ACCESSING VALUES	
Read excel file	pd.read_excel(“file name.xlsx”)
If we need to access sheet other than first	df1=pd.read_excel('A.xlsx',sheet_name='Sheet2')
If CSV fie is not opening	pd.read_csv('superstore.csv',encoding='latin=1')
Extract to excel file	ndf.to_excel('mohit.xlsx', index=False)
Extract to excel without using index	ndf.to_excel(‘Asc_sales.xlsx’,index=False) Index is used if we don’t need data frame index when converting in excel
How many columns and rows are in pandas dataframe?	df.shape #(996, 13)
Data type in each column	df.info() and df.dtypes they give same set of information
Show all the columns headings	df.columns
To access single column	df[‘column Name ‘]
To access multiple column	df[['Customer ID','Customer Name']] Outer brackets to call columns and centre ones are used to call out multiple column provided in list
To check starting 5 rows	print(df.head()) print(df.head(6)). print(df.head(6))
To check last 5 rows	print(df.tail()). print(df.tail(2))
Iloc	df.iloc[0:3,0:3] To find rows and columns separated by comma
loc	df.loc[df['Ship Mode']=='Same Day']
Delete any column	ndf=ndf.drop(columns=['lenght','Email check','copy adhaar','ad_len']) Pass brackets to delete whether its one or multiple in list
Delete any row	df.drop(index=1). Index=1 specify the row # to delete df.drop(index=[1,3,5,7]). To delete multiple indexes
To add completely new column and if column exist it will replace with India every where	df['country']='India'
To create new columns using existing column	df['new discount']=df['Discount']/2

To create new columns using existing columns	df['Net profit']=df['Quantity']*df['Discount']
To return values in single column within specific range	df['New COlumn']=np.arange(1,10) Let's suppose I have added 4 people under Name column then in order to number them I can use df['S.no']=np.arange(1,5)
This will return true where ever city is Noida and else false	df['City']=='Noida'
Renaming columns	df.rename(columns={ 'AREA NAME':'Area name' })
To find unique values in any column of data	df['Courier Status'].unique()
FILTERING	
Loc will give location and condition will do fltering	df.loc[df['City']=='Berlin']
Filtering Multiple columns using and	df.loc[(df['Ship Mode']=='Standard Class') & (df['City']=='Los Angeles')]
Filtering Multiple columns using or	df.loc[(df['City']=='Los Angeles') | (df['City']=='Miami')]
Is in membership used for multiple filters done using list	df.loc[df['City'].isin(['Los Angeles', 'Miami','Costa Mesa'])] Or. l=['Hollywood','Foothill','Topanga','Olympic']. # List df.loc[df['AREA NAME'].isin(l)]
Update in second column if condition is True in first column	df.loc[df['Segment']=='Corporate','Premium']='Member'
FUNCTIONS	

Aggregate functions combine values and then do mathematical functions	
	df['Sales'].sum()
	df['Sales'].max()
	df['Sales'].min()
	df['Sales'].mean()
	df['Order ID'].count()
To find maximum sales at which position	m=df['Sales'].max() df.loc[df['Sales']==m]
To find minimum sales at which position	l=df['Sales'].min() df.loc[df['Sales']==l]
Rounding any floating values in any columns	 df['New lives covered']=df['New lives covered'].round()

Functions in pandas	def f(v):
	    if v>0:
	        return 'profit'
	    else:
	        return 'loss'
	
	df['P/L']=df['Profit'].apply(f)
	df

GROUP BY	
How to use	ndf=df.groupby('Category').agg(revenue=('Profit','sum'))
Value counts is used when we have to count same column for which we have to go group by and apply aggregate functions	df.value_counts('Status')

Group by with multiple conditions or using aggregate function	df.groupby('Category').agg(reveue=('Sales','sum'),total_qty=('Quantity','sum'))

Group by with Category and subcategory and using multiple conditions or using aggregate function	df.groupby(['ship-state','ship-city']).agg(
	
	Ship_status=('Status','count'),
	Amount_spent=('Amount','sum')
	)

Sorting by default sort by lowest to highest	df.sort_values(‘total_profit’)
Descending (Highest to lowest)	df.sort_values('total_profit',ascending=False)
When we have to sort based on two columns	df.sort_values(['City','Sales'],ascending=(True,False)) City in ascending order and sales in Descending
STRING METHODS	
Lower	df['product']=df['product'].str.lower()
Title	df['ship-state'] = df['ship-state'].str.title()
Replace	df['product']=df['product'].str.replace('apple i phone','i phone') df['ship-country']=df['ship-country'].str.replace('DIADIA','INDIA')
startswith	df.loc[df['phone number'].str.startswith('93')]

typecasting	df['Phone Number']=df['Phone Number'].astype(str)
Length	df['lenght']=df['phone copy'].str.len() ndf=df.loc[df['lenght']<10] ndf
Combining Strings	df['first']=df['first'].str.title() df['last']=df['last'].str.title() df['Full_name']=df['first']+' ' +df['last']
If we need to find any words any where	df.loc[df['Review'].str.contains('offer')]
To check presence of @ **to check what location is @**	ndf['Email check']=ndf['Email ID'].str.contains('@') ndf ndf.loc[ndf['Email check']==False]
Splitt	df['CI']=df['Customer_ID'].str.split('-',expand=True)[1] df Expand =True will give two column 0 and 1 . O is before splitter and 1 is after
DUPLICATES	
Find duplicated rows on the basis of every columns	df.duplicated() used to identify duplicate rows meaning value in each and every column is duplicate and will give results in True / False form df.loc[df.duplicated()] will give only very first duplicate rows instances. df.loc[df.duplicated(keep=False)] Keep = False will give all the rows of duplicacy
Find duplicate rows on the basis of any columns provided	df.duplicated(subset='Phone no'). checks for duplicate rows based on the values in the 'Phone no' column and will return very first instances
	df.loc[df.duplicated(subset='Phone no')]
Find duplicate rows for multiple columns provided	df.duplicated(subset=['Phone no','Adhar'])
	df.loc[df.duplicated(subset=['Phone no','Adhar'])
	We can use keep = False in all the above scenarios otherwise will give very first duplicate value only instances
Delete duplicate rows where values in every columns is duplicate	df.drop_duplicates() keep=False. Keep=False will delete all the duplicate rows including initial duplicate row
Delete duplicate rows for any columns provided	df.drop_duplicates(subset='Phone no')
Delete duplicate rows for multiple columns provided	df.loc[df.drop_duplicates(subset=['Phone no','Adhar'])

MISSING VALUES- Find	
Missing values in series or column	df['Aadhar Number’] This will return results in True or False form
Sum of Missing values in series	df['Aadhar Number’].isnull().sum()Isnull This will return True value will wherever we have NAN and then sum will count no of True in 1 and 0 form
Sum of missing values in all the columns column of data frame	df.isnull().sum()
To find location for example provide data of all the people who has not provided phone no	df.loc[df['Phone Number'].isnull()]
Multiple conditions for people those who had not told both phone and aadhar	df.loc[(df['Phone Number'].isnull()) & (df['Aadhar Number'].isnull())]
MISSING VALUES- Fill	

We use dictionary method	d={
	    'Name':'Mr',
	    'Email ID':'srimohit@yahoo.com'
	
	}
	
	ndf=df.fillna(d)
	
	ndf
	We pass one dictionary i.e. column value pair

To fill age	d={
	
	    'age':df['age'].mean()
	
	}

MISSING VALUES- Delete	
Delete we use dropna. **Delete any row where any value is missing**	**df.dropna()** **Default value for how is any**
Delete user where all the value are missing in all the column	**df.dropna(how='all')**
Delete user who don't provide phone no	**df.dropna(subset='Phone Number')**
Delete user who don't provide any of the phone no and Aadhaar no	**df.dropna(subset=['Phone Number','Name'])**
Delete user who don't provide phone no and Aadhaar no iboth	**df.dropna(subset=['Phone Number','Name'],how='all')**
Delete user with less than 3 information out of 5 (Thresh)	**df.dropna(thresh=3)**
Delete user who has less than two documents in Aadhar , name and phone meaning any person with one document will be deleted	**df.dropna(subset=['Aadhar Number','Name','Phone Number'],thresh=2)**
DATE AND TIME	
Convert date to Integer type from string since by default its type is string	**df['Date Rptd']=pd.to_datetime(df['Date Rptd'])** **By Default its converted in YEAR- MONTH-DATE format**
Max and min	**df['Date Rptd'].max()** **df['DATE OCC'].min()**

What all crime happenes on latest day Location	d=df['DATE OCC'].max() df.loc[df['DATE OCC']==d]
Oldest to newest date wise sorting	df=df.sort_values('DATE OCC') df
To pull date of specific date	d=‘2024-04-01’. Or alternatively we can use just d=‘2024 df.loc[df['DATE OCC']==d]
To pull date of specific date range	 ndf=df.loc[(df['DATE OCC']>='2024-01-01') & (df['DATE OCC']<='2024-12-31')] ndf
YEAR	ndf['Year']=ndf['DATE OCC'].dt.year
Month	ndf['Month']=ndf['DATE OCC'].dt.month
Day	ndf['Day']=ndf['DATE OCC'].dt.day ndf['Day']=ndf['DATE OCC’].dt.days
month_name	ndf['month_name']=ndf['DATE OCC'].dt.month_name() ndf
day_name	ndf['month_name']=ndf['DATE OCC'].dt.day_name() ndf df['Quarter']=df['Order Date'].dt.quarter

to find tp 10 crimes in 2023 when you have pulled year only	d=ndf.loc[ndf[‘Year’]==2023]. This will give data for complete 2023
	
	d=d.groupby('Crm Cd Desc').agg(top_10_crime=('Crm Cd Desc','count'))
	d.sort_values('top_10_crime',ascending=False)

Differentiate between weekday and weekend	df['Week']='Weekday' df.loc[(df['day']=='Saturday') | (df['day']=='Sunday'),'Week']='Weekend' df df.loc[df[‘day’] isin list, ‘Week’]=‘Weekend’

JOINING	
Inner will give common values from both data sets	ldf.merge(rdf,on='id',how='inner')
Outer will give common values from both data sets plus non common values	ldf.merge(rdf,on='id',how='outer')
Left will give common values from both data sets plus ll the values from left data set	ldf.merge(rdf,on='id',how='left')
Right will give common values from both data sets plus ll the values from right data set	ldf.merge(rdf,on='id',how='right')
Top to bottom concatenate	new_data=pd.concat([tdf,bdf]) new_data

For resetting index	new_data=pd.concat([tdf,bdf])
	new_data=new_data.reset_index()
	new_data

If we want to delete old index	new_data=pd.concat([tdf,bdf])
	new_data=new_data.reset_index(drop=True)
	new_data