BI Project Report

Hajra Abdul Hai – 14893

Selected dataset

Delivery truck trips data

Selected BI tool

Power BI

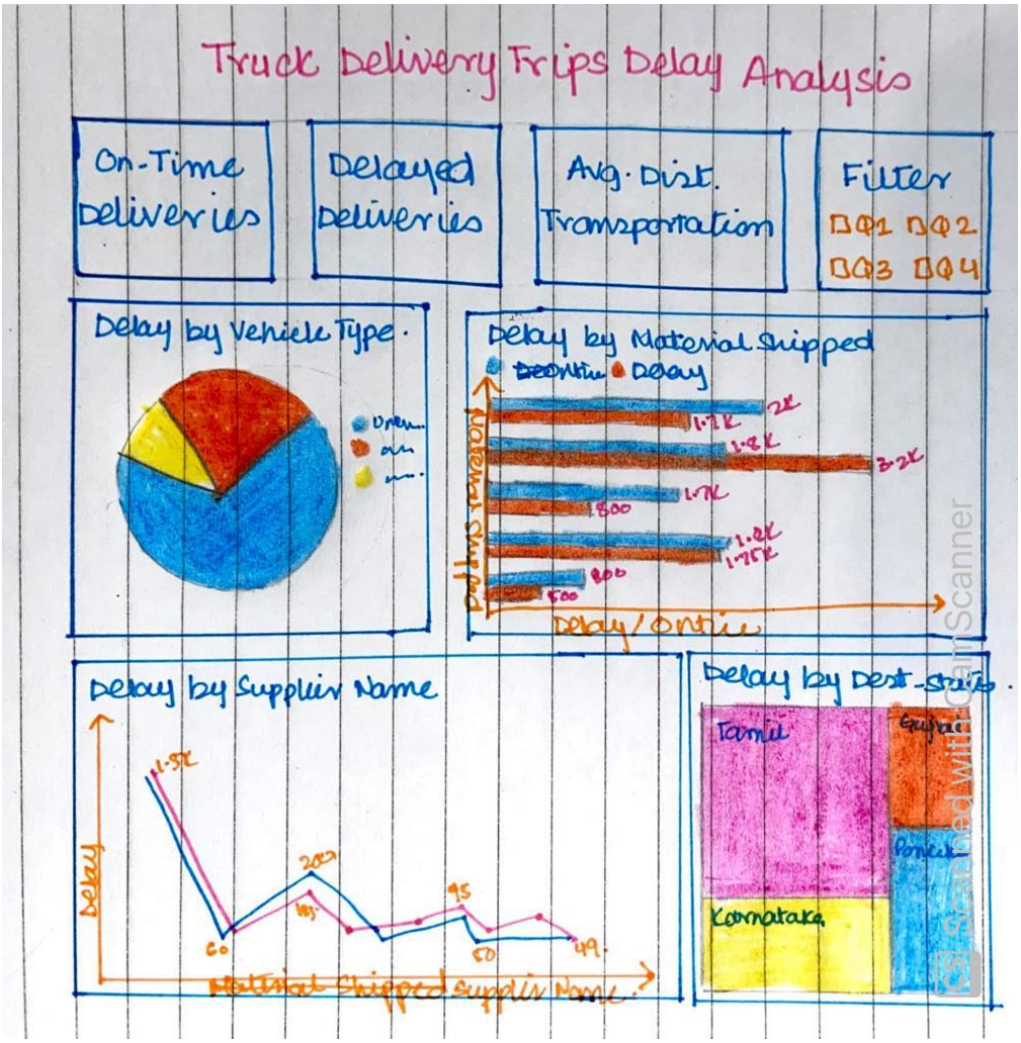
Background Knowledge

- Logistics helps in reducing costs and improves efficiency - With global trade growing more popular, logistics has become the heart of supply chains. Business leaders have realized they can reduce their costs by establishing partnerships with other businesses which offer transportation and warehousing.
- Effective transport improves a supply chain by decreasing (if not avoiding) waste of materials and time.
- Optimizing your transportation and logistics data management is essential to the efficiency of your business.
- Logistics helps delivering your product at the right place timely -Customers nowadays are more likely to impulse shop using a smartphone, and be equally as impatient about receiving their order. With professionally organized logistics, businesses are able to answer short-time requirements.
- Transportation involves fluctuations, factors such as delays and changes in fuel costs need to be taken into account in order to cover all possible scenarios that might jeopardize the efficient movement of goods.
- 70% of all freight tonnage moved in the U.S. goes on trucks.
- Create scorecards for delivery time and distance covered so stakeholders have immediate idea about the delivery trucks.
- Filter for vehicle type so that the user can easily customize their view by the type of vehicle they'd like to know more about.
- Monitoring it over time will also enable you to identify trends and patterns that can translate a certain difficulty or on the contrary a greater efficiency; it can also give you insights on the functioning of your supply chain.

Problem to solve

Why were there an increase in delayed deliveries in the months of July and August?

BI dashboard and story on paper with very brief explanation



The dashboard above displays the 3 KPIs as score cards which are on time deliveries, delay deliveries and avg distance. There is a time filter drilled down to quarters.

The pie chart shows that unknown vehicle type led to the greatest number of delays and the company should be more careful in that region to avoid angry clients.

Then, a line graph for delay w.r.t supplier name provides a visualization that the suppliers had significant number of on time deliveries but at the same time delayed deliveries were not that less. To avoid delayed deliveries the company should collect data specific to that attribute and try to get to the bottom of it.

In order to solve the problem, a stacked bar graph delay against material shipped identifies that some materials have almost the same or a greater number of delayed deliveries as on time deliveries. In this way, company can identify the materials they must take into account to strike a new delivery strategy in the coming year to avoid that many delayed deliveries.

Lastly the tree map displays the top 4 destination states with respect to the number of delayed deliveries that year. Through this map the company gets the idea about the states that need more thought process before delivering. The company should consider more factor like the weather and the route to avoid or even lessen the time of delay.

Wrangle

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import shapiro
from scipy.stats import normaltest
from scipy.stats import anderson
from scipy.stats import chi2_contingency
from scipy.stats import chi2

import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.graphics.gofplots import qqplot

plt.style.use('ggplot')
```

In [2]:
```
data=pd.read_excel("C:/Users/Hajra Hai/Desktop/University/Semester 8/Business Intelligence/Project/Delivery truck trip data.xlsx'
```

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

Out[3]:
```
(6880, 32)
```

In [4]:
```
#Show the features ( –columns) and the data in the dataframe
data.head()
```

Out[4]:

	GpsProvider	BookingID	Market/Regular	BookingID_Date	vehicle_no	Origin_Location	Destination_Location	Org_lat_
0	CONSENT TRACK	MVCV0000927/082021	Market	2020-08-17 14:59:01.000	KA590408	TVSLSL-PUZHAL-HUB,CHENNAI,TAMIL NADU	ASHOK LEYLAND PLANT 1-HOSUR,HOSUR,KARNATAKA	13.1550,80.1
1	VAMOSYS	VCV00014271/082021	Regular	2020-08-27 16:22:22.827	TN30BC5917	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	12.8390,79.9
2	CONSENT TRACK	VCV00014382/082021	Regular	2020-08-27 17:59:24.987	TN22AR2748	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	11.8710,79.7
3	VAMOSYS	VCV00014743/082021	Regular	2020-08-28 00:48:24.503	TN28AQ0781	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	12.8390,79.9
4	VAMOSYS	VCV00014744/082021	Regular	2020-08-28 01:23:19.243	TN68F1722	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	11.8720,79.6

5 rows × 32 columns

In [5]:
```
#Show the features ( –columns) and the data in the dataframe
data.tail()
```

In [6]:
```
#Get a summary on the dataframe including datatypes and shape.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6880 entries, 0 to 6879
Data columns (total 32 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   GpsProvider                     5927 non-null   object
 1   BookingID                       6880 non-null   object
 2   Market/Regular                  6880 non-null   object
 3   BookingID_Date                  6880 non-null   datetime64[ns]
 4   vehicle_no                      6880 non-null   object
 5   Origin_Location                 6880 non-null   object
 6   Destination_Location            6880 non-null   object
 7   Org_lat_lon                     6880 non-null   object
 8   Des_lat_lon                     6880 non-null   object
 9   Data_Ping_time                  5927 non-null   datetime64[ns]
 10  Planned_ETA                     6880 non-null   datetime64[ns]
 11  Current_Location                5916 non-null   object
 12  DestinationLocation             6880 non-null   object
 13  actual_eta                      6843 non-null   datetime64[ns]
 14  Curr_lat                        5927 non-null   float64
 15  Curr_lon                        5927 non-null   float64
 16  ontime                          2548 non-null   object
 17  delay                           4342 non-null   object
 18  OriginLocation_Code             6877 non-null   object
 19  DestinationLocation_Code        6853 non-null   object
 20  trip_start_date                 6880 non-null   datetime64[ns]
```

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
GpsProvider                          953
BookingID                              0
Market/Regular                         0
BookingID_Date                         0
vehicle_no                             0
Origin_Location                        0
Destination_Location                   0
Org_lat_lon                            0
Des_lat_lon                            0
Data_Ping_time                       953
Planned_ETA                            0
Current_Location                     964
DestinationLocation                    0
actual_eta                            37
Curr_lat                             953
Curr_lon                             953
ontime                              4332
delay                               2538
OriginLocation_Code                    3
DestinationLocation_Code              27
trip_start_date                        0
trip_end_date                        194
TRANSPORTATION_DISTANCE_IN_KM        712
vehicleType                          828
Minimum_kms_to_be_covered_in_a_day  4060
Driver_Name                         3429
Driver_MobileNo                     4189
customerID                             0
customerNameCode                       0
supplierID                             0
supplierNameCode                       0
Material Shipped                       0
dtype: int64
```

In [8]:
```
#dropping columns as almost 60% of the enteries are missing
data.drop('Minimum_kms_to_be_covered_in_a_day', axis=1, inplace=True)
```

In [9]:
```
#dropping unnecessary columns
data.drop('Curr_lat', axis=1, inplace=True)
data.drop('Curr_lon', axis=1, inplace=True)
data.drop('Data_Ping_time', axis=1, inplace=True)
```

In [10]:
```
#replacing the null values with mean on the 'TRANSPORTATION_DISTANCE_IN_KM' column
data['TRANSPORTATION_DISTANCE_IN_KM'] = data['TRANSPORTATION_DISTANCE_IN_KM'].fillna(data['TRANSPORTATION_DISTANCE_IN_KM'].mean()
```

In [11]:
```
#finding incorrect values for trip start date
data.sort_values('trip_start_date').head(5)
```

Out[11]:

	GpsProvider	BookingID	Market/Regular	BookingID_Date	vehicle_no	Origin_Location	Destination_Location	Org_lat_lo
6868	JTECH	WDSBKTP44502	Regular	2019-04-15 15:15:13	KA21A5090	Mugabala, Bangalore Rural, Karnataka	Peenya Small Industries, Bangalore, Karnataka	16.560192249175344,80.7922930915995
6264	NaN	WDSBKTP49392	Regular	2019-06-10 13:17:44	WB59B9152	Sonai, Kolkata, West Bengal	Kalyani, Nadia, West Bengal	23.525267916088961,87.2644243485708
5910	NaN	WDSBKTP41957	Regular	2019-03-18 12:19:22	AP26TE1258	Sedarapet, India	Redhills, Chennai, Tamil Nadu, India	12.0001,79.748399499999
6631	NaN	WDSBKTP41973	Regular	2019-03-18 16:24:18	TN20AJ1188	Kanchipuram, Tamil Nadu, India	Periyapatti, Tamil Nadu, India	12.8341735,79.70364
5912	NaN	WDSBKTP41974	Regular	2019-03-18 16:56:02	TN25AT7677	Sedarapet, India	Mylasandra, Bengaluru, Karnataka, India	12.0001,79.748399499999

In [12]:
```
#6868,6264 index rows having years as 1899 in all datetime features, may be it's a mistake
#As we have mistake in those 2 rows let's remove those
data.drop(data.index[[6868,6264]], inplace=True)
```

In [13]:
```
#creating a single column 'ontime/delay' from 'ontime' and 'delay' columns
#'1' represents ontime and '0' represents delay
data['ontime/delay']=data.ontime.replace({np.NaN, 'G'}, {'Delay', 'On-time'})
data['ontime']=data.ontime.replace({np.NaN, 'G'}, {'0', '1'})
data['delay']=data.delay.replace({np.NaN, 'R'}, {'0', '1'})
data.head()
data.head()
data.head()
```

Out[13]:

	GpsProvider	BookingID	Market/Regular	BookingID_Date	vehicle_no	Origin_Location	Destination_Location	Org_lat_
0	CONSENT TRACK	MVCV0000927/082021	Market	2020-08-17 14:59:01.000	KA590408	TVSLSL-PUZHAL-HUB,CHENNAI,TAMIL NADU	ASHOK LEYLAND PLANT 1-HOSUR,HOSUR,KARNATAKA	13.1550,80.1
1	VAMOSYS	VCV00014271/082021	Regular	2020-08-27 16:22:22.827	TN30BC5917	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	12.8390,79.9
2	CONSENT TRACK	VCV00014382/082021	Regular	2020-08-27 17:59:24.987	TN22AR2748	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	11.8710,79.7
3	VAMOSYS	VCV00014743/082021	Regular	2020-08-28 00:48:24.503	TN28AQ0781	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	DAIMLER INDIA COMMERCIAL VEHICLES,KANCHIPURAM,...	12.8390,79.9
4	VAMOSYS	VCV00014744/082021	Regular	2020-08-28 01:23:19.243	TN68F1722	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	LUCAS TVS LTD-PONDY,PONDY,PONDICHERRY	11.8720,79.6

5 rows × 29 columns

In [14]:
```
#dropping null values in 'current location' column since they are just 10%
#feature like gps provider, data ping time, current location, curr_lat, curr_lon having null values in same rows
#all the above mentioned features are dependent on each other and it's not feasible to impute those, so let's drop those rows

data.dropna(how='any', subset=['Current_Location'], inplace=True)
```

In [15]:
```
#finding null values in the dataset
data.isnull().sum()
```

Out[15]:
```
GpsProvider                      0
BookingID                        0
Market/Regular                   0
BookingID_Date                   0
vehicle_no                       0
Origin_Location                  0
Destination_Location             0
Org_lat_lon                      0
Des_lat_lon                      0
Planned_ETA                      0
Current_Location                 0
DestinationLocation              0
actual_eta                      26
ontime                           0
delay                            0
OriginLocation_Code              3
DestinationLocation_Code        27
trip_start_date                  0
trip_end_date                  194
TRANSPORTATION_DISTANCE_IN_KM    0
```

In [16]:
```
data['vehicle_states'] = data.vehicle_no.astype(str).str[:2]
data['Origin_states'] = data['Origin_Location'].str.split(',').apply(lambda x: x[-1])
data['Dest_states'] = data['Destination_Location'].str.split(',').apply(lambda x: x[-1])
```

In [17]:
```
#Performing Camel Case on the following field's values
data['vehicle_states']=data['vehicle_states'].replace(('tn', 'hr'), ('TN', 'HR'))

data['Origin_states']=data['Origin_states'].replace((' Maharashtra', 'TAMIL NADU', ' Gujarat', ' Tamil Nadu',
                                                     'RAJASTHAN', ' Haryana', 'PONDICHERRY',
                                                     ' Karnataka', 'KARNATAKA', 'GUJARAT', 'HARYANA', ' Rajasthan',
                                                     ' Uttar Pradesh', ' Pondicherry', ' West Bengal', ' Odisha',
                                                     ' Jharkhand', ' Bihar', ' Assam', ' Andhra Pradesh', ' Telangana',
                                                     ' Chattisgarh', ' Delhi', ' Kerala', ' Chandigarh', ' India',
                                                     'UTTAR PRADESH'),
                                                    ('Maharashtra', 'Tamil Nadu', 'Gujarat', 'Tamil Nadu',
                                                     'Rajasthan', 'Haryana', 'Pondicherry',
                                                     'Karnataka', 'Karnataka', 'Gujarat', 'Haryana', 'Rajasthan',
                                                     'Uttar Pradesh', 'Pondicherry', 'West Bengal', 'Odisha',
                                                     'Jharkhand', 'Bihar', 'Assam', 'Andhra Pradesh', 'Telangana',
                                                     'Chattisgarh', 'Delhi', 'Kerala', 'Chandigarh', 'India',
                                                     'Uttar Pradesh'))

data['Dest_states']=data['Dest_states'].replace((' Tamil Nadu', 'TAMIL NADU', 'RAJASTHAN', ' Maharashtra',
       'KARNATAKA', 'PONDICHERRY', 'MAHARASHTRA', ' Haryana', ' Gujarat',
       'GUJARAT', 'JHARKHAND', 'Haryana', ' Himachal Pradesh',
       ' Karnataka', ' Assam', 'HARYANA', ' Uttar Pradesh',
       'HIMACHAL PRADESH', ' West Bengal', ' Odisha', ' Rajasthan',
       ' Andhra Pradesh', ' Jharkhand', ' Telangana', ' Punjab', ' Delhi',
       ' Central Development Region', ' Madhya Pradesh', ' Meghalaya',
       ' Chattisgarh', ' Jammu & Kashmir', ' Uttarakhand', ' Chandigarh',
       ' Bihar', ' Pondicherry', ' Kerala', ' Dadra & Nagar Haveli',
       ' Goa', ' Sikkim', ' India'),
       ('Tamil Nadu', 'Tamil Nadu', 'Rajasthan', 'Maharashtra',
       'Karnataka', 'Pondicherry', 'Maharashtra', 'Haryana', 'Gujarat',
       'Gujarat', 'Jharkhand', 'Haryana', 'Himachal Pradesh',
       'Karnataka', 'Assam', 'Haryana', 'Uttar Pradesh',
       'Himachal Pradesh', 'West Bengal', 'Odisha', 'Rajasthan',
       'Andhra Pradesh', 'Jharkhand', 'Telangana', 'Punjab', 'Delhi',
       'Central Development Region', 'Madhya Pradesh', 'Meghalaya',
       'Chattisgarh', 'Jammu & Kashmir', 'Uttarakhand', 'Chandigarh',
       'Bihar', 'Pondicherry', 'Kerala', 'Dadra & Nagar Haveli',
       'Goa', 'Sikkim', 'India'))
```

In [18]:
```
#simplifying the name of states
for i in data.index:
    if data['Origin_states'][i]=='India':
        if data['Origin_Location'][i]=='Sedarapet, India':
            data['Origin_states'][i]='Pondicherry'
        elif data['Origin_Location'][i]=='Kanchipuram, Tamil Nadu, India':
            data['Origin_states'][i]='Tamil Nadu'
        elif data['Origin_Location'][i]=='Karnataka 562114, India':
            data['Origin_states'][i]='Karnataka'
        elif data['Origin_Location'][i]=='Sedarapet, Pondicherry, India':
            data['Origin_states'][i]='Pondicherry'
        elif data['Origin_Location'][i]=='Pondicherry, Puducherry, India':
            data['Origin_states'][i]='Pondicherry'
```

In [23]:
```
#impute null values in trip_end_date
import datetime
import random
df_sub=data[data['trip_end_date'].isna()]
for i in df_sub.index:
    if df_sub['ontime/delay'][i]==0:
        df_sub['trip_end_date'][i]=df_sub['actual_eta'][i]
    else:
        df_sub['trip_end_date'][i]=df_sub['Planned_ETA'][i]-datetime.timedelta(random.randint(0,3))

data=pd.concat([data, df_sub])

#remove duplicates as we have concatinationg those null in trip_end_date
data.dropna(subset=['trip_end_date'], inplace=True)
```

In [20]:
```
from geopy import distance

#let's find the distance between origin and destination
distances_km = []
for row in data.itertuples(index=False):
    distances_km.append(distance.distance(row.Org_lat_lon, row.Des_lat_lon).km)

data['Org_Dest_distance'] = distances_km
```

Missing Value Treatment

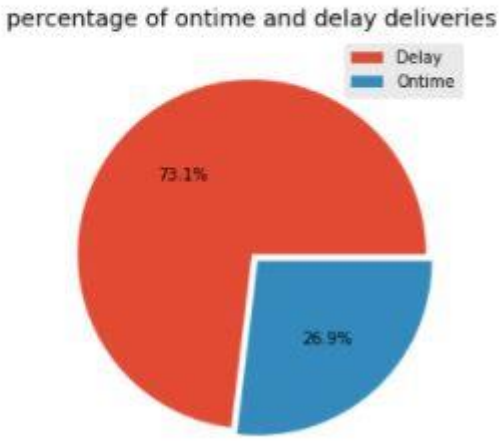
In [21]:
```
#let's check the percentage of null values in each feature
for col in data.columns:
    if data[col].isna().sum()>0:
        print(col, data[col].isna().mean().round(4)*100)
```

```
actual_eta 0.44
OriginLocation_Code 0.05
DestinationLocation_Code 0.45999999999999996
trip_end_date 3.2800000000000002
vehicleType 14.000000000000002
Driver_Name 41.839999999999996
Driver_MobileNo 54.690000000000005
```

In [22]:
```
#let's name unkown for null values in driver name
data['Driver_Name']=data['Driver_Name'].fillna('Unknown')

#name unkown for null values in vehicle type
data['vehicleType']=data['vehicleType'].fillna('Unknown')

#fill pervious date for actual.eta
data['actual_eta']=data['actual_eta'].fillna(method='ffill')
```

Exploring the dataset

In [24]:
```
#pie chart for percentage of ontime and delay deliveries
plt.rcParams['figure.figsize']=(5,5)
plt.pie(data['ontime/delay'].value_counts(), explode = (0, 0.05), autopct='%1.1f%%')
plt.title('percentage of ontime and delay deliveries')
plt.legend(['Delay', 'Ontime'])
plt.show()
```



In [27]:
```
#delay deliveries stats
print('star supliers with more number of delay delivery')
data[data['ontime/delay']=="Delay"][['supplierNameCode', 'TRANSPORTATION_DISTANCE_IN_KM']].groupby(['supplierNameCode']).agg('sum
                                                                                                                                ascending=Fals
```

```
star supliers with more number of delay delivery
```

Out[27]:

supplierNameCode	TRANSPORTATION_DISTANCE_IN_KM
TRANS CARGO INDIA	379651.893077
EKTA TRANSPORT COMPANY	327531.700519
Unknown	289308.225908
KRC Logistics	146383.712565
R.Sai logistics india PVT.LTD	111912.568847
Rajdhani Roadways	89655.419107
SUNITA CARRIERS PRIVATE LIMITED	69063.050000
VJ Logistics	62824.700519
PAWAN R LOGISTICS	58242.712565
PATANJALI PARIVAHAN PRIVATE LIMITED	44266.000000

In [26]:
```
#Distance covered stats for ontime deliveries
print('star supliers with distance covered(ontime)')
data[data['ontime/delay']=="On-time"][['supplierNameCode', 'TRANSPORTATION_DISTANCE_IN_KM']].groupby(['supplierNameCode']).agg('s
                                                                                                                                ascending=Fals
```

```
star supliers with distance covered(ontime)
```

Out[26]:

supplierNameCode	TRANSPORTATION_DISTANCE_IN_KM
EKTA TRANSPORT COMPANY	89964.000000
TRANS CARGO INDIA	63530.000000
KRC Logistics	57991.000000
VJ Logistics	39304.856282
Arvinth Transport	37798.000000
R.Sai logistics india PVT.LTD	31400.000000
EKTA TRAVELS	30225.000000
SR TRANSPORTS	25065.000000
Unknown	24156.706542
Sterling Translogistics Private Limited	23240.000000

In [25]:
```
#ontime deliveries stats
print('star supliers with more number of ontime delivery')
data[data['ontime/delay']=="On-time"][['supplierNameCode', 'TRANSPORTATION_DISTANCE_IN_KM']].groupby(['supplierNameCode']).agg('
                                                                                                                    ascending=Fals
```

```
star supliers with more number of ontime delivery
```

Out[25]:

supplierNameCode	TRANSPORTATION_DISTANCE_IN_KM
SR TRANSPORTS	120
SRI PACHIAMMAN TRANSPORT	93
VJ LOGISTICS	83
A S TRANSPORTS	80
NAMAKKAL SRI ANJINAYA TRANSPORT	74
ARVINTH TRANSPORT	64
EKTA TRANSPORT COMPANY	61
KRC Logistics	47
ESWAR TRANSPORT	46
Sree Sakthi Transport	46

In [28]:
```
#Distance covered stats for delayed deliveries
print('star supliers with distance covered(delay)')
data[data['ontime/delay']=="Delay"][['supplierNameCode', 'TRANSPORTATION_DISTANCE_IN_KM']].groupby(['supplierNameCode']).agg('cou
                                                                                                                    ascending=Fals
```

```
star supliers with distance covered(delay)
```

Out[28]:

supplierNameCode	TRANSPORTATION_DISTANCE_IN_KM
SUNITA CARRIERS PRIVATE LIMITED	324
A S TRANSPORTS	316
Unknown	279
K.RAMACHANDRAN TRANSPORTS	214
EKTA TRANSPORT COMPANY	187
S.B.TRANSPORT COMPANY	183
TRANS CARGO INDIA	171
A P R TRAILLER SERVICE	156
SHRI SAI ENTERPRISES	135
DISTRIBUTION LOGISTICS INFRASTRUCTURE PRIVATE LTD	124

In [29]:
```
#checking the supplier code for the unknown suppliers
data[data['supplierNameCode']=='Unknown']['supplierID'].value_counts()
```

Out[29]:
```
999    316
Name: supplierID, dtype: int64
```

In [30]:
```
#checking whether having driver's mobile number making any impact on ontime delivery
data['Driver_MobileNo'].values[data['Driver_MobileNo'].values>0]=1
data['Driver_MobileNo'].fillna(0, inplace=True)
data[data['Driver_MobileNo']==1]['ontime/delay'].value_counts()
```

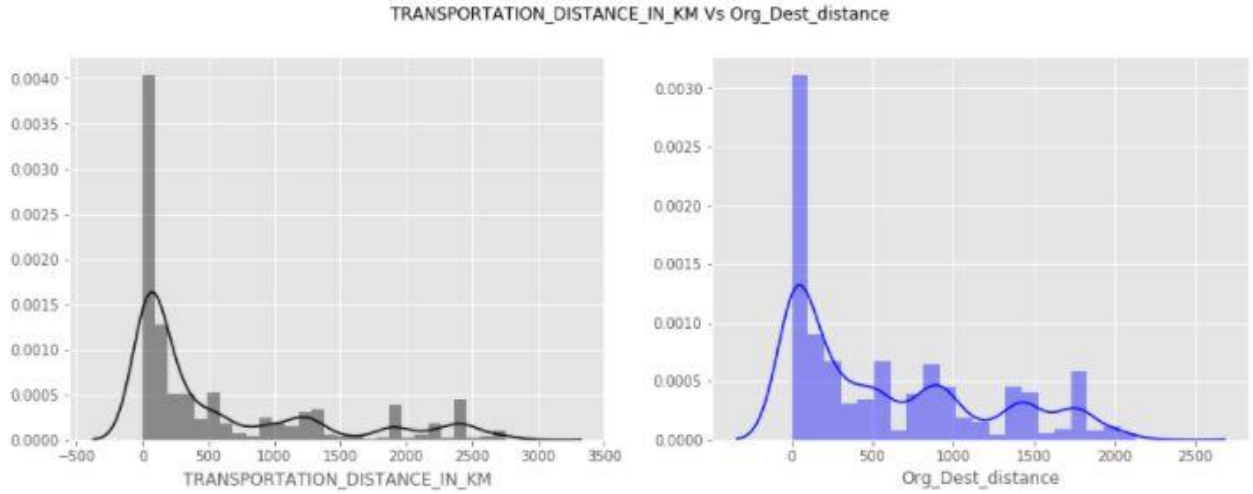
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarning: invalid value encountered in greater
```

Out[30]:
```
Delay      2003
On-time     677
Name: ontime/delay, dtype: int64
```

In [31]:
```
data[data['Driver_MobileNo']==0]['ontime/delay'].value_counts()
```

Out[31]:
```
Delay      2321
On-time     914
Name: ontime/delay, dtype: int64
```

In [32]:
```
#checking the pattern of 'transportation in km' vs 'distance between origin and destination'
plt.rcParams['figure.figsize']=15,5
plt.subplot(121)
sns.distplot(data['TRANSPORTATION_DISTANCE_IN_KM'], color='black')

plt.subplot(122)
sns.distplot(data['Org_Dest_distance'], color='blue')

plt.suptitle('TRANSPORTATION_DISTANCE_IN_KM Vs Org_Dest_distance')
plt.show()
```

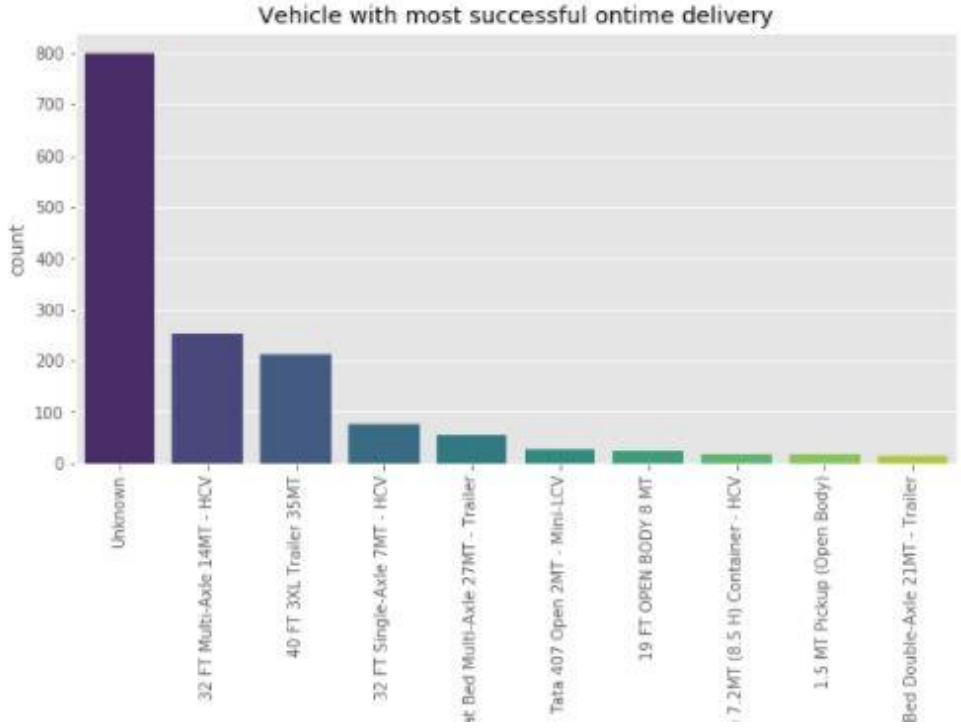


In [33]:
```
#fraudulant entries by suppliers
data[data['Org_Dest_distance']==0][['TRANSPORTATION_DISTANCE_IN_KM', 'supplierNameCode']].groupby(['supplierNameCode']).agg('sum
                                                                                                                    ascending=Fals
```

Out[33]:

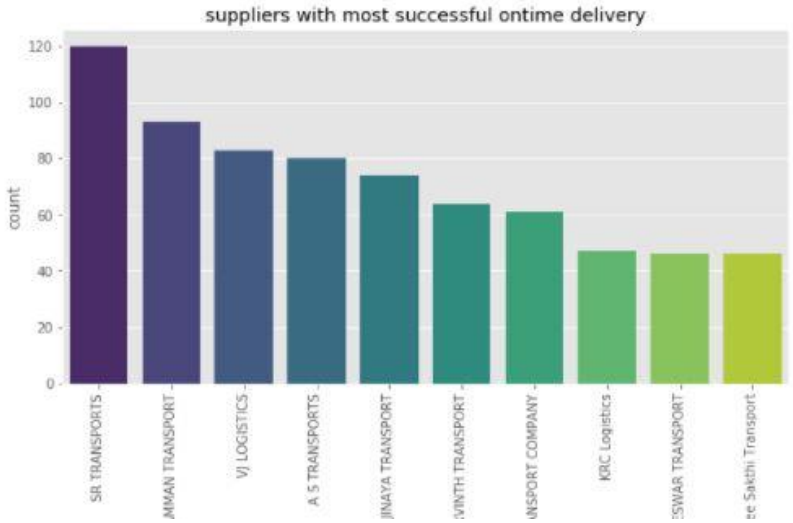
supplierNameCode	TRANSPORTATION_DISTANCE_IN_KM
ARVINTH TRANSPORT	16450.000000
ESWAR TRANSPORT	15381.000000
VJ LOGISTICS	9379.000000
KASAM TRANSPORT SERVICE	7740.000000
SUSEE TRANSPORTER	4661.000000
Sree Sakthi Transport	4430.850259
SR TRANSPORTS	3540.000000
VIRS TEMPO SERVICE	2340.000000
S R LOGISTICS	2340.000000
G.S. TRANSPORT	2100.000000

In [35]:
```
#bar graph of vehicle against ontime deliveries
plt.rcParams['figure.figsize']=10,5
sns.countplot(data[data['ontime/delay']=="On-time"]['vehicleType'],
              order=data[data['ontime/delay']=="On-time"]['vehicleType'].value_counts().head(10).index,
              palette='viridis')
plt.xticks(rotation=90)
plt.title('Vehicle with most successful ontime delivery')
plt.show()
```

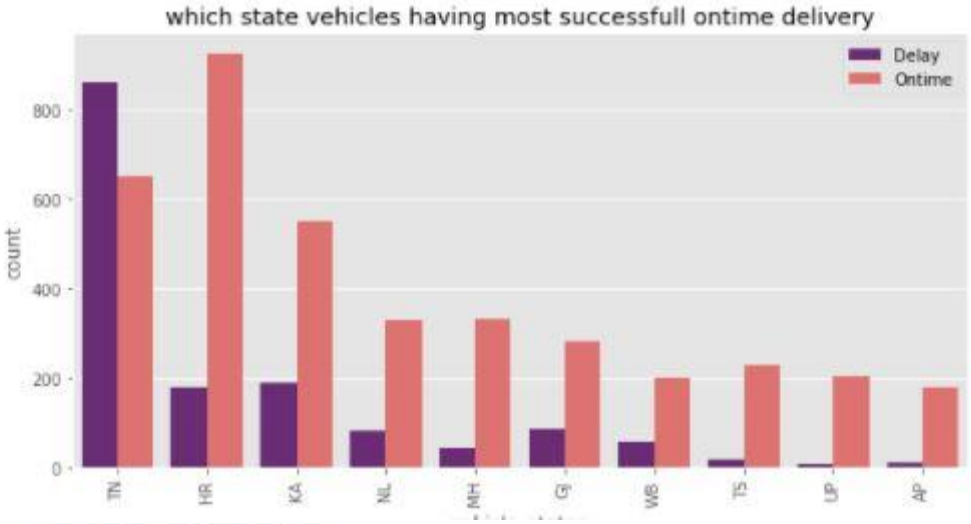


By this result, we can see that we don't have vehicletype data for most of the record. Definetly we shoud have record of vehicle type to get the suppliers having star vehicle. By this we can see the second most successful star vehicle which making more number of ontime delivery is '32 FT Multi-Axle 14MT - HCV'

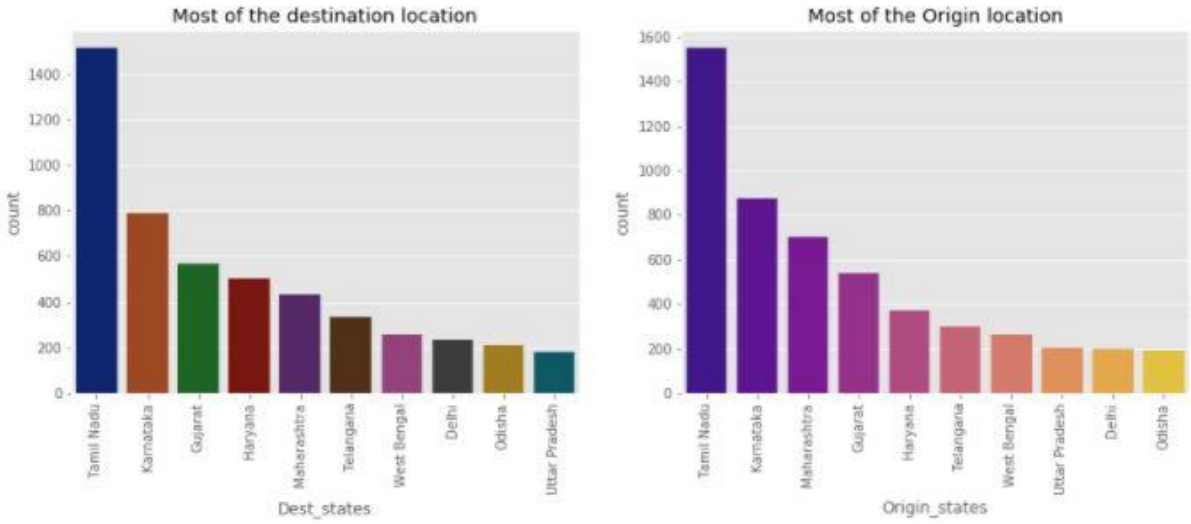
In [36]:
```
#bar graph of suppliers against ontime deliveries
plt.rcParams['figure.figsize']=10,5
sns.countplot(data[data['ontime/delay']=="On-time"]['supplierNameCode'],
              order=data[data['ontime/delay']=="On-time"]['supplierNameCode'].value_counts().head(10).index,
              palette='viridis')
plt.xticks(rotation=90)
plt.title('suppliers with most successful ontime delivery')
plt.show()
```



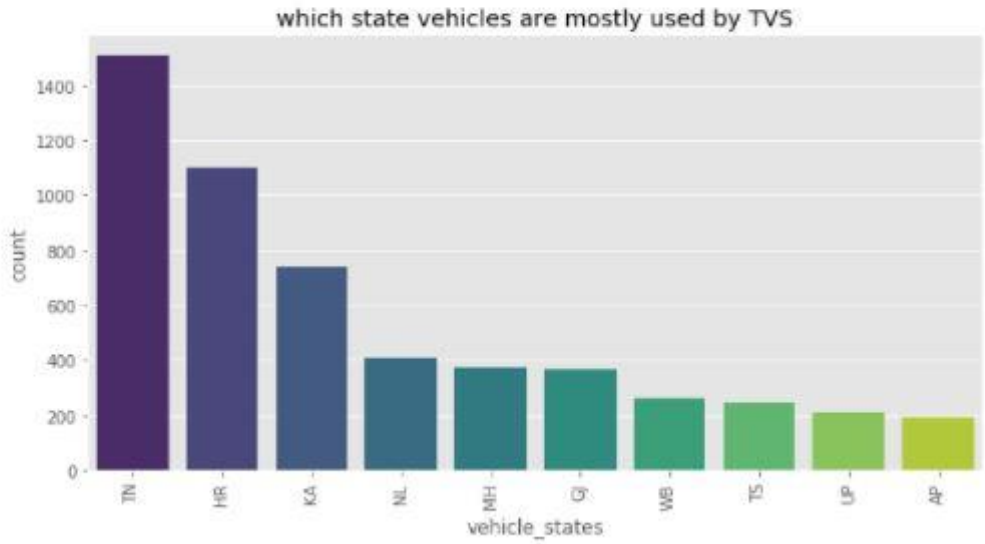
In [37]:
```
#bar graph of state vehicles against ontime deliveries
plt.rcParams['figure.figsize']=10,5
sns.countplot(data['vehicle_states'],
              order=data['vehicle_states'].value_counts().head(10).index,
               hue=data['ontime/delay'],
              palette='magma')
plt.xticks(rotation=90)
plt.title('which state vehicles having most successfull ontime delivery')
plt.legend(['Delay', 'Ontime'])
plt.show()
```



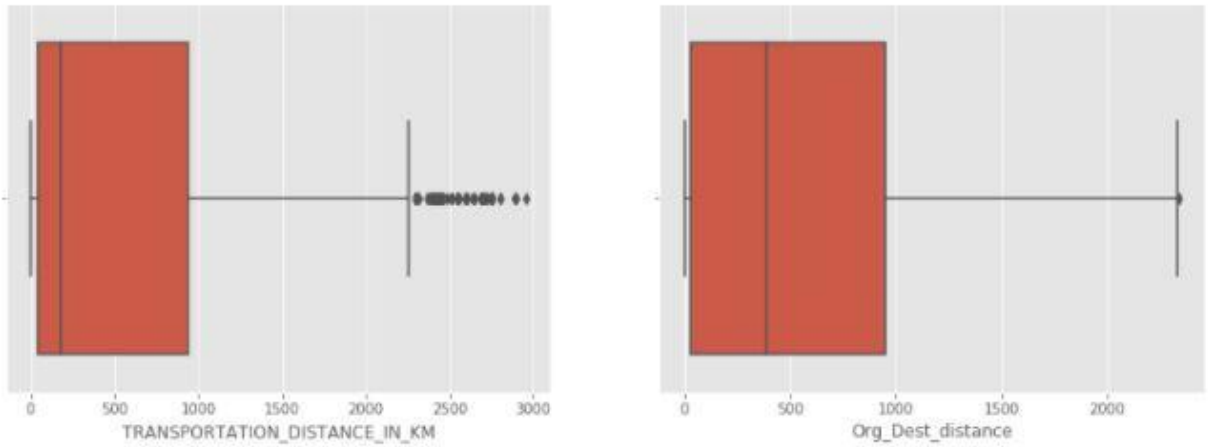
In [38]:
```
plt.rcParams['figure.figsize']=15,5

plt.subplot(121)
sns.countplot(data['Dest_states'],
              order=data['Dest_states'].value_counts().head(10).index,
              palette='dark')
plt.xticks(rotation=90)
plt.title('Most of the destination location')

plt.subplot(122)
sns.countplot(data['Origin_states'],
              order=data['Origin_states'].value_counts().head(10).index,
              palette='plasma')
plt.xticks(rotation=90)
plt.title('Most of the Origin location')

plt.show()
```



In [144]:
```
#bar graph of state vehicle against TVS
plt.rcParams['figure.figsize']=10,5
sns.countplot(data['vehicle_states'],
              order=data['vehicle_states'].value_counts().head(10).index,
              palette='viridis')
plt.xticks(rotation=90)
plt.title('which state vehicles are mostly used by TVS')
plt.show()
```



In [39]:
```
#box plot to find outliers
plt.subplot(121)
sns.boxplot(data['TRANSPORTATION_DISTANCE_IN_KM'])

plt.subplot(122)
sns.boxplot(data['Org_Dest_distance'])
plt.show()
```



we don't have major outliers in our data

Feature Encoding

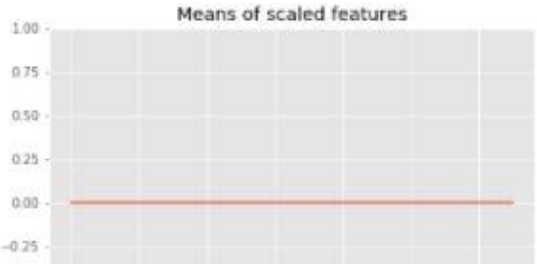
```
0]: #filter usefull data alone
df_cln=data[['Market/Regular ',
       'vehicle_no',
       'Current_Location',
       'TRANSPORTATION_DISTANCE_IN_KM',
       'vehicleType', 'Driver_Name',
       'Driver_MobileNo', 'customerID', 'supplierID',
       'Material Shipped', 'ontime/delay',
       'vehicle_states', 'Origin_states', 'Dest_states', 'Org_Dest_distance']]
```

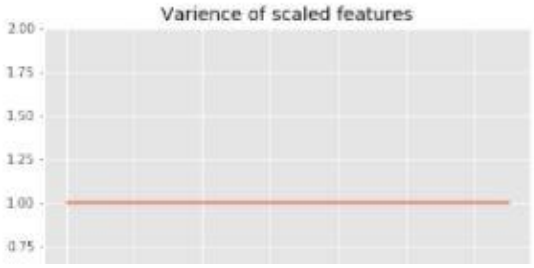
```
1]: df_cln['vehicle_no']=df_cln.vehicle_no.astype("category").cat.codes
df_cln['customerID']=df_cln.customerID.astype("category").cat.codes
df_cln['supplierID']=df_cln.supplierID.astype("category").cat.codes
df_cln['Current_Location']=df_cln.Current_Location.astype("category").cat.codes
df_cln['vehicleType']=df_cln.vehicleType.astype("category").cat.codes
df_cln['Material Shipped']=df_cln['Material Shipped'].astype("category").cat.codes
df_cln['Market/Regular ']=df_cln['Market/Regular '].astype("category").cat.codes
df_cln['Driver_Name']=df_cln['Driver_Name'].astype("category").cat.codes
df_cln['vehicle_states']=df_cln.vehicle_states.astype("category").cat.codes
df_cln['Origin_states']=df_cln['Origin_states'].astype("category").cat.codes
df_cln['Dest_states']=df_cln['Dest_states'].astype("category").cat.codes
```

Scaling Treatment

In [45]:
```
from sklearn.preprocessing import StandardScaler

sc=StandardScaler()
scaled=sc.fit_transform(x)
x_scl=pd.DataFrame(scaled, columns=x.columns)
```

In [46]:
```
#check weathear data is standardized or not
plt.subplot(121)
plt.ylim(-1,1)

means=[]
for i in range(x_scl.shape[1]):
    means.append(np.mean(x_scl.iloc[:,i]))
plt.plot(means, scaley=False)
plt.title('Means of scaled features')

plt.subplot(122)
plt.ylim(0,2)
vars=[]
for i in range(x_scl.shape[1]):
    vars.append(np.var(x_scl.iloc[:,i]))
plt.plot(vars, scaley=False)
plt.title('Varience of scaled features')
plt.show()
```

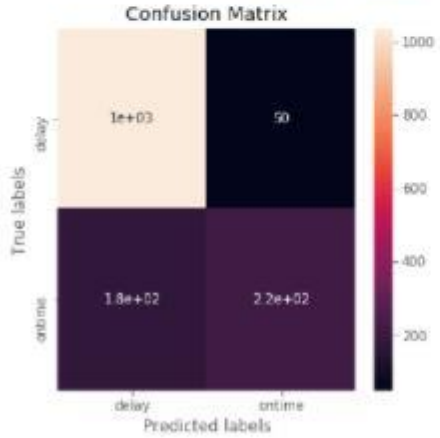




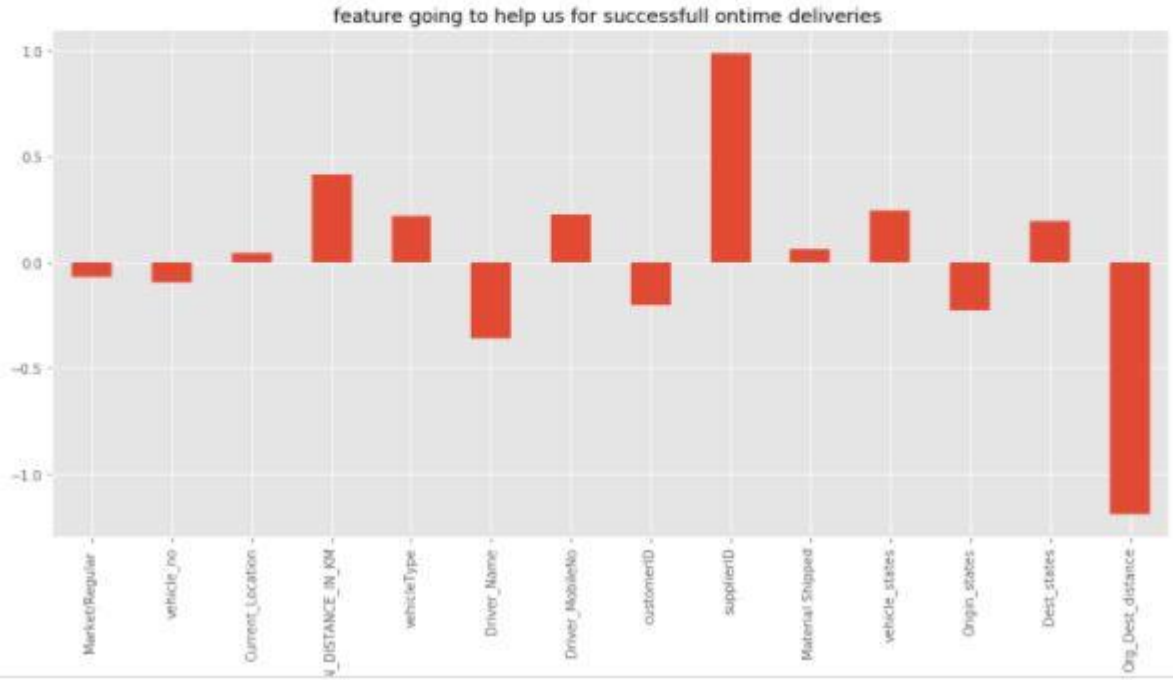
Model Building

In [47]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_scl, y, test_size=0.25)
```

In [48]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR=LogisticRegression()
LR.fit(X_train, y_train)
y_pred_LR=LR.predict(X_test)

plt.rcParams['figure.figsize']=5,5
ax= plt.subplot()
cm = confusion_matrix(y_test, y_pred_LR)
sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells

# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['delay', 'ontime']); ax.yaxis.set_ticklabels(['delay', 'ontime']);
```



In [49]:
```
#selecting most helpfull features
plt.rcParams['figure.figsize']=15,7
plt.style.use('ggplot')
weights=pd.Series(LR.coef_[0], index=['Market/Regular ', 'vehicle_no', 'Current_Location',
       'TRANSPORTATION_DISTANCE_IN_KM', 'vehicleType', 'Driver_Name',
       'Driver_MobileNo', 'customerID', 'supplierID', 'Material Shipped',
       'vehicle_states', 'Origin_states', 'Dest_states', 'Org_Dest_distance'])

params_weight =weights.plot(kind='bar', title='feature going to help us for successfull ontime deliveries ')
fig=params_weight.get_figure()
plt.show()
```



Parameters that impact on ontime delivery:

Current location, Transportation distance, Vehicle state, Vehicle type, Driver mobile number, Supplier, material shipped, Destination state.

[119]:
```
#ANOVA(analysis pf variance)
y = data['ontime']

# Ordinary Least Squares (OLS) model
model = ols('y ~ C(Q("Origin_Location"))', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print ("\nAnova => ontime/delay - Origin Location")
display(anova_table)

model = ols('y ~ C(Q("Destination_Location"))', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print ("\nAnova => ontime/delay - Destination Location")
display(anova_table)

model = ols('y ~ C(Q("Current_Location"))', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print ("\nAnova => ontime/delay - Current Location")
display(anova_table)

model = ols('y ~ C(Q("vehicleType"))', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print ("\nAnova => ontime/delay - vehicle Type")
display(anova_table)

model = ols('y ~ C(Q("supplierID"))', data=data).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print ("\nAnova => ontime/delay - supplierID")
display(anova_table)
```

```
Anova => ontime/delay - Origin Location
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:1832: ValueWarning
e full rank. The number of constraints is 168, but rank is 86
  'rank is %d' % (J, J_), ValueWarning)
```

	sum_sq	df	F	PR(>F)
C(Q("Origin_Location"))	4.758983e-27	168.0	11.951198	4.672434e-117
Residual	3.564850e-27	1504.0	NaN	NaN

```
Anova => ontime/delay - Destination Location
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:1832: ValueWarning
e full rank. The number of constraints is 459, but rank is 135
  'rank is %d' % (J, J_), ValueWarning)
```

	sum_sq	df	F	PR(>F)
C(Q("Destination_Location"))	5.409400e+03	459.0	4.812818e+30	0.0
Residual	3.565318e-27	1456.0	NaN	NaN

```
Anova => ontime/delay - Current Location
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:1832: ValueWarning
e full rank. The number of constraints is 2566, but rank is 1
  'rank is %d' % (J, J_), ValueWarning)
```

	sum_sq	df	F	PR(>F)
C(Q("Current_Location"))	1.023748e-12	2566.0	1.316680e-07	0.999711
Residual	2.824090e-06	932.0	NaN	NaN

The ANOVA output provides an estimate of how much variation in the dependent variable that can be explained by the independent variable.

The first column lists the independent variable along with the model residuals (aka the model error). The Df column displays the degrees of freedom for the independent variable (calculated by taking the number of levels within the variable and subtracting 1), and the degrees of freedom for the residuals. The Sum Sq column displays the sum of squares (a.k.a. the total variation) between the group means and the overall mean explained by that variable. The sum of squares for the 'Origon_Location' variable is 632.081512, while the sum of squares of the residuals is 530.976800. The Mean Sq column is the mean of the sum of squares, which is calculated by dividing the sum of squares by the degrees of freedom. The F-value column is the test statistic from the F test: The larger the F value, the more likely it is that the variation associated with the independent variable is real and not due to chance. The Pr(>F) column is the p-value of the F-statistic. This shows how likely it is that the F-value calculated from the test would have occurred if the null hypothesis of no difference among group means were true. Because the p-value of the independent variable, 'Origon_Location', is significant ($p < 0.05$), it is likely that fertilizer type does have a significant effect on ontime deliveries.

[52]:
```
data_crosstab = pd.crosstab(data['ontime/delay'], data['actual_eta'],
margins = False)

stat, p, dof, expected = chi2_contingency(data_crosstab)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

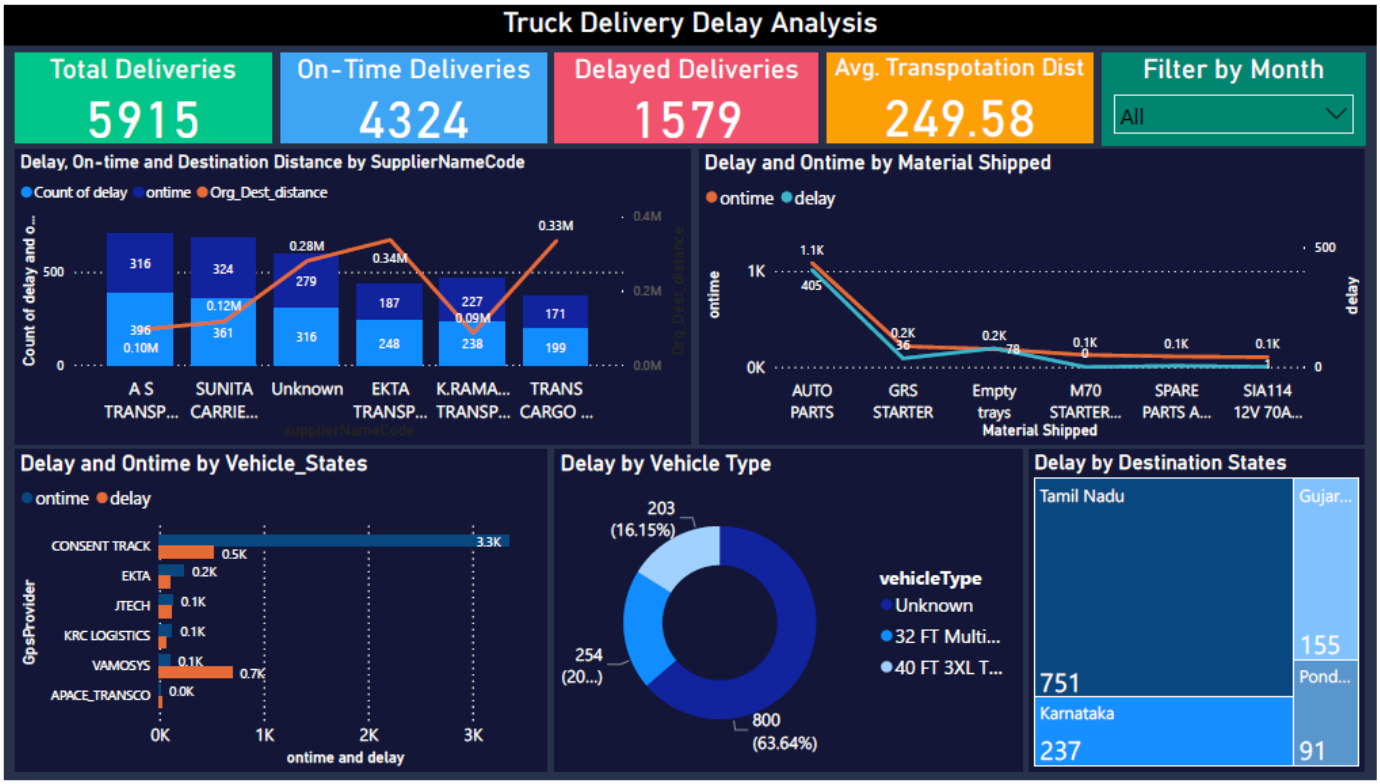
```
significance=0.050, p=0.234
Independent (fail to reject H0)
```

In [53]:
```
data_crosstab = pd.crosstab(data['ontime/delay'], data['TRANSPORTATION_DISTANCE_IN_KM'],
margins = False)

stat, p, dof, expected = chi2_contingency(data_crosstab)

# interpret p-value
alpha = 0.05
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
significance=0.050, p=0.000
Dependent (reject H0)
```

In [54]:
```
#saving the changes made to the data into a new csv
data.to_csv('Delivery_truck_trip_data_clean1.csv')
```

Image of the actual BI dashboard and story



On the dashboard template displayed above, three metrics are displayed, each bringing valuable information to the transportation management. Monitoring your average distance travelled is a primary KPI to measure, as it will impact the rest of the transport's efficiency. Knowing the distance will let you evaluate a certain time per KM and set targets. Optimizing this time will consequently let you load more and transport more. Monitoring it over time will also enable you to identify trends and patterns that can translate a certain difficulty or on the contrary a greater efficiency; it can also give you insights on the functioning of your supply chain.

The management of the routes is another important aspect. The deliveries, as a last-step in the completion of an order placed online, are the demonstration of your company's efficiency and reliability. They should be carried out in the timeframe originally given to your customer and with the correct order undamaged. Without all of these checkboxes ticked, the image of your business might suffer from it.

The other KPIs displayed at the top of the dashboard with the help of scorecards are, 'Ontime deliveries' and 'Delayed deliveries' the company has had. The dashboard focuses on analyzing the increase in delayed deliveries in the months of July and August. To find the trend, on the top left corner, a line and column chart of 'Supplier Name' and 'Destination Distance' w.r.t Ontime and delayed deliveries was displayed, it showed that there were 20% more delayed deliveries than the rest of the months. The most delayed deliveries were for which the Supplier Name was unknown. Therefore, the company must look into the Supplier Name before trusting them for deliveries as this could potentially lead to loss of potential clients.

On its right, a line chart of on-time and delay w.r.t 'Material shipped' was plotted and as expected auto parts which are of high demands have the greatest number of on-time deliveries but at the same time the greatest number of delays in the months of July and August. In addition to that empty trays and empty bins also had the same trend. The potential reason could be that the drivers did not consider them as a high priority. The company should focus on the training of their drivers so that they deliver the goods as soon as possible to avoid any complaints with the clients which would eventually lead to more deliver orders for the company.

Tree map on the bottom right is plotted to display the relation between delayed deliveries and the destination state. The map shows that the greatest number of delays were for the states: Tamil Nadu, Gujrat, Pondicherry and Karnataka. This could be because of many factors such as distance from origin or the weather. The

company should look into this matter as to why are there so delays for these states and try to ship the products from the nearest warehouse.

The pie chart again shows that there are too many delays for the unknown vehicles type and possible explanation could be that the vehicle when arrives at the warehouse for loading it is not appropriate for the product that has to be delivered. Hence, the company should make sure to know the vehicle type beforehand to avoid such situations and ultimately reducing the number of delays.

Lastly, a stacked bar graph shows on-time and delay deliveries w.r.t GPS Provider. It can be clearly seen that those vendors who provide GPS have a high rate of on-time deliveries against those who don't. so, the company should emphasis the importance of providing GPS to their vendors. Hence, it is suggested that the company should know all about the vehicle and the product and take into account any factors that could cause the delay and should train their drivers accordingly.

Stakeholder's Contribution/feedback:

- My dashboard had to many contrasting colors and Marium suggested to select a color theme and follow that.
- For wrangling she asked me to handle the null values since my dataset had too many of them.
- While wrangling the dataset she also explained that use of correlation matrix and Anova would result in better analysis.
- Marium also suggested that I should add a scorecard for total deliveries so that the viewer can get the whole idea at one glance.
- I had added a filter for date by quarter and Marium suggested that I drill that down to months, so that the viewer can get a better understanding in more detail.

My stakeholder contribution role for the other project

- I suggested Marium to change her color theme, since her original colors were not in harmony with each other.
- Her dashboard showcased only bar graphs and to show her full knowledge of Tableau BI, I asked her to add some other types of visuals.
- I suggested her to add another dimension to her line graph for the graph to give more insight.
- In wrangling I asked her to add some visuals to get some ideas about the data she was handling.
- Marium had displayed all four of the KPIs scorecards on the left corner and I suggested that she place them at the top in the middle of the dashboard.
- While Marium was wrangling her data, she could not identify a new KPI so I helped her in identifying the 3rd KPI.