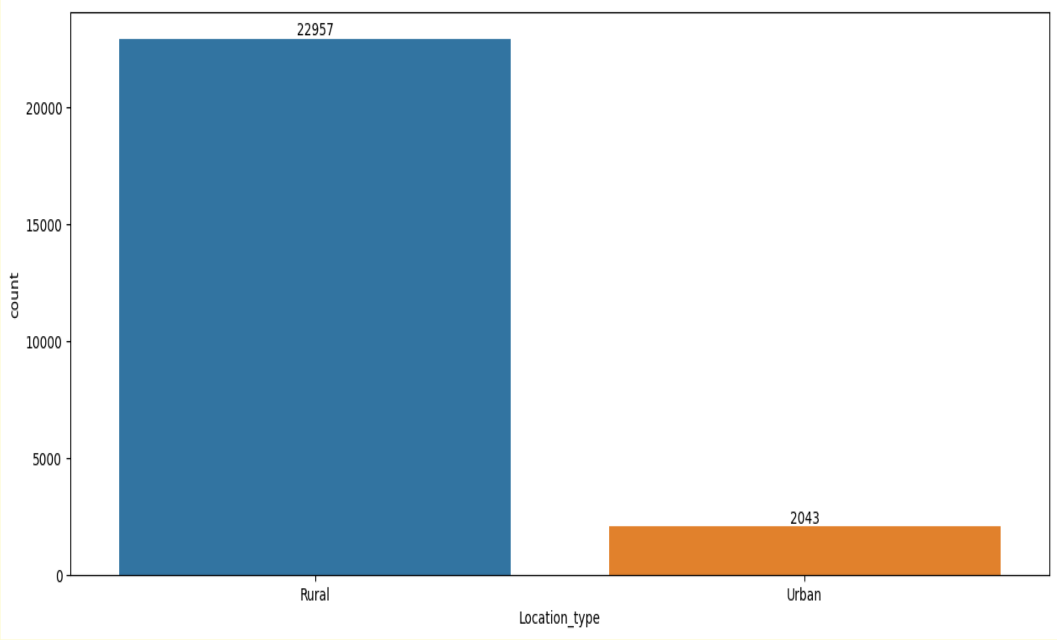
Step 2 Univariate Analysis and Visualization for each columns in the dataset

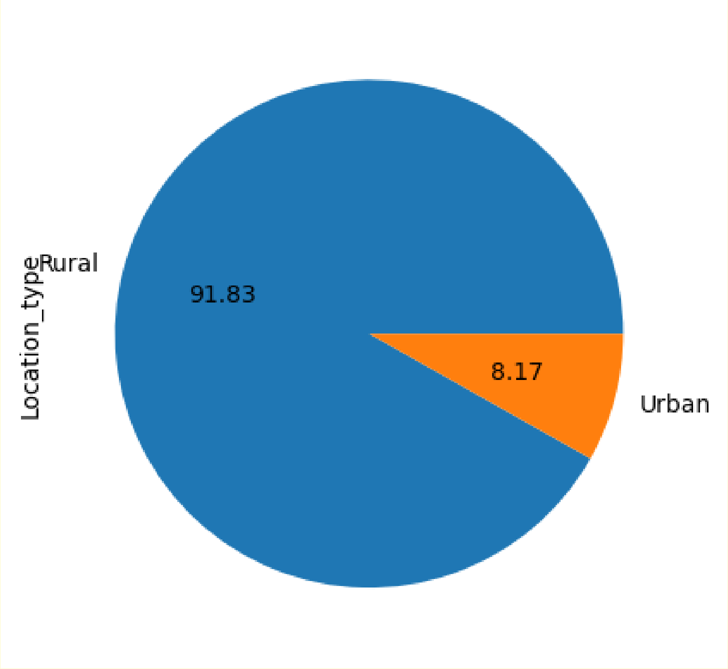
Step 3 Bivariate Analysis between the columns and Visualization

Step 4 Implementation of feature engineering

Step 5 Implementation of Models

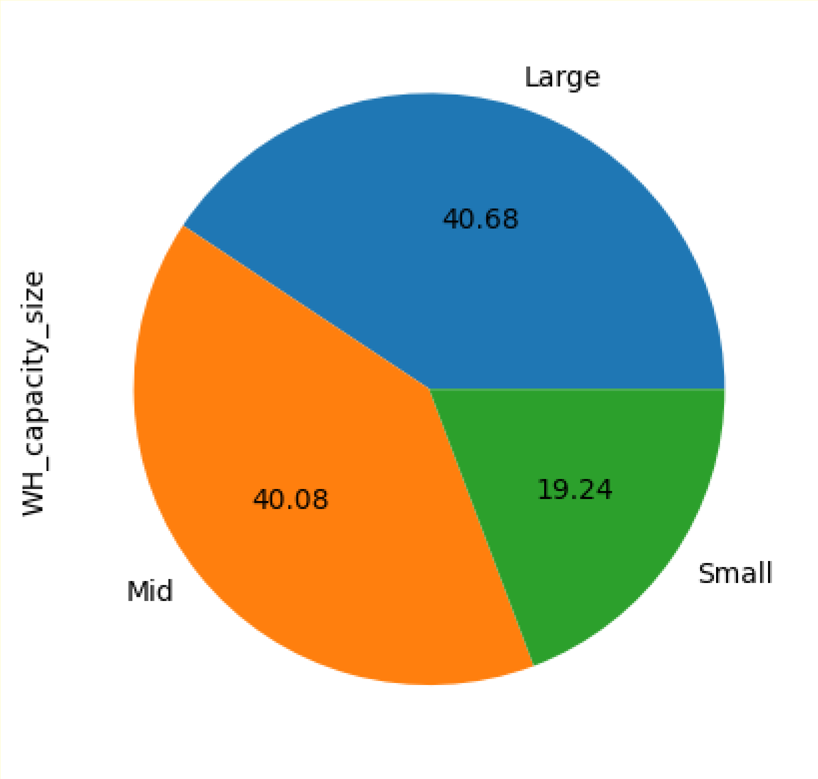
UNI-VARIATE ANALYSIS AND VISUALIZATION





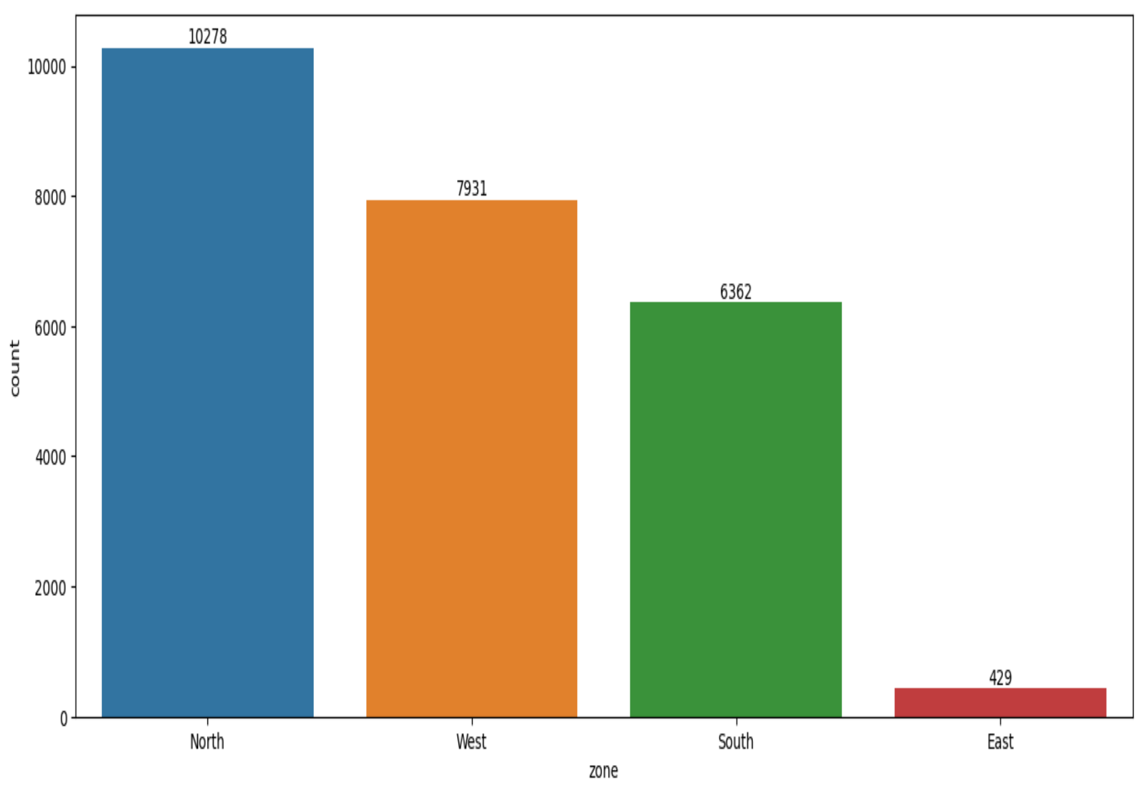
Approx 92% of the location type fall under Rural

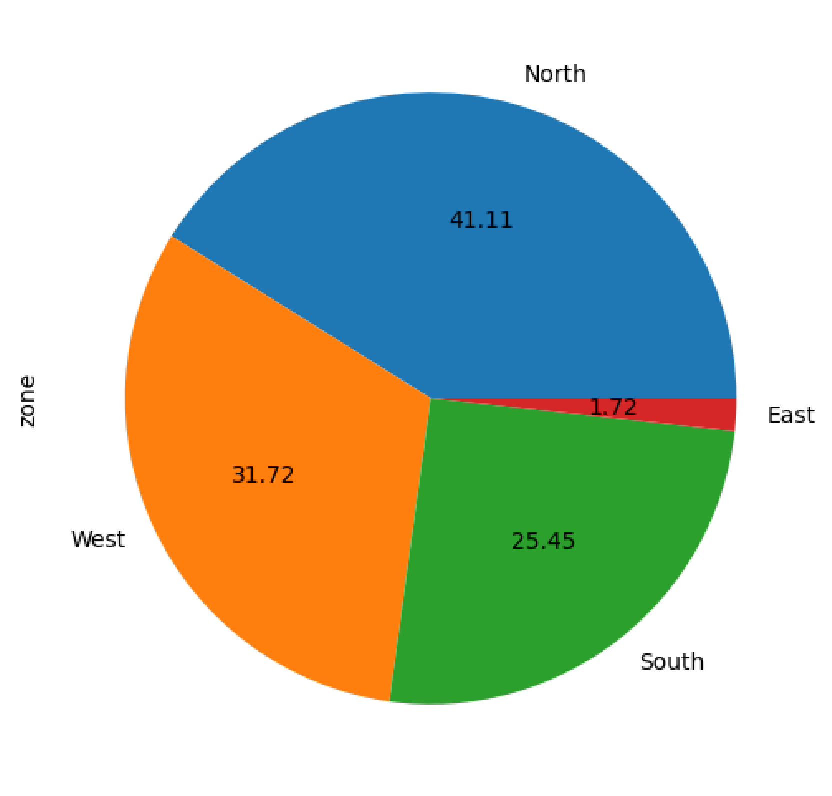
UNI-VARIATE ANALYSIS AND VISUALIZATION



Approx 40.68% of the warehouse size are large,40.08% are of mid-size

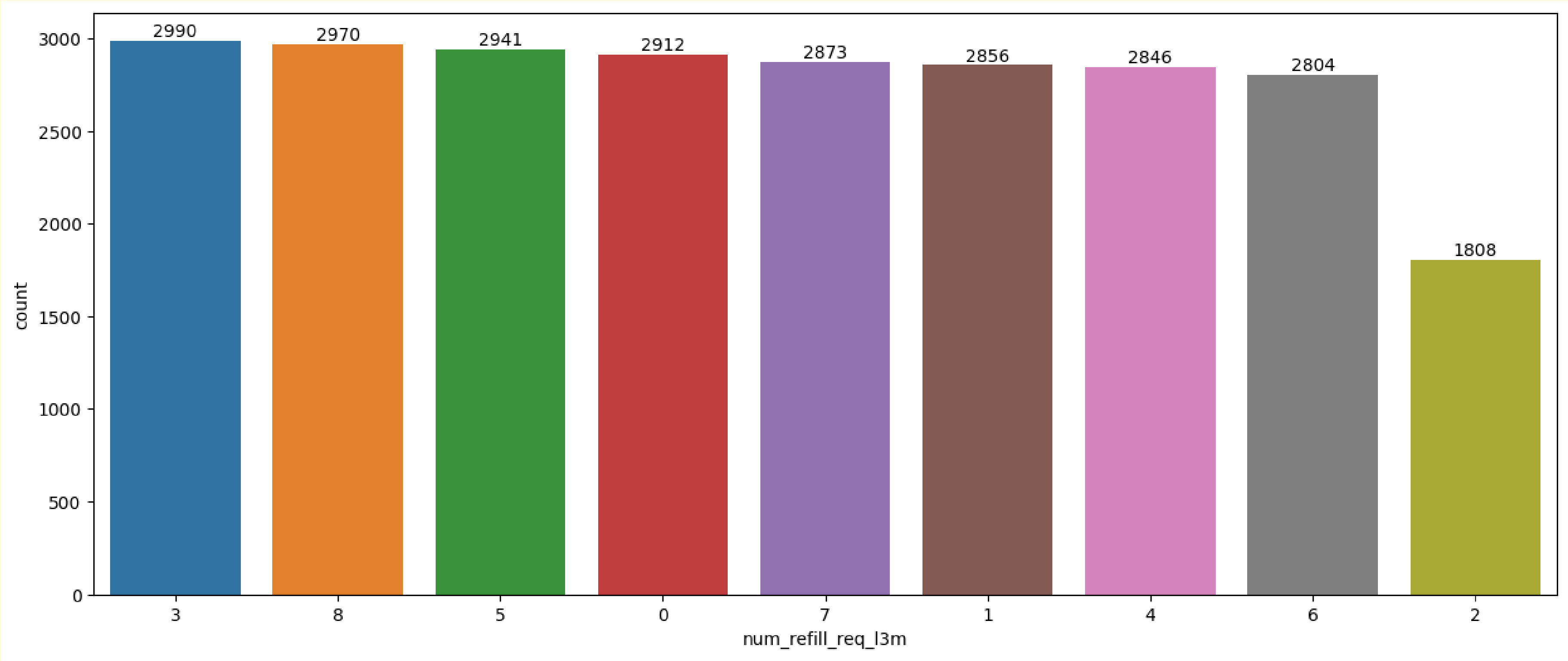
UNI-VARIATE ANALYSIS AND VISUALIZATION





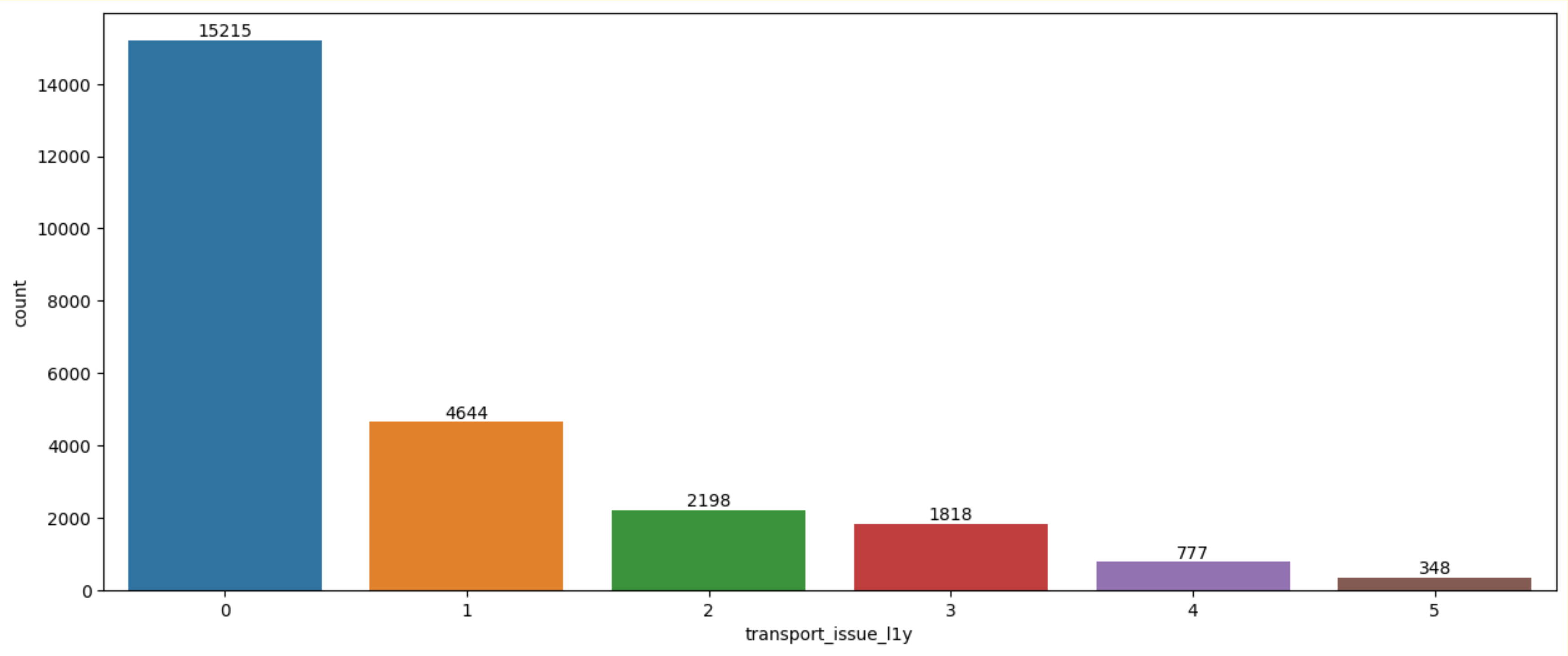
North zone has more number of warehouse followed by west and south

UNI-VARIATE ANALYSIS AND VISUALIZATION



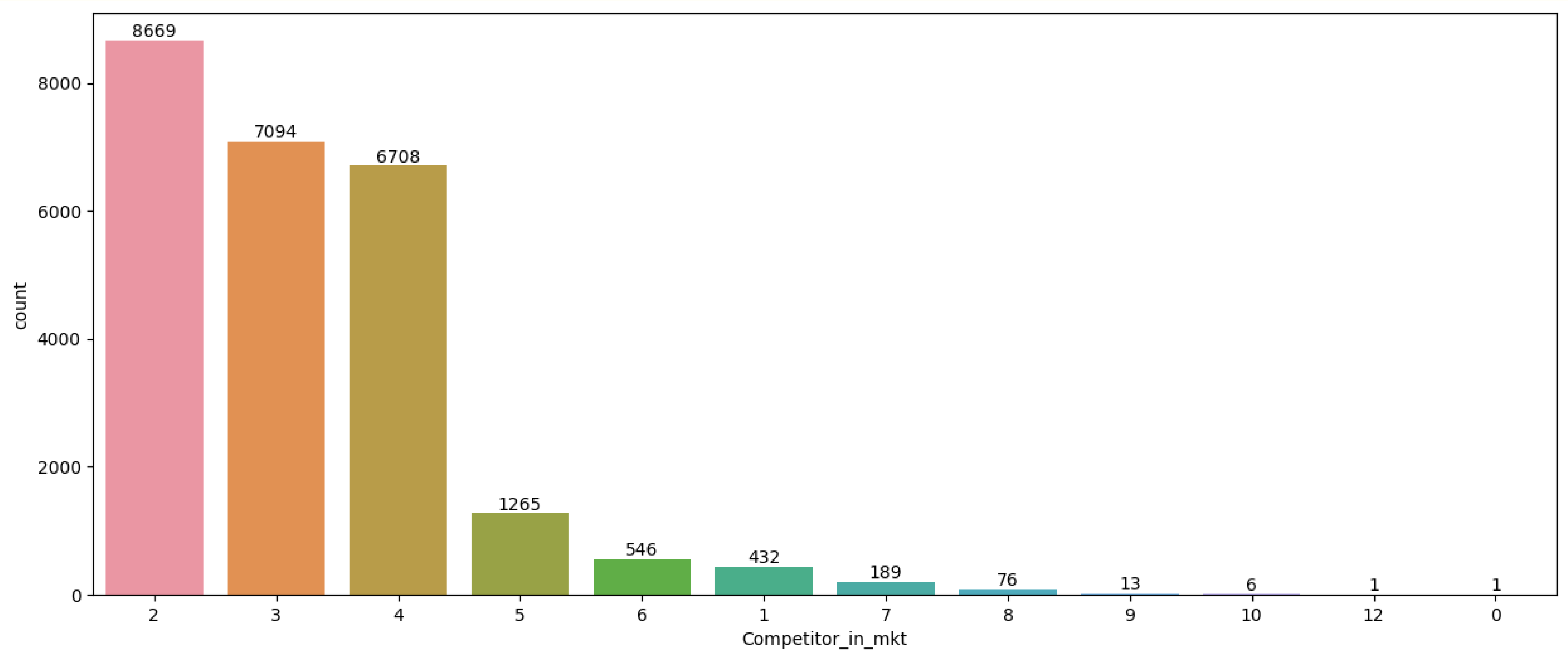
Frequency of refill required is almost similar for the warehouse in last 3 month

UNI-VARIATE ANALYSIS AND VISUALIZATION

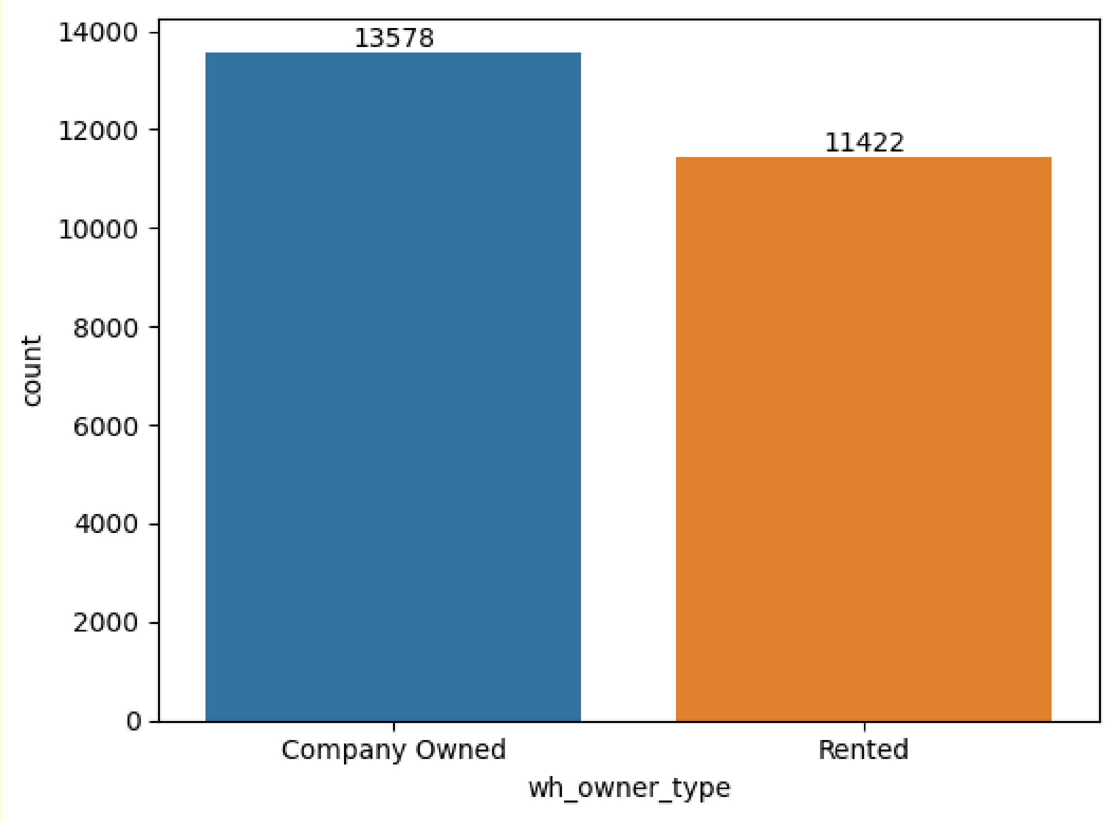


Almost 61% times no transport issue occured in last 1 year

UNI-VARIATE ANALYSIS AND VISUALIZATION

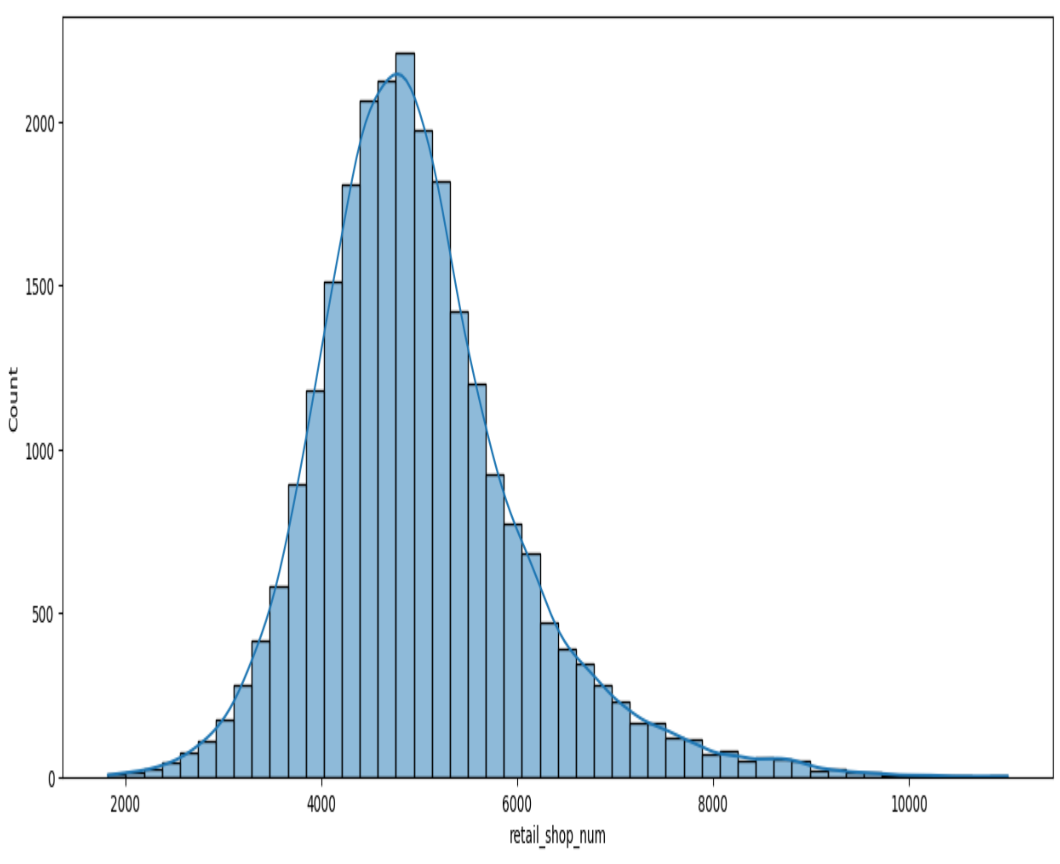


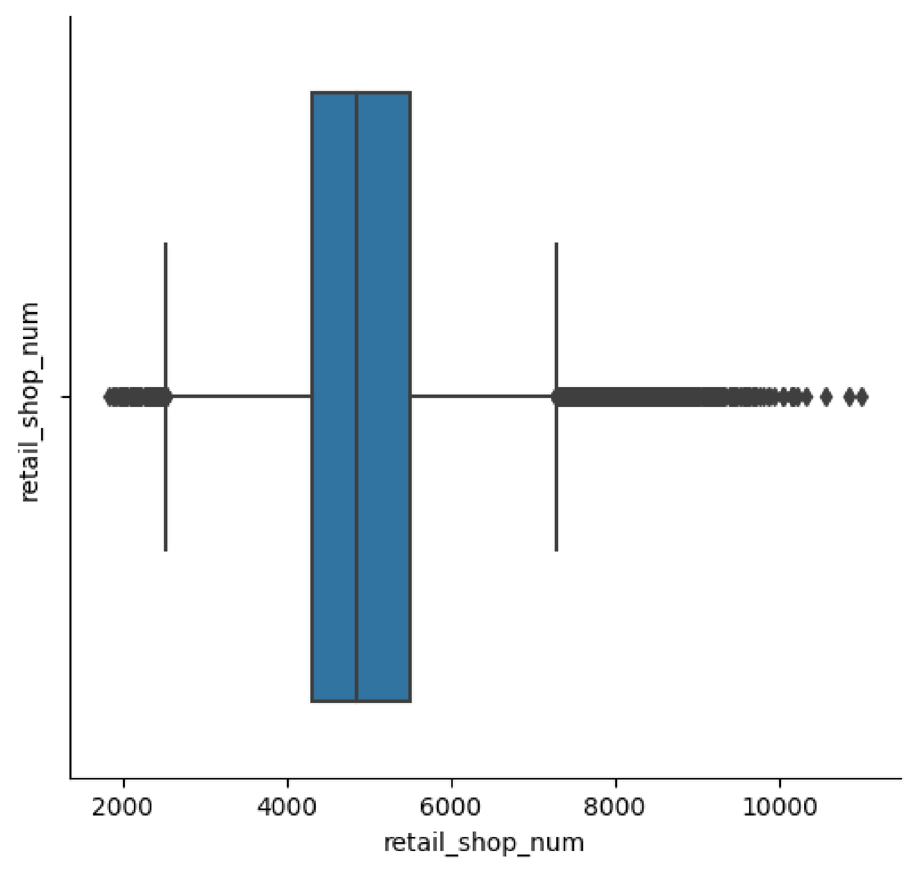
UNI-VARIATE ANALYSIS AND VISUALIZATION



Company owned is more than rented one

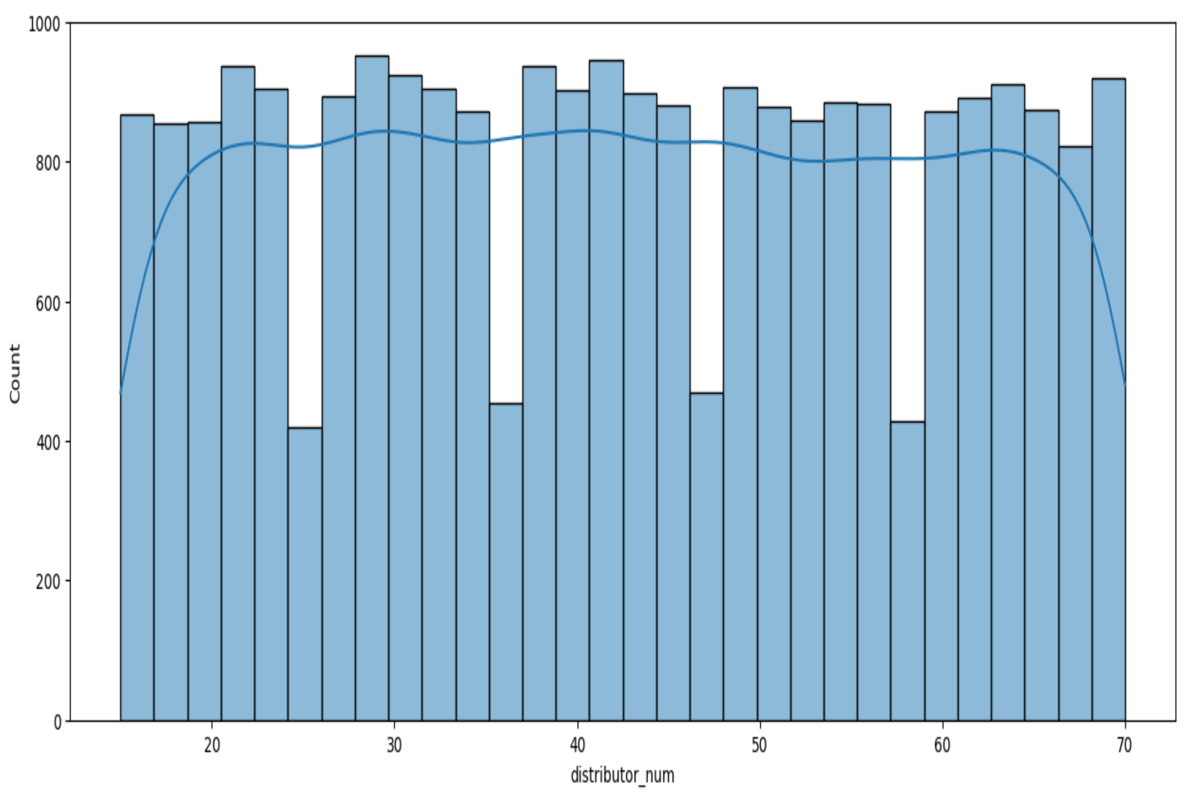
UNI-VARIATE ANALYSIS AND VISUALIZATION

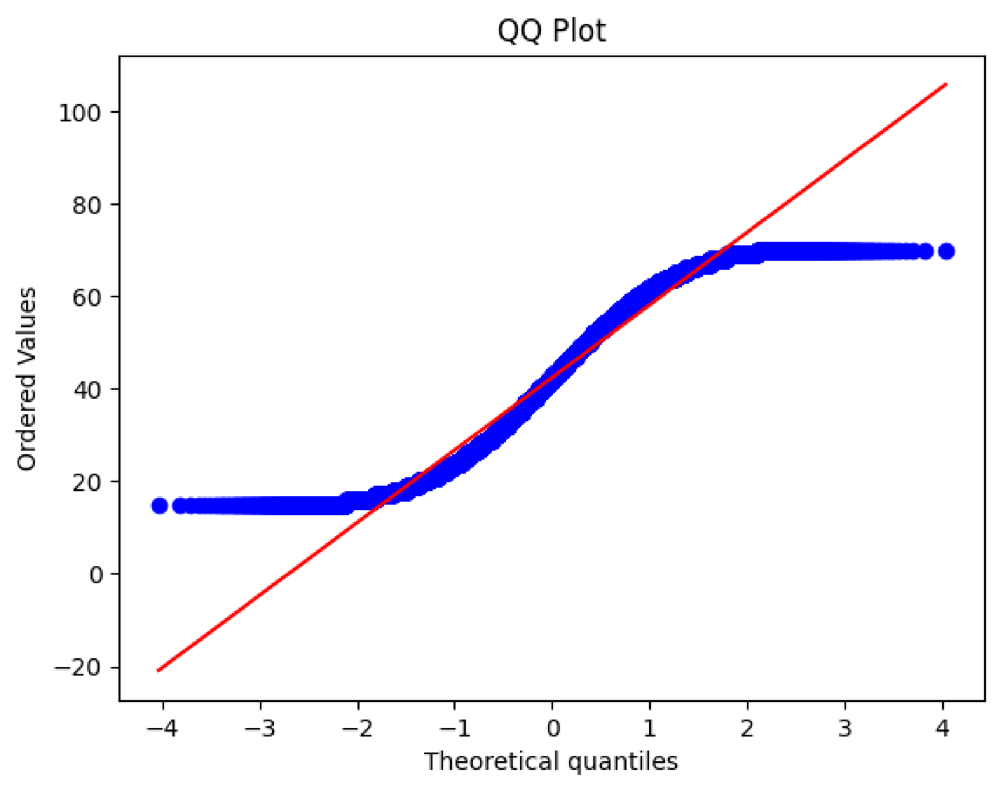




The column 'retail_shop_num' is normally distributed which is a good sign for linear regression problem but there are plenty of outliers in the column

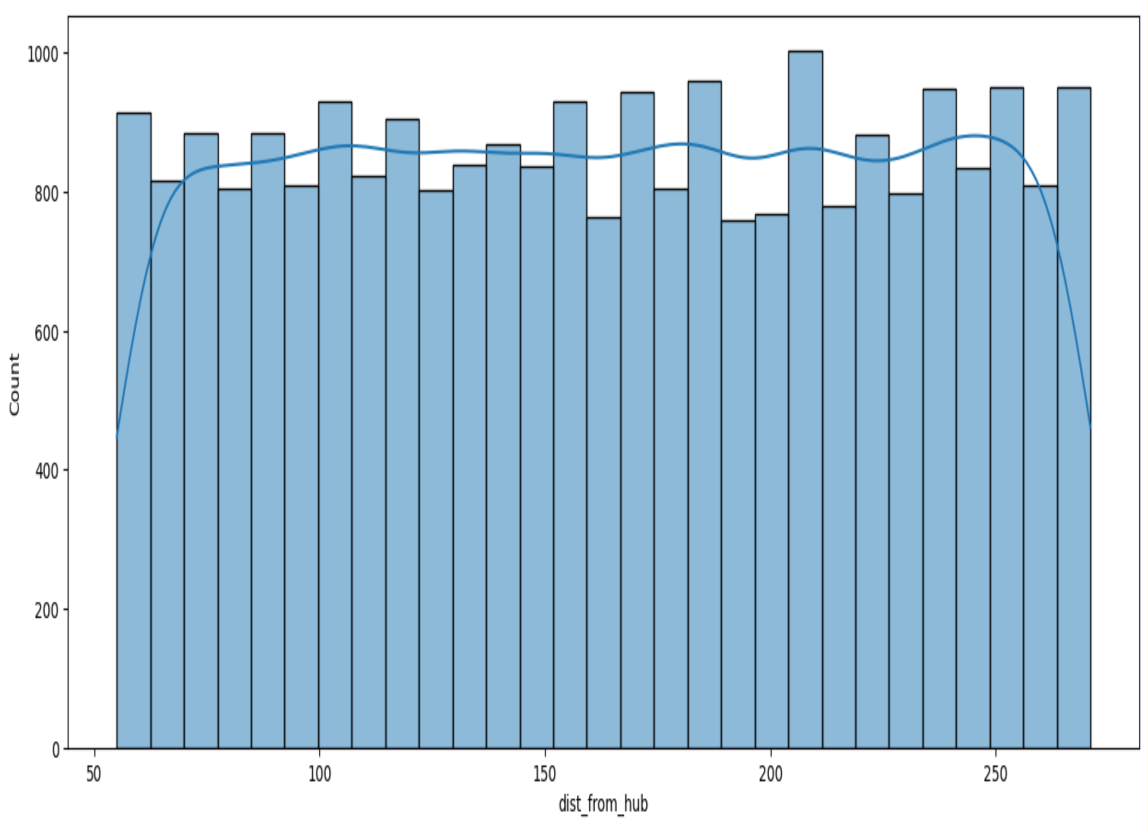
UNI-VARIATE ANALYSIS AND VISUALIZATION

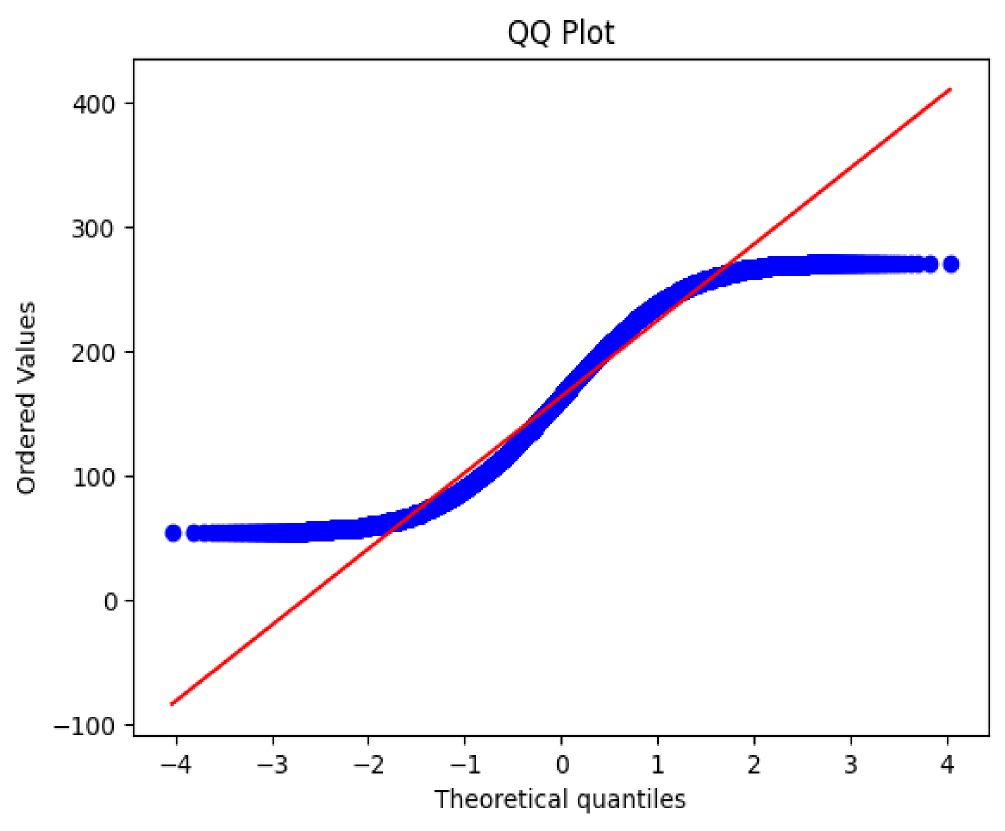




The ‘distributer no.’ is not normally distributed, count is almost similar for all the warehouse, we can covert the column into range values

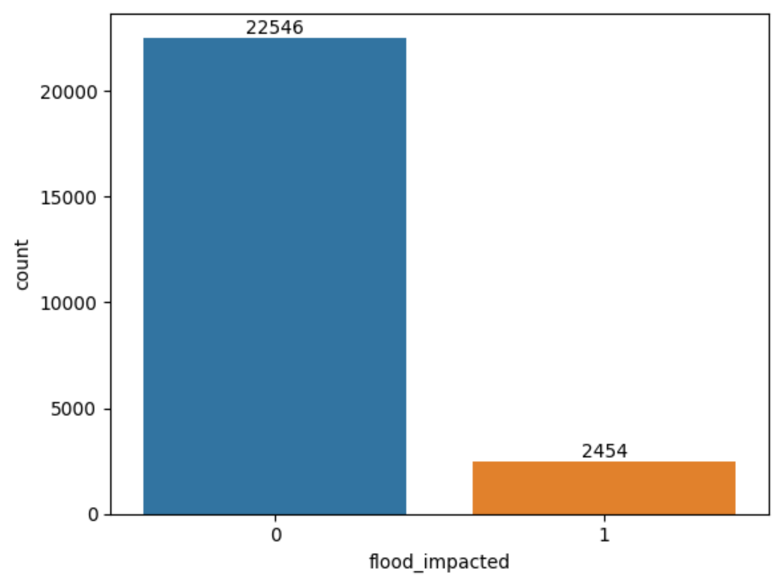
UNI-VARIATE ANALYSIS AND VISUALIZATION

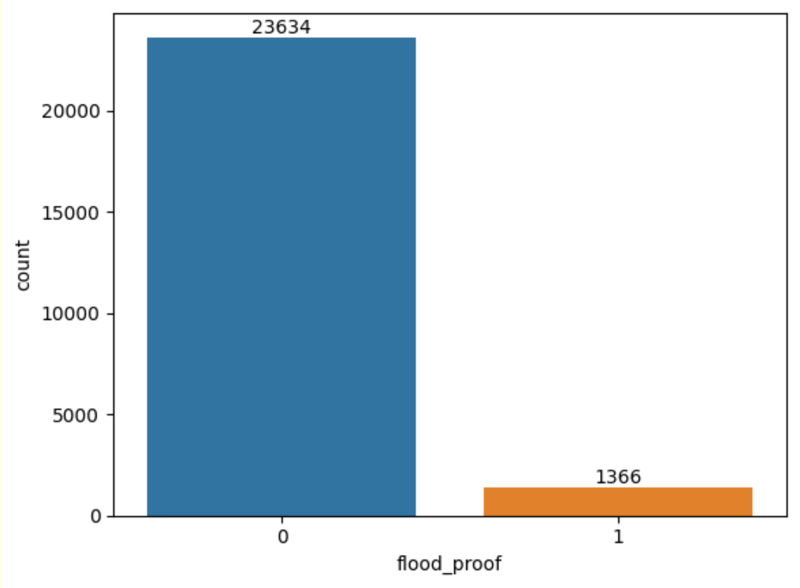


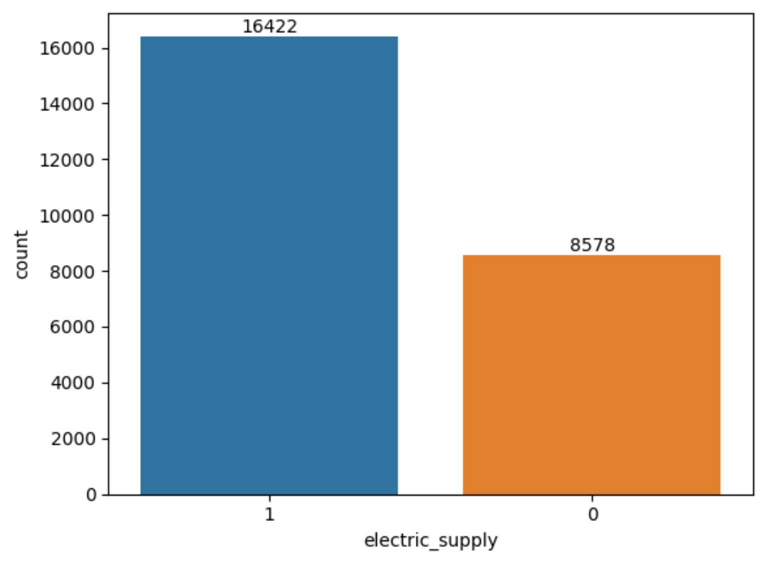


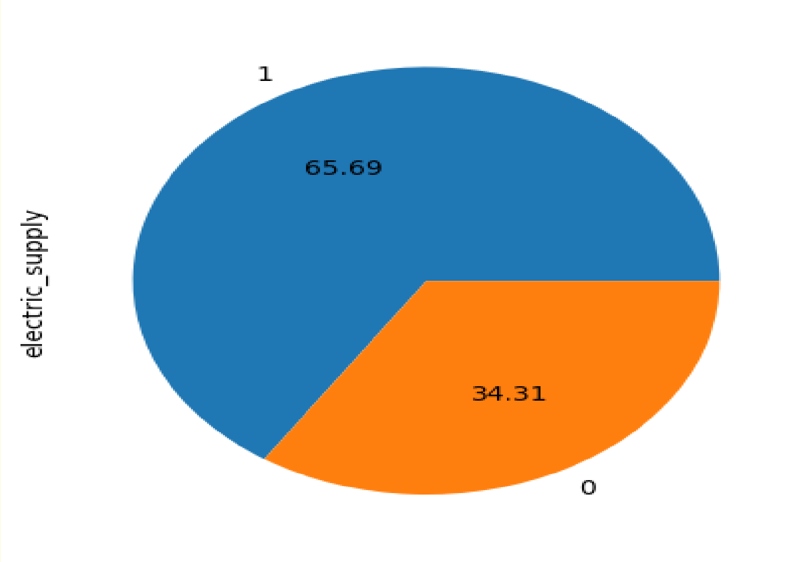
The ‘distance from hub’ is not normally distributed, freq of distance is almost similar for all the warehouse, , we can covert the column into range of values

UNI-VARIATE ANALYSIS AND VISUALIZATION

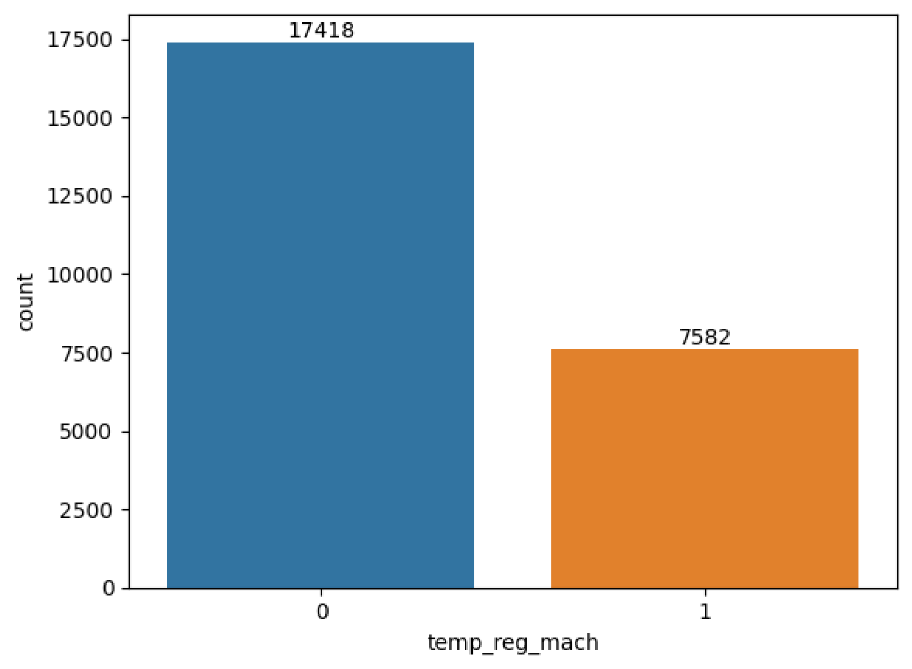


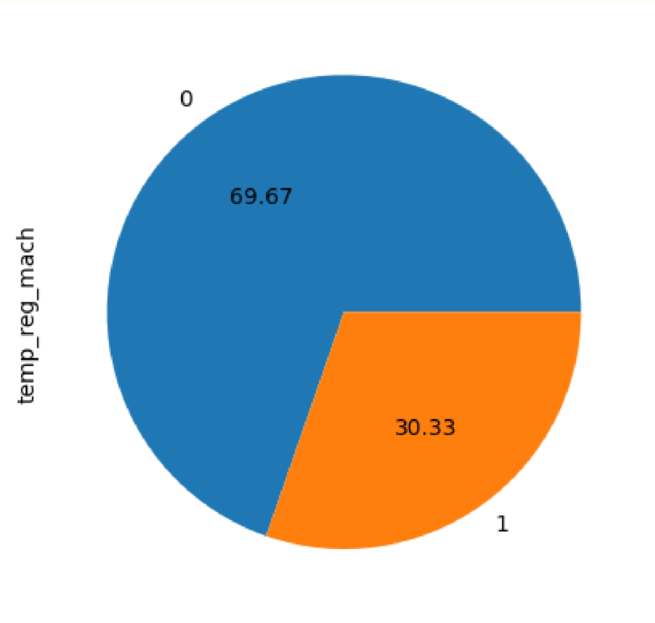






UNI-VARIATE ANALYSIS AND VISUALIZATION

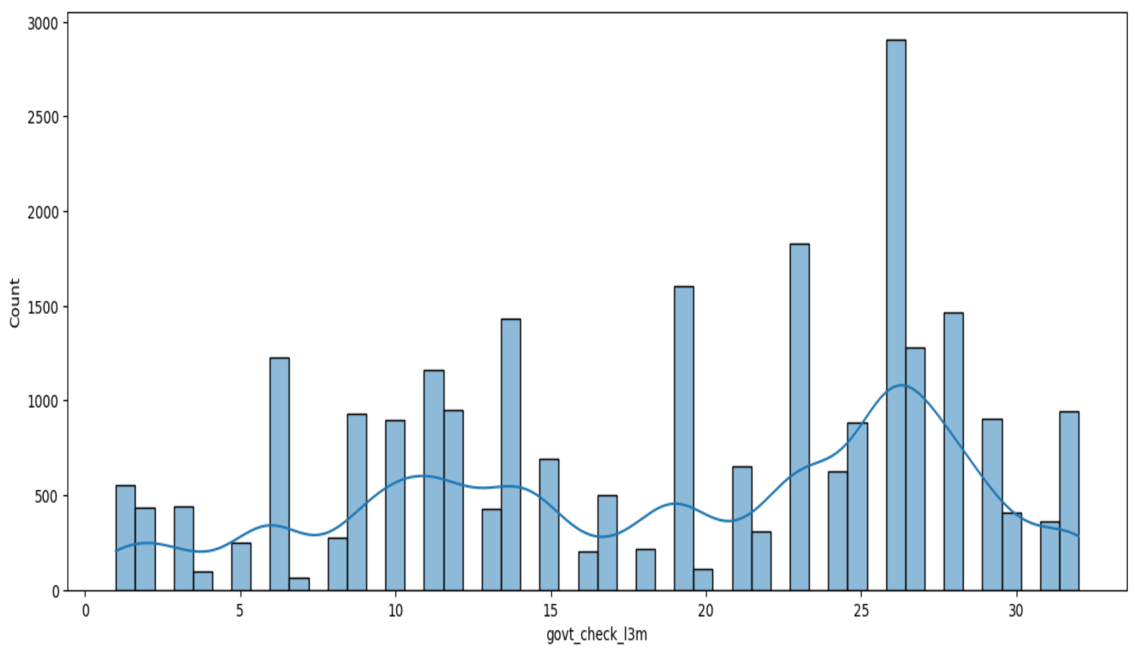


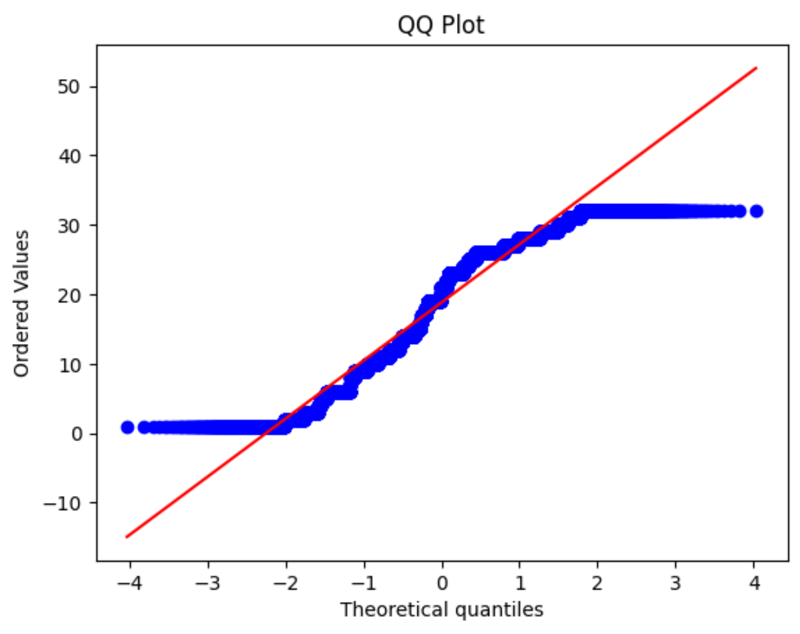


70% of the warehouse contains temperature regulator machine

UNI-VARIATE ANALYSIS AND VISUALIZATION

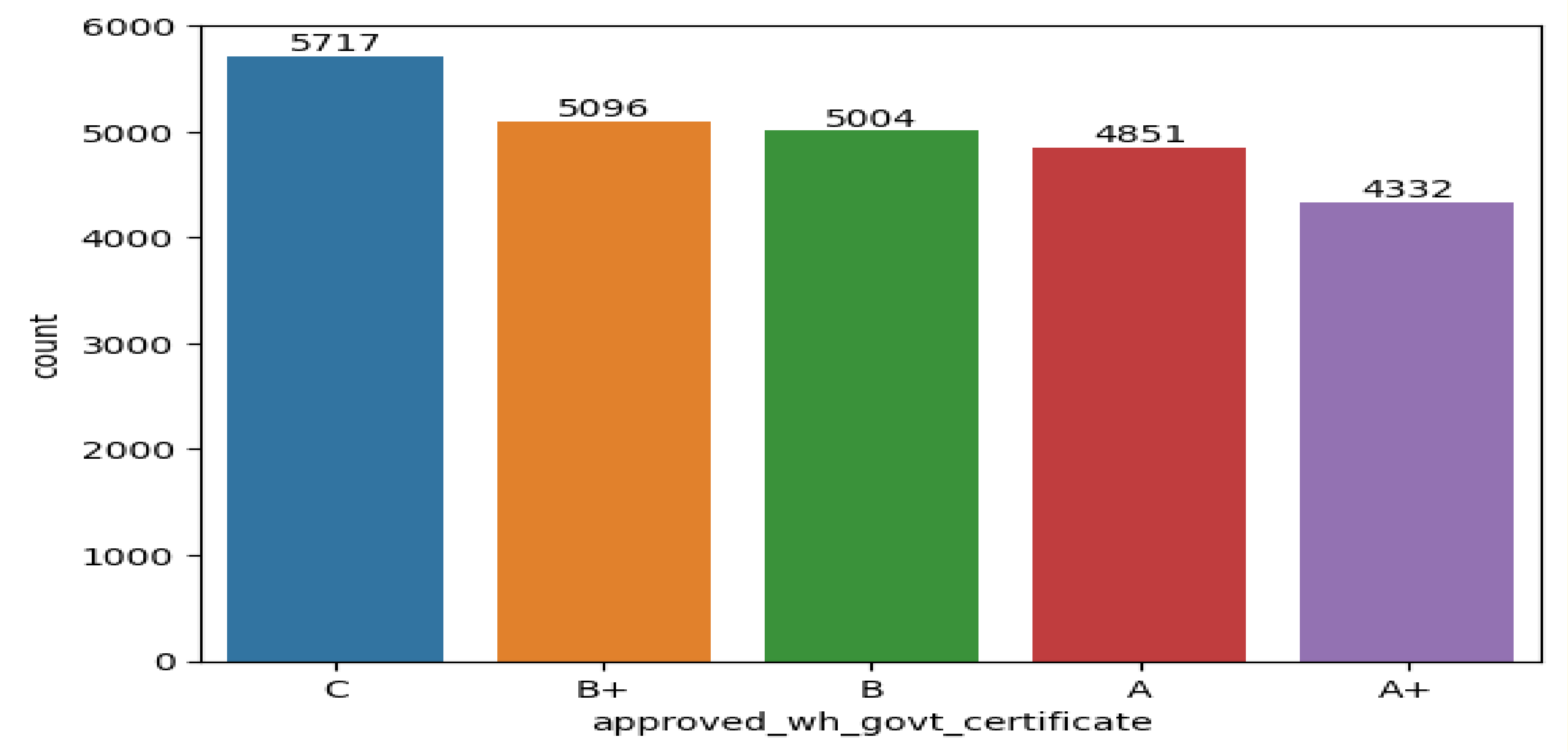
Create a histogram plot with kernel density estimation for the column 'govt_check_l3m'



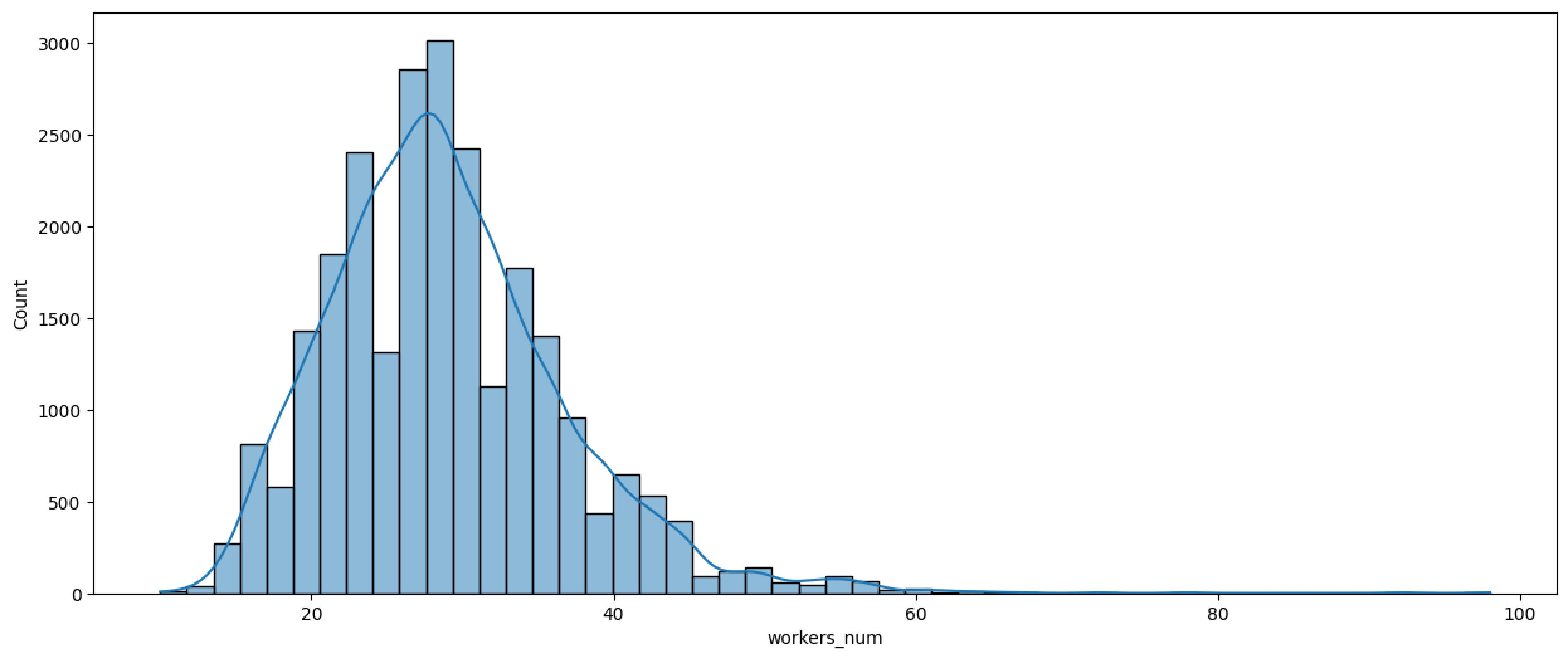


The frequecies of govt check for all the warehouse is almost similar, to be converted as values of ranges

UNI-VARIATE ANALYSIS AND VISUALIZATION



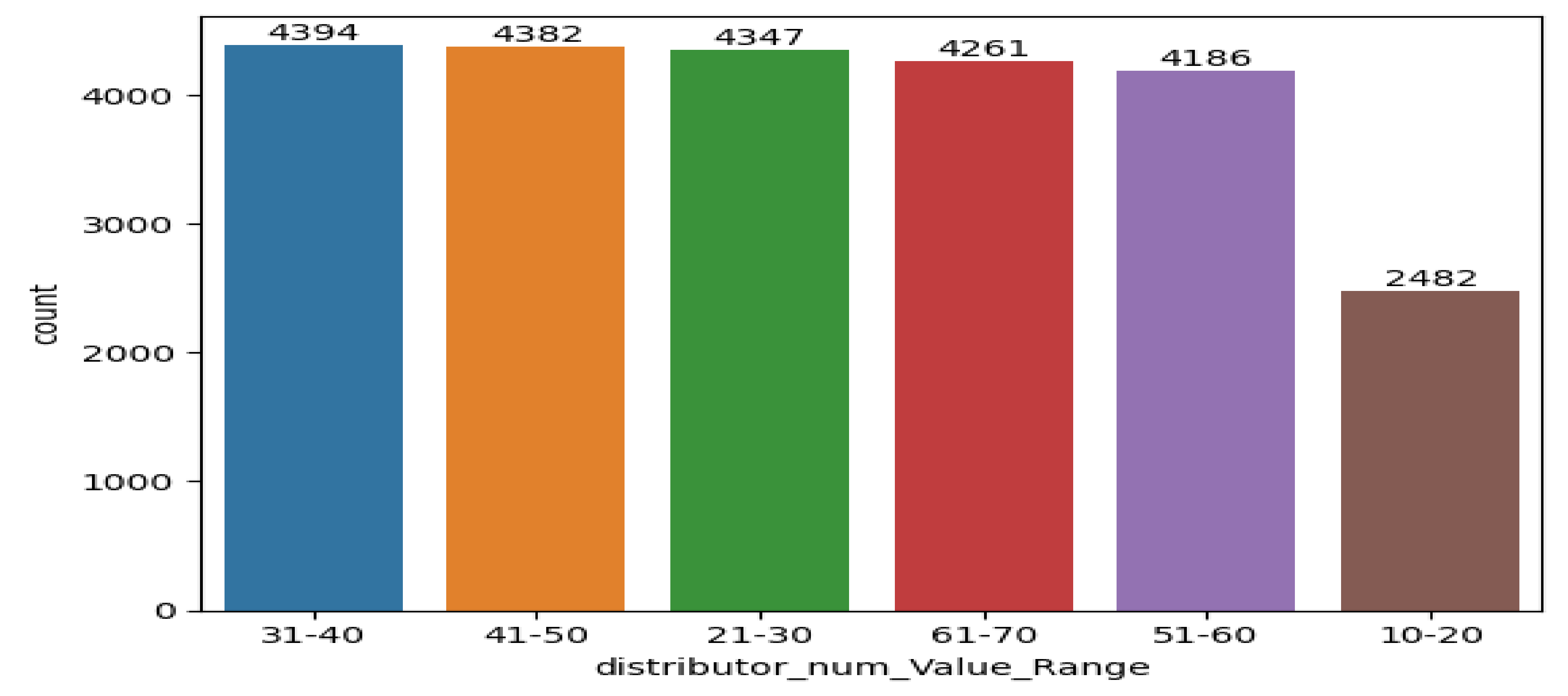
UNI-VARIATE ANALYSIS AND VISUALIZATION




```
Skewness for 'column': 1.0535450798511456
```

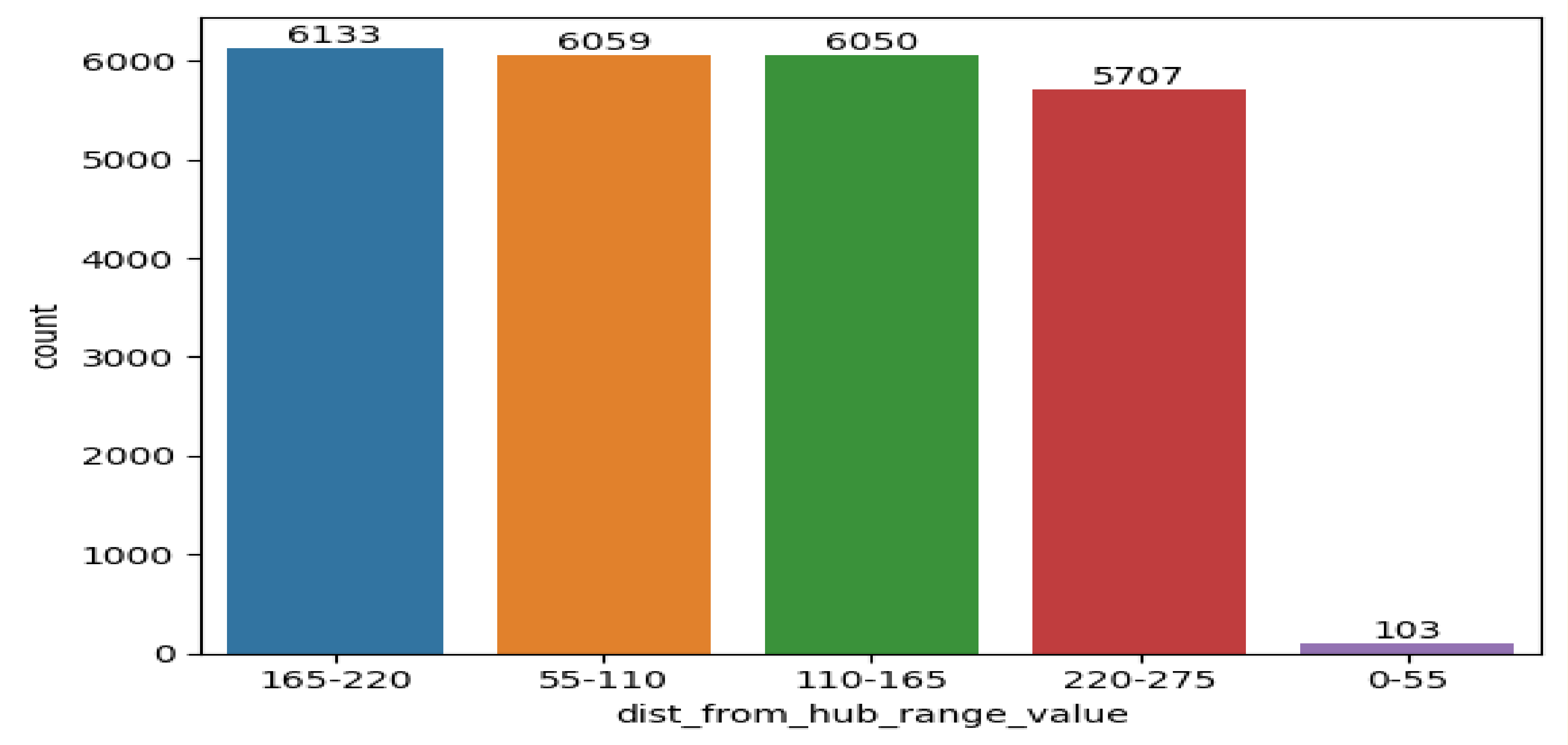
FEATURE ENGINEERING :

Converting the column 'distributor_num' values into ranges



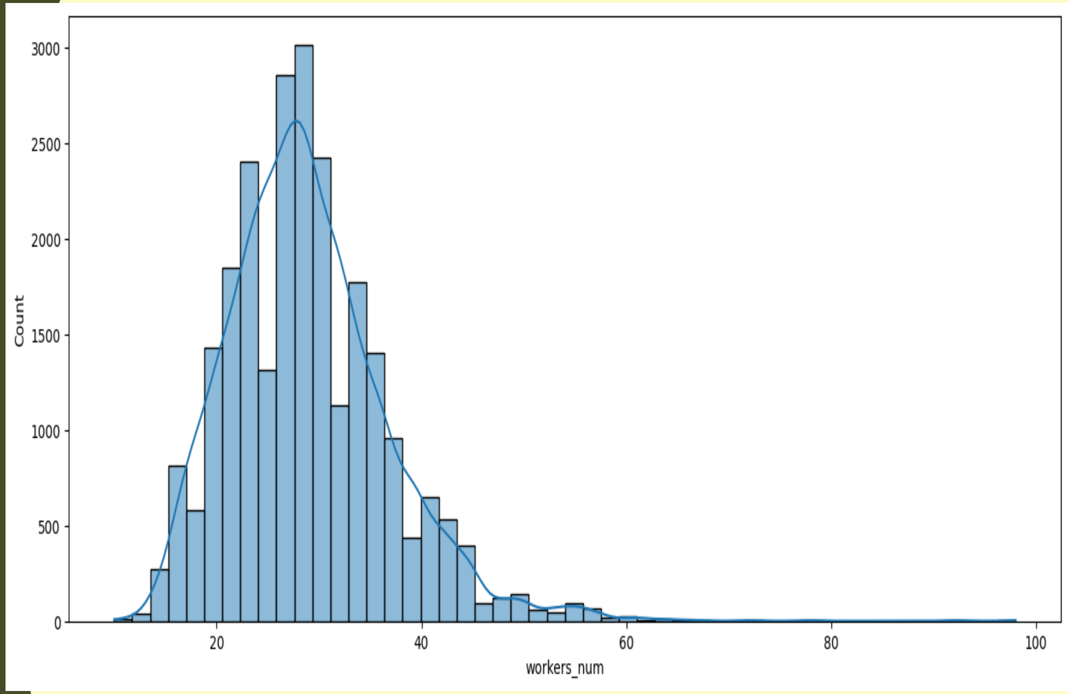
FEATURE ENGINEERING :

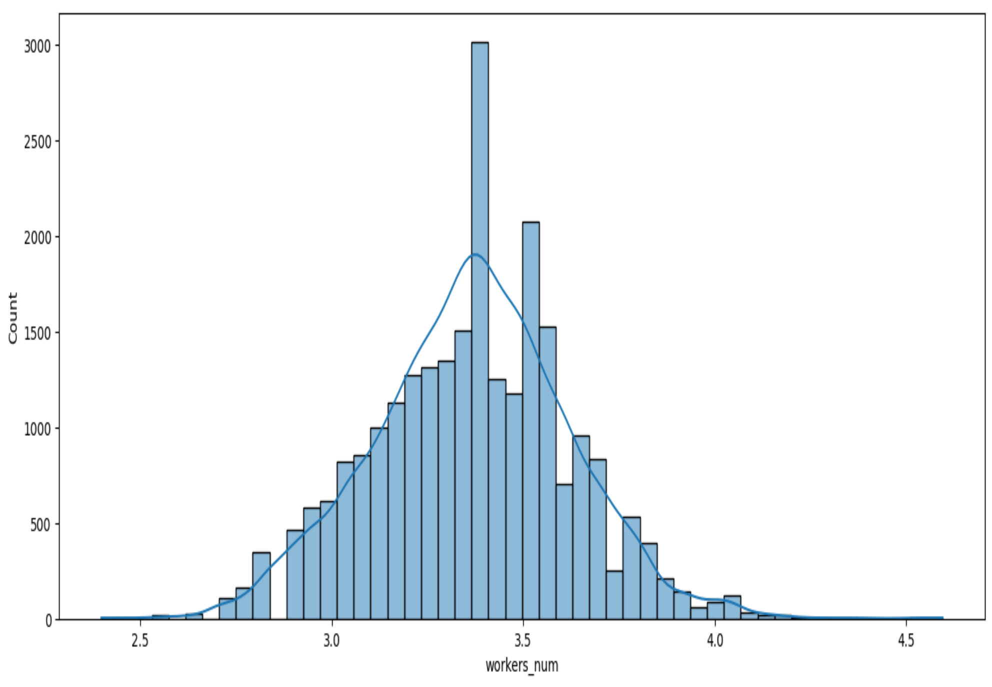
Converting the column 'dist_from_hub' values into ranges



FEATURE ENGINEERING :

Transformed the 'workers_num' column values using the np.log1p function



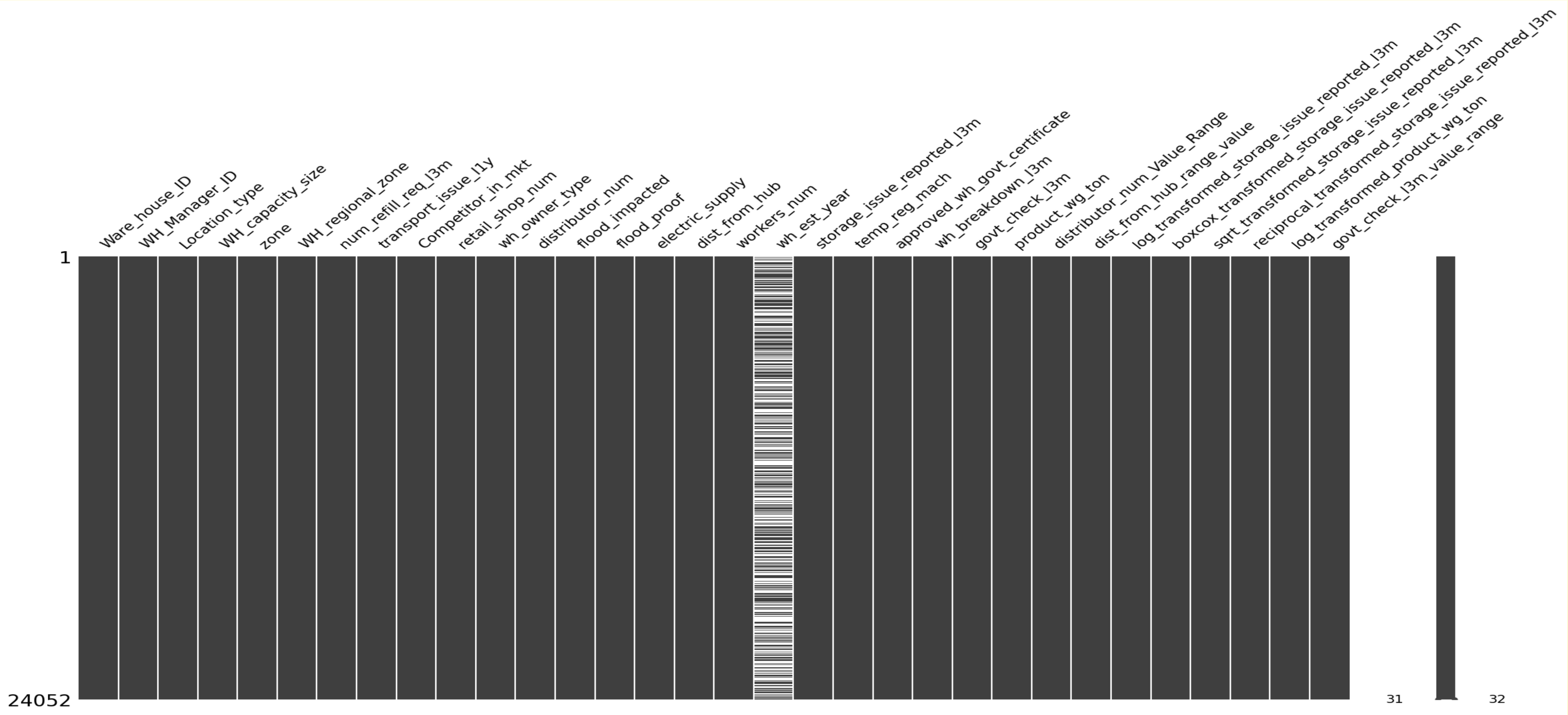



```
Skewness for 'column':
0.026005812895925115
```

```
Skewness for 'column':
 0.026005812895925115
```

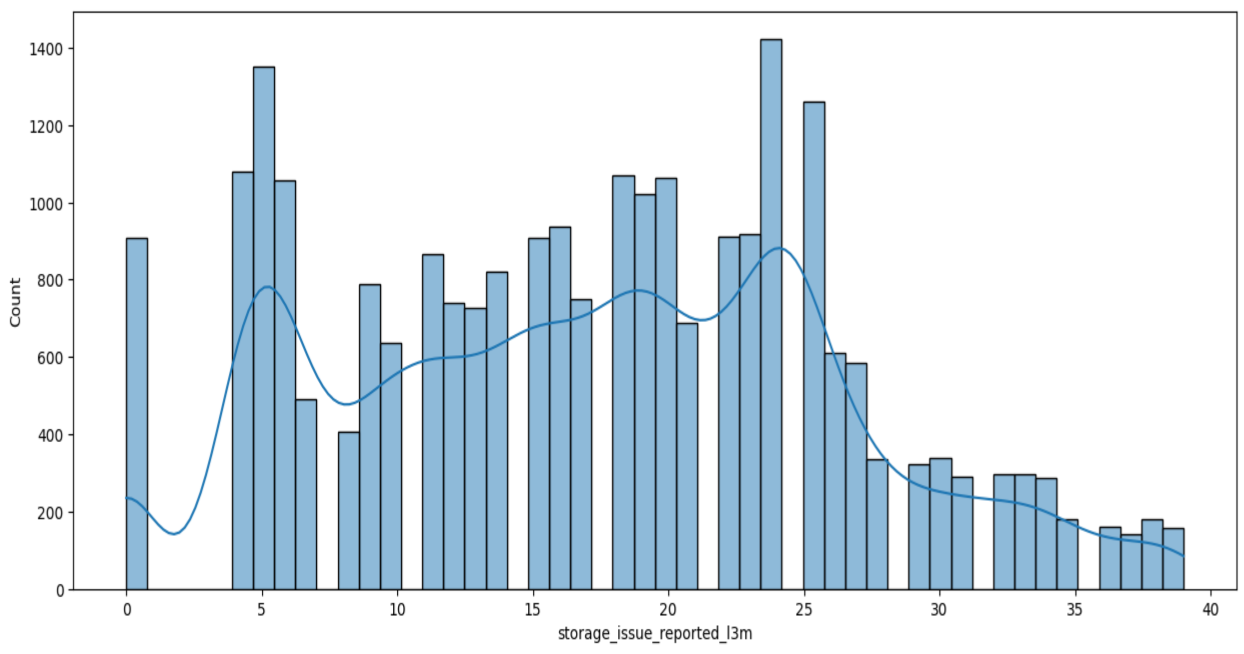
FEATURE ENGINEERING :

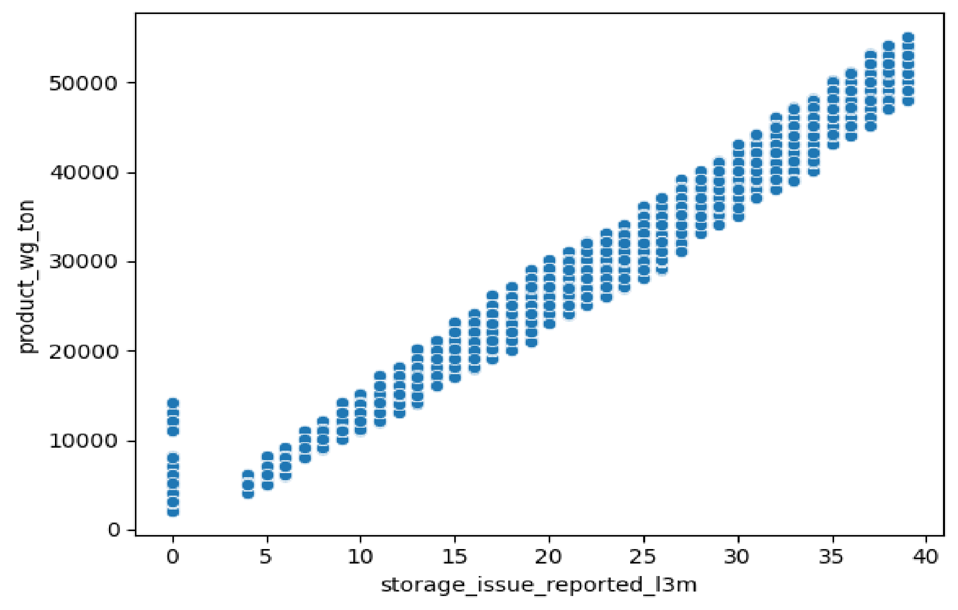
Almost 50% of the data is missing in the column ‘wh_est_year’, have to do some analysis before dropping the column



FEATURE ENGINEERING :

Almost 50% of the data is missing in the column 'wh_est_year', have to do some analysis before dropping the column

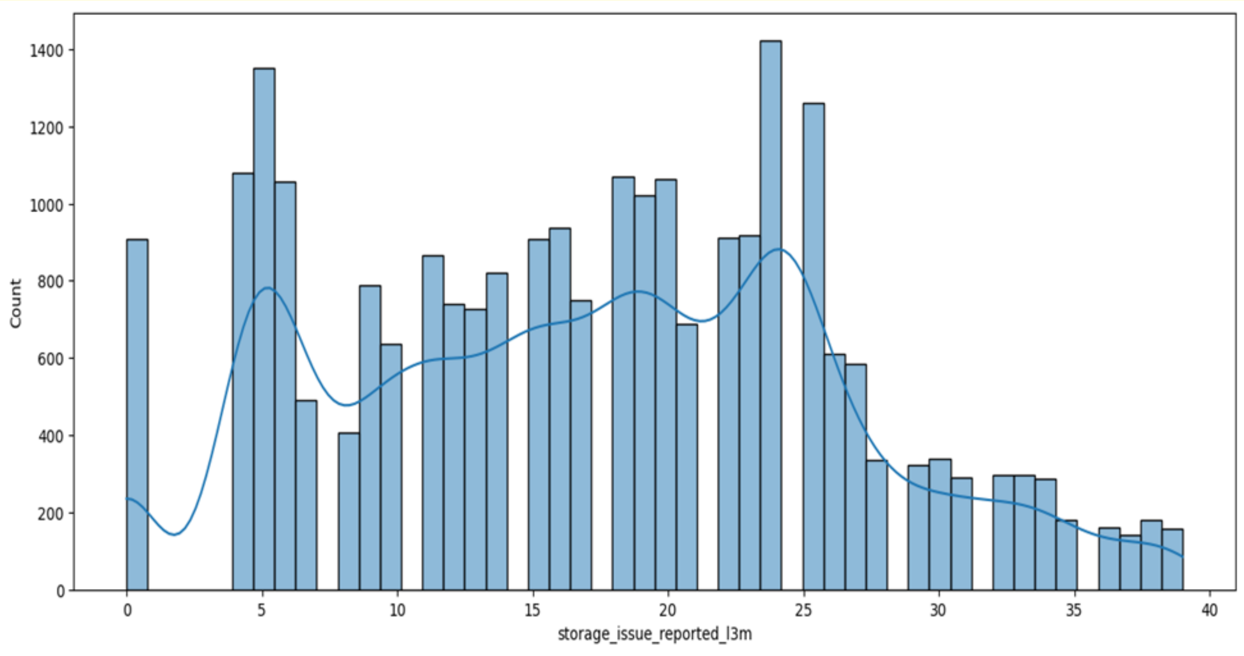


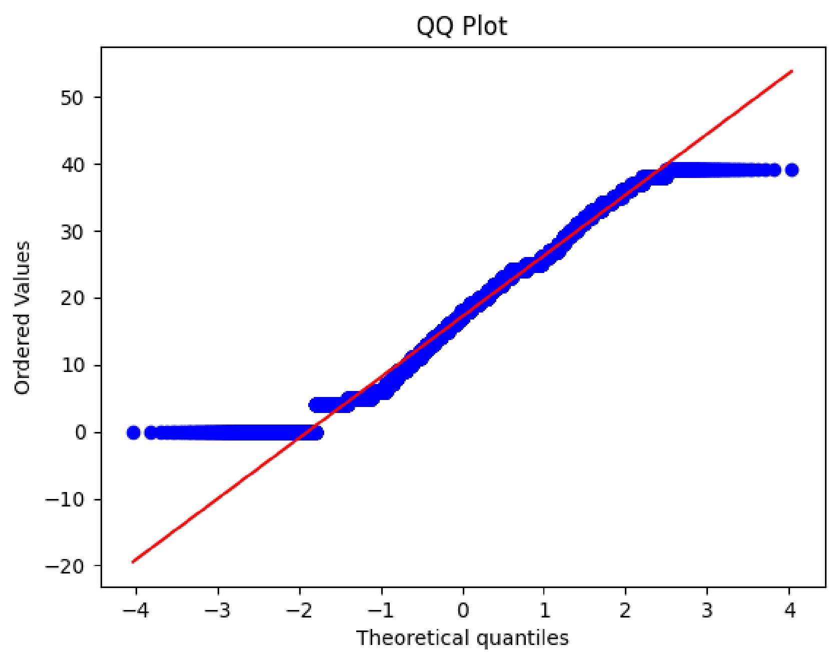


The column 'issue reported' is not normally distributed but showing positively co-related with the target variable

FEATURE ENGINEERING :

Calculated the skewness of the 'storage_issue_reported_l3m' column in the df DataFrame

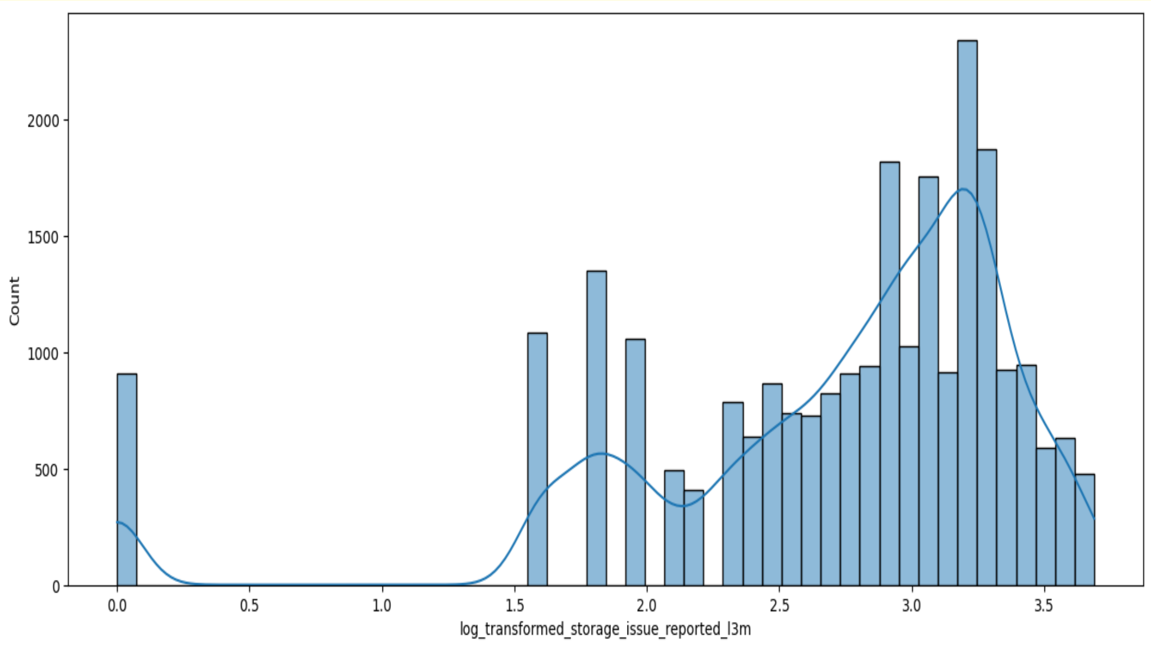


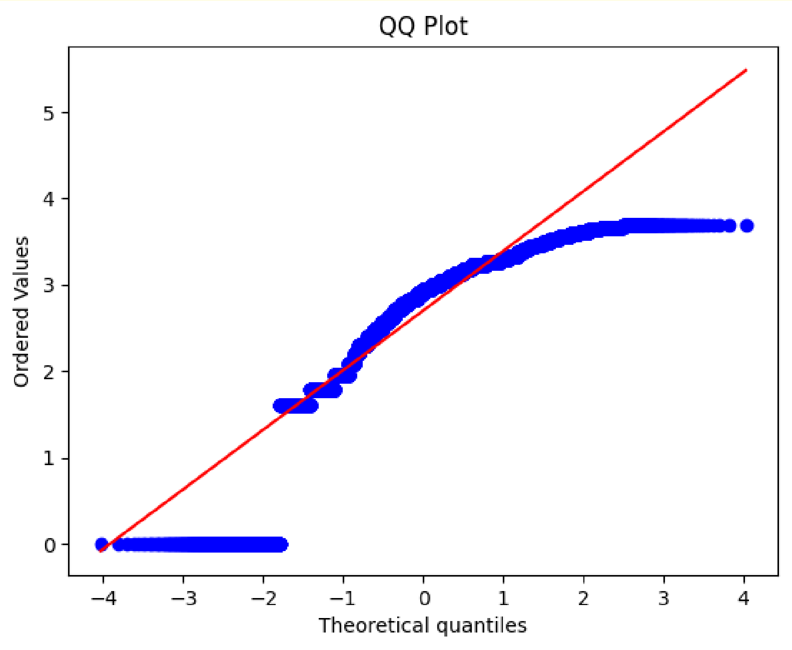



```
Skewness for 'storage_issue_reported_l3m ':
0.11333840770152336
```

FEATURE ENGINEERING :

Creating a histogram plot of the log-transformed values of a given column in a dataframe.



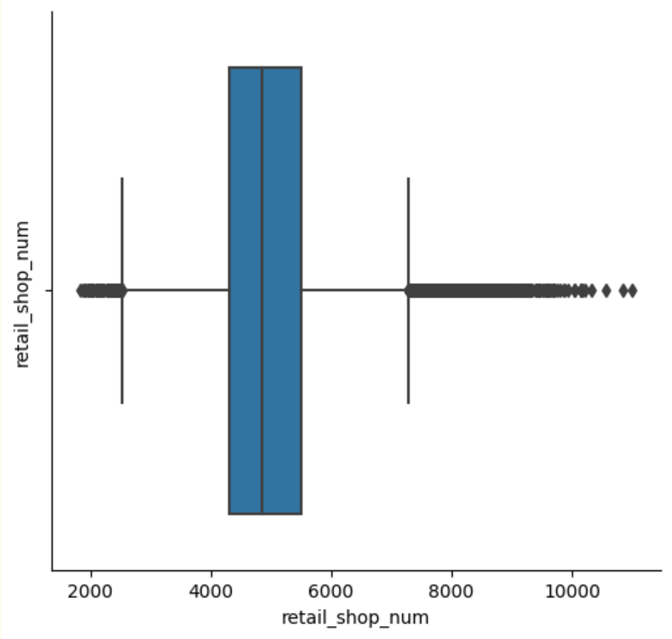


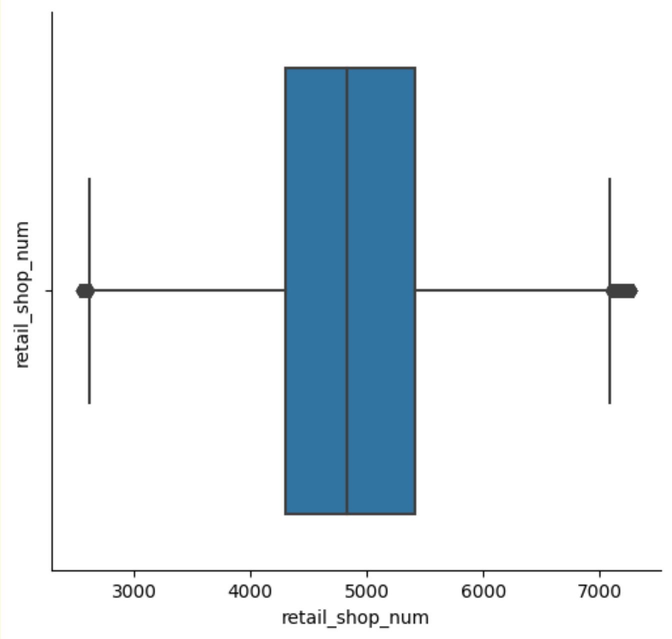

```
Skewness for log_transformed_storage_issue_reported_l3m ':
-1.7128569768953485
```

Log transformation is not showing the desired result

FEATURE ENGINEERING :

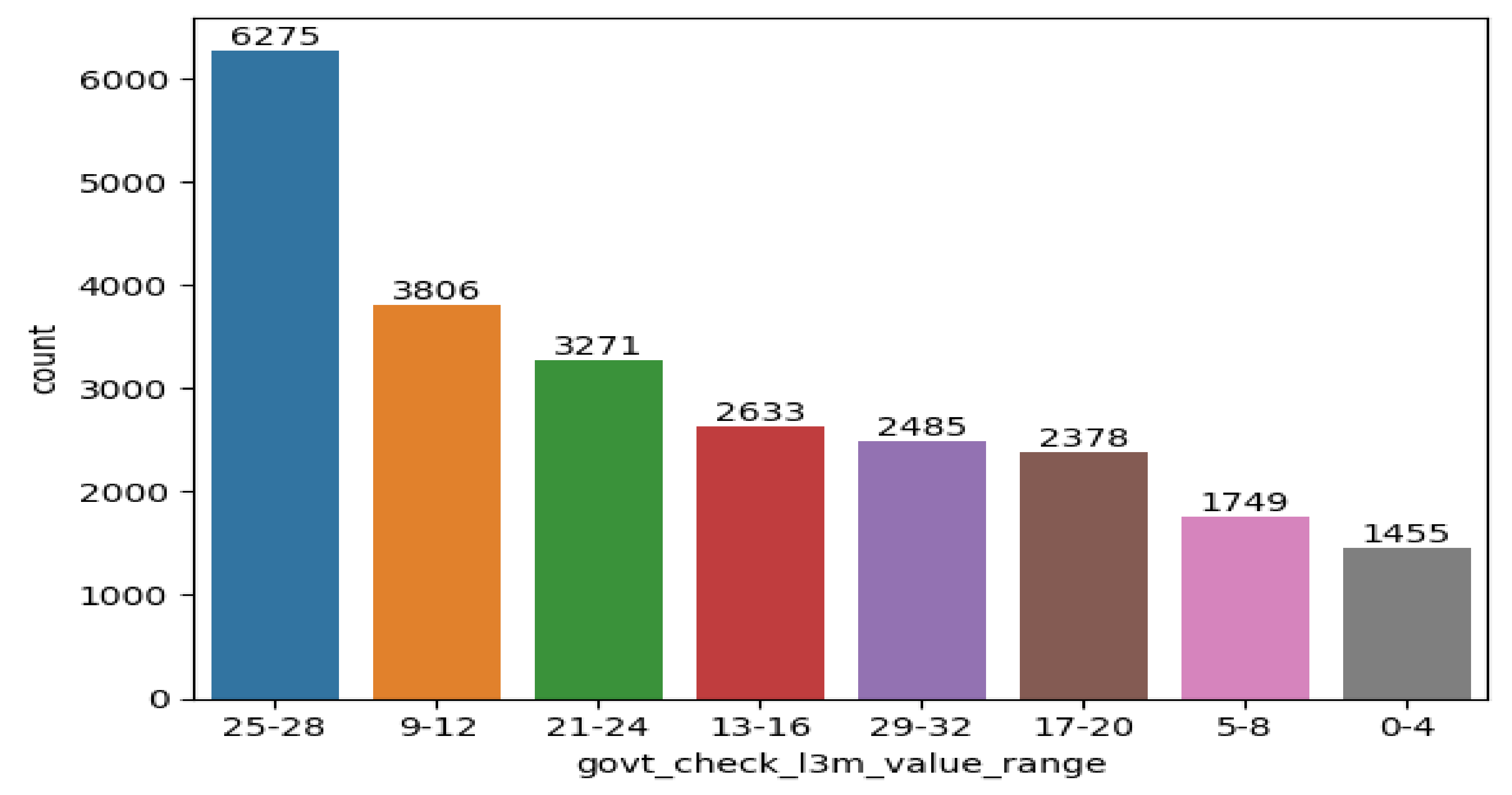
Remove outliers from a specified column ‘retail_shop_num’ in the given dataframe





FEATURE ENGINEERING :

Converting the column 'govt_check_l3m' values into ranges



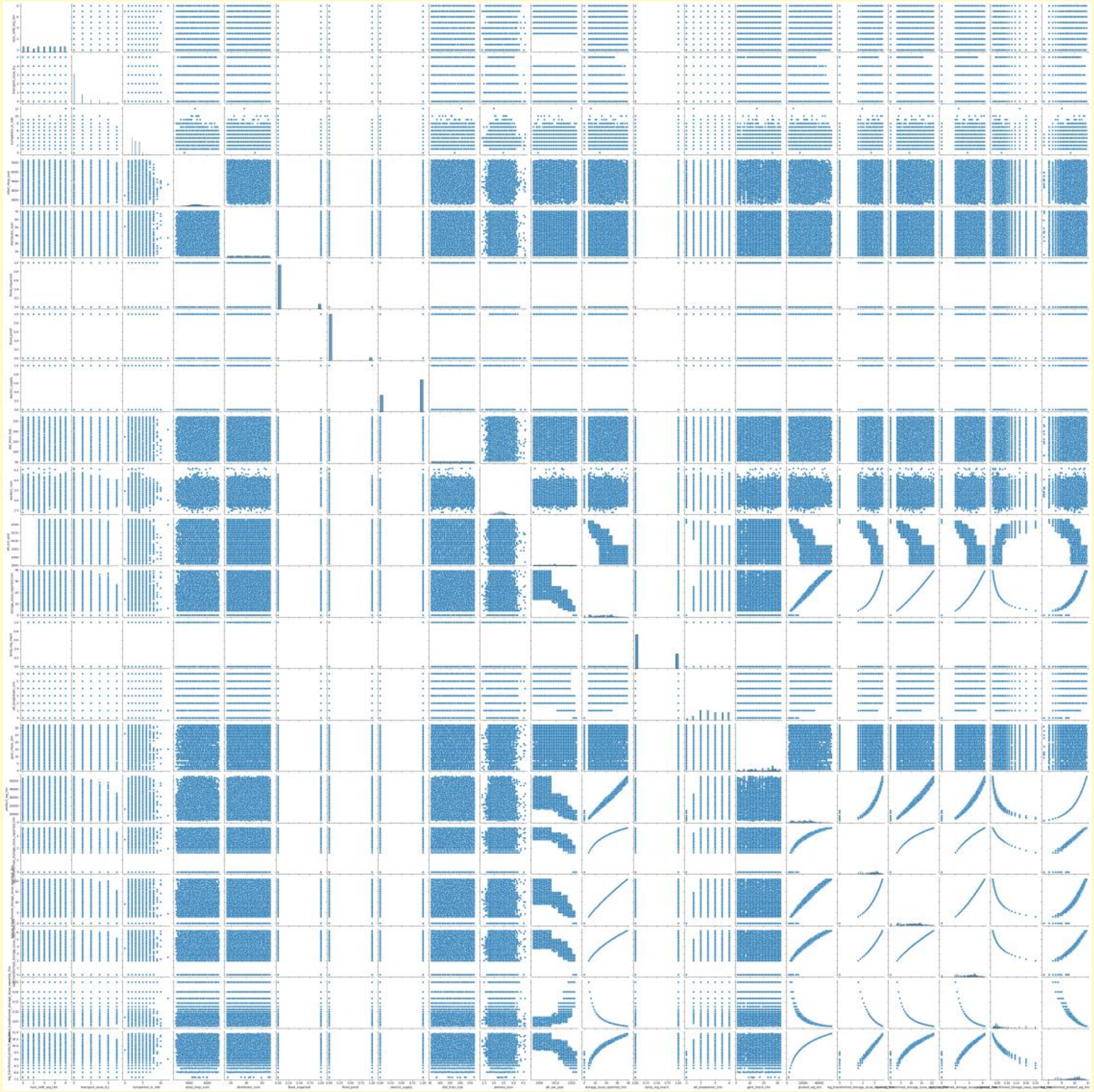
BIVARIATE ANALYSIS AND VISUALIZATION:

Calculated the correlation between the 'product_wg_ton' column of a DataFrame and the other columns.

```
num_refill_req_l3m                              0.000889
transport_issue_l1y                            -0.175692
Competitor_in_mkt                               0.009018
retail_shop_num                                -0.005454
distributor_num                                 0.005664
flood_impacted                                 -0.000007
flood_proof                                     0.003043
electric_supply                                -0.000614
dist_from_hub                                  -0.003984
workers_num                                    -0.010529
wh_est_year                                    -0.828937
storage_issue_reported_l3m                      0.986804
temp_reg_mach                                   0.101968
wh_breakdown_l3m                                0.341207
govt_check_l3m                                 -0.009690
product_wg_ton                                  1.000000
log_transformed_storage_issue_reported_l3m      0.847320
boxcox_transformed_storage_issue_reported_l3m   0.976860
sqrt_transformed_storage_issue_reported_l3m     0.926076
reciprocal_transformed_storage_issue_reported_l3m  -0.824716
log_transformed_product_wg_ton                  0.946093
Name: product_wg_ton, dtype: float64
```

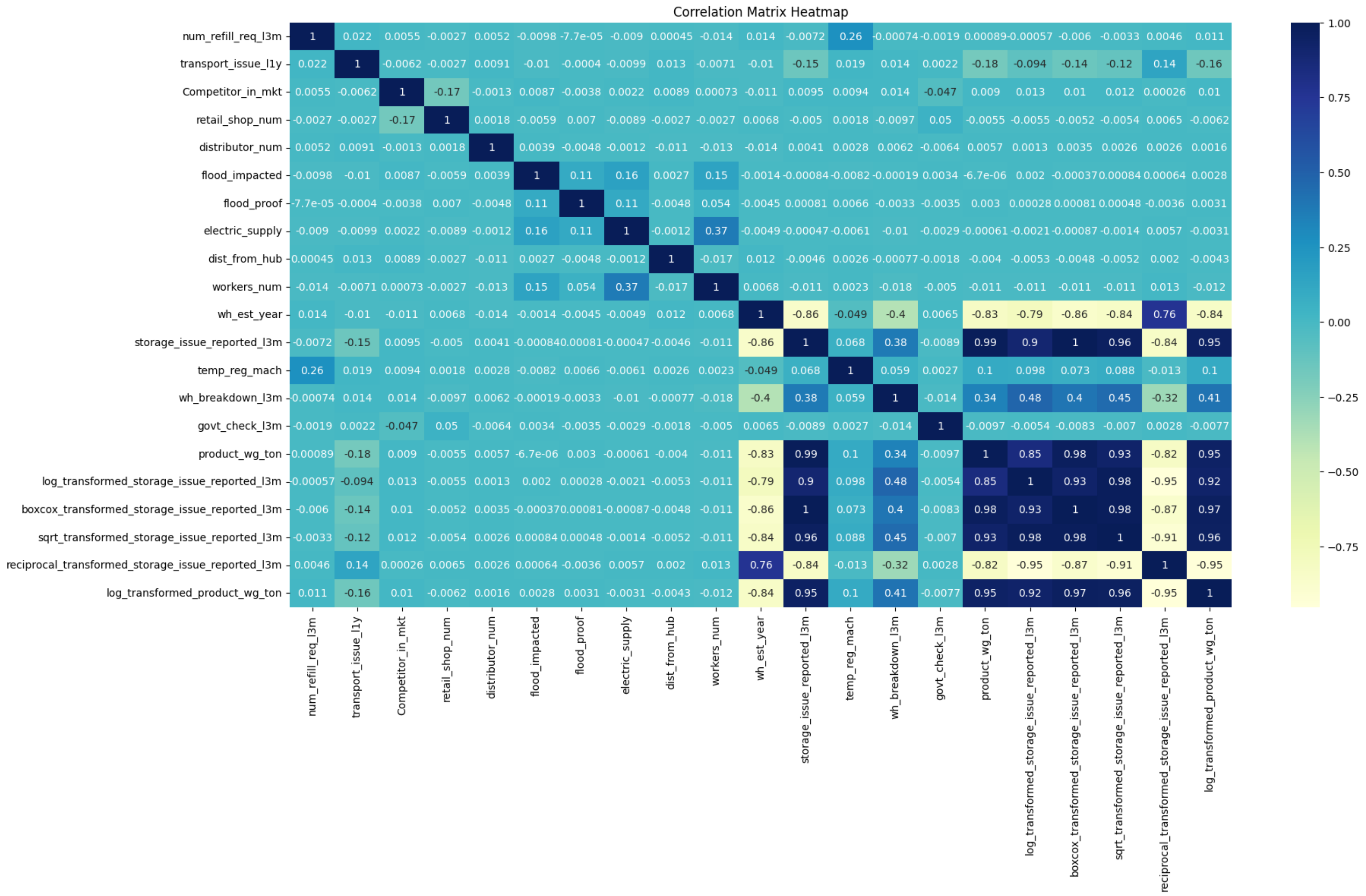
Output showing highcorrelation with the 'wh_est_year' and 'storage_issue_reported_l3m'

BIVARIATE ANALYSIS AND VISUALIZATION:



Plotted pairplot of dataframe columns

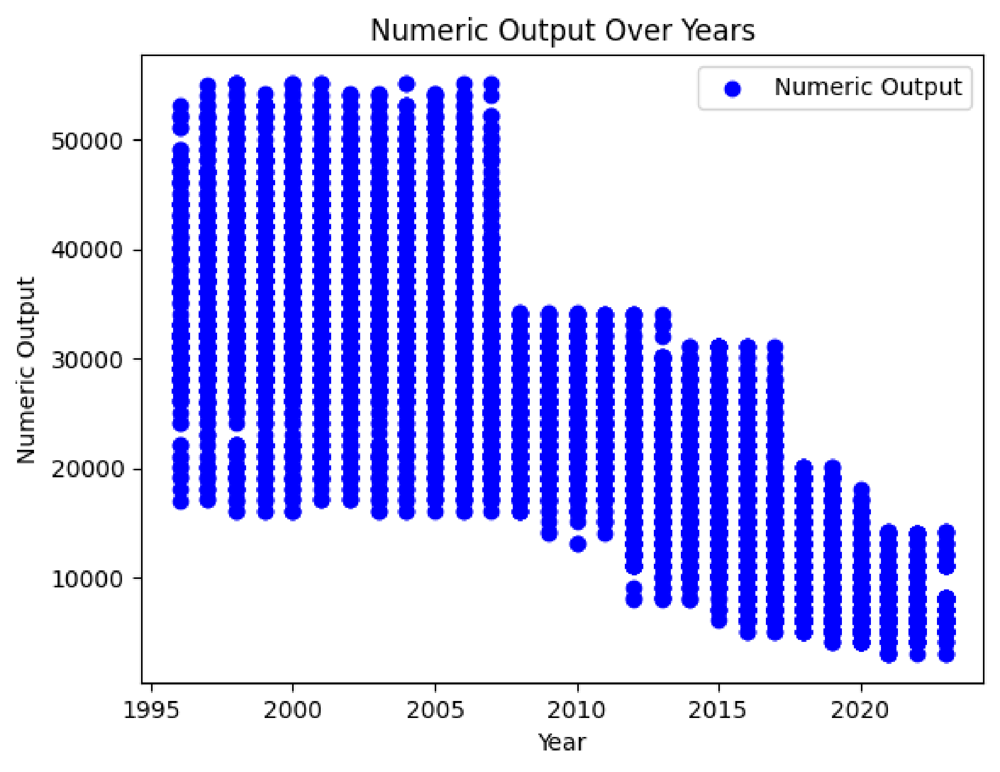
BIVARIATE ANALYSIS AND VISUALIZATION:

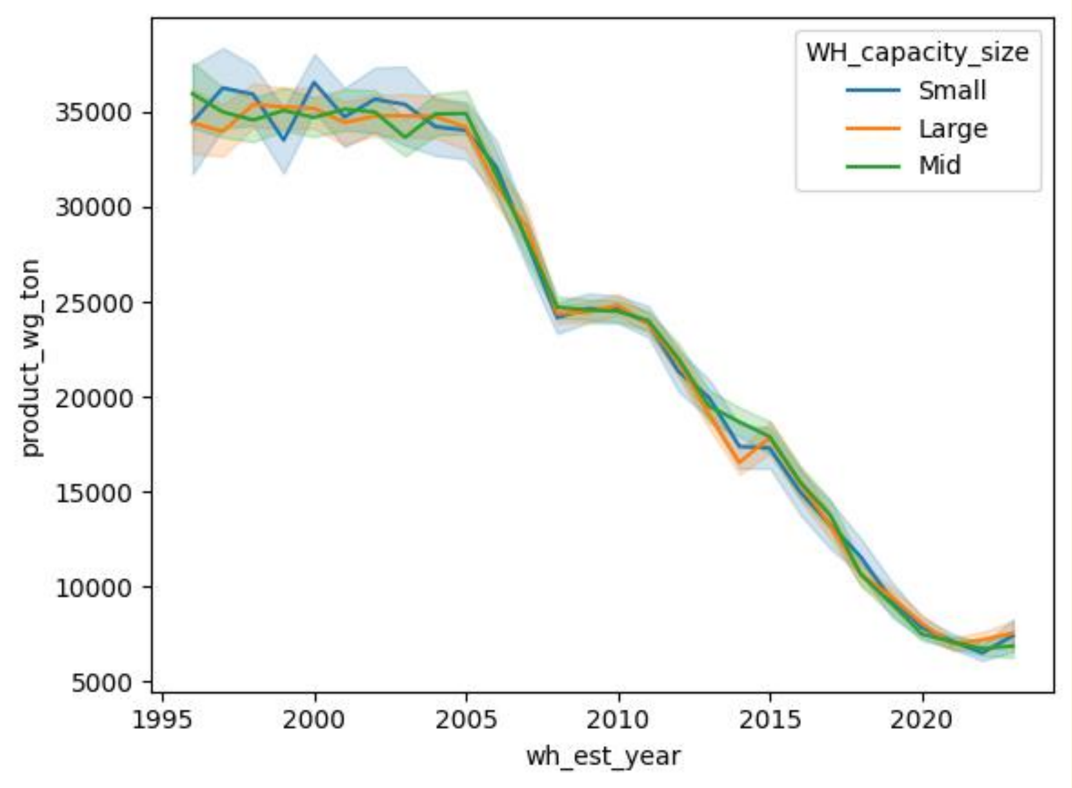


wh_est_year showing collinearity with storage_issue_reported_l3m

BIVARIATE ANALYSIS AND VISUALIZATION:

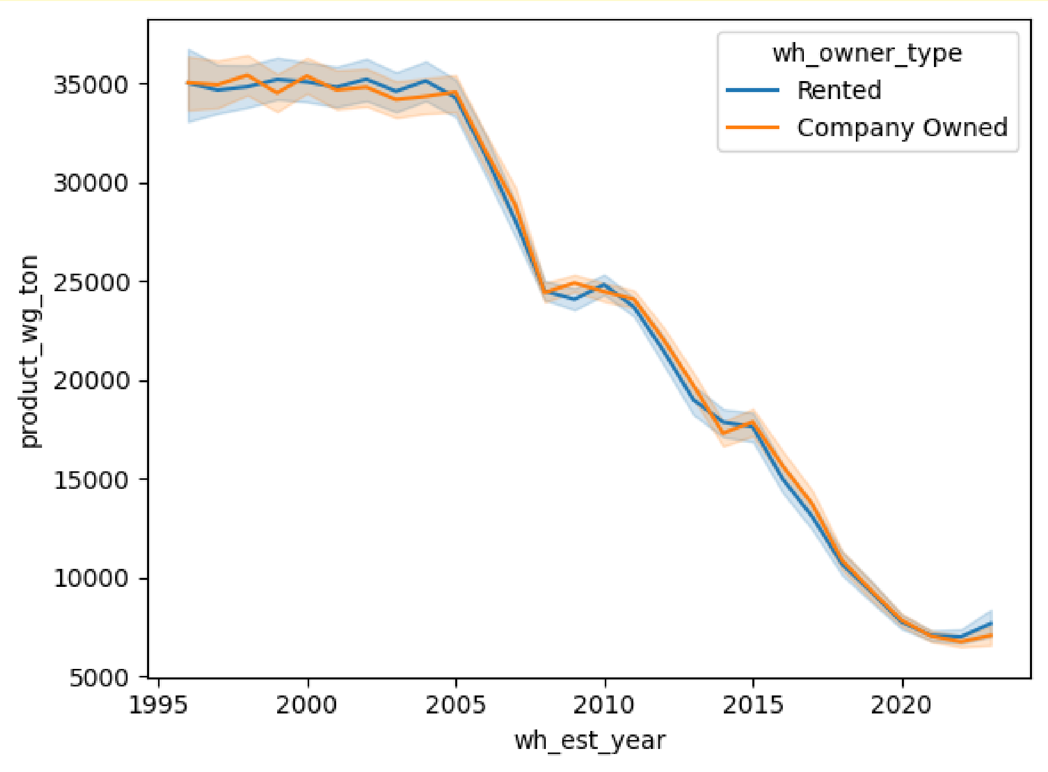
Scatter plot of 'wh_est_year' vs 'product_wg_ton'

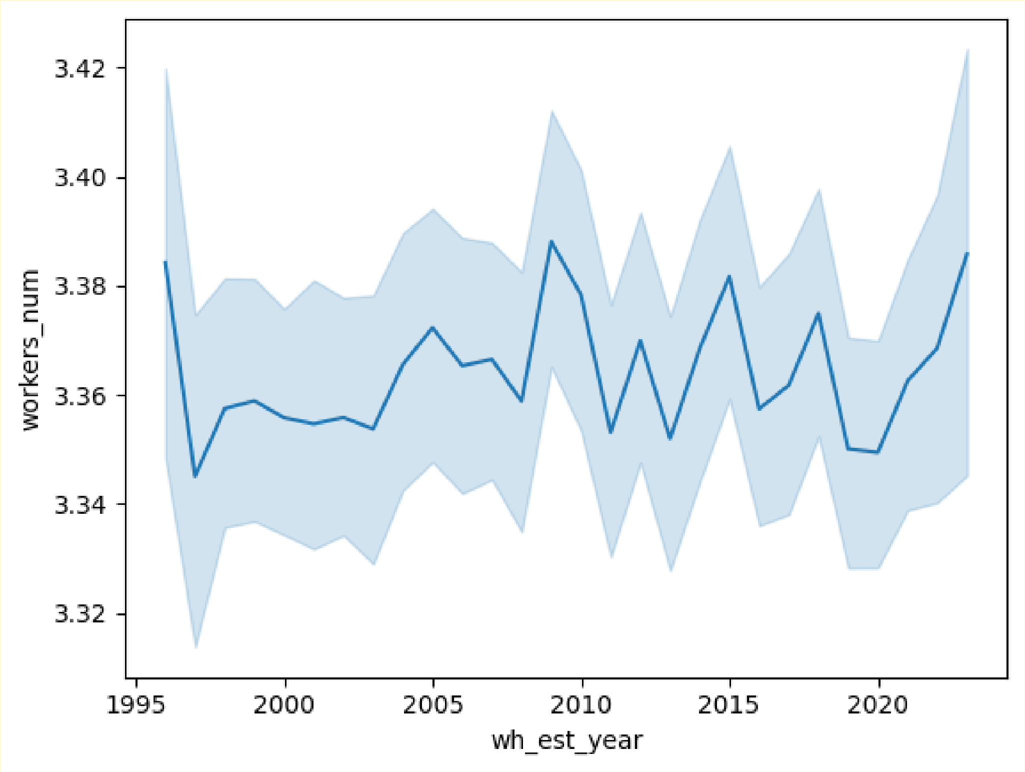




BIVARIATE ANALYSIS AND VISUALIZATION:

Scatter plot of 'wh_est_year' vs 'product_wg_ton'





FEATURE ENGINEERING :

Remove columns 'Ware_house_ID', 'WH_Manager_ID', and 'wh_est_year' and other unnecessary columns from the DataFrame

* Dropping wh_est_year, since it has 50% null value and showing multicollierity with storage_issue_reported_l3m

```
Location_type                                  0
WH_capacity_size                               0
zone                                           0
WH_regional_zone                               0
num_refill_req_l3m                             0
transport_issue_l1y                            0
Competitor_in_mkt                              0
retail_shop_num                                0
wh_owner_type                                  0
distributor_num                                0
flood_impacted                                 0
flood_proof                                    0
electric_supply                                0
dist_from_hub                                  0
workers_num                                    0
storage_issue_reported_l3m                     0
temp_reg_mach                                  0
approved_wh_govt_certificate                   0
wh_breakdown_l3m                               0
govt_check_l3m                                 0
product_wg_ton                                 0
distributor_num_Value_Range                    0
dist_from_hub_range_value                      0
log_transformed_storage_issue_reported_l3m     0
boxcox_transformed_storage_issue_reported_l3m  0
sqrt_transformed_storage_issue_reported_l3m    0
reciprocal_transformed_storage_issue_reported_l3m  0
log_transformed_product_wg_ton                 0
govt_check_l3m_value_range                     0
dtype: int64
```

MODELLING TECHNIQUES USED

- Linear Regression
- Decision Tree
- Random Forest

DATA PARTITIONING USED :

70 % training set, 30% validation set – try to account for overfitting of data

FEATURE ENGINEERING :

Encoding the categorical columns

Create an OrdinalEncoder with the specified order for column 'Encoded_WH_capacity_size', 'approved_wh_govt_certificate'

```
size_order = ['Small', 'Mid', 'Large']
from sklearn.preprocessing import OrdinalEncoder
# Create an OrdinalEncoder with the specified order
ordinal_encoder = OrdinalEncoder(categories=[size_order])

# Apply ordinal encoding to the 'Warehouse_Size' column
X_train['Encoded_WH_capacity_size'] =
ordinal_encoder.fit_transform(X_train[['WH_capacity_size']]
)
X_train
X_test['Encoded_WH_capacity_size'] =
ordinal_encoder.transform(X_test[['WH_capacity_size']])
X_test
```

```
df['approved_wh_govt_certificate'].value_counts()
# Define the reversed order of grades
grade_order = ['C', 'B+', 'B', 'A','A+']

# Create an OrdinalEncoder with the specified order
ordinal_encoder = OrdinalEncoder(categories=[grade_order])

# Apply ordinal encoding to the 'Grade' column
X_train['Encoded_approved_wh_govt_certificate'] =
ordinal_encoder.fit_transform(X_train[['approved_wh_govt_certif
icate']])

# Display the DataFrame with the encoded grades
X_test['Encoded_approved_wh_govt_certificate'] =
ordinal_encoder.transform(X_test[['approved_wh_govt_certificat
e']])
X_test
```

FEATURE ENGINEERING :

Encoding the categorical columns

Applying column transformer to apply One Hot Encoder for the remaining Categorical column

```
from sklearn.preprocessing import OneHotEncoder
transformer = ColumnTransformer(transformers = [
   ('trf1',OneHotEncoder(sparse=False,drop='first'),['zone','WH_regional_zone','Location_type',
                               'num_refill_req_l3m','transport_issue_l1y','Competitor_in_mkt',
                               'wh_owner_type','flood_impacted','flood_proof',
                               'electric_supply','temp_reg_mach','wh_breakdown_l3m','distributor_num_Value_Range',
                                        'dist_from_hub_range_value','govt_check_l3m_value_range'])],remainder='passthrough')
```

LINEAR REGRESSION

```
# Calculate R-squared
r_squared = model.score(X_test_scaled, y_test)

# Calculate adjusted R-squared manually
n = len(y_test)
k = X_test_scaled.shape[1]
adjusted_r_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))
mse = mean_squared_error(y_test, y_pred)

# Display the results
print('R-squared:', r_squared)
print('Adjusted R-squared:', adjusted_r_squared)
print('Mean squared error:', mse)
```

R-squared: 0.9856754322295243
Adjusted R-squared: 0.9855371177632266
Mean squared error: 1939592.1013149295

LASSO REGRESSION

```
# Calculate R-squared
r_squared = lasso_model.score(X_test_scaled, y_test)

# Calculate adjusted R-squared manually
n = len(y_test)
k = X_test_scaled.shape[1]
adjusted_r_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))
mse = mean_squared_error(y_test, y_pred)

# Display the results
print('R-squared:', r_squared)
print('Adjusted R-squared:', adjusted_r_squared)
print('Mean squared error:', mse)
```

R-squared: 0.9856759624653062
Adjusted R-squared: 0.9855376531188335
Mean squared error: 1939520.305701194

CROSS VALIATION-LINEAR REGRESSION

```
# Perforing cross validation to check the overfitting
from sklearn.model_selection import cross_val_score

model2 = LinearRegression()
cross_val_scores = cross_val_score(model2, X_train, y_train, cv=5,
scoring='r2')
mean_r2_score = np.mean(cross_val_scores)

print(f"Cross-validated R-squared scores: {cross_val_scores}")
print(f"Mean R-squared: {np.mean(cross_val_scores):.3f}")

# Calculate adjusted R-squared manually
n = len(y_test)
k = X_test_scaled.shape[1]
adjusted_r_squared = 1 - ((1 - mean_r2_score) * (n - 1) / (n - k - 1))
print('Adjusted R-squared:', adjusted_r_squared)
```

Cross-validated R-squared scores: [0.98489168 0.98492006 0.98576746 0.98473008 0.98530435]
Mean R-squared: 0.985
Mean Adjusted R-squared: 0.9849790768764007

DECISION TREE

HYPERPARAMETER: max_depth=8,min_samples_split=12,max_leaf_nodes = 5

```
# Calculate R-squared
r_squared = model.score(X_test_scaled, y_test)

# Calculate adjusted R-squared manually
n = len(y_test)
k = X_test_scaled.shape[1]
adjusted_r_squared = 1 - ((1 - r_squared) * (n - 1) / (n - k - 1))
mse = mean_squared_error(y_test, y_pred)

# Display the results
print('R-squared:', r_squared)
print('Adjusted R-squared:', adjusted_r_squared)
print('Mean squared error:', mse)
```

Mean Squared Error: 7452651.220737845
R-squared: 0.9449595575230462
Adjusted R-squared: 0.9444281006897255

CROSS VALIATION-DECISION TREE

```
# Perforing cross validation to check the overfitting
from sklearn.model_selection import cross_val_score

model4 =
DecisionTreeRegressor(max_depth=8,min_samples_split=12,max_leaf_n
odes = 5)
cross_val_scores = cross_val_score(model4, X_train, y_train, cv=10,
scoring='r2')
mean_r2_score = np.mean(cross_val_scores)

print(f"Cross-validated R-squared scores: {cross_val_scores}")
print(f"Mean R-squared: {np.mean(cross_val_scores):.3f}")

# Calculate adjusted R-squared manually
n = len(y_test)
k = X_test_scaled.shape[1]
adjusted_r_squared = 1 - ((1 - mean_r2_score) * (n - 1) / (n - k - 1))
print('Adjusted R-squared:', adjusted_r_squared)
```

```
Cross-validated R-squared scores: [0.94475998 0.94576191
0.94180174 0.94151096 0.94636355 0.9441197]
 0.9440495  0.94630972 0.94383012 0.94115034]
Mean R-squared: 0.944
Adjusted R-squared: 0.9434246986748428
Mean Squared Error: 7452651.220737845
```

RANDOM FOREST

HYPERPARAMETER: max_depth=8,min_samples_split=12,max_leaf_nodes = 5,n_estimators=10, random_state=42

```
# Create a Random Forest Regressor model
    rf_regressor = 
RandomForestRegressor(max_depth=8,min_samples_split=12,max
_leaf_nodes = 5,n_estimators=10, random_state=42)

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

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

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
adjusted_r_squared = 1 - ((1 - r2) * (n - 1) / (n - k - 1))
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
print('Adjusted R-squared:', adjusted_r_squared)
```

Mean Squared Error: 6785284.493684421
R-squared: 0.9498882948090748
Adjusted R-squared: 0.9494044286380456

COMPARISON OF R2 SCORE, ADJUSTED R2 SCORE AND MEAN SQUAED ERROR BETWEEN THE MODELLING TECHNIQUES

Algorithm	R-squared	Adjusted R-squared	Mean Squared Error
Linear Regression	0.9856754322295244	0.9856714603543627	1939592.1013149142
Lasso Regression	0.985676737194287	0.9856727656809622	1939415.4049296137
Cross-Validation Linear Regression	0.9856754322295244	0.9856714603543627	1939592.1013149142
Decision Tree	0.9449595575230462	0.9449442960666544	7452651.220737845
Cross-Validation Linear Regression	0.9449595575230462	0.9449442960666544	7452651.220737845
Random Forest	0.9498882948090748	0.949874399978854	6785284.493684421

INFERENCE

From the results obtained, it Linear Regression with Lasso Regularization showing optimum result to predict the optimum product weights

RECOMMENDATIONS

Regularly update the linear regression model as new data becomes available. This ensures that the model remains accurate and continues to reflect changes in the product characteristics and weights.

Integrate the linear regression model with the warehouse management system to automate the process of estimating and optimizing product weights. This allows for real-time decision-making and adaptability to changing conditions.

Utilize warehouse management systems (WMS) and other advanced technologies to automate and optimize inventory management. Automation can enhance accuracy, reduce picking time, and improve overall warehouse efficiency.

THANK YOU