🚘 UBER DATA ANALYSIS : using Machine Learning 🤖

Problem Statement ▶ *Uber is an international company located in 69 countries and around 900 cities around the world. Lyft, on the other hand, operates in approximately 644 cities in the US and 12 cities in Canada alone. However, in the US, it is the second-largest passenger company with a market share of 31%.*

🤔 **How does Uber price work?** If you request a ride on Saturday night, you may find that the price is different from the cost of the same trip a few days earlier. That's because of our dynamic pricing algorithm, which converts prices according to several variables, such as the time and distance of your route, traffic, and the current need of the driver. In some cases, this may mean a temporary increase in price during very busy times.

😳 **Why are Uber rates changing?** As demand increases, Uber uses flexible costs to encourage more drivers to get on the road and help address a number of passenger requests. When we inform you of an increase in Uber fees, we also inform drivers. If you decide to proceed and request your ride, you will receive a warning in the app to make sure you know that ratings have changed.

📌 **Business Problem▶** *Before you start managing and analyzing data, the first thing you should do is think about the PURPOSE. What it means is that you have to think about the reasons why you are going to do any analysis. If you are unsure about this, just start by asking questions about your story such as Where? What? How? Who? Which?*

📍 **How many times have I traveled in the past?**

📍 **How many trips were completed and canceled?**

📍 **What type of product is most often selected?**

📍 **What a measure. fare, distance, amount, and time spent on the ride?**

🤔 **Some explanations** :

> 🖋 **Correlation matrix** is a table showing **correlation coefficients between variables**. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.
>
> - The **magnitude of the correlation coefficient** *indicates* the **strength of the association**. For example, a correlation of $\mathbf{r = 0.9}$ suggests a **strong** , positive association between two variables, whereas a correlation of $\mathbf{r = -0.2}$ suggest a **weak** , negative association.

> - **Pairplot** is used to understand the best set of features to explain a **relationship between two variables** or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or make linear separation in our data-set.
> - **StandardScaler** removes the **mean** and scales each feature/variable to unit variance. This operation is performed feature-wise in an independent way. **StandardScaler** can be influenced by outliers (if they exist in the dataset) since it involves the

estimation of the empirical mean and standard deviation of each feature. In Machine Learning, StandardScaler is used to resize the distribution of values so that the mean of the observed values is 0 and the standard deviation is 1

- $\mathbf{KNN}$: The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, you can use the KNN algorithm for applications that require high accuracy but that do not require a human-readable model. The quality of the predictions depends on the distance measure.
- $\mathbf{RFE}$: **Recursive Feature Elimination**, or RFE for short, is a popular feature selection algorithm. RFE is popular because it is easy to configure and use and because it is **effective at selecting those features** (columns) in a training dataset that are **more or most relevant in predicting the target variable**. RFE can be used to handle problems presented by the two models listed below :
    - **Classification** : Classification predicts the class of selected data points. ...
    - **Regression** : Regression models supply a function describing the relationship between one (or more) independent variables and a response, dependent, or target variable.

🖍 $\mathbf{Linear\ Regression}$ is a **regression model** that estimates the **relationship** between one **independent variable** and one **dependent variable** using a straight line. Both variables should be *quantitative*. When we talk of linearity in **linear regression**, we mean **linearity** in parameters. So evenif the **relationship** between response **variable** & **independent** variable is not a straight line but a curve,we can still fit the relationship through linear regression using higher order variables. $LogY = a + bx$ which is **linear regression**. **Regression analysis** allows us to understand the **strength of relationships between variables**. Using statistical measurements like $\frac{R-squared}{adjustedR-squared}$, regression analysis can tell us how much of the **total variability in the data is explained by our model**.

🖍- $\mathbf{Decision\ Tree}$ is a type of **supervised machine learning** used to **categorize or** make predictions** based on how a previous set of questions were answered. The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization. The goal of using a $\mathbf{Decision\ Tree}$ is to create a training model that can use to **predict the class or value of the target variable** by learning simple decision rules inferred from prior data(training data).

- **One-hot encoding** is an important step for preparing our dataset for use in machine learning. **One-hot encoding** turns your categorical data into a binary vector representation . Pandas get dummies makes this very easy!
    - This means that for each unique value in a column, a new column is created. The values in this column are represented as 1s and 0s, depending on whether the value matches the column header.
    - For example, with the help of the get_dummies function, we turn this table below :

Gender
Male
Female
Male
Male

- To this :

Gender	Male	Female
Male	1	0
Female	0	1
Male	1	0
Male	1	0

> How do we **evaluate our model ?**
>
> - After training the model we then apply the evaluation measures to check how the model is performing. Accordingly, we use the following evaluation parameters to check the performance of the models respectively :
>
> - **Accuracy Score** : Typically, the accuracy of a predictive model is good (above 90% accuracy)
>
> - **Execution time** : Variable that depends on the machine on which the program was executed, but which can give a small idea of the model that executes the fastest.

- **F1 score** : The $F1 - score$ is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0 . **F1 scores are lower than accuracy measures** as they embed precision and recall into their computation.

- **ROC-AUC Curve** : The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The **higher** the AUC, the **better** the performance of the model at distinguishing between the positive and negative classes.

- **Confusion Matrix with Plot** : A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. Note that :
  - **Actual values** are the columns.
  - **Predicted values** are the lines.

	Positive	Negative
Positive	TP	TN
Negative	FP	TN

DATA SET INFORMATION

👉 **trip_completed_at**

👉 **trip_status**

👉 **ride_hailing_app**

👉 **trip_uid**

👉 **driver_uid**

- 👉 rider_uid
- 👉 customer
- 👉 trip_start_time
- 👉 trip_end_time
- 👉 trip_time
- 👉 total_time
- 👉 wait_time
- 👉 trip_type
- 👉 surge_multiplier
- 👉 vehicle_make_model
- 👉 driver_name_en
- 👉 vehicle_make
- 👉 vehicle_model
- 👉 driver_gender
- 👉 driver_photo_url
- 👉 driver_phone_number
- 👉 pickup_lat
- 👉 pickup_long
- 👉 dropoff_lat
- 👉 dropoff_long
- 👉 trip_map_image_url

- 👉 trip_path_image_url
- 👉 city
- 👉 country
- 👉 trip_start_address
- 👉 trip_end_address
- 👉 rub_usd_exchange_rate
- 👉 price_rub
- 👉 price_usd
- 👉 distance_kms
- 👉 temperature_time
- 👉 temperature_value
- 👉 feels_like
- 👉 humidity
- 👉 wind_speed
- 👉 cloudness
- 👉 weather_main
- 👉 weather_desc
- 👉 precipitation

TOPICS

📌 **Reading the data**

📌 **Data cleaning**

📌 **Data Profiling**

📌 **Data Processing**

📌 **Data Visualization**

📌 **Datetime Operation**

Importing libraries :

```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression
#import statsmodels.api as sm
from sklearn.feature_selection import RFE
#from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

READING THE DATA

```
dataset = pd.read_csv("uber_dataset.csv")
dataset.head()
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
0	May 11, 2015 at 6:55PM	Completed	Uber	ee89076fd9da9bddf5f096b0ca42f8d5	05cfeb269e606247fe9d2b6082942c59	3ffa4a7
1	May 11, 2015 at 8:12PM	Completed	Uber	518be51d403944a03c47e8d1f2c87311	4a4e248742f9d5ff517c5bbbb48d0e54	3ffa4a7
2	May 13, 2015 at 11:38AM	Completed	Uber	6e460cc8a12c3c6568d0d4a67ac58393	cb249a2bd807ca78697b4ed0348c37da	3ffa4a7
3	May 16, 2015 at 1:44AM	Completed	Uber	49613a86a04e6c15d72b51d1a2935d81	d3f73f8151c2e8c34b541f961db7f5fa	3ffa4a7
4	May 16, 2015 at 3:18AM	Completed	Uber	9896148fdecdb4c5d977a8691510bdb6	1287d21e6455ee40d4861f6b91c680f4	3ffa4a7

5 rows × 45 columns

View first three row 👇

```
dataset.head(3)
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
0	May 11, 2015 at 6:55PM	Completed	Uber	ee89076fd9da9bddf5f096b0ca42f8d5	05cfeb269e606247fe9d2b6082942c59	3ffa4a7
1	May 11, 2015 at 8:12PM	Completed	Uber	518be51d403944a03c47e8d1f2c87311	4a4e248742f9d5ff517c5bbbb48d0e54	3ffa4a7
2	May 13, 2015 at 11:38AM	Completed	Uber	6e460cc8a12c3c6568d0d4a67ac58393	cb249a2bd807ca78697b4ed0348c37da	3ffa4a7

3 rows × 45 columns

Shape of the dataset 👇

```
dataset.shape
```

```
(678, 45)
```

Dataset size 👇

```
dataset.size
```

```
30510
```

check info about data 👇

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 678 entries, 0 to 677
Data columns (total 45 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   trip_completed_at   678 non-null    object
```

1	trip_status	678 non-null	object
2	ride_hailing_app	678 non-null	object
3	trip_uid	678 non-null	object
4	driver_uid	678 non-null	object
5	rider_uid	678 non-null	object
6	customer	678 non-null	object
7	trip_start_time	678 non-null	object
8	trip_end_time	678 non-null	object
9	trip_time	678 non-null	object
10	total_time	678 non-null	object
11	wait_time	678 non-null	object
12	trip_type	678 non-null	object
13	surge_multiplier	643 non-null	float64
14	vehicle_make_model	678 non-null	object
15	vehicle_license_plate	678 non-null	object
16	driver_name_en	678 non-null	object
17	vehicle_make	678 non-null	object
18	vehicle_model	678 non-null	object
19	driver_gender	678 non-null	object
20	driver_photo_url	678 non-null	object
21	driver_phone_number	678 non-null	object
22	pickup_lat	678 non-null	float64
23	pickup_long	678 non-null	float64
24	dropoff_lat	678 non-null	float64
25	dropoff_long	678 non-null	float64
26	trip_map_image_url	678 non-null	object
27	trip_path_image_url	678 non-null	object
28	city	678 non-null	object
29	country	678 non-null	object
30	trip_start_address	678 non-null	object
31	trip_end_address	678 non-null	object
32	rub_usd_exchange_rate	678 non-null	float64
33	price_rub	678 non-null	object
34	price_usd	678 non-null	float64
35	distance_kms	678 non-null	float64
36	temperature_time	678 non-null	object
37	temperature_value	678 non-null	int64
38	feels_like	678 non-null	int64
39	humidity	678 non-null	float64
40	wind_speed	678 non-null	float64
41	cloudness	678 non-null	object

```
 42  weather_main          678 non-null    object
 43  weather_desc          678 non-null    object
 44  precipitation         678 non-null    object
dtypes: float64(10), int64(2), object(33)
memory usage: 238.5+ KB
```

DATA CLEANING

```
dataset['surge_multiplier'].isnull()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
673    False
674    False
675    False
676    False
677    False
Name: surge_multiplier, Length: 678, dtype: bool
```

```
dataset[dataset['surge_multiplier'].isnull()].head(5)
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
20	June 12, 2015 at 6:01PM	Completed	Gett	c9df3a9edb2c7b37c80f9ffa5e1b8c36	151944a8f9967b3edad02deb712d61f2	3ffa4
135	March 5, 2016 at 4:35PM	Completed	Gett	19564cbd0929d51297a5fce739a5c777	524f217db5cb84a0c343cb7f05f40f52	3ffa4
137	March 17, 2016 at 12:57PM	Completed	Gett	2f7ca9c163bb40c5e5b49771e929075a	af6d53a1a051b89037613fa74b0a2039	3ffa4
[illegible]	April 3, 2016 at [illegible]	Completed	Gett	[illegible]	[illegible]	[illegible]
199	July 29, 2016 at [illegible]	Completed	Gett	1e4023aecad6062b038656b9bec5b433	e75db42065bc8e071d2b57c0c9977376	3ffa4

Filtering based on conditions

dataset[dataset['trip_status'] !='completed'].head(50)

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
0	May 11, 2015 at 6:55PM	Completed	Uber	ee89076fd9da9bddf5f096b0ca42f8d5	05cfeb269e606247fe9d2b6082942c59	3ffa4
1	May 11, 2015 at 8:12PM	Completed	Uber	518be51d403944a03c47e8d1f2c87311	4a4e248742f9d5ff517c5bbbb48d0e54	3ffa4
2	May 13, 2015 at 11:38AM	Completed	Uber	6e460cc8a12c3c6568d0d4a67ac58393	cb249a2bd807ca78697b4ed0348c37da	3ffa4
3	May 16, 2015 at 1:44AM	Completed	Uber	49613a86a04e6c15d72b51d1a2935d81	d3f73f8151c2e8c34b541f961db7f5fa	3ffa4
4	May 16, 2015 at 3:18AM	Completed	Uber	9896148fdecdb4c5d977a8691510bdb6	1287d21e6455ee40d4861f6b91c680f4	3ffa4
5	May 18, 2015 at 11:06AM	Completed	Uber	5c0312a92ff104197d799c42ae67542f	fc6b1516376f15c97e508d904505d27a	3ffa4
6	May 18, 2015 at 11:08PM	Completed	Uber	4ad2e954813b53afeb73ce659ac3376c	1b926e88a8477f7b5d1fad298e00fb11	3ffa4
7	May 19, 2015 at 9:10AM	Completed	Uber	1e3935b05addc654d65e72b8da96fd43	439ae2cf8ae38bc24b2f8dbc3f0b987d	3ffa4
8	May 19, 2015 at 12:37PM	Completed	Uber	0eb9a9f7a3fd598c885c67af75645c06	75a4c47c323bcc96ac5849052b19ed5f	3ffa4
9	May 19, 2015 at 10:33PM	Completed	Uber	b56495d149fea002e04438a3369ab532	176f50c4249ddc7b086d8997349d9ae5	3ffa4
10	May 20, 2015 at 11:08AM	Completed	Uber	613f3deb5164333956 0bc9041184e810	e516233997d15dedf0f12878fd231e23	3ffa4
11	May 31, 2015 at 2:27PM	Completed	Uber	0d486aced52a47da5cf9e09757a964e7	a81952458c5e4c7b62a0bacf591dd690	3ffa4
12	June 1, 2015 at 11:45PM	Completed	Uber	d12da2c7ae96bab237bba8823c3f9c74	f065223fa83baa7b2e0037f61b8ebac7	3ffa4

13	June 3, 2015 at 4:14PM	Completed	Uber	36695e9088a840d3f7476e86294aa846	b897afbe685f70aa5114bf4f6a19392b	3ffa4
14	June 3, 2015 at 5:20PM	Completed	Uber	30bd4a26ca276b8f04a01d87ea3777fc	ef33153fbce5b15f93d9319528dc598f	3ffa4
15	June 4, 2015 at 4:34AM	Completed	Uber	3e955966177bc5b2d97fc9a239c561ef	b6584dd094679d58c938511d478169d6	3ffa4
16	June 5, 2015 at 4:31PM	Completed	Uber	4669e3d96401bf985ecf6ee9fd78a816	c85086781a9a41cb76345374fb11360a	3ffa4
17	June 6, 2015 at 2:59PM	Completed	Uber	a7e321acb3162ea301b42939c7ca7394	d505ef8cb3f776ca6acccd10b3fdbf78	3ffa4
18	June 7, 2015 at 9:48PM	Completed	Uber	8e9c9603d9abce84a08458da0c718698	97e28e41c95e7cbf1e36d6737d0eeefe	3ffa4
19	June 12, 2015 at 5:25PM	Completed	Uber	e01e7067e5cbbda3a9eb621a1268e8d0	767a9c87d435285ab26f73a14abbc8d2	3ffa4
21	June 14, 2015 at 7:44AM	Cancelled	Uber	b970e99cf7d4e163c9808ed945f9e5be	a89bc69e8452fcec3f650b688f6e3473	3ffa4
22	June 14, 2015 at 11:18AM	Completed	Uber	945269c9502d99bc6065d9c8166760b1	2f5f40db716f7d90f664a61e42a57c46	3ffa4
23	June 14, 2015 at 1:39PM	Completed	Uber	04f14fe72ad64ecd1f62d8bfb5fb2ba1	6b3642ae75aaf7e19a5b525ffeeeaa69	3ffa4
24	June 14, 2015 at 8:31PM	Completed	Uber	fdbd229175bc4d808d52df176debf210	69f4e9a8758cf75e5dbc8fb9b80a5bd2	3ffa4
25	June 14, 2015 at 10:41PM	Completed	Uber	d4984dfe402ec968a2365649c478de41	e1992ce90ac08ca0d5afa13da214e187	3ffa4
	June 14, 2015 at					

26	June 14, 2015 at 11:19PM	Completed	Uber	57bd0828f1a6d5e812b0ca38cab4231b	b2cadbd6ea7ae0f6b30e2968652830d7	3ffa4
27	June 14, 2015 at 12:46AM	Completed	Uber	268145a88a53bbb369a1e812be234908	a45d6a74641c9079c0a0f91d34446aa1	3ffa4
28	June 15, 2015 at 10:48AM	Completed	Uber	9622fd4e1819f8656c418ffdfcd359a8	78aab4ce1d6f0c1e0acf0a1b0b933ce6	3ffa4
29	June 15, 2015 at 11:10AM	Completed	Uber	72801dd197d168a38ebaa88875492d60	3613586d80fac02ff4bbbb4ce24fed87	3ffa4
30	June 15, 2015 at 12:37PM	Completed	Uber	f374f9aad40c4f0a40daf9ac5902c7c2	a4c34b4a8caa9702feab8271a387f4e6	3ffa4
31	June 15, 2015 at 1:57PM	Completed	Uber	a37a5b276b8e1052844a1c87ccad8619	4ef6986a59b30bf2d117db7ce2ee37ff	3ffa4
32	June 15, 2015 at 2:19PM	Completed	Uber	8503882e3710dd9da0a8450c1ef0e781	93b85d4beea9fe816e151eaa13be1617	3ffa4
33	June 15, 2015 at 3:30PM	Completed	Uber	e47f8622a543e241ed5eca1f7eb132ea	abc979ec5c04d6528b77c7c897348a59	3ffa4
34	June 15, 2015 at 8:21PM	Completed	Uber	e4a75fb0e38b78f004306f507e9d1dd1	bd364948535b2c2f46335f4e4c9c2fa4	3ffa4
35	June 15, 2015 at 11:52PM	Completed	Uber	7357d950162556ec2632b238ab8dc09f	301b8ee2b154ec6b8351cc5873ab8152	3ffa4
36	June 16, 2015 at 12:08PM	Completed	Uber	d3e0518f0d060a6ecba2eb8a76607a46	121a2fc51989885257e5c2748789be0f	3ffa4
37	June 16, 2015 at 12:20PM	Completed	Uber	02b75305097bdd0d3c96c1c200cc4daa	121a2fc51989885257e5c2748789be0f	3ffa4
38	June 16, 2015 at 1:27PM	Completed	Uber	6320e1c1f99ef5a9d216b9d89e05cad0	3d3f28ed813e3758629e4bcb292a60c0	3ffa4
39	June 16, 2015 at 10:13PM	Completed	Uber	cb06523d1847d26a43f897c6ac32afd2	66cb6d76e534f519761e68a2ff12b53b	3ffa4
40	June 17, 2015 at 6:59AM	Completed	Uber	94ff538a9dcf0c637e60681582036a6e	9d9dbe8400b3d1c13672822764ca10db	3ffa4

41	June 17, 2015 at 8:59AM	Completed	Uber	9e4c25f0043af40d117e2b6a1a765131	1ca0a7c5fbd2a75b7dcc6498952665ce	3ffa4
42	June 17, 2015 at 1:09PM	Completed	Uber	ad25b01455efdc88f1a1f299bf742629	c7a97f8bd6d3b77efea17756780a6e47	3ffa4
43	June 17, 2015 at 2:05PM	Completed	Uber	03210d1c2a6c153ca27d799e061828c2	6410050a3993adfcfc29b9f422bceb51	3ffa4
44	June 17, 2015 at 2:59PM	Completed	Uber	e912743c9ab9bc7aad704d7e29e4cf70	fc8157aa6901aec3388acb9728ccd713	3ffa4
45	June 17, 2015 at 5:35PM	Completed	Uber	f981dcaa5c7120490cdd38357b78c79c	d3f73f8151c2e8c34b541f961db7f5fa	3ffa4
46	June 29, 2015 at 3:28PM	Completed	Uber	910b27c47f7185f5cbc29f46e0266d78	a7d863211c776977bd215afaa7ccd183	3ffa4
47	July 7, 2015 at 9:05PM	Completed	Uber	e4004a6b46e6e1556f76f63f295a3f37	bbe2b7c592ac0d8c3d2ad22875240994	3ffa4
48	July 8, 2015 at 2:16AM	Completed	Uber	e49979ce9755949bbcd19a57bcbc2dd6	e44cceba36c401b2c41333d6981dfdf6	3ffa4
49	July 9, 2015 at 2:33AM	Completed	Uber	4d92ff7d2ecbae299bb2c85aeeaca576	72d4ed2f0ccfb8ff1e50b7f8bbe461d1	3ffa4
50	September 6, 2015 at 4:26PM	Completed	Uber	0697b6dd10f0c3bc63019e22948451d9	bb10f17be28fe5fb311a4ebccf826d77	3ffa4

50 rows × 49 columns

DATA PROFILING

Unique trip_start_address ▶

```
dataset.trip_start_address.unique()
```

```
'Prospekt Yuriya Gagarina, 28, Sankt-Peterburg, Russia, 196135',
'Rubinstein St, 38, Sankt-Peterburg, Russia, 191002',
'English Embankment, 70, Sankt-Peterburg, Russia, 190121',
'Shosse Revolyutsii, 3к1, Sankt-Peterburg, Russia, 195027',
'Ligovsky Ave, 30 A, Sankt-Peterburg, Russia, 191040',
'Shosse Revolyutsii, 3, Sankt-Peterburg, Russia, 195027',
"Industrial'nyy Prospekt, 19, Sankt-Peterburg, Russia, 195426",
'Ulitsa Kollontay, 1, Sankt-Peterburg, Russia, 193230',
'Prospekt Kosygina, 4, Sankt-Peterburg, Russia, 195279',
'Ligovsky Ave, 16, Sankt-Peterburg, Russia, 191040',
'Prospekt Energetikov, 9к1, Sankt-Peterburg, Russia, 195248',
'Litovskaya Ulitsa, 2, Sankt-Peterburg, Russia, 194353',
'Okhtinskiy Park, Leningrad Oblast, 188664',
'Ligovsky Ave, 30литА, Sankt-Peterburg, Russia, 191040',
'Kirochnaya Ulitsa, 24, Sankt-Peterburg, Russia, 191123',
"Ulitsa Krasnogo Tekstil'shchika, 12, Sankt-Peterburg, Russia, 191124",
'Rue Joukovski, 30, Sankt-Peterburg, Russia, 191014',
'Grazhdanskiy Prospekt, 41Б, Sankt-Peterburg, Russia, 195220',
'Sredneokhtinskiy Prospekt, 18, Sankt-Peterburg, Russia, 195027',
'Bogatyrskiy Prospekt, 47к1, Sankt-Peterburg, Russia, 197372',
'Degtyarnyy Pereulok, 7, Sankt-Peterburg, Russia, 191015',
'Piskarovskiy Prospekt, 5, Sankt-Peterburg, Russia, 195027',
'Prospekt Nastavnikov, 34, Sankt-Peterburg, Russia, 195279',
"Tul'skaya Ulitsa, 11, Sankt-Peterburg, Russia, 191124",
'Prospekt Shaumyana, 4к1, Sankt-Peterburg, Russia, 195027',
"Ulitsa Krasnogo Tekstil'shchika, 2к2, Sankt-Peterburg, Russia, 191124",
"Ulitsa Krasnogo Tekstil'shchika, 7, Sankt-Peterburg, Russia, 191124",
"Prospekt Bol'shevikov, 8к1, Sankt-Peterburg, Russia, 193231",
'Ulitsa Marata, 5, Sankt-Peterburg, Russia, 191025',
```

```
'Magnitogorskaya Ulitsa, 11a, Sankt-Peterburg, Russia, 195112',
'Ligovsky Ave, 23, Sankt-Peterburg, Russia, 191036',
"Industrial'nyy Prospekt, 35к1, Sankt-Peterburg, Russia, 195279",
'Kirochnaya Ulitsa, 7, Sankt-Peterburg, Russia, 191014',
'Ulitsa Marata, 7, Sankt-Peterburg, Russia, 191025',
'Bukharestskaya street, 36 корпус 1, Sankt-Peterburg, Russia, 192071',
'Fontanka river embankment, 20, Sankt-Peterburg, Russia, 191028',
"Nevsky pr., 114-116, Sankt-Peterburg, Leningradskaya oblast', Russia, 191025",
'Kirochnaya Ulitsa, 47, Sankt-Peterburg, Russia, 191015',
'Zona Vyleta, Vnukovo, Moskva, Russia',
'ulitsa Bakhrushina, 31, Moskva, Russia, 115054',
'Yakornaya Ulitsa, 5A, Sankt-Peterburg, Russia, 195027',
'Prospekt Chernyshevskogo, 9, Sankt-Peterburg, Russia, 191123',
'Prospekt Udarnikov, 42, Sankt-Peterburg, Russia, 195279',
'Ulitsa Kollontay, 21, Sankt-Peterburg, Russia, 193231',
'Botkinskaya Ulitsa, 1Б, Sankt-Peterburg, Russia, 195009',
'Ulitsa Komsomola, 39, Sankt-Peterburg, Russia, 195009',
'Nevsky pr., 33литБ, Sankt-Peterburg, Russia, 191011',
'prospekt Engelsa, 154, Sankt-Peterburg, Russia, 194358',
'Ulitsa Mayakovskogo, 3A, Sankt-Peterburg, Russia, 191025',
'Divenskaya Ulitsa, 18, Sankt-Peterburg, Russia, 197046',
'Shpalernaya ulitsa, 34, Sankt-Peterburg, Russia, 191123',
'Brantovskaya Doroga, Sankt-Peterburg, Russia, 195027',
"Murmanskoye Shosse, 12км, St Petersburg, Leningradskaya oblast', Russia, 193315",
'Ulitsa Kapitana Voronina, 10A, Sankt-Peterburg, Russia, 194100',
'Zanevskiy Prospekt, 67 корпус 2, Sankt-Peterburg, Russia, 195277',
"Industrial'nyy Prospekt, 40 корпус 1, Sankt-Peterburg, Russia, 195279",
'Soyuznyy Prospekt, 8 корпус 1, Sankt-Peterburg, Russia, 193318',
'Kirochnaya Ulitsa, 9, Sankt-Peterburg, Russia, 191014'],
```

```
dataset.trip_start_address.nunique()
```

```
309
```

unique trip_end_address ▶

```
dataset.trip_end_address.unique()
```

```
'ul. Vedeneeva, 4, Sankt-Peterburg, Russia, 195427',
'Nevsky pr., 78, Sankt-Peterburg, Russia, 191025'
```

'Nevsky pr., 78, Sankt-Peterburg, Russia, 191025',
"ul. Melnikova, 42, Yekaterinburg, Sverdlovskaya oblast', Russia, 620109",
"ul. Melnikova, Yekaterinburg, Sverdlovskaya oblast', Russia, 620109",
"ул. 8 Марта, 145, Yekaterinburg, Sverdlovskaya oblast', Russia, 620144",
'Bolshaya Morskaya ul., 53, Sankt-Peterburg, Russia, 190000',
'ul. Krasnogo Tekstilshchika, 15, Sankt-Peterburg, Russia, 191124',
'Staro-Petergofskiy pr., 12-14, Sankt-Peterburg, Russia, 190020',
'Ulitsa Sofyi Kovalevskoy, 14к6A Sankt-Peterburg 195256',
'Pushkinskaya ul., 9, Sankt-Peterburg, Russia, 191040',
'Millionnaya ulitsa, 17, Sankt-Peterburg, Russia, 191186',
'pr. Kultury, 19к3, Sankt-Peterburg, Russia, 195274',
'ul. Belinskogo, 8, Sankt-Peterburg, Russia, 191014',
'Nevsky pr., 20 Sankt-Peterburg 191186',
'Parkovaya ul., 18, Sestroretsk, g. Sankt-Peterburg, Russia, 197706',
'pr. Khudozhnikov, 33к4, Sankt-Peterburg, Russia, 194295',
"41K-68, Kudrovo, Leningradskaya oblast', Russia, 193315",
'Kirishskaya ul., 11, Sankt-Peterburg, Russia, 195299',
'Skobelevskiy pr., 5, Sankt-Peterburg, Russia, 194017',
'pr. Nauki, 19, Sankt-Peterburg, Russia, 195257',
'Paradnaya Ulitsa, 3, Sankt-Peterburg, Russia, 191014',
"Murmanskoye sh., 12, Kudrovo, Leningradskaya oblast', Russia, 193315",
'ul. Rustaveli, 59, Sankt-Peterburg, Russia, 195299',
'nab. Admirala Lazareva, 22/10, Sankt-Peterburg, Russia, 197110',
'pr. Kultury, 41, Sankt-Peterburg, Russia, 195276',
'Baykonurskaya ul., 14A, Sankt-Peterburg, Russia, 197227',
'Uralskaya ulitsa, 29, Sankt-Peterburg, Russia, 199155',
'Kazanskaya ul., 7-9, Sankt-Peterburg, Russia',
'pr. Nauki, 21, Sankt-Peterburg, Russia, 195220',
'Millionnaya ulitsa, 6-10, Sankt-Peterburg, Russia, 191186',
'pr. Udarnikov, 47, Sankt-Peterburg, Russia, 195030',
'Prospekt Marshala Blyukhera, 1Б, Sankt-Peterburg, Russia, 194100',
'Industrialnyy pr., 71, Sankt-Peterburg, Russia, 195279',
'sh. Revolyutsii, 65, Sankt-Peterburg, Russia, 195279',
'Zanevskiy prosp., 73, Sankt-Peterburg, Russia, 195277',
'pl. Ostrovskogo, 2A Sankt-Peterburg 191023',
'Ligovsky Ave, 74, Sankt-Peterburg, 191040',
'nab. Obvodnogo Kanala, 13, Sankt-Peterburg, Russia, 191167',
'ul. Zhaka Dyuklo, 20, Sankt-Peterburg, Russia, 194214',
'Industrialnyy pr., 25, Sankt-Peterburg, Russia, 195279',
'Zanevskiy prosp., 67к2, Sankt-Peterburg, Russia, 195277',
'pr. Kosygina, 26к1, Sankt-Peterburg, Russia, 195426',
'Brantovskaya dor., Sankt-Peterburg, Russia, 195027'

```
 'Brantovskaya dor., Sankt-Peterburg, Russia, 195027',
 'Granitnaya ul., Sankt-Peterburg, Russia, 195277',
 'Ligovskiy pr., 43, Sankt-Peterburg, Russia, 191040',
 'Millionnaya ulitsa, 15, Sankt-Peterburg, Russia, 191186',
 'Pushkinskaya ul., 11, Sankt-Peterburg, Russia',
 'ul. Odoyevskogo, 33, Sankt-Peterburg, Russia, 199155',
 'pr. Solidarnosti, 5, Sankt-Peterburg, Russia, 193312',
 'Khasanskaya ul., 17к1, Sankt-Peterburg, Russia, 195298',
 'Kronverkskiy prospekt, 47, Sankt-Peterburg, Russia, 197101',
 'Shafirovskiy Prospekt, 12 Sankt-Peterburg 195279',
 'Irinovskiy Prospekt, 32 Sankt-Peterburg 195030',
 'Baykonurskaya Ulitsa, 14A Sankt-Peterburg 197227',
 'Sofyi Kovalevskoy ul., 3, Sankt-Peterburg, Russia, 195256',
 'Kirishskaya Ulitsa, 11 Sankt-Peterburg 195299',
 'Ulitsa Moldagulovoy, 6 Sankt-Peterburg 195027',
 'Obukhovskoy Oborony Ave, 53 Sankt-Peterburg 192029',
 'Obukhovskoy Oborony Ave, 51 Sankt-Peterburg 192029'
```

```
dataset.trip_end_address.nunique()
```

```
268
```

Identify popular start address points

```
dataset['trip_start_address'].value_counts().head(10)
```

```
Paradnaya Ulitsa, 3, Sankt-Peterburg, Russia, 191014                    173
Sverdlovskaya naberezhnaya, 44Д/4Б, Sankt-Peterburg, Russia, 195027      40
Sofyi Kovalevskoy ul., 14к6A, Sankt-Peterburg, Russia, 195256            23
Pulkovo Airport (LED), Unnamed Road, Sankt-Peterburg, Russia, 196210     20
Sofyi Kovalevskoy ulitsa, 14 корпус 6A, Sankt-Peterburg, Russia, 195256  16
Irinovskiy Prospekt, 32 Sankt-Peterburg 195030                           10
ul. Kollontay, 1, Sankt-Peterburg, Russia, 193230                         7
Yakornaya Ulitsa, 5A, Sankt-Peterburg, Russia, 195027                     6
Magnitogorskaya ul., 11, Sankt-Peterburg, Russia, 195027                  6
Ulitsa Dzhona Rida, 2, Sankt-Peterburg, Russia, 193318                    5
Name: trip_start_address, dtype: int64
```

Identify popular end address points

```
dataset['trip_end_address'].value_counts().head(10)
```

```
Paradnaya Ulitsa, 3, Sankt-Peterburg, Russia, 191014                    183
Sverdlovskaya naberezhnaya, 44Д/4Б, Sankt-Peterburg, Russia, 195027       59
Sofyi Kovalevskoy ul., 14к6А, Sankt-Peterburg, Russia, 195256             29
Pulkovo Airport (LED), Unnamed Road, Sankt-Peterburg, Russia, 196210      28
Sofyi Kovalevskoy ulitsa, 14 корпус 6А, Sankt-Peterburg, Russia, 195256   15
Kirishskaya ul., 11, Sankt-Peterburg, Russia, 195299                      13
Yakornaya Ulitsa, 5A, Sankt-Peterburg, Russia, 195027                     12
Irinovskiy Prospekt, 32 Sankt-Peterburg 195030                             9
pr. Solidarnosti, 5, Sankt-Peterburg, Russia, 193312                       6
Irinovskiy pr., 34, Sankt-Peterburg, Russia, 195030                        5
Name: trip_end_address, dtype: int64
```

for same cases if trip start and trip ending addresses are same ▶

```
dataset[dataset['trip_start_address']==dataset['trip_end_address']]
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
14	June 3, 2015 at 5:20PM	Completed	Uber	30bd4a26ca276b8f04a01d87ea3777fc	ef33153fbce5b15f93d9319528dc598f	3ffa4a
21	June 14, 2015 at 7:44AM	Cancelled	Uber	b970e99cf7d4e163c9808ed945f9e5be	a89bc69e8452fcec3f650b688f6e3473	3ffa4a
121	December 26, 2015 at 2:05PM	Completed	Uber	6ea41ffb88c316d552d2cf9c81226d96	8e78bc6f8711a06d85f0ec8ecf4e9065	3ffa4a
122	December 26, 2015 at 4:00PM	Cancelled	Uber	7fc047ee4a265d6740fcb31293291272	164cdace271ca4165ded891d0c2fdd14	3ffa4a
155	May 5, 2016 at 10:55AM	Cancelled	Uber	bfd97f6803ba48c9741bdb9a1ad28e87	164cdace271ca4165ded891d0c2fdd14	3ffa4a

```
dataset[dataset['trip_start_address']==dataset['trip_end_address']].shape
```

(9, 45)

[illegible]	December 13, 2016 at	Cancelled	Uber	[illegible]	[illegible]	[illegible]

```
dataset[dataset['trip_start_address']==dataset['trip_end_address']].head(5)
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
14	June 3, 2015 at 5:20PM	Completed	Uber	30bd4a26ca276b8f04a01d87ea3777fc	ef33153fbce5b15f93d9319528dc598f	3ffa4a7
21	June 14, 2015 at	Cancelled	Uber			

DATA PREPROCESSING

```
dataset.dropna(inplace=True)
```

```
dataset.drop_duplicates(inplace=True)
```

5 rows × 49 columns

DATA VISUALIZATION

```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)

unique_values = {}
for col in object_cols:
  unique_values[col] = dataset[col].unique().size
unique_values
```

```
{'trip_completed_at': 643,
 'trip_status': 2,
 'ride_hailing_app': 2,
 'trip_uid': 643,
 'driver_uid': 593,
 'rider_uid': 1,
```

```
 'customer': 1,
 'trip_start_time': 642,
 'trip_end_time': 642,
 'trip_time': 548,
 'total_time': 78,
 'wait_time': 451,
 'trip_type': 6,
 'vehicle_make_model': 119,
 'vehicle_license_plate': 1,
 'driver_name_en': 174,
 'vehicle_make': 36,
 'vehicle_model': 117,
 'driver_gender': 2,
 'driver_photo_url': 1,
 'driver_phone_number': 1,
 'trip_map_image_url': 1,
 'trip_path_image_url': 1,
 'city': 3,
 'country': 1,
 'trip_start_address': 287,
 'trip_end_address': 250,
 'price_rub': 390,
 'temperature_time': 642,
 'cloudness': 99,
 'weather_main': 9,
 'weather_desc': 13,
 'precipitation': 3}
```

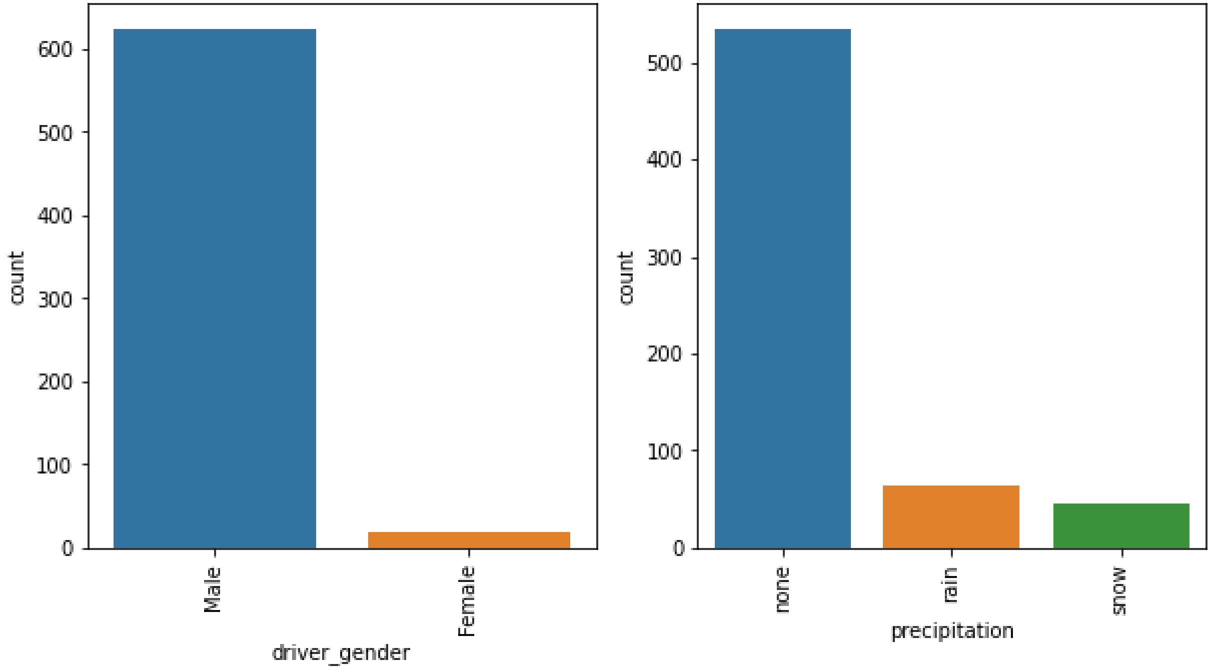
Make a plot for 👇

driver_gender vs precipitation

```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.countplot(dataset['driver_gender'])
```

```
plt.xticks(rotation=90)

plt.subplot(1,2,2)
sns.countplot(dataset['precipitation'])
plt.xticks(rotation=90)
```

(array([0, 1, 2]), <a list of 3 Text major ticklabel objects>)

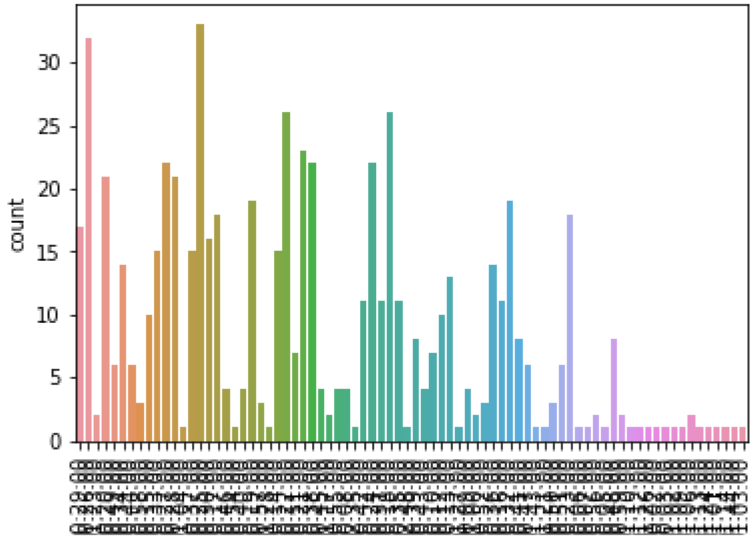


Make a plot 👇

day-night vs count

```
sns.countplot(dataset['total_time'])
plt.xticks(rotation=90)
```

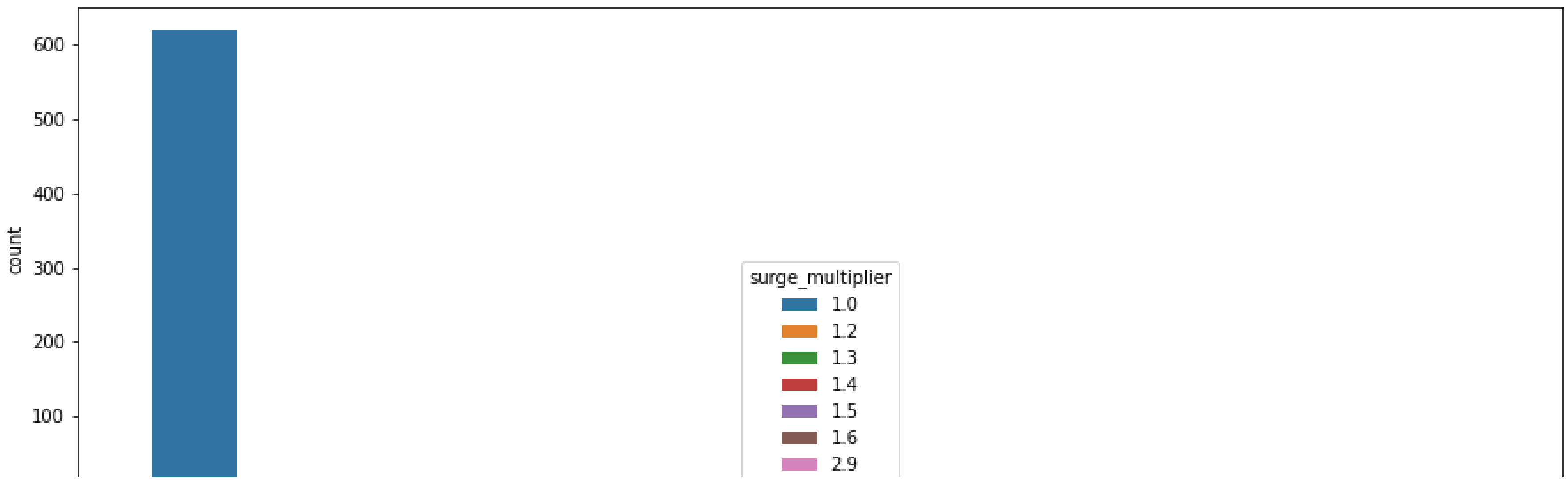
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
        34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 75, 76, 77]),
 <a list of 78 Text major ticklabel objects>)



Plot for 👇

trip_status & surge_multiplier

```
plt.figure(figsize=(15, 5))
sns.countplot(data=dataset, x='trip_status', hue='surge_multiplier')
plt.xticks(rotation=90)
plt.show()
```

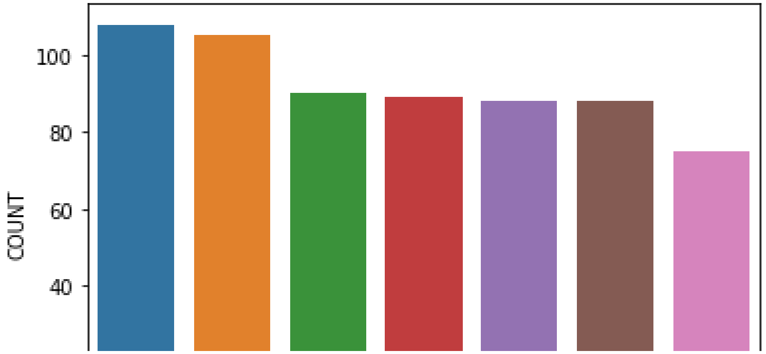


Make a plot for DAY vs COUNT

```
dataset['DAY'] = dataset.trip_start_time.dt.weekday
day_label = {
  0: 'Mon', 1: 'Tues', 2: 'Wed', 3: 'Thus', 4: 'Fri', 5: 'Sat', 6: 'Sun'
}
dataset['DAY'] = dataset['DAY'].map(day_label)
```

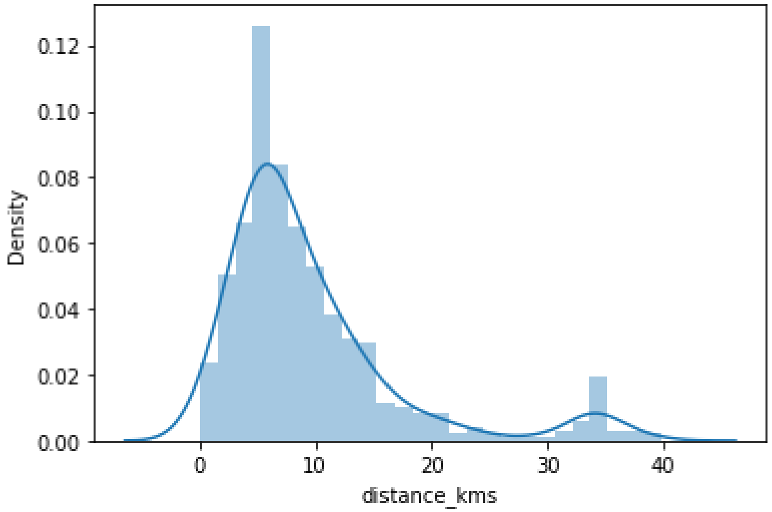
```
day_label = dataset.DAY.value_counts()
sns.barplot(x=day_label.index, y=day_label);
plt.xlabel('DAY')
plt.ylabel('COUNT')
```

Text(0, 0.5, 'COUNT')

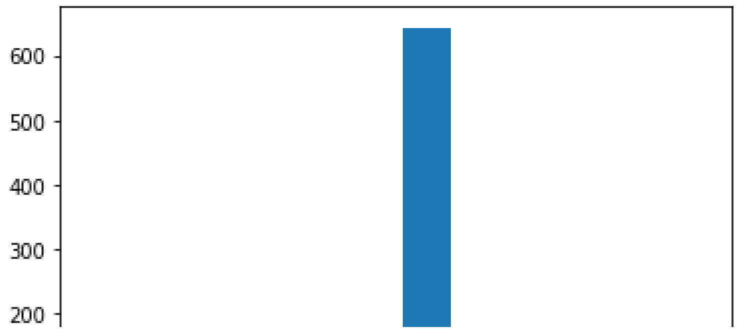



```
sns.distplot(dataset[dataset['distance_kms']<40]['distance_kms'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f6096c5f670>




```
plt.hist(dataset["country"], bins=10, rwidth=0.8)
plt.show()
```



dataset['price_usd'].value_counts()

```
3.41    5
2.83    5
2.79    5
2.76    5
3.62    5
       ..
3.44    1
2.37    1
4.91    1
4.68    1
4.55    1
Name: price_usd, Length: 419, dtype: int64
```

dataset['distance_kms'].value_counts()

```
0.01     6
5.89     5
10.04    4
5.66     4
5.65     4
        ..
2.64     1
12.33    1
4.70     1
7.23     1
9.03     1
Name: distance_kms, Length: 467, dtype: int64
```

```
dataset['weather_main'].head()
```

```
0     partly-cloudy-day
1     partly-cloudy-day
2     partly-cloudy-day
3   partly-cloudy-night
4   partly-cloudy-night
Name: weather_main, dtype: object
```

```
dataset['weather_desc'].head()
```

```
0    Mostly Cloudy
1    Mostly Cloudy
2    Mostly Cloudy
3    Partly Cloudy
4    Partly Cloudy
Name: weather_desc, dtype: object
```

```
data = dataset.select_dtypes(include =['int64','float64'])
data.head()
```

	surge_multiplier	pickup_lat	pickup_long	dropoff_lat	dropoff_long	rub_usd_exchange_rate	price_usd	distance_kms	te
0	1.0	60.031438	30.329826	59.963131	30.307655	51.28	5.17	9.29	
1	1.0	59.963014	30.307313	60.031351	30.329495	51.28	4.97	9.93	
2	1.0	60.031529	30.329416	59.924281	30.387561	49.50	13.01	18.01	
3	2.9	59.959883	30.311159	59.934680	30.308489	49.53	25.99	5.10	
4	1.4	59.934813	30.308553	60.031470	30.329402	49.53	13.43	21.92	

```
categorical_cols =dataset.select_dtypes(include=['object'])
categorical_cols.head()
```

	trip_completed_at	trip_status	ride_hailing_app	trip_uid	driver_uid	
0	May 11, 2015 at 6:55PM	Completed	Uber	ee89076fd9da9bddf5f096b0ca42f8d5	05cfeb269e606247fe9d2b6082942c59	3ffa4a7
1	May 11, 2015 at 8:12PM	Completed	Uber	518be51d403944a03c47e8d1f2c87311	4a4e248742f9d5ff517c5bbbb48d0e54	3ffa4a7
2	May 13, 2015 at 11:38AM	Completed	Uber	6e460cc8a12c3c6568d0d4a67ac58393	cb249a2bd807ca78697b4ed0348c37da	3ffa4a7
3	May 16, 2015 at 1:44AM	Completed	Uber	49613a86a04e6c15d72b51d1a2935d81	d3f73f8151c2e8c34b541f961db7f5fa	3ffa4a7
4	May 16, 2015 at 3:18AM	Completed	Uber	9896148fdecdb4c5d977a8691510bdb6	1287d21e6455ee40d4861f6b91c680f4	3ffa4a7

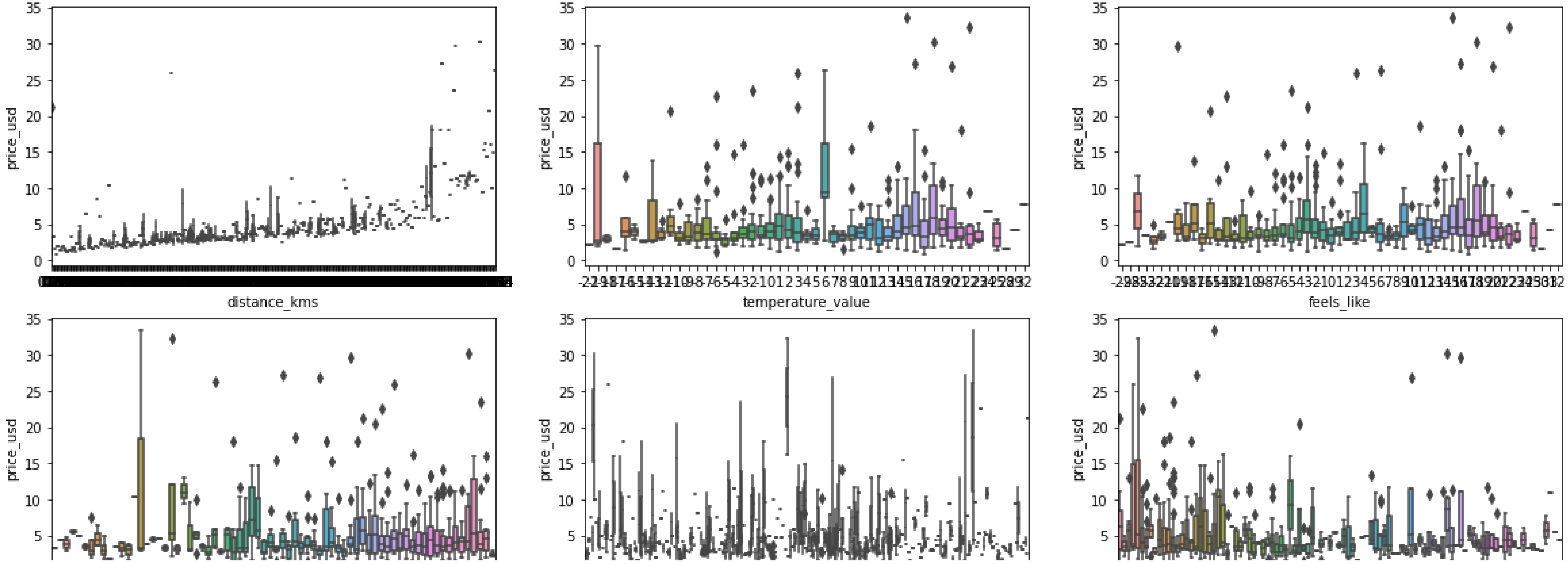
5 rows × 33 columns

BOX SUBPLOTS

```
plt.figure(figsize=(20, 12))

plt.subplot(3, 3, 1) # 3 rows, 3 columns, 1st subplot = left
sns.boxplot(x='distance_kms', y='price_usd', data=dataset)
```

```
plt.subplot(3, 3, 2) # 3 rows, 3 columns, 2nd subplot = middle
sns.boxplot(x='temperature_value', y='price_usd', data=dataset)

plt.subplot(3, 3, 3) # 3 rows, 3 columns, 3rd subplot = right
sns.boxplot(x='feels_like', y='price_usd', data=dataset)

plt.subplot(3, 3, 4)
sns.boxplot(x='humidity', y='price_usd', data=dataset)

plt.subplot(3, 3, 5)
sns.boxplot(x='wind_speed', y='price_usd', data=dataset)

plt.subplot(3, 3, 6)
sns.boxplot(x='cloudness', y='price_usd', data=dataset)

plt.subplot(3, 3, 7)
sns.boxplot(x='price_rub', y='price_usd', data=dataset)
```

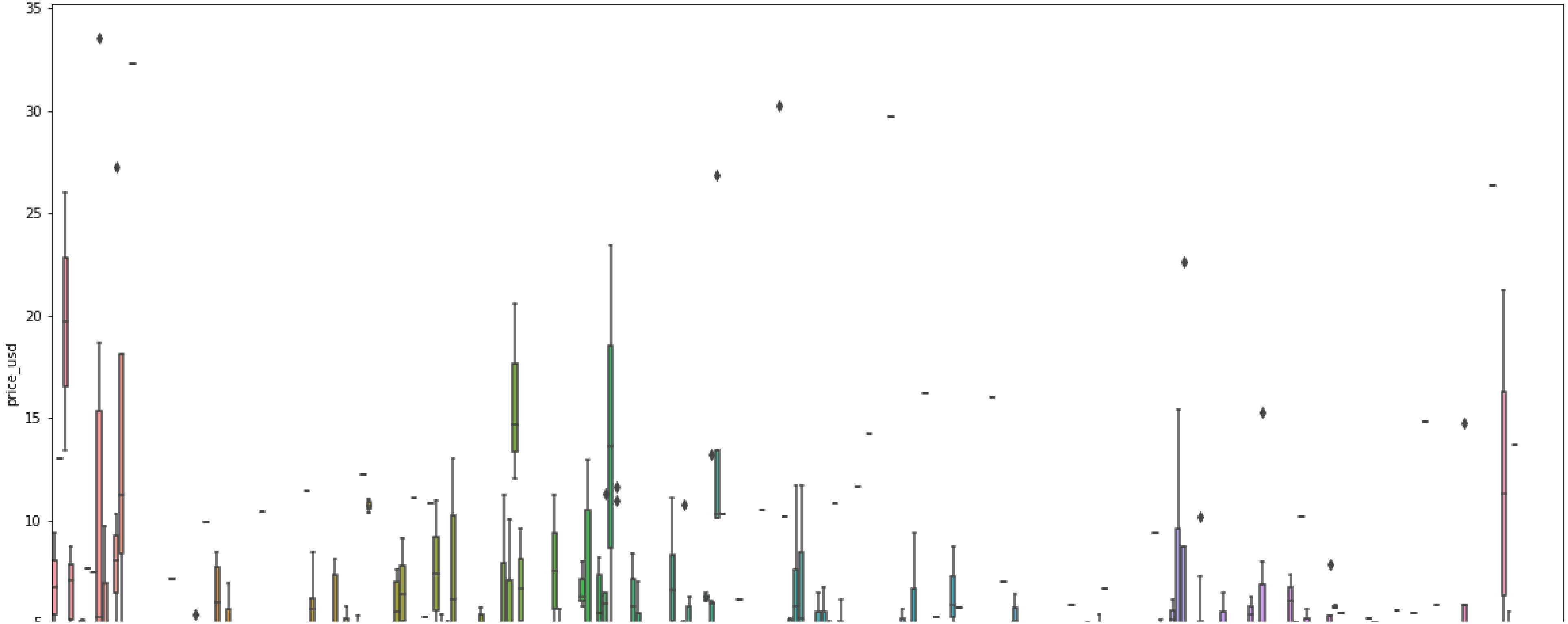
<matplotlib.axes._subplots.AxesSubplot at 0x7f6091bbb9a0>



DATA PROCESSING BOX PLOT

```
plt.figure(figsize=(20, 10))
sns.boxplot(x='rub_usd_exchange_rate', y='price_usd', data=dataset)
```

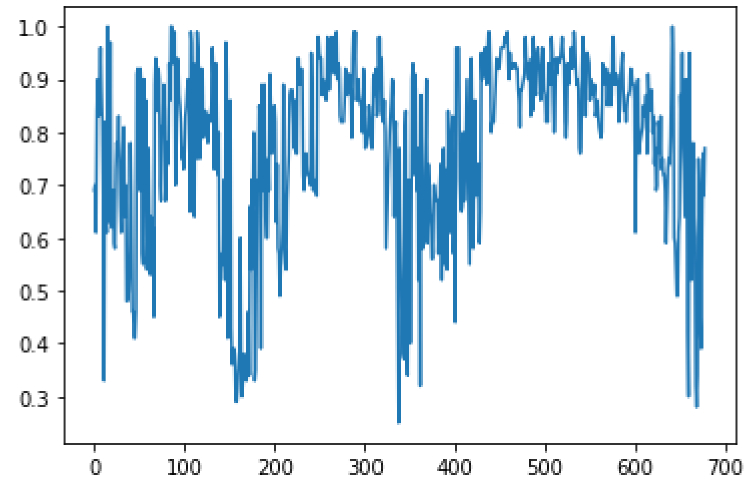
<matplotlib.axes._subplots.AxesSubplot at 0x7f608cb966d0>



Plot of **HUMIDITY**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("uber_dataset.csv")
plt.plot(data['humidity'])
plt.show()
print(data.describe())

a=data.shape
print(a)
##The Name of the driver and the the total number of trips done by that driver
n=pd.DataFrame(data['driver_name_en'].value_counts())
```

```
print(n)

##The Name of the city and the number of times the cities were visited in trips
c=pd.DataFrame([data['city'].value_counts()],index=[1])
print(c)

data.head()
```



	surge_multiplier	pickup_lat	pickup_long	dropoff_lat	dropoff_long	\
count	643.000000	678.000000	678.000000	678.000000	678.000000	
mean	1.011353	59.756784	31.740062	59.758561	31.739366	
std	0.096085	0.800432	6.063872	0.799396	6.062101	
min	1.000000	55.605800	30.168447	55.599648	29.947507	
25%	1.000000	59.932134	30.364931	59.933576	30.366456	
50%	1.000000	59.941415	30.366456	59.941415	30.368724	
75%	1.000000	59.960492	30.420179	59.961987	30.417860	
max	2.900000	60.126500	60.809913	60.126957	60.809916	

	rub_usd_exchange_rate	price_usd	distance_kms	temperature_value	\
count	678.000000	678.000000	678.000000	678.000000	
mean	60.517463	5.061593	10.057788	5.327434	
std	4.526606	4.251843	8.735132	9.996551	
min	49.260000	0.840000	0.010000	-22.000000	
25%	57.230000	2.760000	4.912500	-2.000000	
50%	59.010000	3.735000	7.290000	3.000000	
75%	64.637500	5.670000	11.665000	14.000000	
max	79.360000	33.550000	46.740000	32.000000	

	feels_like	humidity	wind_speed
count	678.000000	678.000000	678.000000
mean	3.002950	0.778083	3.541519
std	11.912408	0.164886	1.737701
min	-29.000000	0.250000	0.050000
25%	-5.750000	0.690000	2.350000
50%	0.000000	0.820000	3.500000

```
50%    0.000000   0.820000   3.500000
75%    14.000000  0.900000   4.660000
max    32.000000  1.000000   8.670000
(678, 45)
         driver_name_en
Aleksandr        54
Sergey           40
Aleksey          38
Andrey           37
```

BY DATE AND TIME

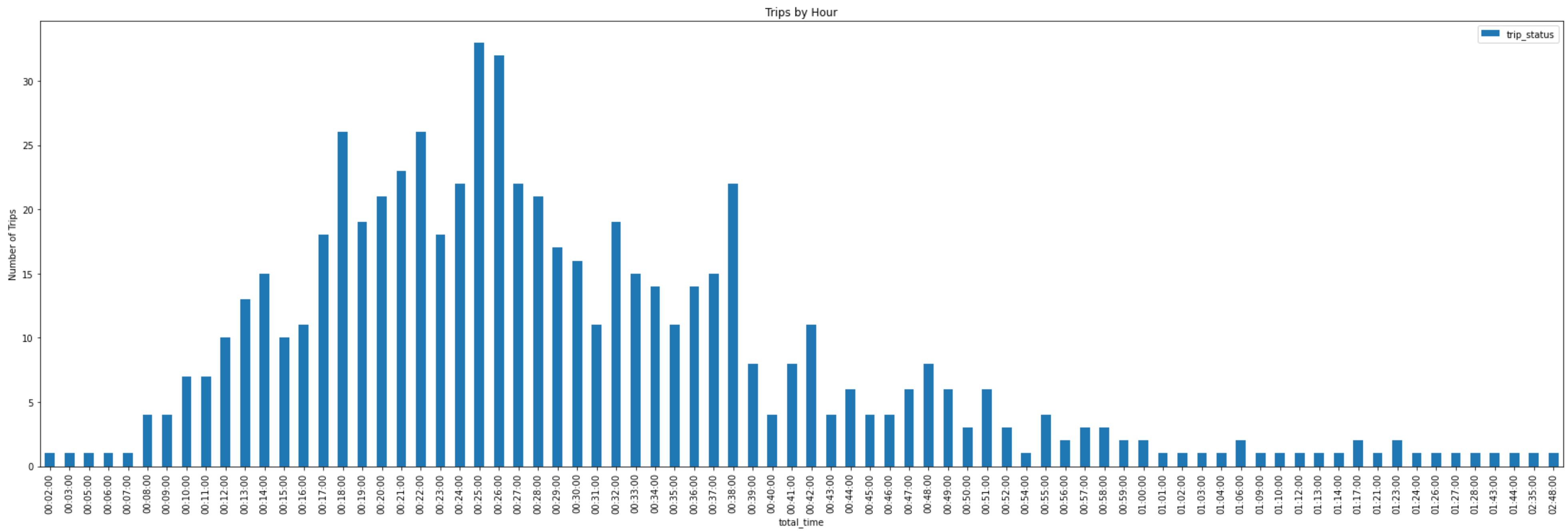
```
Mahamad          1
```

```
df_hour_grouped = dataset.groupby(['total_time']).count()

#Creating the sub dataframe
df_hour = pd.DataFrame({'trip_status':df_hour_grouped.values[:,0]}, index = df_hour_grouped.index)

df_hour.head()
```

	trip_status
total_time	
00:02:00	1
00:03:00	1
00:05:00	1
00:06:00	1
00:07:00	1

May 16, 2015 at

```
df_hour.plot(kind='bar', figsize=(30,9))

plt.ylabel('Number of Trips')
plt.title('Trips by Hour')

plt.show()
```




```
#Grouping by Month
df_month_grouped = dataset.groupby(['trip_completed_at'], sort=False).count()

#Creating the sub dataframe
df_month = pd.DataFrame({'trip_status':df_month_grouped.values[:,0]}, index = df_month_grouped.index)

df_month
```

	trip_status
trip_completed_at	
May 11, 2015 at 6:55PM	1
May 11, 2015 at 8:12PM	1
May 13, 2015 at 11:38AM	1
May 16, 2015 at 1:44AM	1
May 16, 2015 at 3:18AM	1
...	...
April 23, 2018 at 12:11PM	1
April 24, 2018 at 02:58PM	1
April 26, 2018 at 03:57PM	1

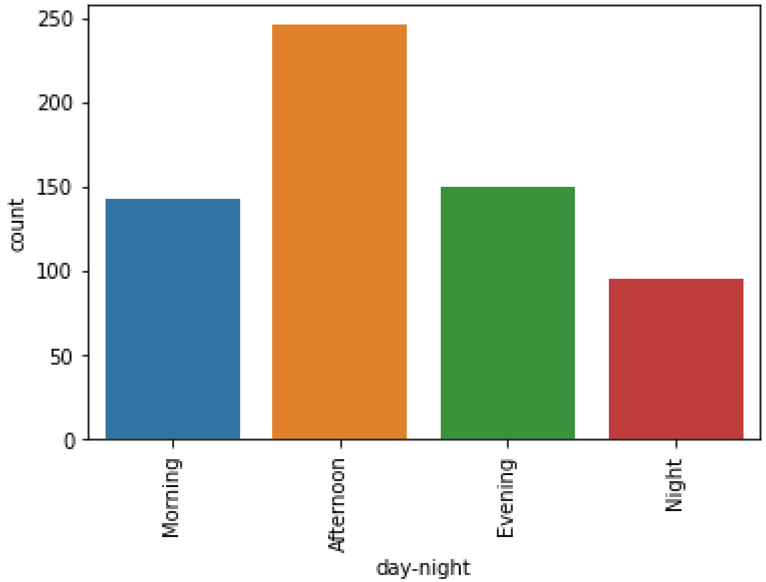
DATE TIME OPERATION

643 rows × 1 columns

```
dataset['trip_start_time'] = pd.to_datetime(dataset['trip_start_time'],
                              errors='coerce')
dataset['trip_end_time'] = pd.to_datetime(dataset['trip_end_time'],
                              errors='coerce')



from datetime import datetime

dataset['date'] = pd.DatetimeIndex(dataset['trip_start_time']).date
dataset['time'] = pd.DatetimeIndex(dataset['trip_start_time']).hour

#changing into categories of day and night
dataset['day-night'] = pd.cut(x=dataset['time'],
```

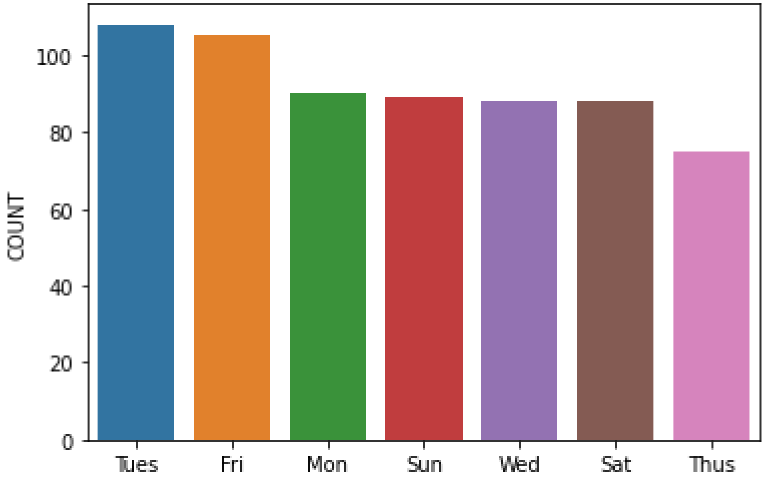
```
                bins = [0,10,15,19,24],
                labels = ['Morning','Afternoon','Evening','Night'])

sns.countplot(dataset['day-night'])
plt.xticks(rotation=90)
```

(array([0, 1, 2, 3]), <a list of 4 Text major ticklabel objects>)




```
dataset['DAY'] = dataset.trip_start_time.dt.weekday
day_label = {
   0: 'Mon', 1: 'Tues', 2: 'Wed', 3: 'Thus', 4: 'Fri', 5: 'Sat', 6: 'Sun'
}
dataset['DAY'] = dataset['DAY'].map(day_label)


day_label = dataset.DAY.value_counts()
sns.barplot(x=day_label.index, y=day_label);
plt.xlabel('DAY')
plt.ylabel('COUNT')
```

Text(0, 0.5, 'COUNT')

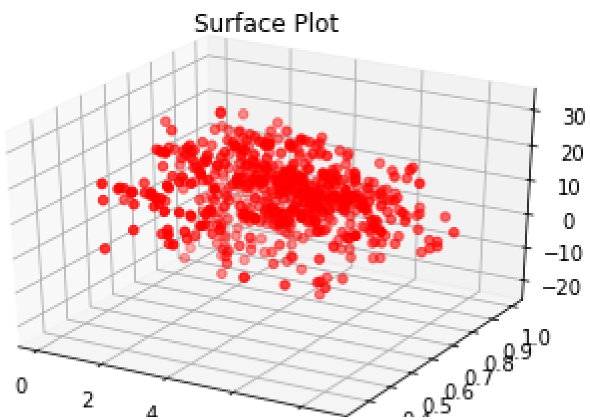


DATA VISUALIZATION USING 3D PLOT

wind_speed , humidity and temperature_valure relationship presentation through 3d scatter plot

3D SURFACE PLOT

```
fig=plt.figure()
d=fig.add_subplot(111,projection='3d')
x=dataset['wind_speed'].values
y=dataset['humidity'].values
z=dataset['temperature_value'].values
d.scatter(x,y,z,c='r')
d.set_title('Surface Plot')
f=plt.show()
print(f)
```

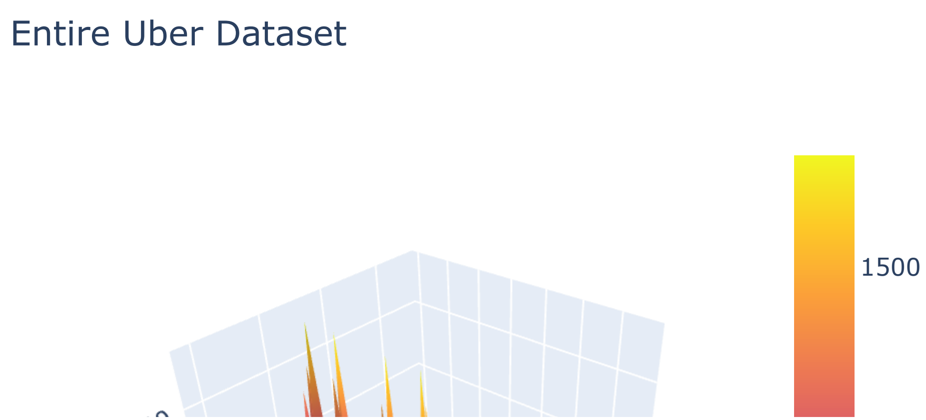


Scatter Plot

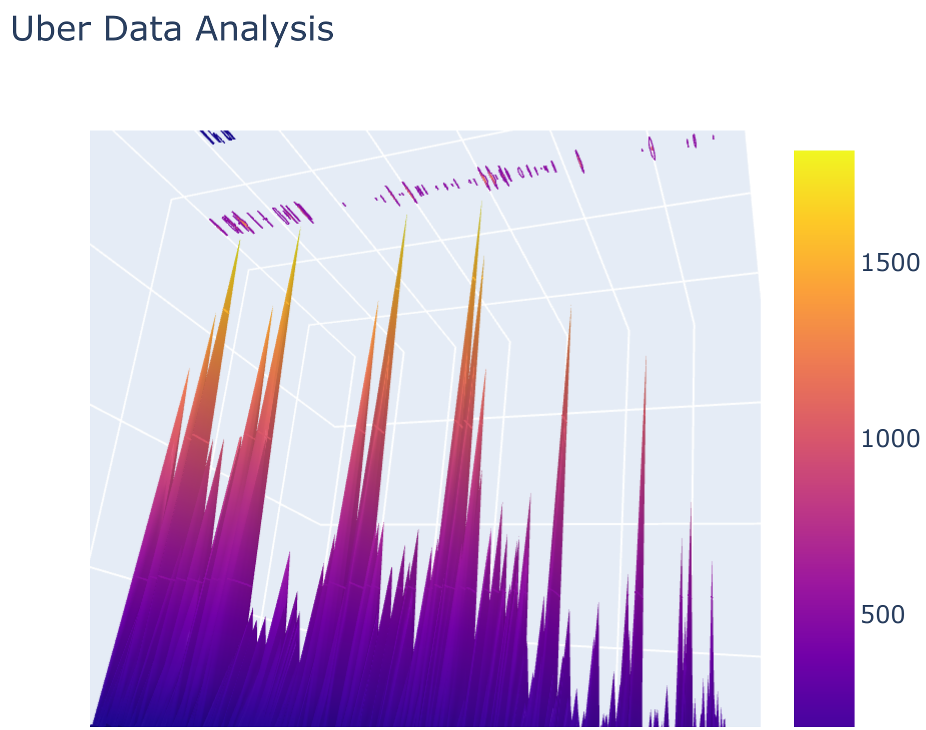
```
fig=plt.figure()
k=plt.axes(projection='3d')
x1=dataset['surge_multiplier'].values
y1=dataset['temperature_value'].values
z1=dataset['feels_like'].values
k.scatter(x1,y1,z1,c='Indigo')
k.set_title('3d Scatter Plot')
g=plt.show()
print(g)
```

SURFACE PLOT


```
import plotly.graph_objects as go

import pandas as pd

# Read data from a csv
z_data = pd.read_csv('uber_dataset.csv')

fig = go.Figure(data=[go.Surface(z=z_data.values)])

fig.update_layout(title='Entire Uber Dataset', autosize=False,
             width=500, height=500,
             margin=dict(l=65, r=50, b=65, t=90))

fig.show()
```




```
import plotly.graph_objects as go

import pandas as pd

# Read data from a csv
z_data = pd.read_csv('uber_dataset.csv')

fig = go.Figure(data=[go.Surface(z=z_data.values)])
fig.update_traces(contours_z=dict(show=True, usecolormap=True,
                              highlightcolor="black", project_z=True))
fig.update_layout(title='Uber Data Analysis', autosize=False,
              scene_camera_eye=dict(x=1.87, y=0.88, z=-0.64),
              width=500, height=500,
              margin=dict(l=65, r=50, b=65, t=90)
)

fig.show()
```



PROJECT ANALYSIS

Uber uses a mixture of internal and external data to estimate fares. Uber calculates fares automatically using street traffic data, GPS data and its own algorithms that make alterations based on the time of the journey. It also analyses external data like public transport routes to plan various services.By doing this project ,we can now easily get the idea of how Uber analyse data based on previous dataset.