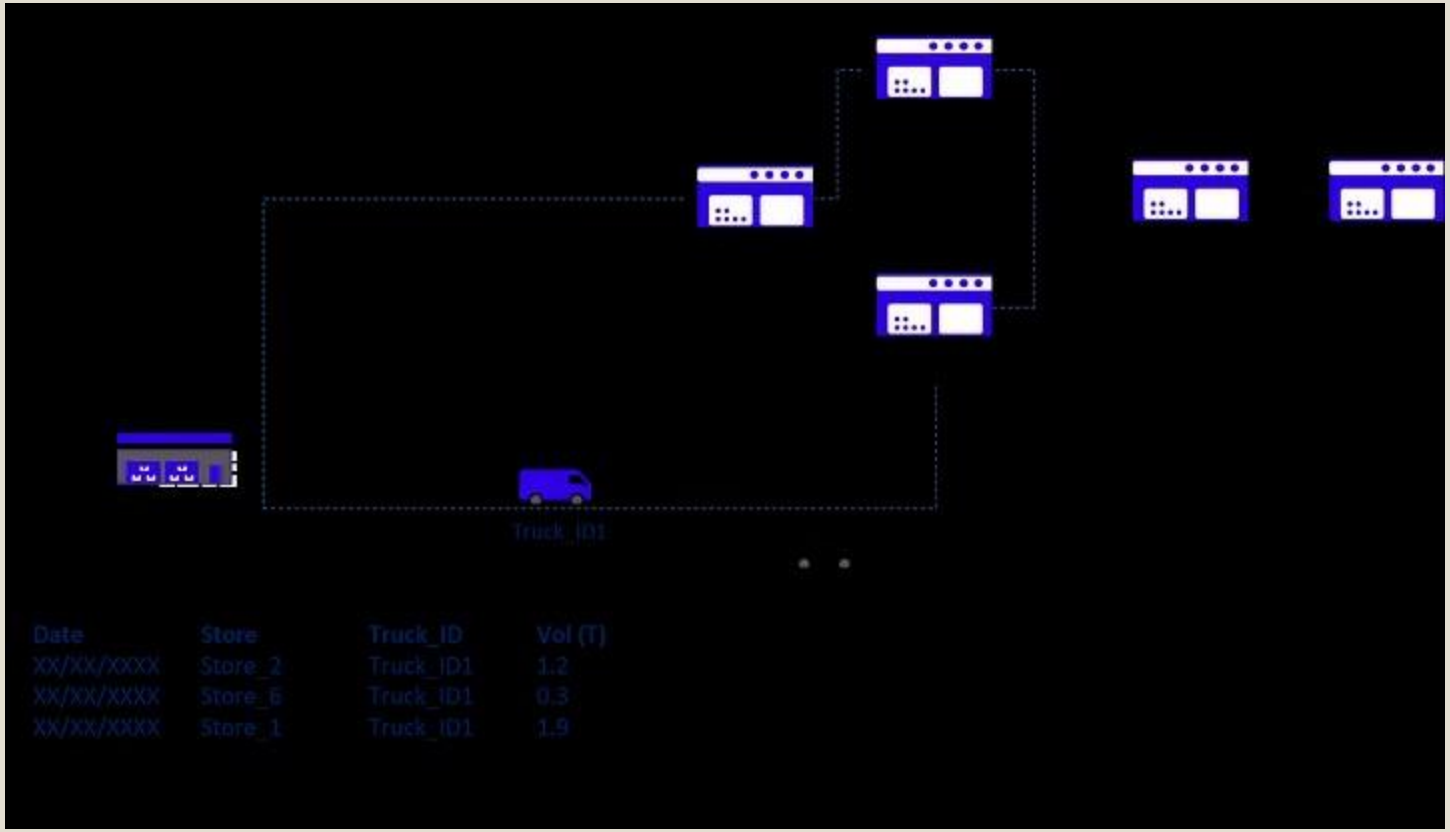
PUBLIC TRANSPORT ANALYSIS

T Mahalakshmi
A Shiny Angel
A Sharon rose mystica

- Processing Data: extract unstructured transportation records and process them to build your optimization model
- Improving Visibility: using Python visualization libraries to get clarity on current routing and truck loading rate
- Simulating Scenarios: build a model to simulate multiple routing scenarios and estimate the impact on average cost per ton

- . Problem Statement
- Retail Stores Distribution with Full Truck Load (FTL)
-
- 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


Truck_ID1
Date Store Truck_ID Vol (T)
XX/XX/XXXX Store_2 Truck_ID1 1.2
XX/XX/XXXX Store_6 Truck_ID1 0.3
XX/XX/XXXX Store_1 Truck_ID1 1.9

Raw dataset

City_En, 3.5T (Rmb) , 5T (Rmb) , 8T (Rmb) ,3.5T (Rmb/Ton),5T (Rmb/Ton),8T (Rmb/Ton)

City_1, 485 , 650 , 800 , 139 , 130 , 100

City_2, 640 , 700 , 820 , 183 , 140 , 103

City_3, 690 , 780 , 890 , 197 , 156 , 111

City_4, 810 ," 1,000 "," 1,150 ", 231 , 200 , 144

City_5," 1,300 "," 1,568 "," 1,723 ", 371 , 314 , 215

City_6," 1,498 "," 1,900 "," 2,100 ", 428 , 380 , 263

City_7, 980 ," 1,250 "," 1,450 ", 280 , 250 , 181

City_8," 1,350 "," 1,450 "," 1,500 ", 386 , 290 , 188

City_9," 1,350 "," 1,450 "," 1,500 ", 386 , 290 , 188

City_10, 850 ," 1,000 "," 1,200 ", 243 , 200 , 150

Data processing

- Understand the Current Situation
- 1. Import Datasets
- Before starting to think about the optimization model, 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.

- 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
- 9/1/2016,Truck_ID4,Store_ID8,3.5,12,125,88,1.138
- 9/1/2016,Truck_ID5,Store_ID9,5,18,201,95,2.19

Store address

- Code,city,Long,Lat,address
- Store_ID1,City_Store1,31.952792,118.8192708,Address_1
- Store_ID2,City_Store2,31.952792,118.8192718,Address_2
- Store_ID3,City_Store3,31.675948,120.7468221,Address_3
- Store_ID4,City_Store4,31.664448,120.7700006,Address_4
- Store_ID5,City_Store5,31.750971,119.9478857,Address_5
- Store_ID6,City_Store6,31.791351,119.9232302,Address_6
- Store_ID7,City_Store7,31.79233,119.9768294,Address_7
- Store_ID8,City_Store8,31.982972,119.5832084,Address_8
- Store_ID9,City_Store9,31.996161,119.6341775,Address_9
- Store_ID10,City_Store10,31.885547,121.1886473,Address_10
- Store_ID11,City_Store11,30.310079,120.1515734,Address_11
- Store_ID12,City_Store12,31.383616,121.2569408,Address_12
- Store_ID13,City_Store13,31.387863,121.2797154,Address_13

Transport cost

- City_En, 3.5T (Rmb) , 5T (Rmb) , 8T (Rmb) ,3.5T (Rmb/Ton),5T (Rmb/Ton),8T (Rmb/Ton)
- City_1, 485 , 650 , 800 , 139 , 130 , 100
- City_2, 640 , 700 , 820 , 183 , 140 , 103
- City_3, 690 , 780 , 890 , 197 , 156 , 111
- City_4, 810 ," 1,000 "," 1,150 ", 231 , 200 , 144
- City_5," 1,300 "," 1,568 "," 1,723 ", 371 , 314 , 215
- City_6," 1,498 "," 1,900 "," 2,100 ", 428 , 380 , 263
- City_7, 980 ," 1,250 "," 1,450 ", 280 , 250 , 181
- City_8," 1,350 "," 1,450 "," 1,500 ", 386 , 290 , 188
- City_9," 1,350 "," 1,450 "," 1,500 ", 386 , 290 , 188
- City_10, 850 ," 1,000 "," 1,200 ", 243 , 200 , 150

Listing of stores delivered by each route

```
# Create Transport Plan
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]
```

```
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
```

- Date,TruckID,List_Code,Capacity(T),List_Loading,Count,Total_tons(T),Store1,Store2,Store 3,Store4,Box1,Box2,Box3,Box4,Tons1,Tons2,Tons3,Tons4,Occupation(%),Available(T)
- 9/1/2016,Truck_ID1,['Store_ID6'],3.5,[2.91],1,2.91,ID6,0,0,0,243,0,0,0,2.91,0,0,0,83.14,0.59
- 9/1/2016,Truck_ID2,"['Store_ID34', 'Store_ID22', 'Store_ID9']",3.5,"[0.3, 1.37, 0.47]",3,2.14,ID34,ID22,ID9,0,31,116,44,0,0.3,1.37,0.47,0,61.14,1.36
- 9/1/2016,Truck_ID3,['Store_ID18'],3.5,[1.5],1,1.5,ID18,0,0,0,174,0,0,0,1.5,0,0,0,42.86,2
- 9/1/2016,Truck_ID4,['Store_ID37'],3.5,[2.3],1,2.3,ID37,0,0,0,179,0,0,0,2.3,0,0,0,65.71,1.2
- 9/1/2016,Truck_ID5,"['Store_ID34', 'Store_ID48']",3.5,"[2.14, 0.51]",2,2.65,ID34,ID48,0,0,168,46,0,0,2.14,0.51,0,0,75.71,0.85

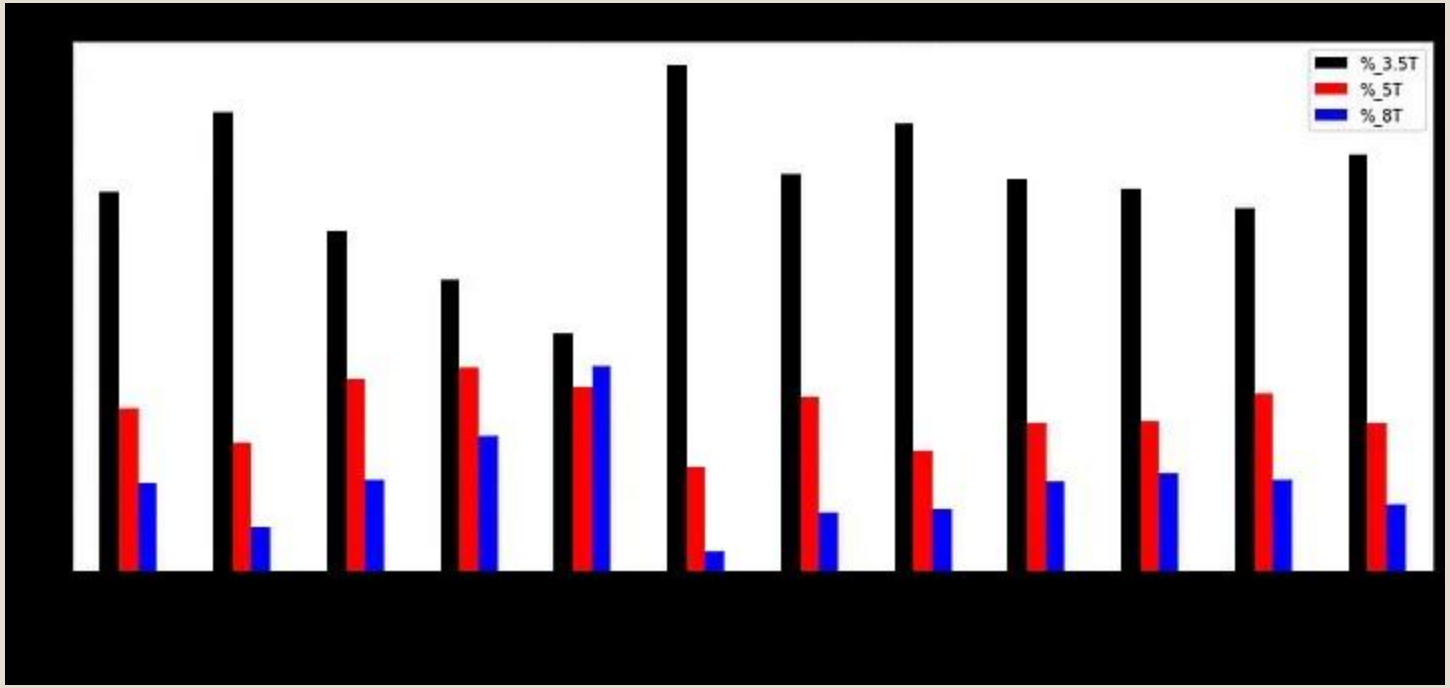
Add cities covered by each route

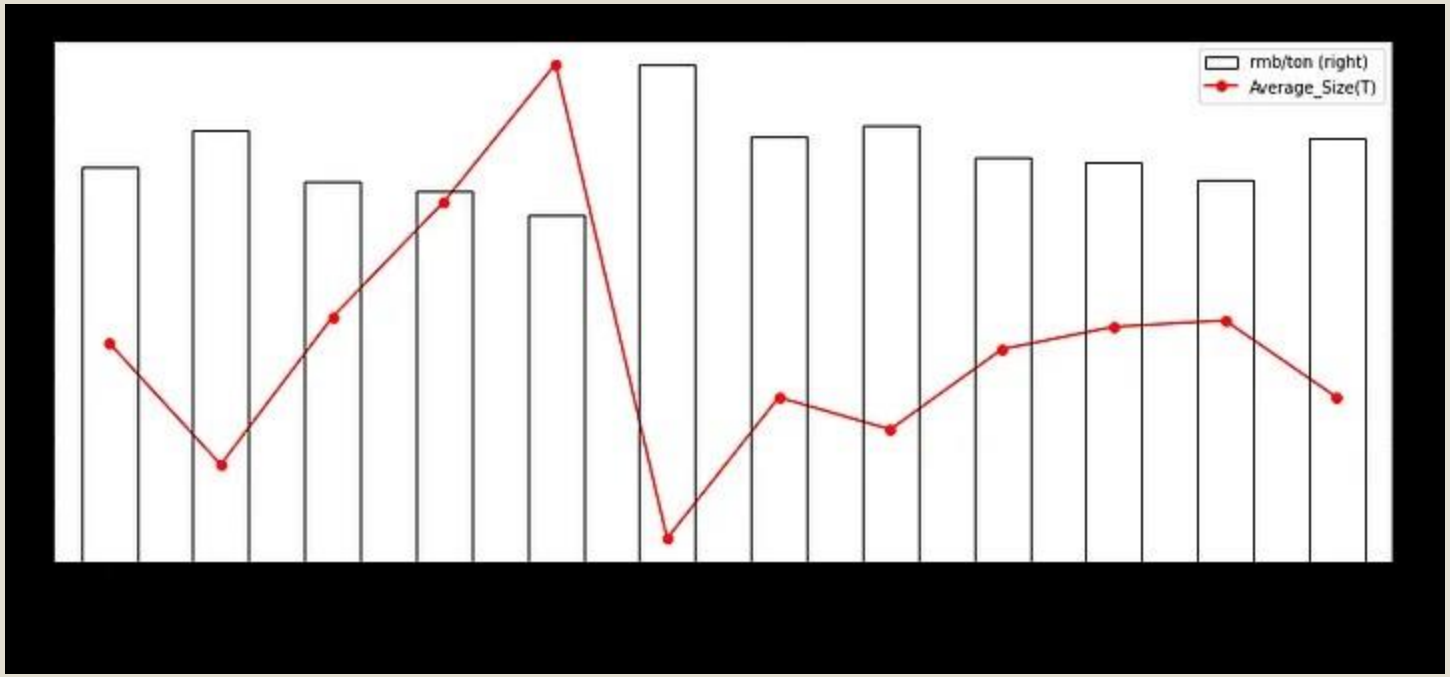
```
Pricing Functions
def f_max city (list_cities, list_price):
        return list_cities[list_price.index (max (list_price))] # Index of Maximum Price
def inner_stops(list_cities, max_city):
        return list_cities.count (max_city) – 1
def outer_stops(list_cities, max_city):
        return len(list_cities) – (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
```

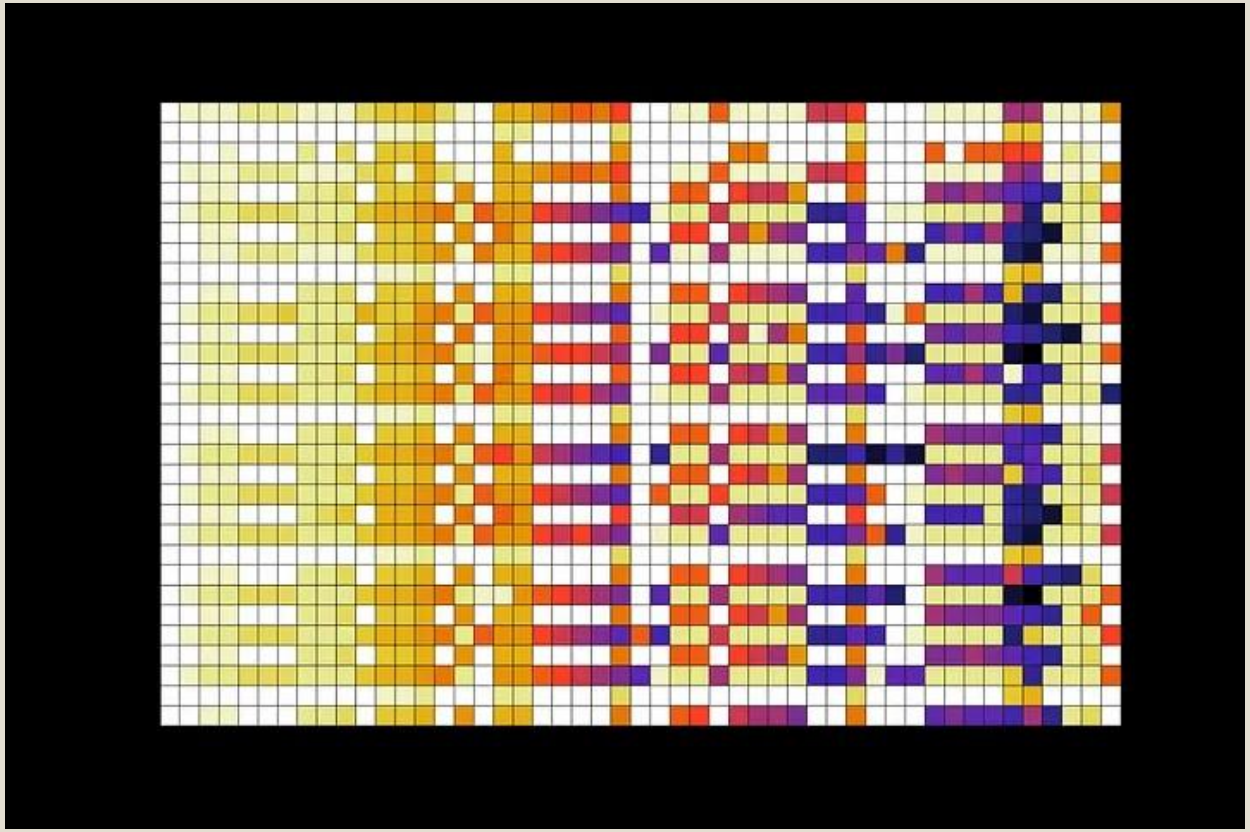
- Mapping First City

```
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
```


%_3.5T
%_5T
%_8T


rmb/ton (right)
Average_Size(T)

Transportation Plan Visualisation



UNDER THE GUIDANCE OF

ARUNA C