

# KSRTC BUS SCHEDULING

## Problem

To create a bus scheduling system from the given input by considering several constraints.

## Constraints

1. Hours of work :
    - a) Steering Duty
    - b) Sign on & Sign off time (30 minutes(15+15))
    - c) Short breaks taken in each terminals (Under 15 minutes)
  2. Compulsory 30 minutes break after 5 hours of continuous work.
  3. In each schedule, There shouldn't be more than of 10 Hours of Work and 12 Hours of Spread over.
  4. Working hours of more than 8 is considered as Overtime and will be charged double of the payment.
- 
- Minimize Overtime, Spill over,
  - Hours of work maximum 10 hour, compulsory 8 hours of steering duty.
  - Maximum 12 hours spill over.
  - Max 5 hours duty in single stretch(30 minutes compulsory break with payment)
  - Ending time - Starting time + 30 minutes(break) = steering duty.
  - Breaks which goes beyond 30 minutes is not payed(eg. If 35 minutes , no payment for that 5 minutes).
  - In rare cases we can take a split of upto 15 minutes(To adjust the duty). But this split should occur only at most 1 time in a schedule(trip).

## TASKS PERFORMED

### 1. Importing xlsx file into a dataframe

```
# set up the data set
import pandas as pd
data = pd.read_excel(r"input.xlsx")
df = pd.DataFrame(data)
#Renaming coloumns
df.rename(columns={'Unnamed: 0': 'Sl No.',
                  'Unnamed: 1': 'Departure Time',
                  'Unnamed: 2': 'Departure Place',
                  'Unnamed: 3': 'Route of Operation',
                  'Unnamed: 4': 'Arrival Place',
                  'Unnamed: 5': 'Arrival Time act',
                  'Unnamed: 6': 'KM',
                  'Unnamed: 7': 'Running Time'}, inplace=True)
df.drop(index=0, inplace=True)
df.reset_index(drop=True, inplace=True)
#Dropping coloumns 'Duplicates' & 'Remarks'
df = df.drop(columns=['Unnamed: 8', 'Unnamed: 9'])
display(df)
```

✓ 7.1s

	SI No.	Departure Time	Departure Place	Route of Operation	Arrival Place	Arrival Time act	KM	Running Time
0	1	03:40:00	PSL	NH	KLKV	03:45:00	3.5	00:05:00
1	2	07:55:00	PSL	NH	KLKV	08:05:00	3.5	00:10:00
2	3	06:40:00	PSL	NH	KLKV	06:50:00	3.5	00:10:00
3	4	04:30:00	PSL	NH	KLKV	04:40:00	3.5	00:10:00
4	5	12:45:00	PSL	NH	KLKV	12:55:00	3.5	00:10:00
...	...	...	...	...	...	...	...	...
722	723	20:40:00	TVM	NH-KLKV	PSL	22:10:00	37.2	01:30:00
723	724	08:15:00	PCD	KTNI-KDPM-TVM-NTA	KLKV	10:35:00	53	02:20:00
724	725	10:45:00	KLKV	KRKM	VLRD	11:30:00	17	00:45:00
725	726	11:40:00	VLRD	KRKM-KLKV	PSL	12:30:00	20.5	00:50:00

Imported the xlsx file and made some changes such as renaming the columns and resetting the index, as it started from 1 not 0. Deleted the extra columns 'Duplicates' and 'Remarks'.

## 2. Sorting the data set with respect to the departure place and time.

```
#Sorting the routes starting from PSL with respect to their departure time in ascending order(creating new sorted set)
sorted_set = df.sort_values('Departure Time', ascending=True)
start_set = sorted_set[sorted_set['Departure Place'] == 'PSL']
display(start_set)
```

✓ 0.0s

	SI No.	Departure Time	Departure Place	Route of Operation	Arrival Place	Arrival Time act	KM	Running Time
0	1	03:40:00	PSL	NH	KLKV	03:45:00	3.5	00:05:00
14	15	04:20:00	PSL	NH	KLKV	04:25:00	3.5	00:05:00
3	4	04:30:00	PSL	NH	KLKV	04:40:00	3.5	00:10:00
692	693	05:00:00	PSL	MKD-KLD-PLKDA-NTA-TVM-MC-SKRM-CHPY	PCD	07:45:00	66.5	02:45:00
644	645	05:00:00	PSL	KRK-MJ	TVM	06:35:00	39	01:35:00
...	...	...	...	...	...	...	...	...
61	62	14:35:00	PSL	NH	KLKV	14:45:00	3.5	00:10:00
62	63	14:40:00	PSL	NH	KLKV	14:50:00	3.5	00:10:00
54	55	15:05:00	PSL	NH	KLKV	15:15:00	3.5	00:10:00
64	65	15:20:00	PSL	NH	KLKV	15:30:00	3.5	00:10:00
530	531	15:40:00	PSL	KRK-M	VLRD	16:20:00	17	00:40:00

104 rows × 8 columns

Created a new dataset having only departure place as PSL and sorted it according to its time in ascending order.

## 3. Retrieving the first leg from the sorted set

```
'''Creates the first row of output'''
"""Getting the first row from the start_set(created before) and add it to the route_plot.
   This is the first row of the output """
# route_plot = start_set[start_set['SI No.'] == start_set.iloc[0]['SI No.']]
route_plot = start_set.iloc[0].to_frame().T
display(route_plot)
hr_of_work = convertToSec(route_plot.iat[0,7])
print(hr_of_work)
```

✓ 0.1s

	SI No.	Departure Time	Departure Place	Route of Operation	Arrival Place	Arrival Time act	KM	Running Time
0	1	03:40:00	PSL	NH	KLKV	03:45:00	3.5	00:05:00

300

Got the first route of operation from the sorted set by setting the index as 0.

#### 4. Making the first schedule without considering the constraints.

```
"""Using loop to iterate over and find the corresponding leg of the rout """
i = 0
while((last_leg := route_plot.iloc[len(route_plot) - 1])['Arrival Place'] != 'PSL'):
    temp_set = df[(df['Departure Place'] == last_leg['Arrival Place']) & (df['Departure Time'] > last_leg['Arrival Time act'])]
    temp_set = temp_set.sort_values('Departure Time', ascending=True)
    new_leg = temp_set.iloc[i].to_frame().T
    #calculating hr of work ==> for that convert the time to seconds and add

    # hr_of_work = time(hr_of_work.hour+(new_leg.iat[0,7]).hour, hr_of_work.minute + new_leg.iat[0,7].minute, hr_of_work.second+ne
    route_plot = pd.concat([route_plot, new_leg], ignore_index=True)
    i = 0
display(route_plot)
#print(hr_of_work)
✓ 0.0s
```

	SI No.	Departure Time	Departure Place	Route of Operation	Arrival Place	Arrival Time act	KM	Running Time
0	1	03:40:00	PSL	NH	KLKV	03:45:00	3.5	00:05:00
1	124	03:50:00	KLKV	NH	TVM	04:55:00	33.7	01:05:00
2	313	05:05:00	TVM	NH-UDA	KNVLA	06:05:00	32	01:00:00
3	641	06:15:00	KNVLA	UDA	MC	08:00:00	39	01:45:00
4	336	08:10:00	MC	NH	KLKV	10:00:00	40	01:50:00
5	695	10:05:00	KLKV	PVR-VZM-BYPASS-TVM-NH-KLKV	PSL	13:35:00	81.5	03:30:00

Made the first schedule from the datasheet without considering any constraints. Set departure place and last arrival place as PSL. Using a while loop, found the next legs whose departure place is same as of last leg's arrival place and departure time is greater than last leg's arrival time.