Public transportation

optimization

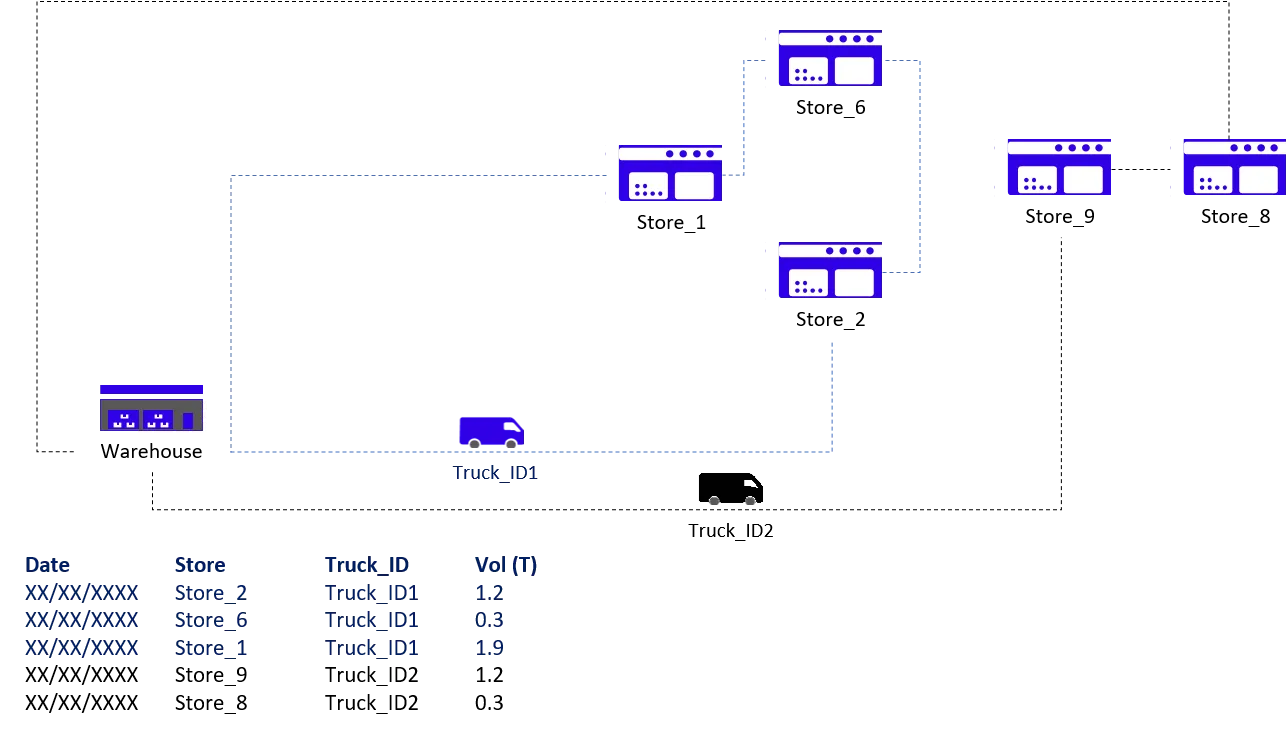


How do you make a transport plan with Python?

**1. Problem Statement**

Retail Stores Distribution with [**Full Truck Load (FTL)**](https://www.youtube.com/watch?v=PYkN24PMKd8)

* **1 Warehouse**delivering stores by using **three** types of Trucks  
  (3.5T, 5T, 8T)
* **49 Stores** delivered
* **12 Months** of Historical Data with **10,000 Deliveries**
* **7 days**a week of Operations
* **23 Cities**
* **84 Trucks** in your fleet



## 2. Objective: Reduce the Cost per Ton

Method: Shipment Consolidation

In this scenario, you are using 3rd party carriers that charge full trucks per destination:



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **s.no** | **City\_En** | **3.5T (Rmb)** | **5T (Rmb)** | **8T (Rmb)** | **3.5T (Rmb/Ton)** | **5T (Rmb/Ton)** | **8T (Rmb/Ton)** |
| **1** | City\_1 | 485 | 650 | 800 | 139 | 130 | 100 |
| **2** | City\_2 | 640 | 700 | 820 | 183 | 140 | 103 |
| 3 | City\_3 | 690 | 780 | 890 | 197 | 156 | 111 |
| 4 | City\_4 | 810 | 1,000 | 1,150 | 231 | 200 | 144 |
| 5 | City\_5 | 1,300 | 1,568 | 1,723 | 371 | 314 | 215 |
| 6 | City\_6 | 1,498 | 1,900 | 2,100 | 428 | 380 | 263 |
| 7 | City\_7 | 980 | 1,250 | 1,450 | 280 | 250 | 181 |
| 8 | City\_8 | 1,350 | 1,450 | 1,500 | 386 | 290 | 188 |
| 9 | City\_9 | 1,350 | 1,450 | 1,500 | 386 | 290 | 188 |
| 10 | City\_10 | 850 | 1,000 | 1,200 | 243 | 200 | 150 |

The table above shows rates applied by carriers for each city delivered for each type of truck. Observing**costs per ton are lower for larger trucks**, one lever of improvement is**maximizing shipments consolidation when building routes**.

Thus, the [**Route Transportation Planning Optimization's**](https://www.youtube.com/watch?v=lhDBTlsGDVc&t=1s) main target will be to cover a maximum number of stores per route.

# II. Data Processing: Understand the Current Situation

## 1. Import Datasets

Before starting to think about the [optimization model](https://www.youtube.com/watch?v=lhDBTlsGDVc&t=1s), your priority is to understand the current situation.

Starting with unstructured data coming from several sources, we’ll need to build a set of data frames to model our network and provide visibility on the loading rate and list of stores delivered for each route.

**Records of Deliveries per Store**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Truck\_ID** | **Store\_ID** | **FTL** | **Order** | **BOX** | **SKU** | **Loading (Tons)** |
| 9/1/2016 | Truck\_ID1 | Store\_ID1 | 3.5 | 16 | 311 | 83 | 2.404 |
| 9/1/2016 | Truck\_ID1 | Store\_ID2 | 3.5 | 18 | 178 | 83 | 1.668 |
| 9/1/2016 | Truck\_ID2 | Store\_ID3 | 3.5 | 10 | 74 | 54 | 0.81 |
| 9/1/2016 | Truck\_ID2 | Store\_ID4 | 3.5 | 19 | 216 | 88 | 2.413 |
| 9/1/2016 | Truck\_ID3 | Store\_ID5 | 3.5 | 10 | 117 | 54 | 1.119 |
| 9/1/2016 | Truck\_ID3 | Store\_ID6 | 3.5 | 15 | 294 | 92 | 2.962 |
| 9/1/2016 | Truck\_ID4 | Store\_ID7 | 3.5 | 5 | 42 | 19 | 0.421 |

### **Store Address**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Code** | **city** | **Long** | **Lat** | **address** | **Code** | **city** | **Long** |
| Store\_ID1 | City\_Store1 | 31.952792 | 118.8192708 | Address\_1 | Store\_ID1 | City\_Store1 | 31.952792 |
| Store\_ID2 | City\_Store2 | 31.952792 | 118.8192718 | Address\_2 | Store\_ID2 | City\_Store2 | 31.952792 |
| Store\_ID3 | City\_Store3 | 31.675948 | 120.7468221 | Address\_3 | Store\_ID3 | City\_Store3 | 31.675948 |
| Store\_ID4 | City\_Store4 | 31.664448 | 120.7700006 | Address\_4 | Store\_ID4 | City\_Store4 | 31.664448 |
| Store\_ID5 | City\_Store5 | 31.750971 | 119.9478857 | Address\_5 | Store\_ID5 | City\_Store5 | 31.750971 |
| Store\_ID6 | City\_Store6 | 31.791351 | 119.9232302 | Address\_6 | Store\_ID6 | City\_Store6 | 31.791351 |
| Store\_ID13 | City\_Store13 | 31.387863 | 121.2797154 | Address\_13 | Store\_ID13 | City\_Store13 | 31.387863 |

### **Transportation Costs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **s.no** | **City\_En** | **3.5T (Rmb)** | **5T (Rmb)** | **8T (Rmb)** | **3.5T (Rmb/Ton)** | **5T (Rmb/Ton)** |
| **1** | City\_1 | 485 | 650 | 800 | 139 | 130 |
| **2** | City\_2 | 640 | 700 | 820 | 183 | 140 |
| 3 | City\_3 | 690 | 780 | 890 | 197 | 156 |
| 4 | City\_4 | 810 | 1,000 | 1,150 | 231 | 200 |
| 5 | City\_5 | 1,300 | 1,568 | 1,723 | 371 | 314 |
| 6 | City\_6 | 1,498 | 1,900 | 2,100 | 428 | 380 |
| 7 | City\_7 | 980 | 1,250 | 1,450 | 280 | 250 |

## 2. Listing of stores delivered by each route

Let us process the initial data frame to list all stores delivered for each route.

## 1 Route = 1 Truck ID + 1 Date

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | #CreateTransportPlan | |  | def transport\_plan(data, dict\_trucks, capacity\_dict): | |  | # List of Stores per Truck for each DAY | |  | df\_plan = pd.DataFrame(data.groupby(['Date', 'TruckID'])['Code'].apply(list)) | |  | df\_plan.columns = ['List\_Code'] | |  | # List of Box Quantity | |  | df\_plan['List\_BOX'] = data.groupby(['Date', 'TruckID'])['BOX'].apply(list) | |  | # Mean of FTL | |  | df\_plan['FTL'] = data.groupby(['Date', 'TruckID'])['FTL'].mean() | |  | df\_plan['Capacity(T)'] = df\_plan['FTL'].map(capacity\_dict) | |  | df\_plan['List\_Loading'] = data.groupby(['Date', 'TruckID'])['Loading(T)'].apply(list) | |  | df\_plan['Count'] = df\_plan['List\_Loading'].apply(lambda t: len(t)) | |  | df\_plan['Total\_tons(T)'] = data.groupby(['Date', 'TruckID'])['Loading(T)'].sum() | |  |  | |  | # Distribute: one shipment per col | |  | # Stores | |  | d = df\_plan['List\_Code'].apply(pd.Series) | |  | for col in d: | |  | df\_plan["Store%d" % (col+1)] = d[col] | |  | # Boxes number | |  | d = df\_plan['List\_BOX'].apply(pd.Series) | |  | for col in d: | |  | df\_plan["Box%d" % (col+1)] = d[col] | |  | # Shipments Tonnage | |  | d = df\_plan['List\_Loading'].apply(pd.Series) | |  | for col in d: | |  | df\_plan["Tons%d" % (col+1)] = d[col] | |  |  | |  | # Fill NaN + Drop useless columns | |  | df\_plan.fillna(0, inplace = True) | |  | if 1 == 0: | |  | df\_plan.drop(['List\_Code'], axis = 1, inplace = True) | |  | df\_plan.drop(['List\_BOX'], axis = 1, inplace = True) | |  | df\_plan.drop(['List\_Loading'], axis = 1, inplace = True) | |  |  | |  | return df\_plan |      |  | | --- | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |

## 3. Add cities covered by each route

Let us now calculate Transportation Costs invoiced by carriers for each route:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | ##PricingFunctions | |  | def f\_maxcity(list\_cities, list\_price): | |  | return list\_cities[list\_price.index(max(list\_price))] # Index of Maximum Price | |  | def inner\_stops(list\_cities, max\_city): | |  | return list\_cities.count(max\_city) - 1 | |  | def outer\_stops(list\_cities, max\_city): | |  | return len(list\_cities) - (list\_cities.count(max\_city)) | |  | def total\_price(max\_price, inner\_stops, outer\_stops, inner\_price, outer\_price): | |  | return max\_price + inner\_stops \* inner\_price + outer\_stops \* outer\_price | |  |  | |  | # Calculate Price | |  | def plan\_price(df\_strinfo, df\_plan, inner\_price, outer\_price): | |  |  | |  | # Dictionnary Ville | |  | dict\_ville = dict(zip(df\_strinfo.Code.values, df\_strinfo.City.values)) | |  |  | |  | # Price per Truck Size： 3.5T, 5T, 8T | |  | dict\_35, dict\_5, dict\_8 = [dict(zip(df\_strinfo.City.values, df\_strinfo[col].values)) for col in ['3.5T', '5T', '8T']] | |  |  | |  | # Mapping Cities | |  | f\_ville = lambda t: [dict\_ville[i] for i in t] # literal\_eval(t) | |  |  | |  | # Mapping Price | |  | f\_35 = lambda t: [dict\_35[i] for i in t] | |  | f\_5 = lambda t: [dict\_5[i] for i in t] | |  | f\_8 = lambda t: [dict\_8[i] for i in t] | |  |  | |  | # Mapping Price | |  | df\_plan['List\_City'] = df\_plan['List\_Code'].map(f\_ville) | |  | df\_plan['List\_Price35'] = df\_plan['List\_City'].map(f\_35) | |  | df\_plan['List\_Price5'] = df\_plan['List\_City'].map(f\_5) | |  | df\_plan['List\_Price8'] = df\_plan['List\_City'].map(f\_8) | |  |  | |  | # Maximum Price City | |  | f\_maxprice = lambda t: max(t) # Maximum Price | |  |  | |  | # Mapping First City | |  | df\_plan['Max\_Price35'] = df\_plan['List\_Price35'].map(f\_maxprice) | |  | df\_plan['Max\_Price5'] = df\_plan['List\_Price5'].map(f\_maxprice) | |  | df\_plan['Max\_Price8'] = df\_plan['List\_Price8'].map(f\_maxprice) | |  | df\_plan['Max\_City'] = df\_plan.apply(lambda x: f\_maxcity(x.List\_City, x.List\_Price35), axis = 1) | |  |  | |  | # Inner City Stop | |  | df\_plan['Inner\_Stops'] = df\_plan.apply(lambda x: inner\_stops(x.List\_City, x.Max\_City), axis = 1) | |  | df\_plan['Outer\_Stops'] = df\_plan.apply(lambda x: outer\_stops(x.List\_City, x.Max\_City), axis = 1) | |  |  | |  | # Total Price | |  | df\_plan['Price35'] = df\_plan.apply(lambda x: total\_price(x.Max\_Price35, x.Inner\_Stops, x.Outer\_Stops, | |  | inner\_price, outer\_price), axis = 1) | |  | df\_plan['Price5'] = df\_plan.apply(lambda x: total\_price(x.Max\_Price5, x.Inner\_Stops, x.Outer\_Stops, | |  | inner\_price, outer\_price), axis = 1) | |  | df\_plan['Price8'] = df\_plan.apply(lambda x: total\_price(x.Max\_Price8, x.Inner\_Stops, x.Outer\_Stops, | |  | inner\_price, outer\_price), axis = 1) | |  |  | |  | return df\_plan | |

**Visualization: % Deliveries per Truck Size**

## 

## 

**Insights**

* **Average Truck Size:** a large majority of small trucks
* **Cost per ton:** the inverse proportion of cost per ton and average truck size