Uber Data Analysis
(Data Science Mini Project)

* Download dataset then open jupyter notebook (using cmd → python -m notebook) and upload the dataset.
Click new → click Python 3(ipykernel)
* Here, we have to clean, process & visualize data, for that we have to first import the libraries.

→ import pandas as pd

1) For data handling & to work with tables (rows & cols) like Excel.
2) Supports csv, excel, sql, etc
3) Data cleaning that basically helps to remove missing & incorrect data
4) Allows filtering, sorting & grouping
5) Works with other libraries for graphs & charts

→ import numpy as np

1) Work with array & numerial operations
2) Helps in mathematical & statistical data.

→ matplotlib.pyplot as plt

1) For creating basic charts & graphs
2) Helps in visualizing data.

→ seaborn as sns

1) Built on top of matplotlib
2) Makes charts & graphs attractive
3) Used for statistical visualization.

* To read our csv file →
  dataset = pd.read_csv("UberDataset.csv")
* Then call the variable dataset.

* dataset.shape → It counts rows & cols
  (ex: 1156 , 7)

* dataset.info() → Gives information
  1) Mtlb konsa col^n kis type ka hai
  2) ex: date ⇔ object
     purpose object
     miles float ...

* Change Start date & end date from object to datetime format.
This comes under preprocessing the data.

Note: To create heading write →
Data Preprocessing → select markdown

* In purpose col^n we want to replace nan with NOT.

→ dataset['Purpose'].fillna("Not", inplace = True)

→ dataset.head : Gives first 5 rows from data.

• Here, inplace = True: ensures that changes are applied directly to dataset without needing to reasign it.

* dataset['START_DATE'] = pd.to_datetime(dataset['START_DATE'], errors = 'coerce')

→ This changes Start_date to datetime from object.

→ errors = 'coerce' ensures that if there is some different values it will be replaced with NaT (Not a Time)

* Why use errors?

⇒ It avoids errors while executions.

⇒ Instead of stopping execution, it assigns NaT for non-convertible values.

* Now add 2 new cols:

```
from datetime import datetime
dataset['Date'] = pd.DatetimeIndex(dataset
        ['Start_date']).date

dataset['Time'] = pd.DatetimeIndex(dataset
        ['Start_date']).hour
```

OR

* Instead of pd.DatetimeIndex() we can directly use .dt

ex: dataset['Date'] = dataset['Startdate'].dt.date

* Divide into 4 categories :

(Morning , Afternoon , Evening , Night)

Here, 0 to 10am 10 to 3pm 3 to 7pm 7 to 12 am.

* First create new col → Day Night

dataset['Day Night'] = pd.cut(x = dataset['Time'],
bins = [~~Nan~~ 0, 10, 15 , 19 , 24],
labels = ['Morning', 'Afternoon', 'Eve.' , 'Night'])

→ pd.cut : categorizes continuos numerical data into discrete bins (ranges) by splitting values into different groups.

→ bins: groups the data

ex: 0 to 10 → Morning, 10 to 15 → Aft.... & so on.

→ labels: assigns name to each bin (group)

ex : 0 to 10 → Morning & so on...

Group (bin) (label)

Q.1 In which category most Uber rides are booked? Also purpose?.

* Here, make graph to visualize

```
plt.figure(figsize=(20,5))
plt.subplot(1,2,1)
sns.countplot(dataset['Category'])
plt.subplot(1,2,2)
sns.countplot(dataset['Purpose'])
```

→ plt.figure: Matplotlib module to create visualization

→ ~~figure~~ figsize=(20,5) : Sets size in inches
(w, h)

→ subplot() : Creates subplot (Small chart inside large figure)

→ (1,2,1) : Grid layout.
1 → No. of rows
2 → No. of cols
1 → First chart in grid (Position)

→ countplot(): Creats barchart to show count (of category col.)

Q.2 : At what time people book cabs the most ?

```
sns.countplot(dataset['Day_Night'])
```

Q.3 Most booked in which month & week?

→ First make 2 new cols (Month & Week) from date col

```
dataset['Month'] = pd.datetimeIndex(['Start_date']).month
month_label = {1.0: 'Jan', 2.0: 'Feb' ....
               12.0: 'Dec'}
dataset['Month'] = dataset.Month.map(month_lable)
mon = dataset.Month.value_counts(sort_false)
```

→ Extracted month (1 to 12) from Start-date

→ Made dictionary

→ map(month) : replaced numbers with names

ex: 1.0 → Jan & so on....

__/__/____

→ .value_counts() : Counts the appearence of every unique month.

→ sort = false : prevents sorting & keeps the order same.

* df = pd.DataFrame({
"Months" : mon.value,
"Value Count" : dataset.groupby('Month', sort=False)['Miles'].max()
)}

p = sns.lineplot(data = df)
p.set(xlabel = "Months", ylabel = "Value Counts")

→ Here, created dataframe :
- "Months" → has mon.values
  ↓
  contains count of month from 'Month'

- "Value Counts" → groups data by month finds max miles

→ lineplot → creates line graph

→ xlabel is label for Month on X-axis,
ylabel —"— Value Counts on y-axis.

* Now create Weekday col.

```
dataset['Weekday'] = dataset.Startdate.dt.
                                    weekday.
day_label = {0:'Mon', 1:'Tue'... }.
dataset['Weekday'] = dataset['Weekday'].
                        map(day_label).
```

```
day_label = dataset.Weekday.value_counts()
sns.barplot(x= day_label.index,
            y= day_label)
plt.xlabel('Weekday')
plt.ylabel('Count')
```

Q.4 Calculation for miles (betn 0 to 50).

→ `sns.boxplot(dataset['Miles'])`

```
sns.boxplot(dataset[dataset['Miles']<50]
                    ['Miles'])
```

`sns.distplot(—"—)` #Shows density graph.