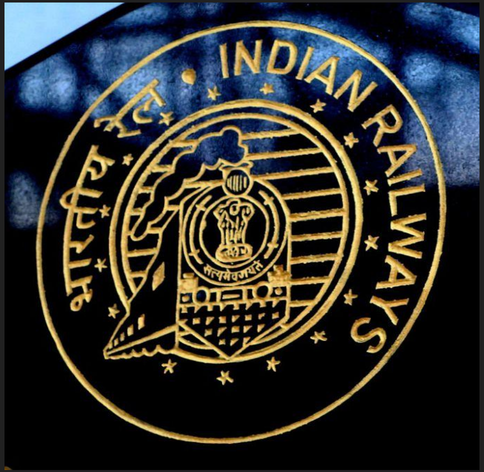
Data science project on

Indian Railways

- Indian Railways (IR) is a statutory body under the ownership of Ministry of Railways, Government of India that operates India's national railway system.It manages the **fourth largest** national railway system in the world by size, with a total route length of 126,511 km (78,610 mi) as of 31 December 2021.

- On **16th April 1853**, the first passenger train ran between Bori Bunder (Bombay) and Thane, a distance of 34 km. Beginning from then Railways has become the most important mode of transportation.

- We may not be able to analyse all of them at this point of time so we tried interpreting some parts from this vast pool.



OUR PROJECT:

- Indian railway dataset (https://www.kaggle.com/sripaadsrinivasan/indian-railways-dataset) is available here.Initial goal is to identify task (that generates some meaningful insight) and prepare that dataset accordingly.

Tasks in the project:

1. Data collection
2. Data Preprocessing and Cleaning
3. Data Visualization
4. Data Statistics(Summary of statistics)
5. Hypothesis Testing
6. Prediction Testing(Using Machine Learning Model)

❑ Data collection:

```
In [5]:  import json
         import csv
         import pandas as pd

         with open("Trains.json") as file:
             data1 = json.load(file)

         with open("Trains.csv",'w') as file:
             csv_file = csv.writer(file)
             csv_file.writerow(['coordinates','type','third_ac','arrival','from_station_code','Train_name','zone','chair_car','first_c

             for data in data1['features']:
                 csv_file.writerow([data['geometry']['coordinates'],data['type'],data['properties']['third_ac'],data['properties']['ar
```

```
In [6]:  df = pd.read_csv('Trains.csv')
```

- ★ Data collected from: https://www.kaggle.com/sripaadsrinivasan/indian-railways-dataset
- ★ Data collected(file): Trains.json
- ★ In this we have imported json, csv and pandas library as pd.

❑ Data Preprocessing and Cleaning:

```
In [6]:  df = pd.read_csv('Trains.csv')

In [7]:  df = df.rename(columns = {"coordinates":"Coordinates","third_ac":"Third_AC","arrival":"Arrival", "from_station_code":"From_st

In [10]: df = df.drop(['Type','type.1','Coordinates','Classes'], axis = 1)
```

★ Capitalising the 1st letter of each column name as a part of cleaning

★ Dropping: 1.Type
2.type.1
3.Coordinates
4.Classes

★ We dropped the above columns as which are of no use in this project

★ For this we used “df.drop(“Column_name”)” command.

```
In [11]: df.head()
```

Out[11]:

	Third_AC	Arrival	From_station_code	Train_name	Zone	Chair_car	First_class	Duration_m	Sleeper	From_station_name	Number	Departure	Returr
0	0	12:15:00	JAT	Jammu Tawi Udhampur Special	NR	0	0	35.0	0	JAMMU TAWI	04601	10:40:00	
1	0	08:35:00	UHP	UDHAMPUR JAMMUTAWI DMU	NR	0	0	50.0	0	UDHAMPUR	04602	06:45:00	
2	0	17:50:00	JAT	JAT UDAHMPUR DMU	NR	0	0	35.0	0	JAMMU TAWI	04603	16:15:00	
3	0	19:50:00	UHP	UDHAMPUR JAMMUTAWI DMU	NR	0	0	30.0	0	UDHAMPUR	04604	18:20:00	
4	1	12:30:00	BDTS	Mumbai BandraT-Bikaner SF Special	NWR	0	0	55.0	1	MUMBAI BANDRA TERMINUS	04728	14:35:00	

- “df.head()” is used to show starting five rows of the data.
- Columns we have now are:

  1.Third_AC,Chair_car, First_class, sleeper -> 0 - doesn’t exist
  1 - exist

  2.Arrival(time-24hrs format), From_station_code, Train_name, Zone, Duration_m (in minutes), From_station_name, Number(train number), Departure(time-24hrs format).

In [11]:
```
df.head()
```
Out[11]:

Duration_m	Sleeper	From_station_name	Number	Departure	Return_train	To_station_code	Second_AC	To_station_name	Duration_h	First_AC	Distance
35.0	0	JAMMU TAWI	04601	10:40:00	04602	UHP	0	UDHAMPUR	1.0	0	53.0
50.0	0	UDHAMPUR	04602	06:45:00	04601	JAT	0	JAMMU TAWI	1.0	0	53.0
35.0	0	JAMMU TAWI	04603	16:15:00	04604	UHP	0	UDHAMPUR	1.0	0	53.0
30.0	0	UDHAMPUR	04604	18:20:00	04603	JAT	0	JAMMU TAWI	1.0	0	53.0
55.0	1	MUMBAI BANDRA TERMINUS	04728	14:35:00	04727	BKN	1	BIKANER JN	21.0	0	1212.0

★ Columns we have now are:

1.Second_AC, First_AC, sleeper -> 0 - doesn't exist
1 - exist

2.Duration_m (in minutes), From_station_name, Number(train number), Departure(time-24hrs format),Return_train, To_station_code, To_station_name, Duration_h(in hours),Distance(in km).

```
In [12]:  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5208 entries, 0 to 5207
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Third_AC           5208 non-null   int64  
 1   Arrival            5208 non-null   object 
 2   From_station_code  5208 non-null   object 
 3   Train_name         5207 non-null   object 
 4   Zone               5193 non-null   object 
 5   Chair_car          5208 non-null   int64  
 6   First_class        5208 non-null   int64  
 7   Duration_m         5193 non-null   float64
 8   Sleeper            5208 non-null   int64  
 9   From_station_name  5208 non-null   object 
 10  Number             5208 non-null   object 
 11  Departure          5208 non-null   object 
 12  Return_train       4609 non-null   object 
 13  To_station_code    5208 non-null   object 
 14  Second_AC          5208 non-null   int64  
 15  To_station_name    5208 non-null   object 
 16  Duration_h         5193 non-null   float64
 17  First_AC           5208 non-null   int64  
 18  Distance           5193 non-null   float64
dtypes: float64(3), int64(6), object(10)
memory usage: 773.2+ KB
```

★ Info about all the columns is obtained by using the command "df.info()"

★ We have 19 different columns showing the count of non-null data and the data type in the respective columns.

★ From this we can conclude that there are some NULL values in the data which will hinder our analysis.

```
In [110]:  df = df.dropna()

In [111]:  df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 4593 entries, 0 to 5205
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Third_AC           4593 non-null   int64
 1   Arrival            4593 non-null   object
 2   From_station_code  4593 non-null   object
 3   Train_name         4593 non-null   object
 4   Zone               4593 non-null   object
 5   Chair_car          4593 non-null   int64
 6   First_class        4593 non-null   int64
 7   Duration_m         4593 non-null   float64
 8   Sleeper            4593 non-null   int64
 9   From_station_name  4593 non-null   object
 10  Number             4593 non-null   object
 11  Departure          4593 non-null   object
 12  Return_train       4593 non-null   object
 13  To_station_code    4593 non-null   object
 14  Second_AC          4593 non-null   int64
 15  To_station_name    4593 non-null   object
 16  Duration_h         4593 non-null   float64
 17  First_AC           4593 non-null   int64
 18  Distance           4593 non-null   float64
dtypes: float64(3), int64(6), object(10)
memory usage: 717.7+ KB
```

- ★ In order to get rid of the NULL values we used “df.dropna()” command.
- ★ We have used “df.info()” command again to make sure whether we got rid of all NULL values in the data.
- ★ With the obtained info it is evident that we got rid of all the NULL data.

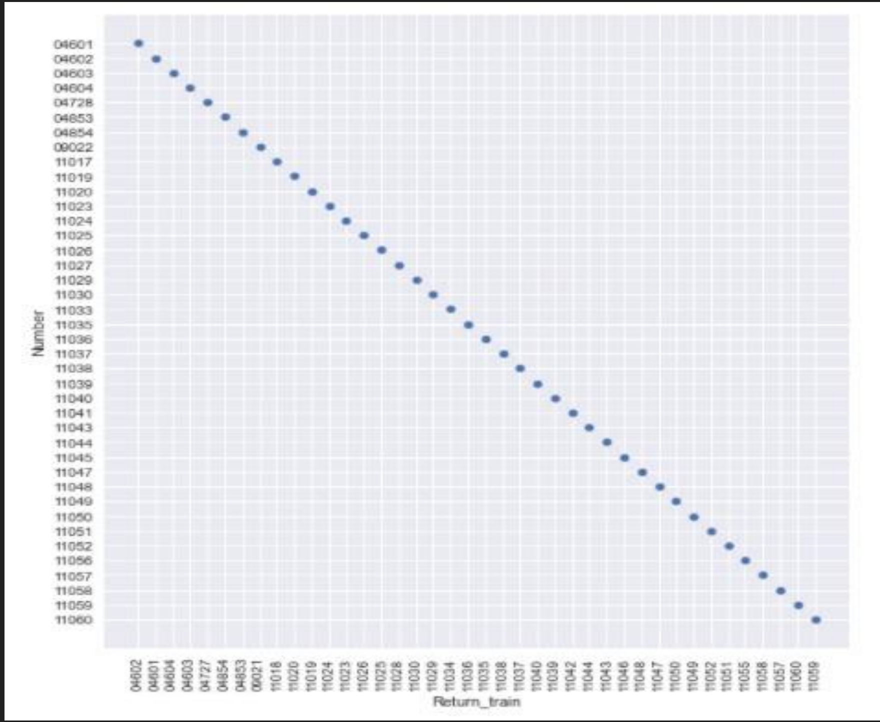
❑ Data Visualization:

★ Now we have some graphs visualising what all we have in our data

Plot between train number and return train number

In [66]:

```python
from matplotlib import pyplot as plt
import seaborn as sns
sns.relplot(y="Number", x="Return_train", data=df[:40], height=8, kind="scatter")
plt.xticks(rotation =90)
```



1

- ★ Here is a plot of train numbers of trains in their to and fro(return) journey
- ★ Train number on y-axis and return train number on x-axis

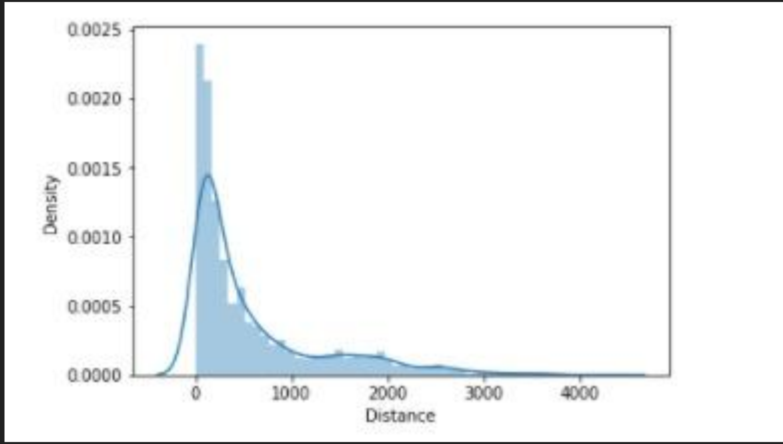
Density Curve of Distance

In [10]:

```
sns.distplot(df['Distance'])
```

```
C:\Python399\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will b
e removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibi
lity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

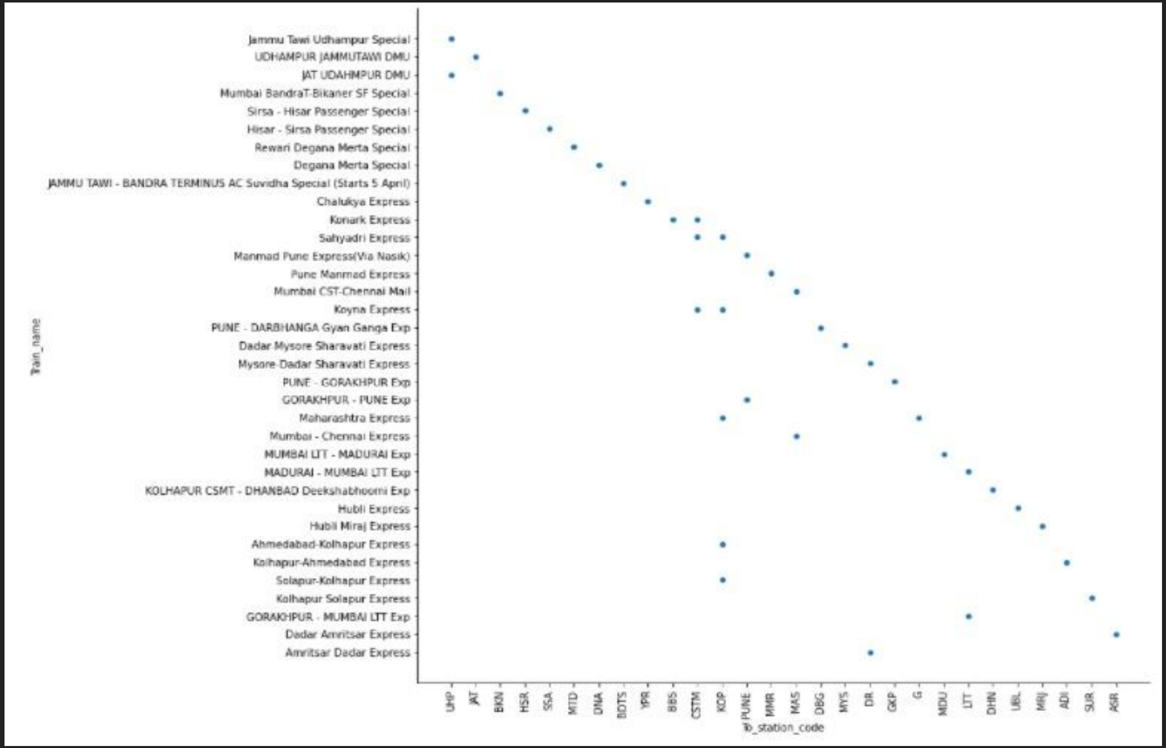
2



- ★ Here we have density curve of distance
- ★ Distance on x-axis and density of trains of y-axis
- ★ From this curve we can say that density of trains in 0 to 1000 is comparatively more than the farther distances

Plot between train and their final destination

```
In [11]:
sns.relplot(y="Train_name", x="To_station_code", data=df[:40], height=10, aspect = 1.5, kind="scatter")
plt.xticks(rotation =90)
```



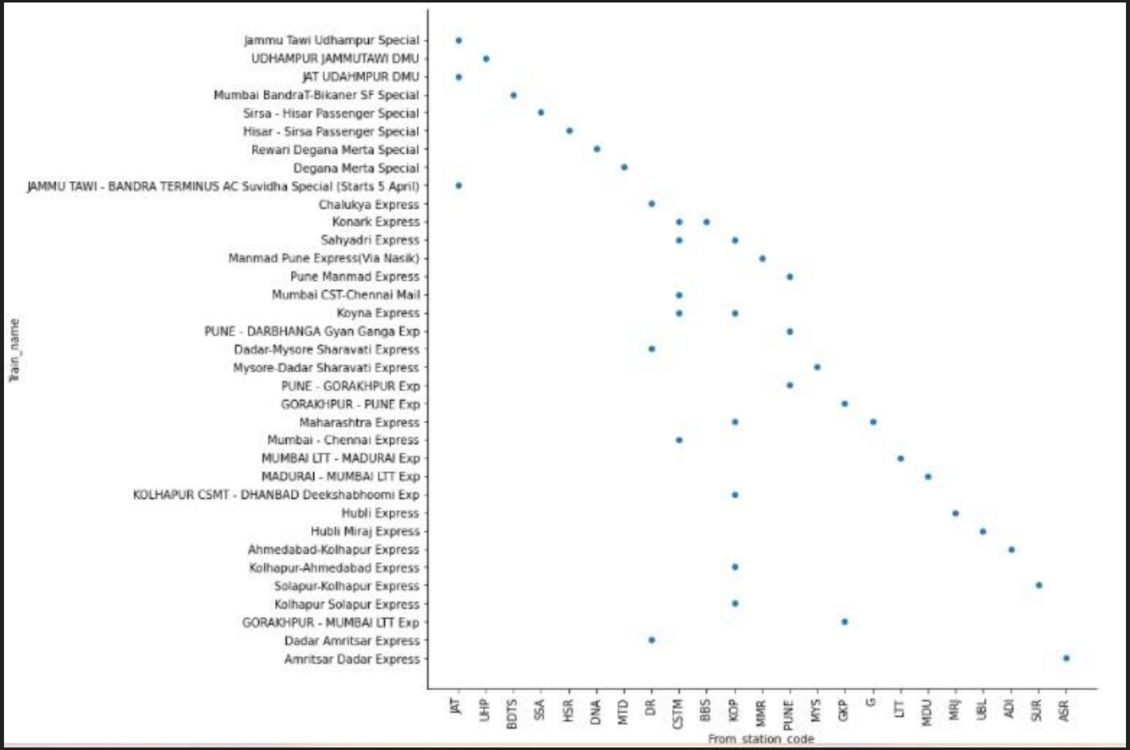
3

- ★ Here is a general plot between train name and it’s final destination
- ★ Destination on x-axis and train names on y-axis

Plot between train and their station of origin

In [12]:

```
sns.relplot(y="Train_name", x="From_station_code", data=df[:40], height=9, aspect=1.5, kind="scatter")
plt.xticks(rotation =90)
```



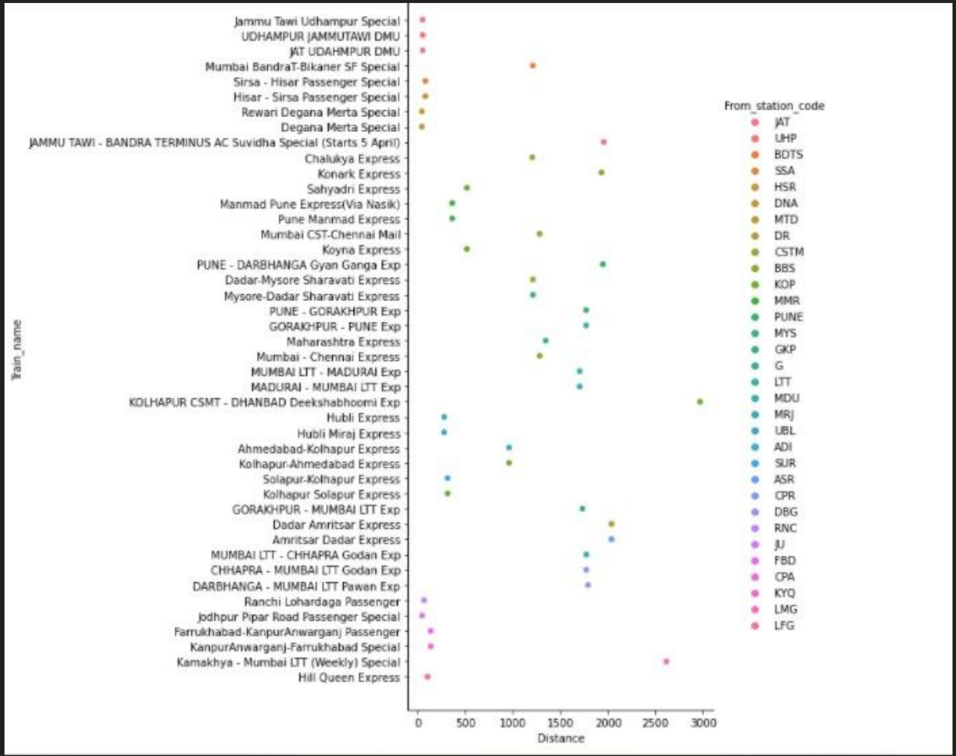
4

- ★ Here is the plot between train and it’s station of origin
- ★ Station of origin on x-axis and train name on y-axis

Plot between train and distance travelled by them

In [14]:

```
plt.rcParams.update({'font.size': 10})
sns.relplot(x="Distance", y="Train_name", data=df[:40], height = 9, aspect = 1.5,kind="scatter"); # seaborn is mainly us
plt.xlabel("Distance (km)")
plt.ylabel('Train Name')  #matplot is always use in the axis manipulation

```

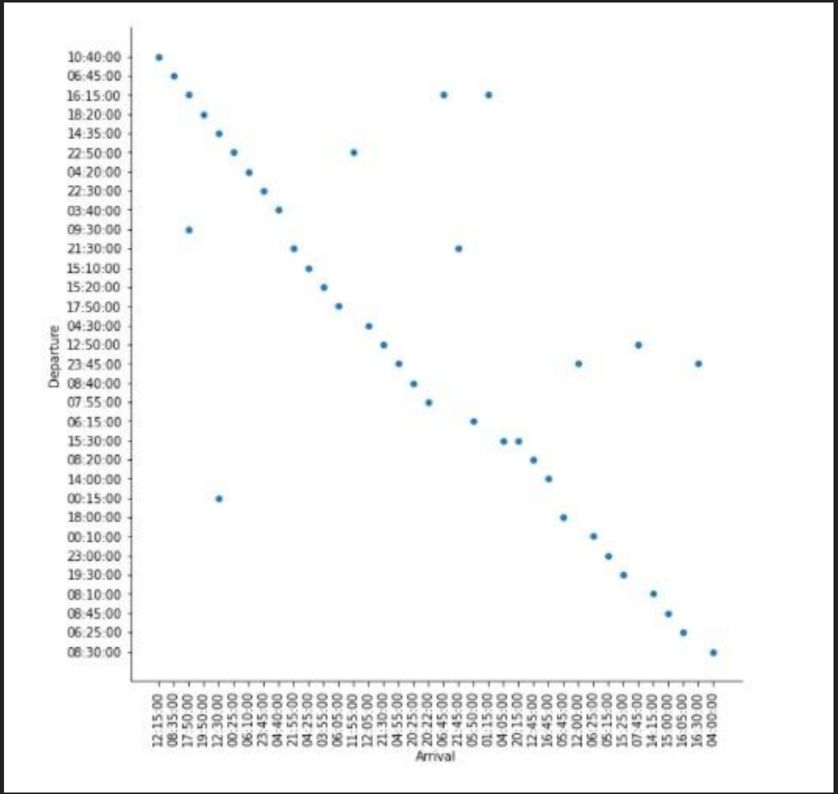


5

- ★ Here is a plot between train and distance travelled by it
- ★ Distances on x-axis and train names on y-axis
- ★ From this plot we can summarize that more number of trains travelled less than 1000 and very less number of trains travelled 2000

Plot between arrival and departure time of trains

```
In [15]:
sns.relplot(x="Arrival", y="Departure", data=df[:40], height = 8);
plt.xticks(rotation =90)
```



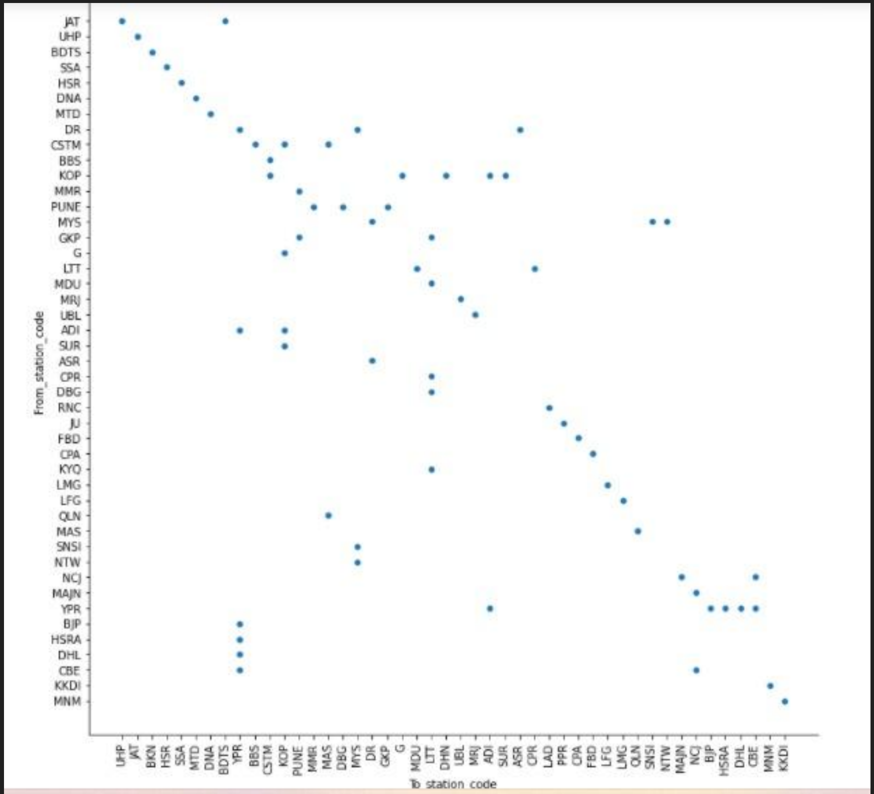
6

- ★ Here is a plot between arrival and departure time of each train
- ★ Arrival times on x-axis and departure times on y-axis
- ★ This plot is approximately representing a straight line showing that the train which left early will reach back early

Plot showing number of trains between two stations

In [16]:

```
sns.relplot(y="From_station_code", x="To_station_code", data=df[:80],height = 10, kind="scatter")
plt.xticks(rotation =90)
```



7

- ★ Here is a plot between number of trains that travel between any 2 station
- ★ To station codes on x-axis and from station codes on y-axis.
- ★ From this plot we can see in between which two stations there are more number of trains.

8

Calculation of total ac coach in a train

In [17]:
```
df.shape
df['Total_AC_coach']=sum([df['Third_AC'] + df['Second_AC'] +df['First_AC']])  #add of the first_ac -> third_ac

df.info()

df['Total_AC_coach']
```

★ We have calculated total number of ac coach in each train which includes all kinds of ac coaches.

Histograms showing various data

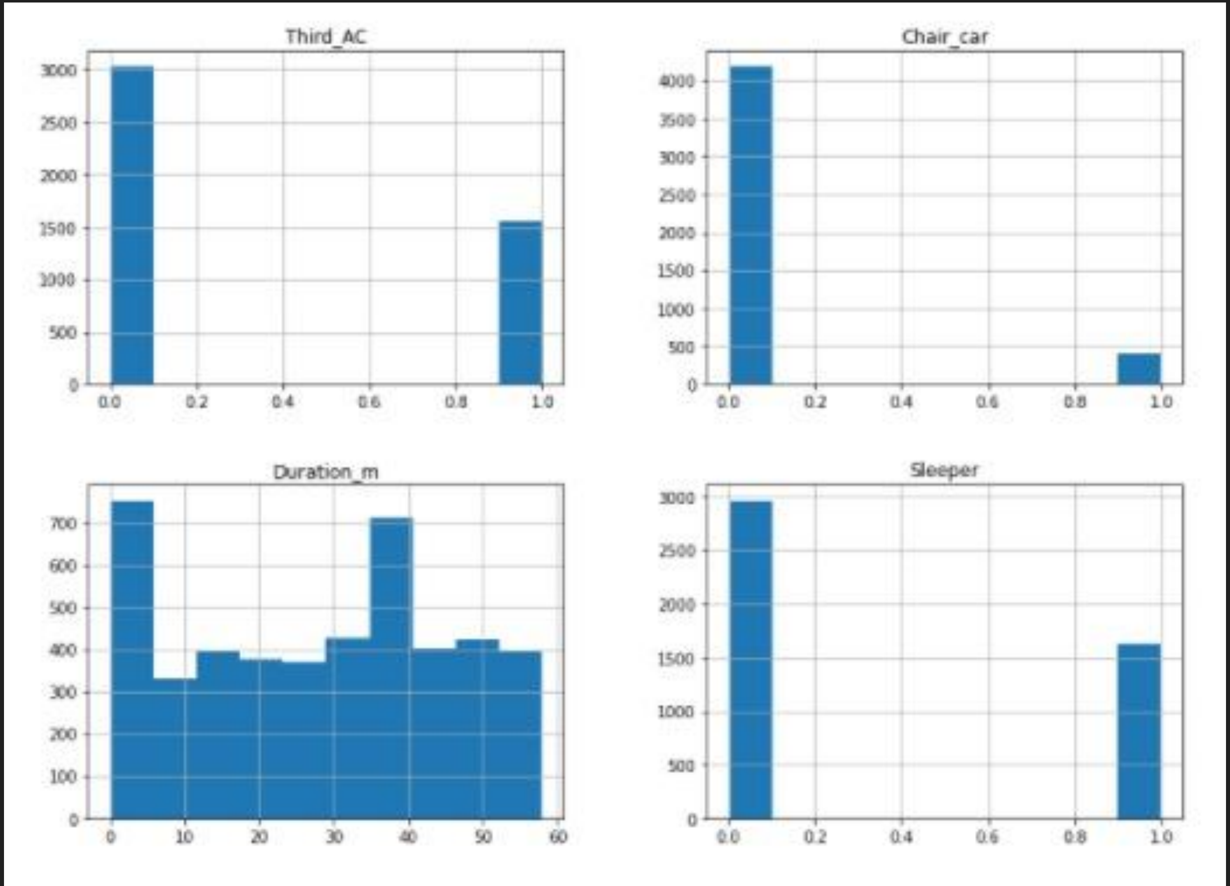
```
Plot (i),(ii),(iii),(v),(vi),(viii),(x) -> are showing data about avaliability of different types of coaches in trains
```

```
Plot (iv),(vii)-> are showing relation between duration  and no of trains
```

```
Plot (ix) -> is showing realation between distance and no of trains
```

In [18]:
```
df.hist(figsize = (20,20))
```

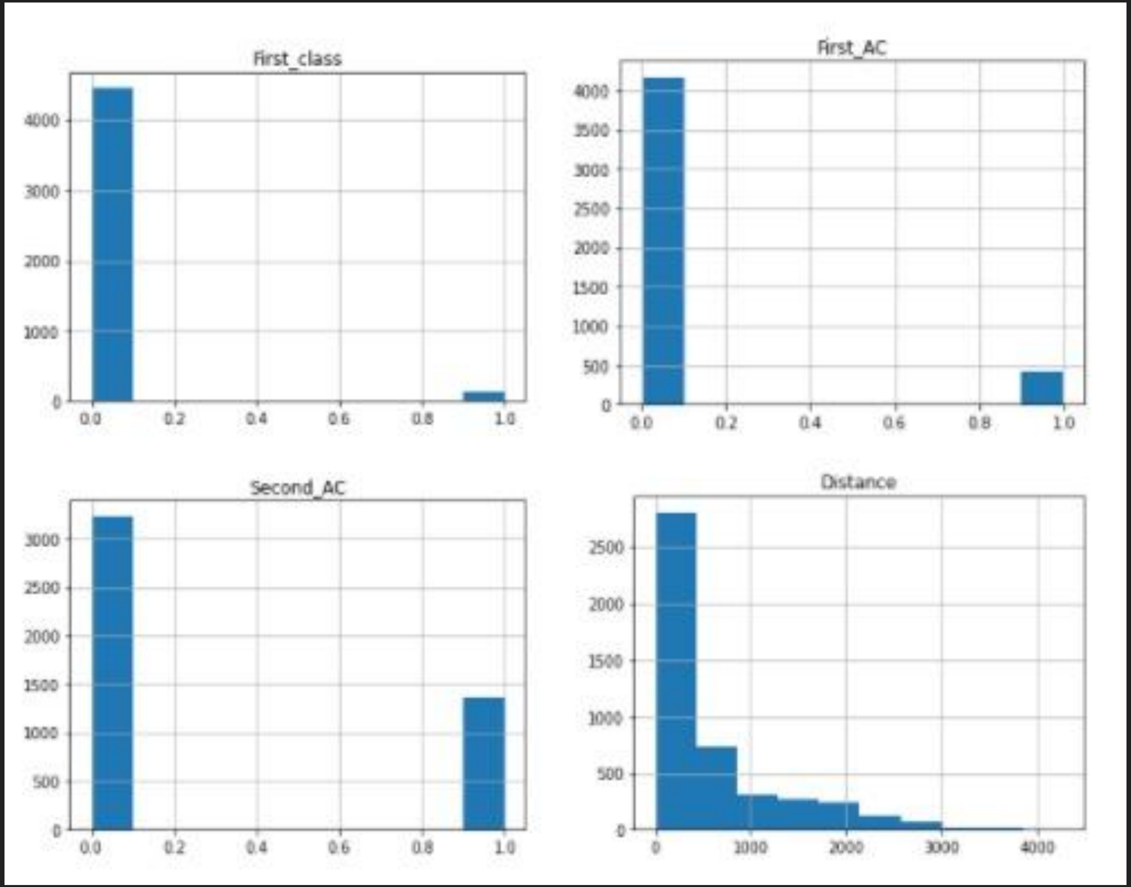
★ Histograms here shows about different availabilities, time and distance.



9

- ★ Here in the plots of Third_AC, Chair_car,Sleeper bar at ‘0’ represent non existence and bar at ‘1’ represents existence of that kind of coach

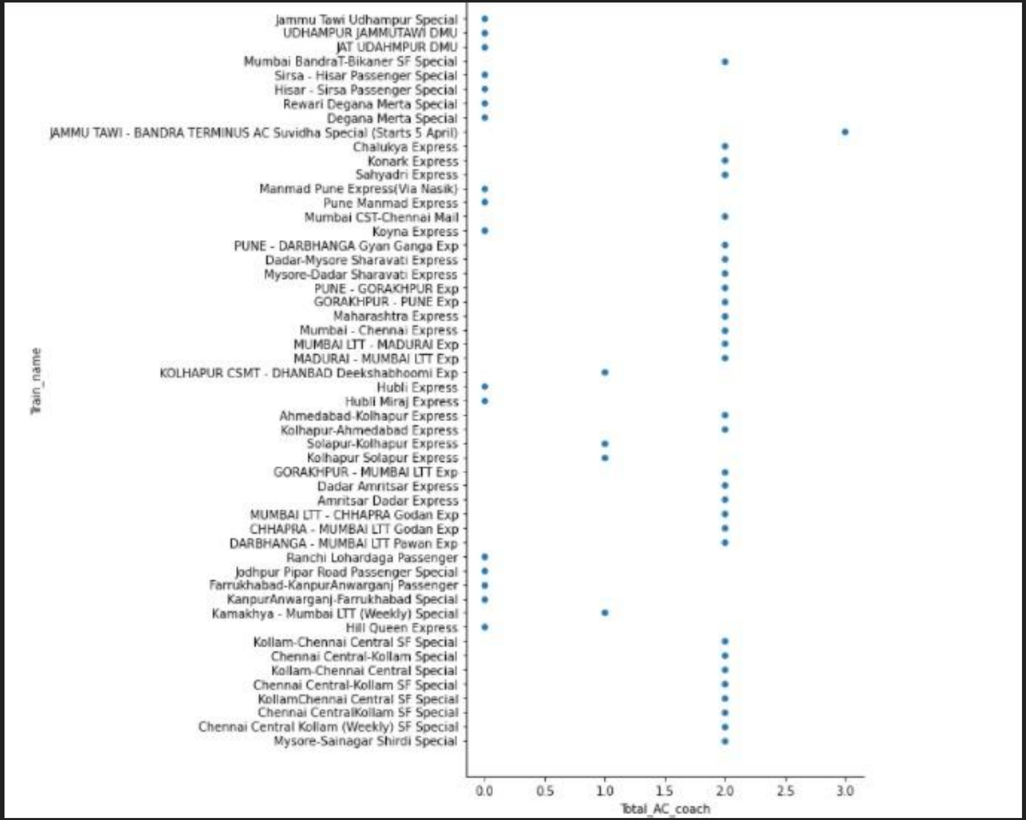
- ★ The histogram of duration in minutes shows number of trains in that particular duration.



- Here in the plots of First_class, First_AC,Second_AC bar at ‘0’ represent non existence and bar at ‘1’ represents existence of that kind of coach
- The histogram of distance shows number of trains in that particular length of distance.

```
1 # Plot showing number of AC coaches in trains
```

```
In [20]: sns.relplot(y="Train_name", x="Total_AC_coach", data=df[:60], height = 10, kind="scatter")
```



10

- ★ Here is a plot that shows total number of AC coaches in trains including all kinds of AC coaches
- ★ Total number of AC coaches on x-axis and train names on y-axis.
- ★ From the plot we can say that most number of trains has 2 AC coaches.

★ Now we have calculated some values like time and velocity which will help in next part of the analysis.

★ Here is the part of code finding total time in hours

Defining total time

```
In [21]:  1 df["Total_time"] = df["Duration_h"] + df["Duration_m"]/60

In [22]:  1 df["Total_time"].head()

Out[22]: 0     1.583333
         1     1.833333
         2     1.583333
         3     1.500000
         4    21.916667
```

★ Here is the part of code finding velocity using Distance and total time calculated just before.

Defining Velocity

```
In [23]:  1 df["Velocity"] = df["Distance"] / df["Total_time"]

In [24]:  1 df["Velocity"]

Out[24]: 0       33.473684
         1       28.909091
         2       33.473684
         3       35.333333
         4       55.300380
                   ...
         5201    27.471698
         5202    44.038005
         5203    37.333333
         5204    38.285714
         5205    61.500000
```

In [25]:

```
df
```

ation_m	Sleeper	From_station_name	...	Return_train	To_station_code	Second_AC	To_station_name	Duration_h	First_AC	Distance	Total_AC_coach	Total_time	Velocity
35.0	0	JAMMU TAWI	...	04602	UHP	0	UDHAMPUR	1.0	0	53.0	0	1.583333	33.473684
50.0	0	UDHAMPUR	...	04601	JAT	0	JAMMU TAWI	1.0	0	53.0	0	1.833333	28.909091
35.0	0	JAMMU TAWI	...	04604	UHP	0	UDHAMPUR	1.0	0	53.0	0	1.583333	33.473684
30.0	0	UDHAMPUR	...	04603	JAT	0	JAMMU TAWI	1.0	0	53.0	0	1.500000	35.333333
55.0	1	MUMBAI BANDRA TERMINUS	...	04727	BKN	1	BIKANER JN	21.0	0	1212.0	2	21.916667	55.300380

★ Now we have this total_time and velocity as extra columns which will be helpful to our upcoming hypothesis.

❑ Data Statistics:

```
In [26]:   1 df["Distance"].describe()

Out[26]: count    4593.000000
         mean      580.905508
         std       709.308473
         min         0.000000
         25%       104.000000
         50%       263.000000
         75%       765.000000
         max      4279.000000
         Name: Distance, dtype: float64

In [27]:   1 df["Total_time"].describe()

Out[27]: count    4593.000000
         mean       11.869922
         std        12.768564
         min         0.216667
         25%         3.083333
         50%         6.583333
         75%        16.000000
         max        85.500000
         Name: Total_time, dtype: float64

In [28]:   1 df["Velocity"].describe()

Out[28]: count    4593.000000
         mean       42.378615
         std        15.433769
         min         0.000000
         25%        32.538462
         50%        41.647059
         75%        52.424908
         max       540.000000
         Name: Velocity, dtype: float64
```

★ Now using the ‘describe’ function we have obtained various measures of central tendency and of dispersion.

★ We have obtained all those measures of Distance,Total time, Velocity.

★ We will use in the upcoming parts of the analysis.

★ Measures of central tendency and of dispersion of velocity are used to draw the probability distribution curve of it.

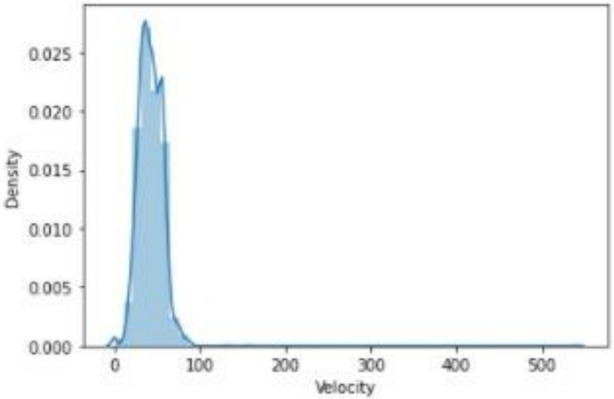
Density Curve of Velocity

In [29]:

```
1 sns.distplot(df['Velocity'])
```

```
C:\Python399\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will b
e removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibi
lity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]: <AxesSubplot:xlabel='Velocity', ylabel='Density'>



★ Measures of central tendency and of dispersion of velocity are used to draw this density curve of velocity.

❏ Hypothesis Testing:

★ Our Hypothesis: **The Velocity of Train having AC but not sleeper is higher than the mean velocity**

★ Here is the code to obtain Distribution curve for train having AC but not sleeper.

H0: μ = 42.38

H1: μ > 42.38

Distribution Curve of (where train having AC but not Sleeper)

In [31]:

```
import numpy as np
import matplotlib.pyplot as plt
def pdf(x):
    mean = np.mean(x)
    std = np.std(x)
    y_out = 1/(std * np.sqrt(2 * np.pi)) * np.exp( - (x - mean)**2 / (2 * std**2))
    return y_out

# To generate an array of x-values
x = (df[(df["Total_AC_coach"] != 0) & (df["Sleeper"] == 0)]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[(df["Total_AC_coach"] != 0) & (df["Sleeper"] == 0)]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
63.987482909138656
```

In [32]:
```python
df[(df["Total_AC_coach"] != 0) & (df["Sleeper"] == 0)]["Velocity"].describe()


# df['Velocity']
```

Out[32]:
```
count    182.000000
mean      63.987483
std       12.471842
min       32.800000
25%       55.088272
50%       63.970449
75%       74.185171
max       89.489362
Name: Velocity, dtype: float64
```

N = 182,X = 63.99,std = 12.47

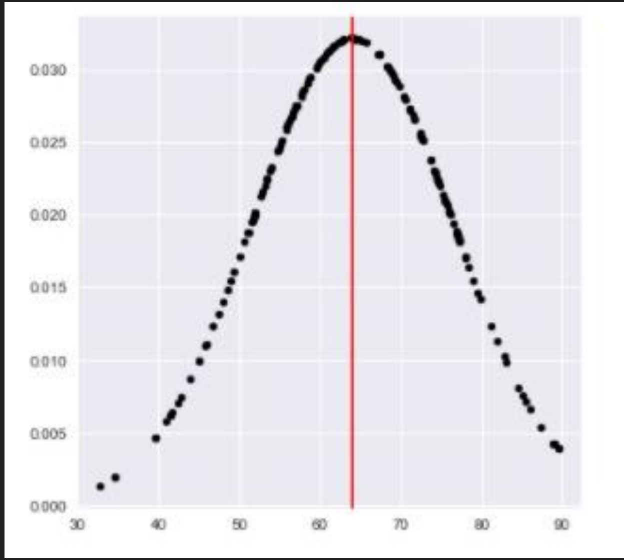
In [33]:
```python
import math
X = 63.99
μ = 42.38
std = 12.47
N = 182
Z = (X -μ)*math.sqrt(N)/std
```

In [34]:
```python
Z
```

Out[34]:
```
23.378896450797463
```

In [35]:
```python
# for alpha(0.05) and for One tail test
# Z(crital) = 1.65
if(Z>1.65):
    (print("H0 is rejected in favor of H1"))
else:
    (print("Do not reject H0"))
```

```
H0 is rejected in favor of H1
```



- ★ Probability distribution curve where train has ac but not sleeper.
- ★ We obtained that H0 is rejected in favor of H1.This means that our hypothesis statement is true.

- ★ The Velocity of Train having AC but not sleeper is higher than the mean velocity

❑ Zone wise analysis:

★ For better understanding we did zone wise analysis and also did drop a “?” zone as a part of cleaning.

```
In [36]:  1  df["Zone"].unique()

Out[36]: array(['NR', 'NWR', '?', 'WR', 'CR', 'SER', 'NER', 'NFR', 'SR', 'SWR',
                'SCR', 'ECoR', 'SECR', 'ER', 'KR', 'ECR', 'NCR', 'WCR'],
               dtype=object)

In [37]:  1  # for NR , NWR, WR , CR, SR for ( 5 zones)
          2

In [38]:  1   df = df.drop(df.index[df['Zone'] == "?"])

In [39]:  1  df["Zone"].unique()

Out[39]: array(['NR', 'NWR', 'WR', 'CR', 'SER', 'NER', 'NFR', 'SR', 'SWR', 'SCR',
                'ECoR', 'SECR', 'ER', 'KR', 'ECR', 'NCR', 'WCR'], dtype=object)
```

★ Here We have different zones like “NR”, “NWR”, “WR”, “CR”, “SR”, “SWR”, “KR” and many other zones

★ We are considering few zones(NR, NWR, WR, CR, SR) to analyse.

❑ Zone NR:

For Zone = "NR"

H0: μ = 42.38

H1: μ >42.38

Distribution Curve of (Zone "NR")

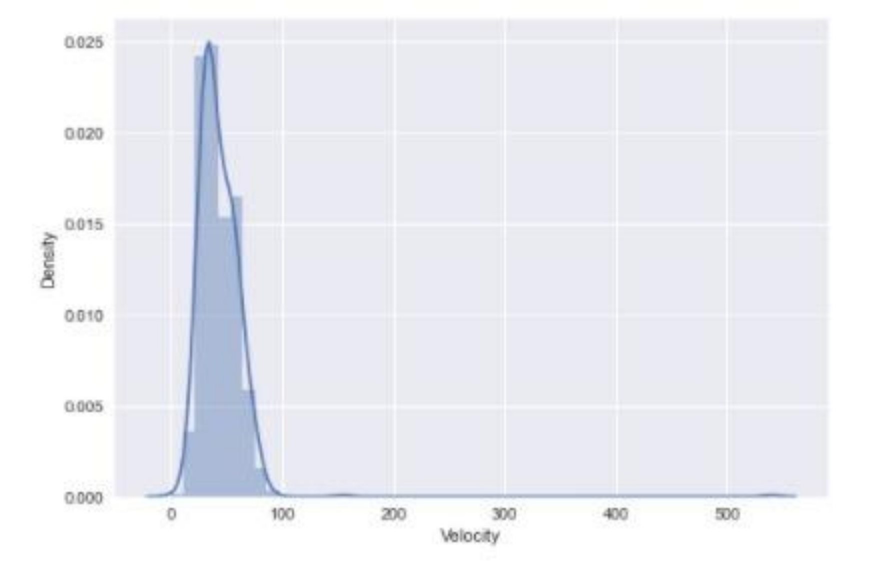
In [40]:

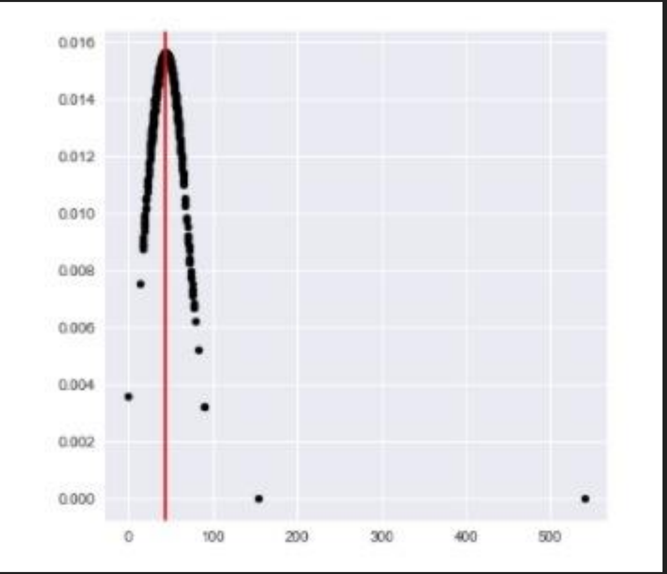
```
sns.distplot(df[df["Zone"] == "NR"]["Velocity"]) #density curve
# To generate an array of x-values
x = (df[df["Zone"] == "NR"]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[df["Zone"] == "NR"]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
C:\Python399\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will b
e removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibi
lity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
43.9172086461152
```

★ Here is the code for obtaining distribution curve for velocity of trains in Zone “NR”





- ★ Here is the distribution curve of velocity of trains in Zone “NR”
- ★ From this we can say that the more number of trains has velocity less than 100 km/h.

```
In [41]: df[df["Zone"] == "NR"]["Velocity"].describe()

Out[41]: count    597.000000
         mean      43.917209
         std       25.615002
         min        0.000000
         25%       31.411765
         50%       39.750000
         75%       54.834879
         max      540.000000
         Name: Velocity, dtype: float64
```

N = 597, X = 43.92, std = 25.62

```
In [42]: import math
         X = 43.92
         µ = 42.38
         std = 25.62
         N = 597
         Z = (X -µ)*math.sqrt(N)/std
```

```
In [43]: Z

Out[43]: 1.4686853437330787
```

```
In [44]: # for alpha(0.05) and for One tail test
         # Z(crital) = 1.65
         if(Z>1.65):
             (print("H0 is rejected in favor of H1"))
         else:
             (print("Do not reject H0"))

         Do not reject H0
```

- ★ Measures of central tendency and of dispersion of velocity of trains in the zone “NR” are obtained by using describe function.
- ★ From those we will calculate Z and then we get less than 1.65(critical).
- ★ So now we can not decide or conclude anything regarding this Zone “NR”
- ★ We can not show any results or conclude just by this analysis for Zone “NR”

❑ Zone NWR:

For Zone = "NWR"

H0: μ = 42.38

H1: μ > 42.38

Distribution Curve of (Zone "NWR")

In [45]:

```python
# To generate an array of x-values
x = (df[df["Zone"] == "NWR"]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[df["Zone"] == "NWR"]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
44.67100721198337
```

★ Here is the code for obtaining distribution curve for velocity of trains in Zone “NWR”

In [46]:
```
df[df["Zone"] == "NWR"]["Velocity"].describe()
```

Out[46]:
```
count    224.000000
mean      44.671007
std       10.178068
min       22.936709
25%       37.125000
50%       43.859723
75%       52.215005
max       70.434783
Name: Velocity, dtype: float64
```

N = 224, X = 44.67, std = 10.18

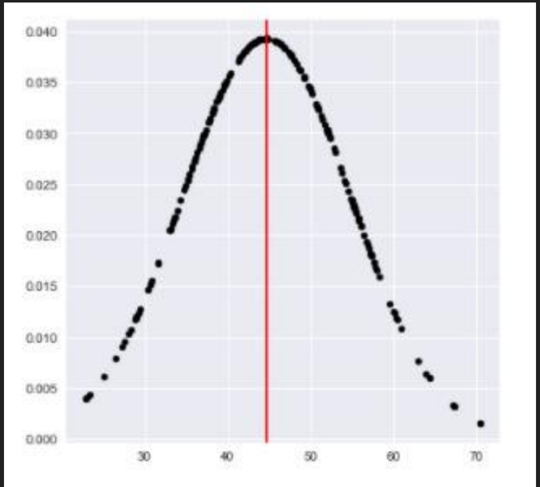
In [47]:
```
import math
X = 44.67
μ = 42.38
std = 10.18
N = 224
Z = (X -μ)*math.sqrt(N)/std
```

In [48]:
```
Z
```

Out[48]:
```
3.3667565484134863
```

In [49]:
```
# for alpha(0.05) and for One tail test
# Z(crital) = 1.65
if(Z>1.65):
    (print("H0 is rejected in favo0r of H1"))
else:
    (print("Do not reject H0"))
```

```
H0 is rejected in favo0r of H1
```



- ★ Here is the distribution curve of velocity of trains in Zone “NWR”.
- ★ Measures of central tendency and of dispersion of velocity of trains in the zone “NWR” are obtained by using describe function.
- ★ Z obtained is greater than 1.65 so H0 is rejected in favour of H1.
- ★ So velocity of trains in Zone “NWR” is higher than the mean velocity.

Zone WR:

For Zone = "WR"

H0: μ = 42.38

H1: μ >42.38

Distribution Curve of (Zone "WR")

In [50]:

```python
# To generate an array of x-values
x = (df[df["Zone"] == "WR"]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[df["Zone"] == "WR"]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
42.47283421335518
```

- ★ Here is the code for obtaining distribution curve for velocity of trains in Zone “WR”

```
In [51]: df[df["Zone"] == "WR"]["Velocity"].describe()
```

```
Out[51]: count    433.000000
         mean      42.472834
         std       13.771783
         min       11.506849
         25%       32.666667
         50%       42.302158
         75%       53.504587
         max       87.410526
         Name: Velocity, dtype: float64
```

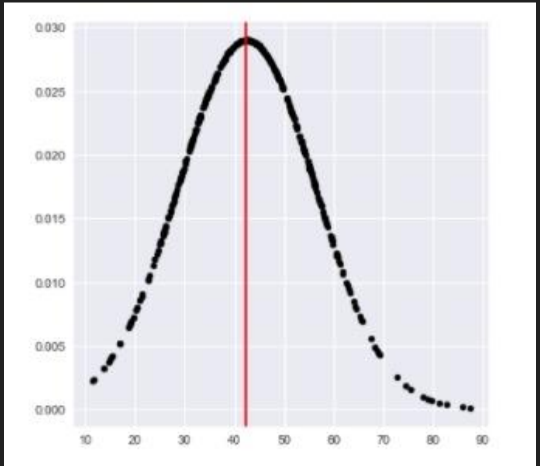
N = 433, X = 42.47, std = 13.77

```
In [52]: import math
         X = 42.47
         μ = 42.38
         std = 13.77
         N = 433
         Z = (X -μ)*math.sqrt(N)/std
```

```
In [53]: Z
```

```
Out[53]: 0.13600426174303243
```

```
In [54]: # for alpha(0.05) and for One tail test
         # Z(crital) = 1.65
         if(Z>1.65):
             (print("H0 is rejected in favor of H1"))
         else:
             (print("Do no reject H0"))

Do no reject H0
```



- ★ Here is the distribution curve of velocity of trains in Zone “WR”.
- ★ Measures of central tendency and of dispersion of velocity of trains in the zone “WR” are obtained by using describe function.
- ★ Z obtained is less than 1.65 so H0 is not rejected.
- ★ So we can not conclude anything about this zone with this kind of analysis.

❑ Zone CR:

For Zone = "CR"

H0: μ = 42.38

H1: μ > 42.38

Distribution Curve of (Zone "CR")

In [55]:

```python
x = (df[df["Zone"] == "CR"]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[df["Zone"] == "CR"]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
46.94728888767952
```

★ Here is the code for obtaining distribution curve for velocity of trains in Zone “CR”

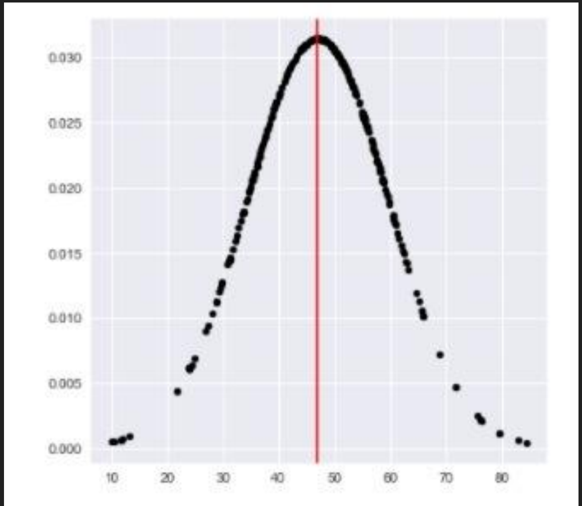
```
In [56]: df[df["Zone"] == "CR"]["Velocity"].describe()

Out[56]: count    333.000000
         mean      46.947289
         std       12.711611
         min       10.080000
         25%       38.270270
         50%       48.410023
         75%       55.400000
         max       84.352941
         Name: Velocity, dtype: float64
```

N = 333, X = 46.95, std = 12.71

```
In [57]: import math
         X = 46.95
         μ = 42.38
         std = 12.71
         N = 333
         Z = (X -μ)*math.sqrt(N)/std
```

```
In [58]: Z

Out[58]: 6.5613433745388345
```

```
In [59]: # for alpha(0.05) and for One tail test
         # Z(crital) = 1.65
         if(Z>1.65):
             (print("H0 is rejected in favour of H1"))
         else:
             (print("Do not reject H0 "))

         H0 is rejected in favour of H1
```



- ★ Measures of central tendency and of dispersion of velocity of trains in the zone “CR” are obtained by using describe function.
- ★ Z obtained is greater than 1.65 so H0 is rejected in favour of H1.
- ★ So velocity of trains in Zone “CR” is higher than the mean velocity.

❑ Zone SR:

For Zone = "SR"

H0: μ = 42.38

H1: μ > 42.38

Distribution Curve of (Zone "SR")

In [60]:
```
x = (df[df["Zone"] == "SR"]["Velocity"])

# To generate an array of
# y-values using corresponding x-values
y = pdf(x)

mean_value = (df[df["Zone"] == "SR"]["Velocity"]).mean()
print(mean_value)
# Plotting the bell-shaped curve
plt.style.use('seaborn')
plt.figure(figsize = (6, 6))
plt.scatter( x, y, marker = 'o', s = 25, color = 'black')
plt.axvline(x = mean_value, color = 'red')
plt.show()
```

```
45.705920863802326
```

★ Here is the code for obtaining distribution curve for velocity of trains in Zone “SR”

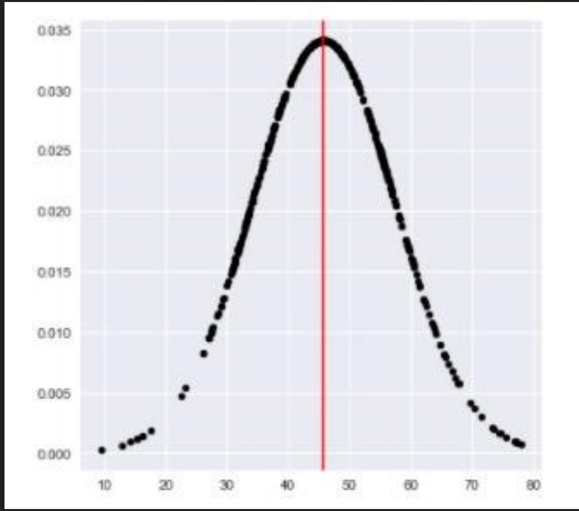
In [61]:
```
df[df["Zone"] == "SR"]["Velocity"].describe()
```
Out[61]:
```
count    498.000000
mean      45.705921
std       11.709679
min        9.517241
25%       36.758152
50%       45.578443
75%       55.068037
max       77.946269
Name: Velocity, dtype: float64
```

N = 498, X = 45.71, std = 11.71

In [62]:
```
import math
X = 45.71
µ = 42.38
std = 11.71
N = 498
Z = (X -µ)*math.sqrt(N)/std
```

In [63]:
```
Z
```
Out[63]:
```
6.346028377687721
```

In [64]:
```
# for alpha(0.05) and for One tail test
# Z(crital) = 1.65
if(Z>1.65):
    (print("H0 is rejected in favour of H1"))
else:
    (print("Do not reject H0"))
```
```
H0 is rejected in favour of H1
```



- ★ Measures of central tendency and of dispersion of velocity of trains in the zone “SR” are obtained by using describe function.
- ★ Z obtained is greater than 1.65 so H0 is rejected in favour of H1.
- ★ So velocity of trains in Zone “SR” is higher than the mean velocity.

- So far we have already done task 1 to 5, now we are here with task 6 that is prediction

❏ Prediction Task:

- We did a prediction interpreting Distance travelled by a train and the kinds of coaches (classes) in that train
- We used linear regression in multiple ways to draw the final output instead of classifiers.

In [142]:

```python
from matplotlib import pyplot as plt
import seaborn as sns
```

In [143]:

```python
import numpy as np
import pandas as pd
import csv

df = pd.read_csv("C:/Users/Yash/Desktop/DS Project/Trains.csv")

```

In [144]:

```python
df.shape
```

Out[144]:

```
(5208, 23)
```

- ★ Just like we did previously we have imported numpy, pandas, csv using import function.
- ★ By using the shape function we got the number of rows and columns in the data.

In [145]:

```
1 df.head()
```

Out[145]:

	coordinates	type	third_ac	arrival	from_station_code	Train_name	zone	chai
0	[[74.880117, 32.706975], [74.953339, 32.762368...	Feature	0	12:15:00	JAT	Jammu Tawi Udhampur Special	NR	
1	[[75.154881, 32.92664], [75.14542599999999, 32...	Feature	0	08:35:00	UHP	UDHAMPUR JAMMUTAWI DMU	NR	
2	[[74.880117, 32.706975], [74.953339, 32.762368...	Feature	0	17:50:00	JAT	JAT UDAHMPUR DMU	NR	
3	[[75.154881, 32.92664], [75.14542599999999, 32...	Feature	0	19:50:00	UHP	UDHAMPUR JAMMUTAWI DMU	NR	
4	[[72.840535, 19.061911], [72.840078, 19.069166...	Feature	1	12:30:00	BDTS	Mumbai BandraT-Bikaner SF Special	NWR	

5 rows × 23 columns

- ★ By using “df.head()” we got the starting 5 rows of the data with 23 columns.
- ★ Columns are just like the ones we had previously.
- ★ Those are,

1.Third_AC,Chair_car, First_class, sleeper -> 0 - doesn’t exist ,1 - exist

2. Arrival(time-24hrs format), From_station_code, Train_name, Zone, Duration_m (in minutes), From_station_name, Number(train number), Departure(time-24hrs format).

In [147]:

```
df
```

Out[147]:

	coordinates	type	third_ac	arrival	from_station_code	Train_name	zone
0	[[74.880117, 32.706975], [74.953339, 32.762368...	Feature	0	12:15:00	JAT	Jammu Tawi Udhampur Special	NR
1	[[75.154881, 32.92664], [75.14542599999999, 32...	Feature	0	08:35:00	UHP	UDHAMPUR JAMMUTAWI DMU	NR
2	[[74.880117, 32.706975], [74.953339, 32.762368...	Feature	0	17:50:00	JAT	JAT UDAHMPUR DMU	NR
...	...	...	...	...	...	...	...
5206	[[85.044629, 25.581921], [85.07942999999999, 2...	Feature	0	04:30:00	DNR	Danapur Giridih Express	?
5207	[[84.11877999999999, 23.843836], [84.146829, 2...	Feature	0	12:30:00	BRWD	Tribeni Link Express	?

5208 rows × 24 columns

- ★ Just by using “df ” we got all the 5208 rows and 24 columns of the data.
- ★ Next to this we got rid of null values using drop function.
- ★ Then we have checked the number of rows and columns again after dropping.

In [153]:

```
df.corr()
```

Out[153]:

	third_ac	chair_car	first_class	duration_m	sleeper	second_ac	duration_h
third_ac	1.000000	-0.184928	-0.030780	-0.027773	0.824601	0.859804	0.706766
chair_car	-0.184928	1.000000	-0.006725	0.005497	-0.221219	-0.186833	-0.120235
first_class	-0.030780	-0.006725	1.000000	0.052699	0.019334	-0.036248	-0.065602
duration_m	-0.027773	0.005497	0.052699	1.000000	-0.033400	-0.027935	-0.052561
sleeper	0.824601	-0.221219	0.019334	-0.033400	1.000000	0.787980	0.687435
second_ac	0.859804	-0.186833	-0.036248	-0.027935	0.787980	1.000000	0.664302
duration_h	0.706766	-0.120235	-0.065602	-0.052561	0.687435	0.664302	1.000000
first_ac	0.390899	-0.012260	-0.005972	-0.016649	0.263780	0.434543	0.168346
distance	0.739609	-0.090658	-0.069837	-0.030656	0.669399	0.696220	0.978181
total_time	0.706797	-0.120223	-0.064468	-0.029932	0.687321	0.664290	0.999744

★ By using correlation function(“df.corr()”) we have correlated distance and kinds of coaches.

In [192]:

```
import sklearn
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor

X = df[['first_ac','sleeper','third_ac','second_ac','chair_car']]

y = df[['distance']]
```

In [212]:

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

- ★ We used 80% data for training and 20% data for testing of the model.
- ★ We defined features using variable X and distance by using Y which are our variables for linear regression Curve.

```python
lm = linear_model.LinearRegression()
model = lm.fit(df[['first_ac','sleeper','third_ac','second_ac','chair_car']], df[['distance']])
model.coef_
```

```
array([[-204.20040091,  190.24071854,  719.1457321 ,  359.89536261,
         172.55982897]])
```

- ★ In statistics, linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables)
- ★ By using the fit function we got the equation mentioned below with features first ac, sleeper, third ac, second ac, chair car
- ★ We are predicting distance in our model
- ★ From this above code for linear regression, we get equation coefficient.

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \epsilon_i$$

```
In [218]:
1  model.intercept_
Out[218]:
array([166.82796928])
```

★ By using linear regression, we obtained the line equation mentioned before and we got the intercept value 166.82

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

```
In [54]:  1  r2_score = model.score(X_test,y_test)
          2  print(r2_score)

0.653143508636860 9
```

★ For our model R^2 value comes out to be 0.65 for the test data.

Conclusion:

- The Velocity of Train having AC but not sleeper is higher than the mean velocity.
- The velocities of trains in zones “SR”, “CR”, “NWR” are higher than the mean velocity.
- To analyse the Zones “WR”, “NR” we don’t have enough data so we can not conclude about the trains velocity in the zones.
- IN PREDICTION, we can predict the distance travelled by a train if we have data regarding the classes of coaches in it with 0.65 as R-square value.

TEAM MEMBERS:

- Lakshya agarwal (Team Leader)
- Deepanshu Sharma
- Divya

THANK YOU!