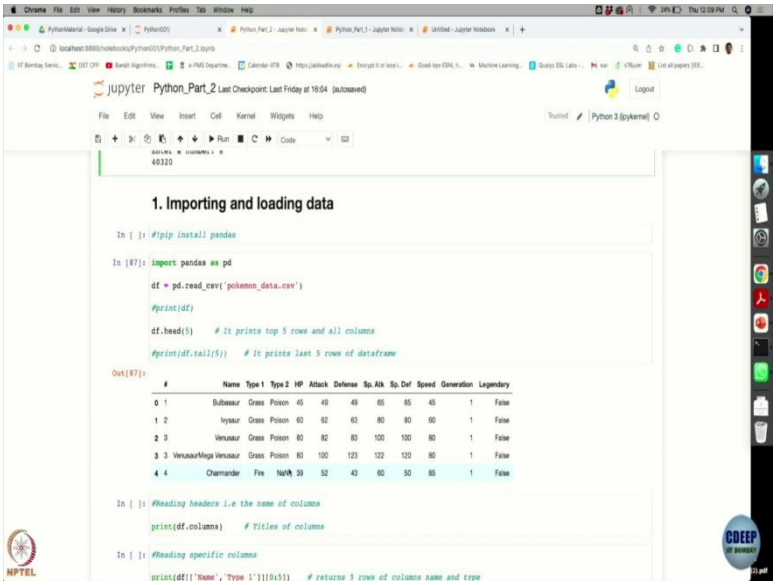
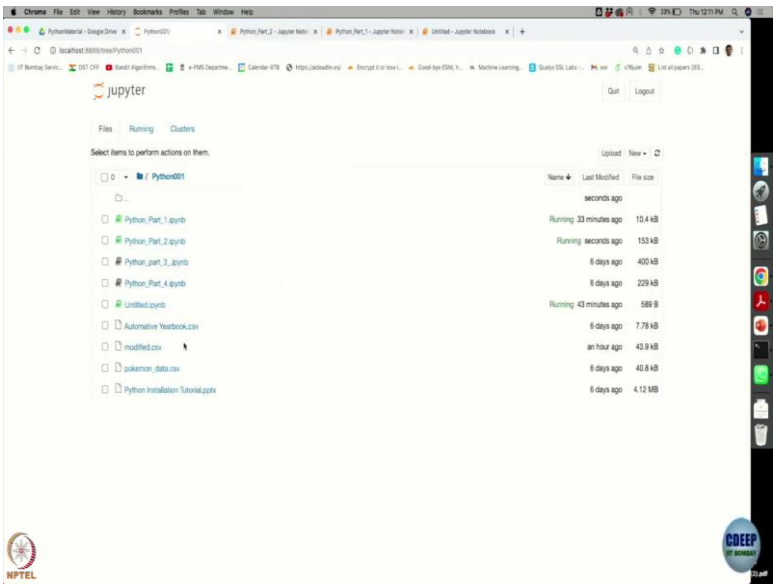
Engineering Statistics
Professor Manjesh Hanawal
Industrial Engineering and Operational Research
Indian Institute of Technology, Bombay
Lecture 50
Python- numpy and pandas function II

(Refer Slide Time: 00:15)





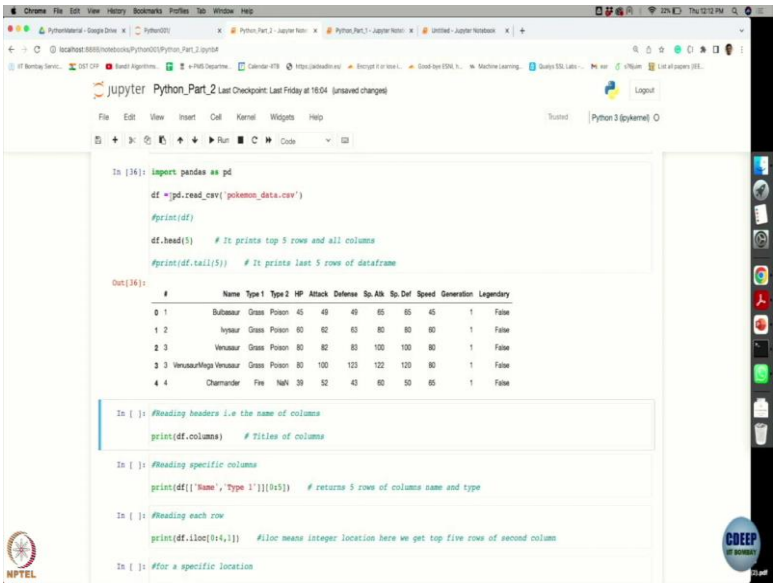
Now, these are the basic operations that we can do and there are many more, we are not going to go into all you can explore, but you just see that how to do define variables and how to create list arrays, and how to do addition, subtractions like basic operations, not only on the numbers, but also on the strings. And then you saw that how to write conditions like if else and how to do looping… looping's like for loop and a while.

Now, we are often we have to deal with the data. And that data may be coming in some files, which we need to read and do operations on that. To reading the files, Python has this very handy library called pandas. And when we want, so if you want to use it, again, we need to first load it. And also, like it is not necessary that all the Python, if you have just installed the basic version of the Python it has already pandas in built it, but maybe you want to get that library, in your installation.

And by the way, I think if you install Panda, and all sorry, Anaconda and all maybe that will already have these packages readily installed. So, let us first get this pandas library, imported as pd. And now I have a file called pokemon underscore data. So, that I have stored here, it is in the same directory where I am running this Jupyter Notebook.

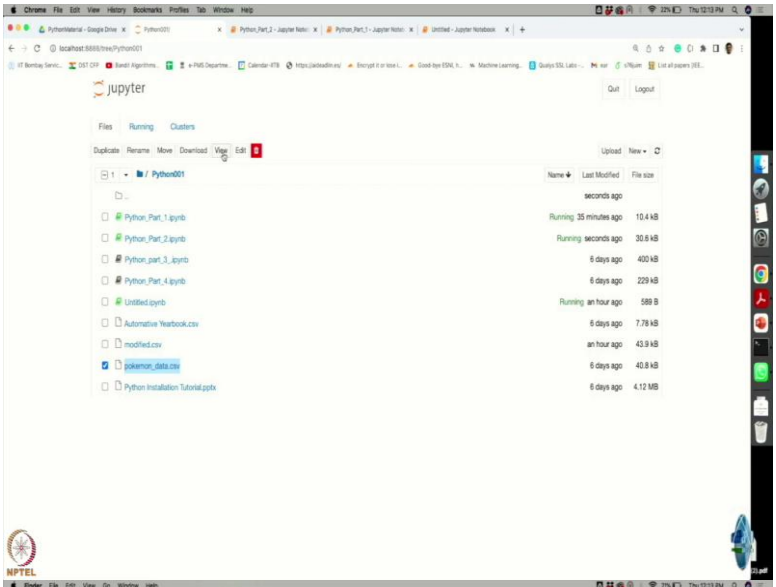
And here we have so many options available to play with our data. First, first thing is we need to read all the data and get into a data frame or like get it into some array. So, Python provides a very handy way of getting this data in terms of data frames. First, I can read all the content of my CSV file and put it into a variable which I am denoting as df here. To read this all I need to do is simply write pd dot read dot csv, read dot csv is the function I need to call to read all the content of my file.

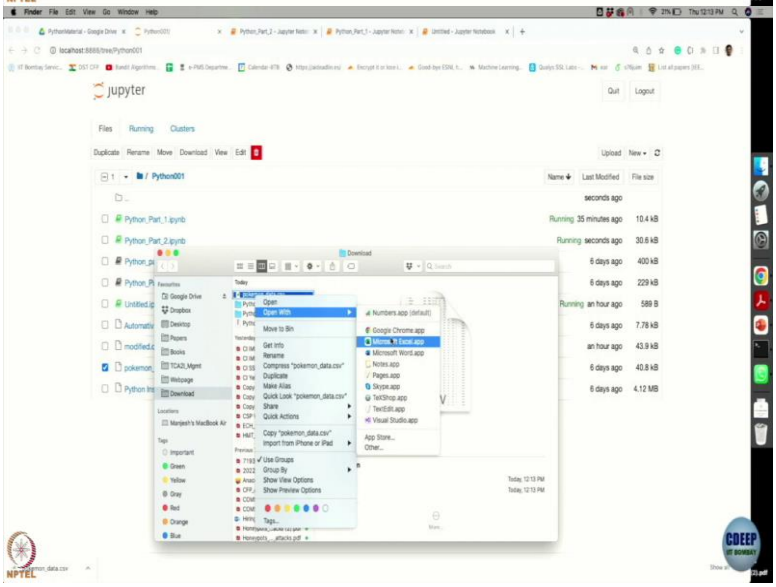
(Refer Slide Time: 02:57)

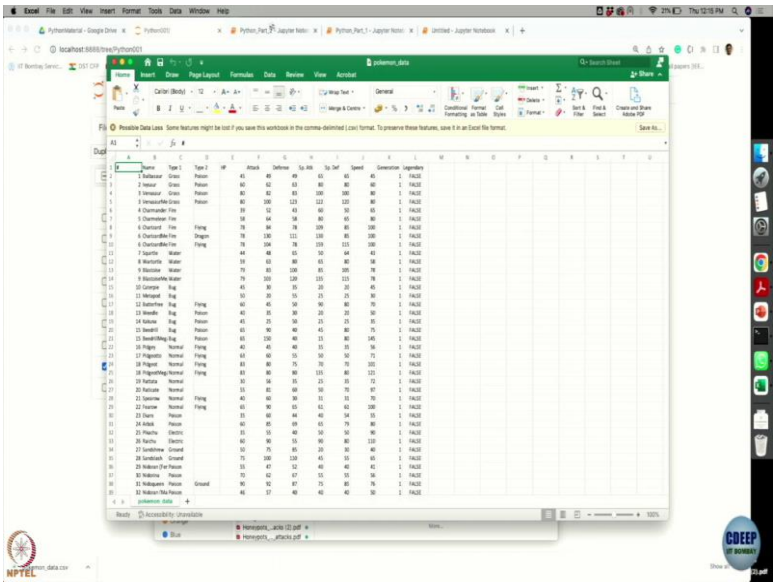


Now, suppose I want to see, maybe let us clear all my cells here… all the things are fine I have to only get started here now I have imported my pandas library and now when I read this all the data is now coming into this variable df here and now in the df suppose now I can do various operations, one of the operations I want to see that, I want to see what all the headers…

(Refer Slide Time: 03:38)





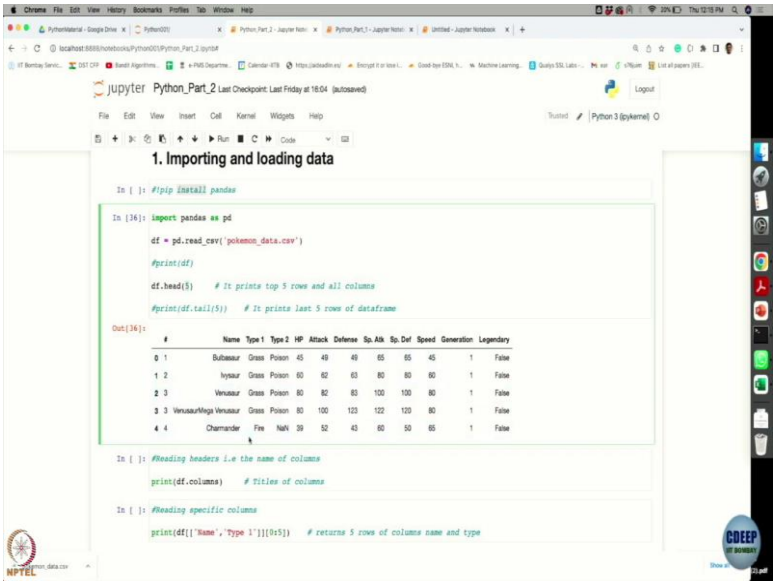


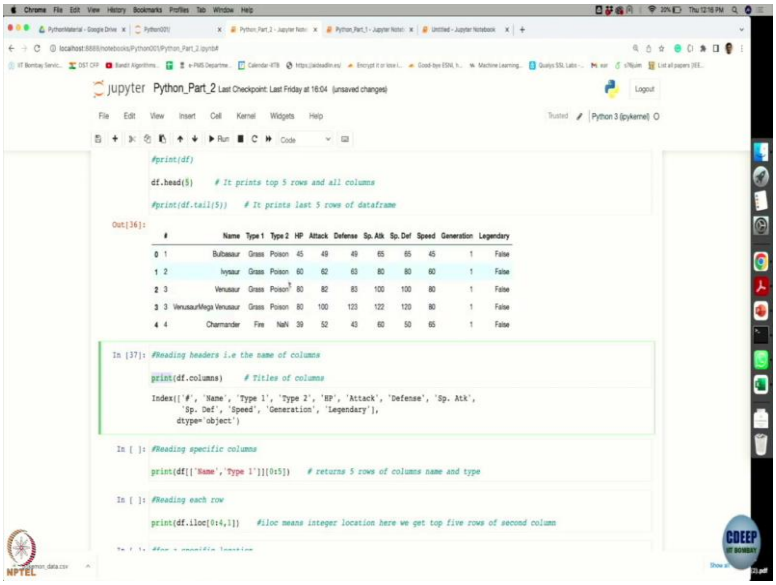
Maybe let us open this file here. This is a file let me see I can open this, let me not open like this let me first download and open it, let me see if I can open it in Excel. Now, you see that whenever I have such a data set, the first row will be usually what type of column it has. And then first row is what are column and usually the first column will be basically index, indexing each of the rows and after that for each row you will have attributes for all of these columns.

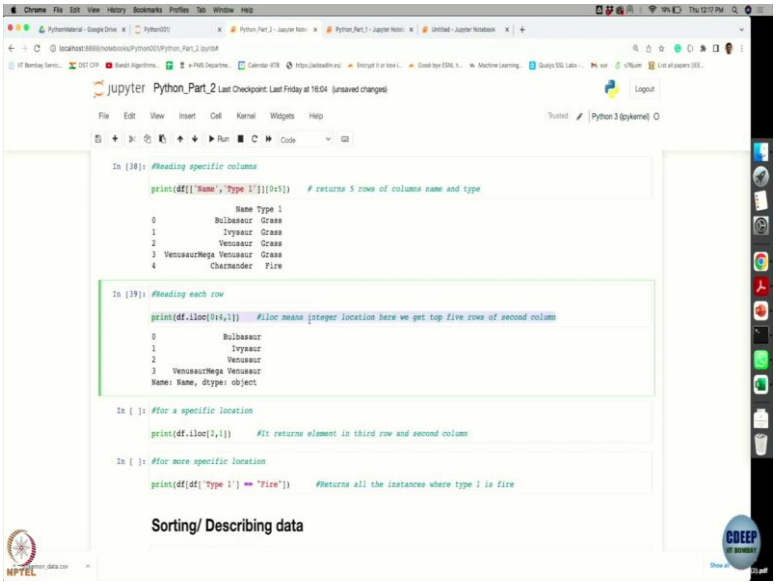
So, when we have such a data loaded, first we want to see that what are the headers in this? What type data is stored in this file, so we want to read this first row and here you see that so this is a pokemon data set I do not know much I have not seen this pokemon, it looks like a famous cartoon there are various very lots of pokemons and they come in different types maybe type 1 and type 2 and under type one also they have different values like grass, fire, water and all.

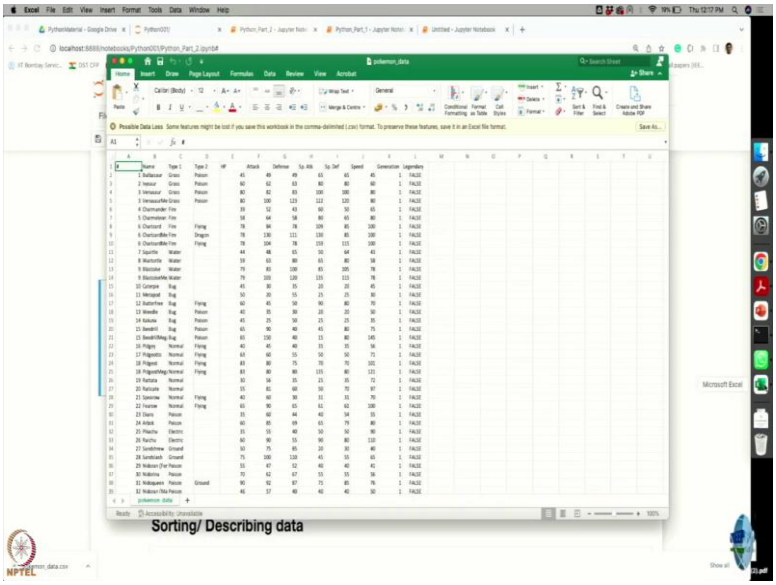
And the type 2 it is like a poison Flying Dragon and all and they have some other characters like what is their HP, attack, defence I do not know what is SP attack maybe Special Attack or special defence whatever, all this information is stored here.

(Refer Slide Time: 06:07)




jupyter Python_Part_2 Last Checkpoint: Last Friday at 16:04 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
#print(df)
df.head(5) # It prints top 5 rows and all columns
#print(df.tail(5)) # It prints last 5 rows of dataframe
Out[36]:
Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed Generation Legendary
0 1 Bulbasaur Grass Poison 45 49 49 65 65 45 1 False
1 2 Ivysaur Grass Poison 60 62 63 80 80 60 1 False
2 3 Venusaur Grass Poison 80 82 83 100 100 80 1 False
3 3 VenusaurMega Venusaur Grass Poison 80 100 123 122 120 80 1 False
4 4 Charmander Fire NaN 39 52 43 60 50 65 1 False
In [37]: #Reading headers i.e the name of columns
print(df.columns) # Titles of columns
Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
'Sp. Def', 'Speed', 'Generation', 'Legendary'],
dtype='object')
In [ ]: #Reading specific columns
print(df[['Name','Type 1']][0:5]) # returns 5 rows of columns name and type
In [ ]: #Reading each row
print(df.iloc[0:4,1]) #iloc means integer location here we get top five rows of second column
CDEEP IIT BOMBAY
NPTEL


jupyter Python_Part_2 Last Checkpoint: Last Friday at 16:04 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel)
In [38]: #Reading specific columns
print(df[['Name','Type 1']][0:5]) # returns 5 rows of columns name and type
Name Type 1
0 Bulbasaur Grass
1 Ivysaur Grass
2 Venusaur Grass
3 VenusaurMega Venusaur Grass
4 Charmander Fire
In [39]: #Reading each row
print(df.iloc[0:4,1]) #iloc means integer location here we get top five rows of second column
0 Bulbasaur
1 Ivysaur
2 Venusaur
3 VenusaurMega Venusaur
Name: Name, dtype: object
In [ ]: #for a specific location
print(df.iloc[2,1]) #It returns element in third row and second column
In [ ]: #for more specific location
print(df[df['Type 1'] == "Fire"]) #Returns all the instances where type 1 is fire
Sorting/ Describing data
CDEEP IIT BOMBAY
NPTEL


Excel File Edit View Insert Format Tools Data Window Help
pokemon_data
Sorting/ Describing data
Microsoft Excel
NPTEL

And now when I want to when I use this function dot head and give 5, what is going to show me? It will show me the all the 5 header and show me the first top 5 rows and the first 5 top rows… this is this is like gives me a first immediate glimpse of what this data is containing, like what kind of attributes its have like maybe I can treat this like attributes type 1, type 2 HP, attack, defence these are like attributes of the data.

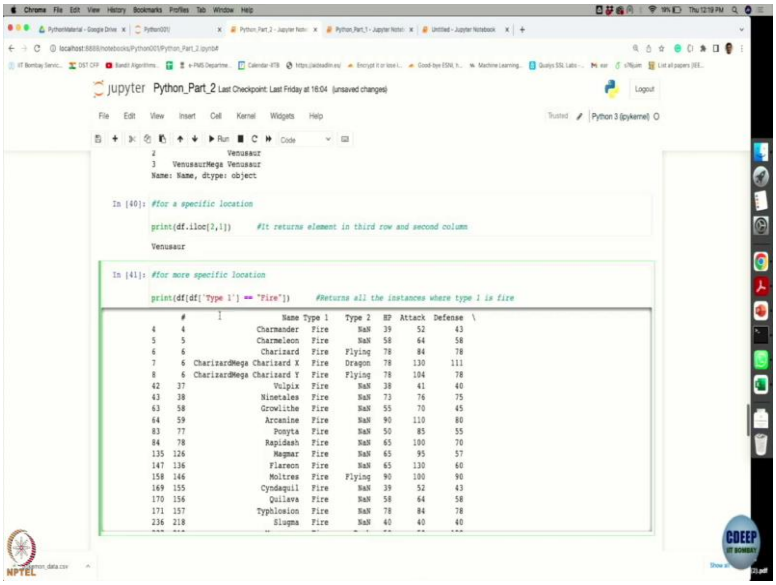
And now in the defence in this df variable I can now see that what are its columns, now it shows it is simply going to say when you did this dot head it kind of nicely showed me in a little formatted way what are the columns and showed me information about these columns by displaying the first few rows.

But when you do this df column, it will simply showing me the first row, so it will giving me some information about what the first row is about. And now I can do a lot of operations, all that data is saved in my variable df now, suppose I am only interested in the two columns which has this name and type 1 as their attributes and I want to see the first 5 rows of these 2 columns, so I had to just use this function, so this is the specifying how many rows I want to see.

And this one is choosing specifying which columns I would be interested in. So, here it is just showing me this column, name column and type 1 column and only showing the first 5 rows and similarly if I want to see the first 4 rows and column, so here is 0 to 4 rows are shown. So, notice that here only integer locations are shown here.

And in our data, he had the first row did not corresponded to an integer it was like something hash here. So, the integer is starting from 1 here I think that is where it started. And it is and it showed the next 4 rows, and it showed me the first column here. Sorry here index is 1 that means which means show me the values from the second column.

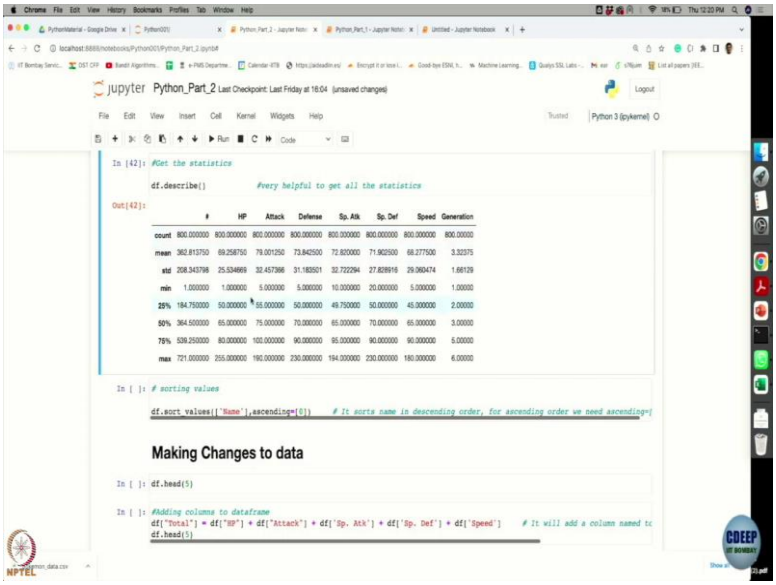
(Refer Slide Time: 09:16)



Similarly, if I am interested in only particular value, let us say I am interested in the index 2 on the row and index 1 on the column, I can just specify that using my iloc function. I look here stands for I think integer location. And that is in my data is simply very sort, now I may be interested in knowing… I am now let us say I am interested in only on one column here, type 1 column and I want to see that in this column where the value this attribute type is taking value 5.

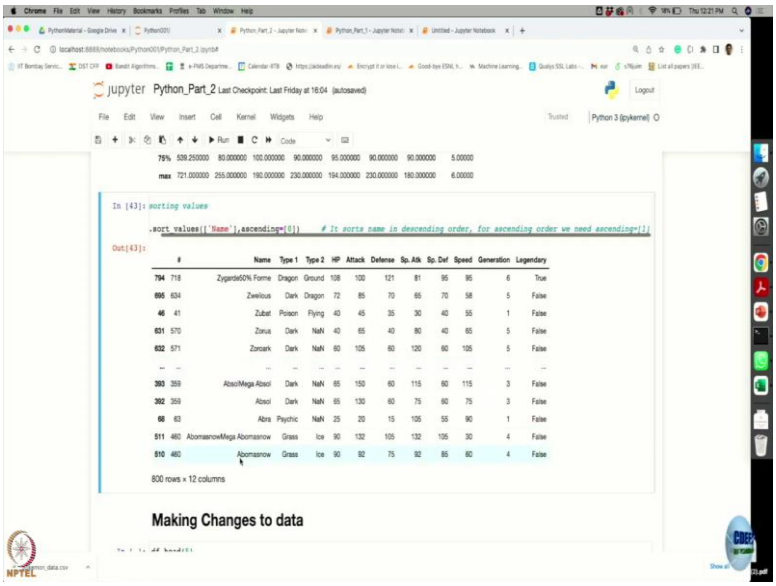
So, then I need to get this first when I do this df type 1 that means it is focusing on my column with the type 1 as the tribute and it is looking in that all there the values matching to be 5 and it will show me only those. Now, if you look into it is only displaying me those rows where type 1 is all 5 and this is happening in rows fourth, fifth sixth, seventh eighth 42, 43 like that.

(Refer Slide Time: 10:49)



Now, you can do sorting or maybe get some summary of your data and there is a very convenient function called a describe. So, here this describe function gives me kind of all these basic statistics like count, mean, standard deviation, minimum value, first quartile, second quartile, third quartile and a max value of each of my attributes here which are numerical. So, not that type 1 and type 2 features are not numerical wherever that features that corresponds to the numerical values for this all these basic statistics is shown.

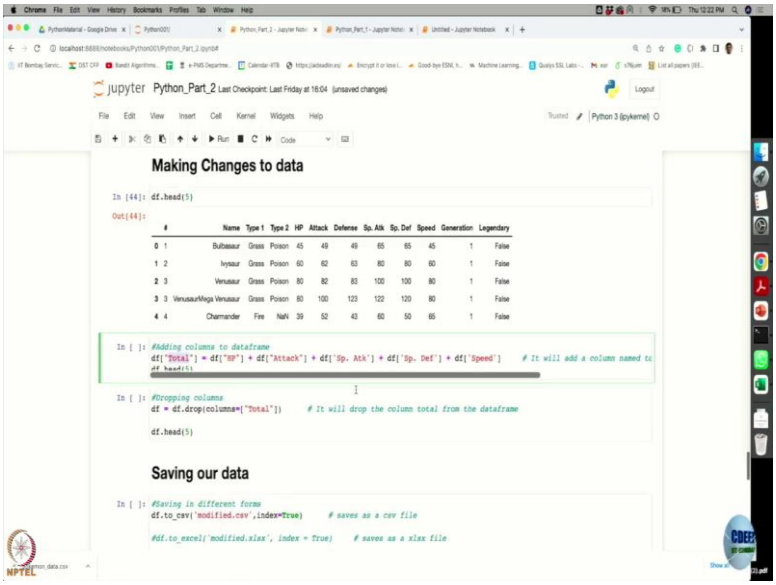
(Refer Slide Time: 11:26)

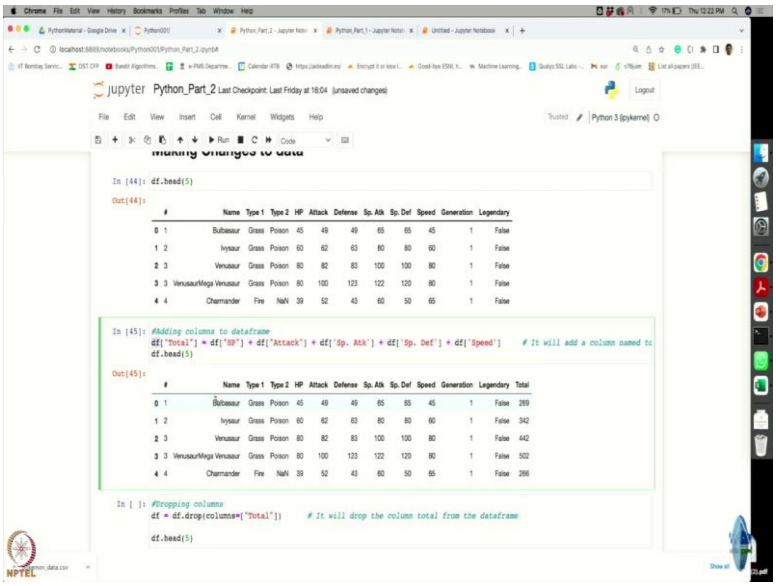


And now, you can also get the values in ascending and descending value by using the sort value function let us see what I get by executing this. So, here I am using the name attribute and I want it to be put in ascending order. So, here 0 means, it is going to going to put in a

descending order 0 means descending order if you want it to be an ascending, it is 1. So, here the variable is ascending, but its value is 0 or 1 based on that it is going to put it in ascending or descending, you see that even though the name has the string value it has put them in a descending, where to starting the first one here has a Z and as you go down you will see that things are ordered starting from Z and all the way to A in dictionary way.

(Refer Slide Time: 12:25)

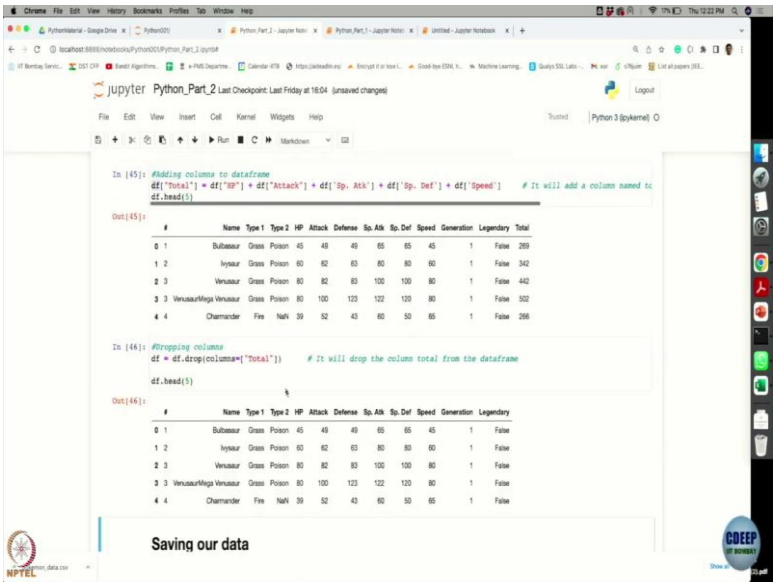


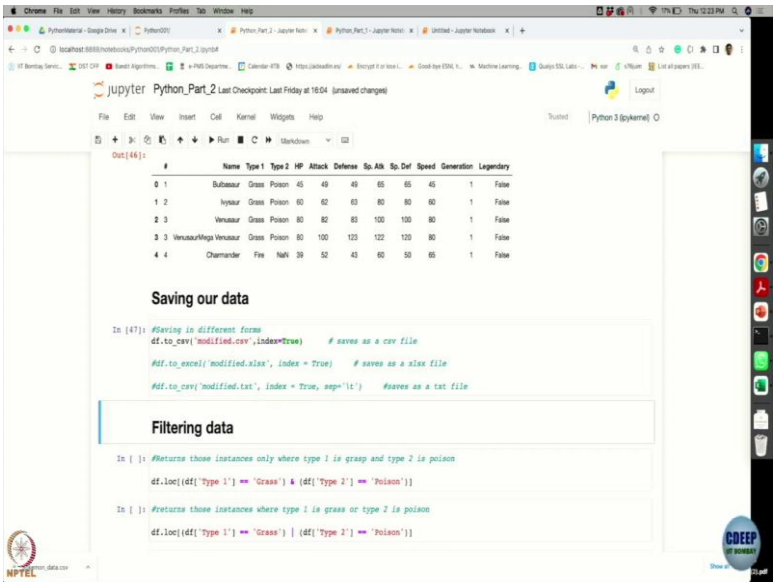


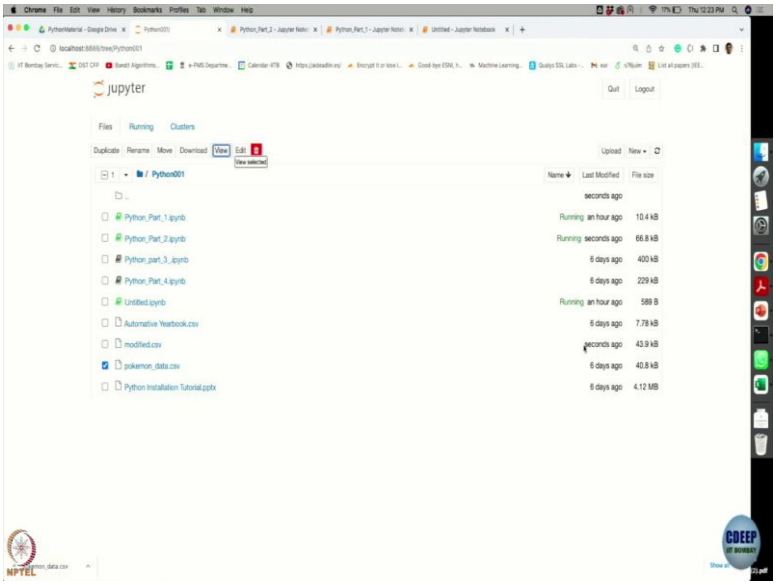
And you can also make changes to our data like we did in our lists. So, for suppose we have these many heads here like attributes here suppose I want to add another one. Let us say I want to add another one total, which is basically the sum of all the values that are happening in this HP column, attack column and defence column and all I can do this, and if I do this

now, and when I see this now head now we will see that that total also, so we have basically earlier this total column was not there, but I added this column and I got this information.

(Refer Slide Time: 13:15)



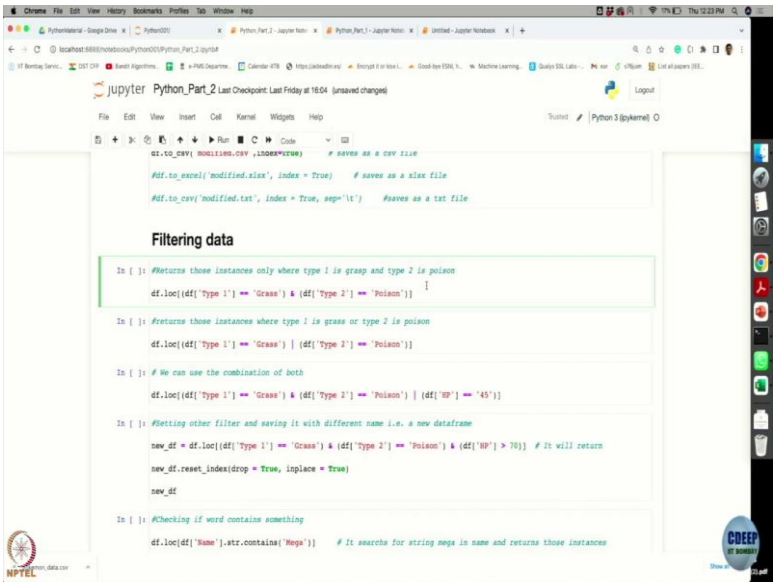


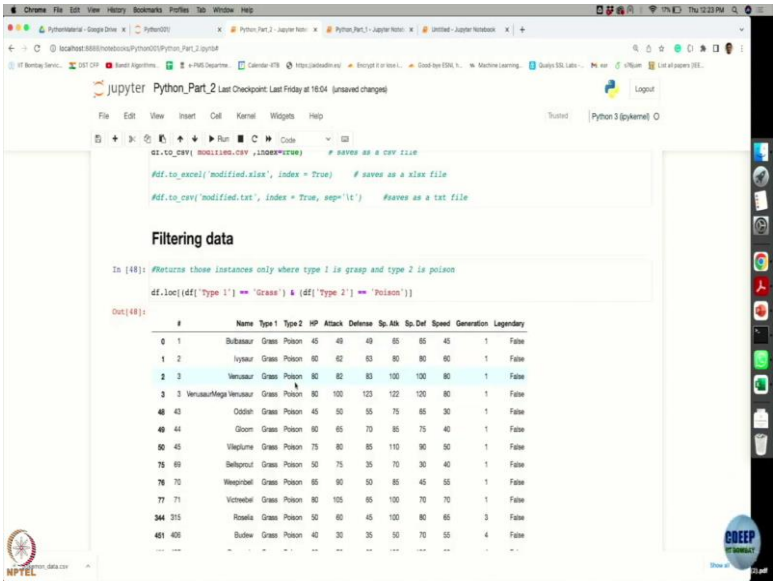


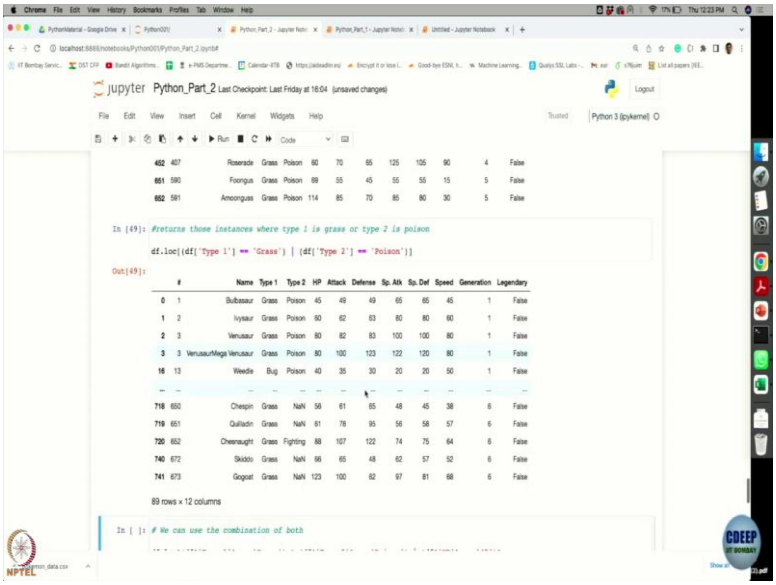
Similarly, you can drop columns like this column I am dropping the new column, I add a total I am simply dropping by using this drop column. And whatever the modification you do to your data now, you can store it as a new file suppose like you have data df, now you have initially loaded it from a file now you have done a lot of modification or some modification on that.

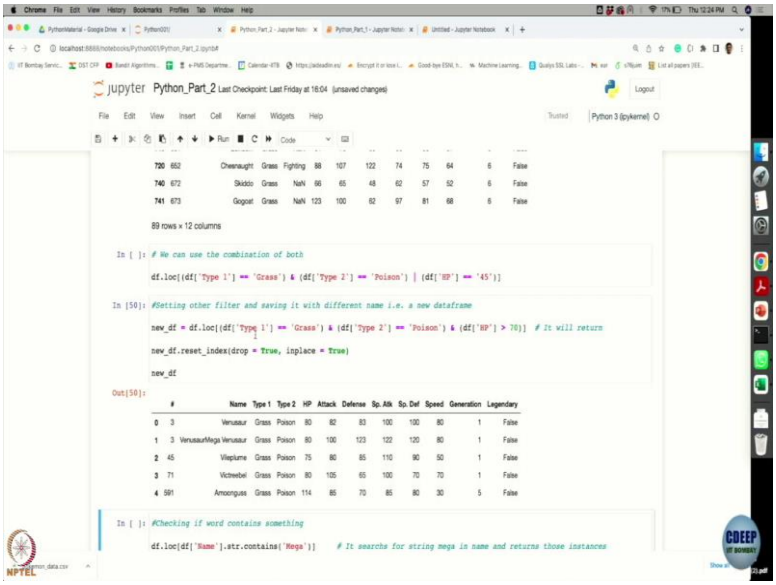
Now, you want to save it as another csv file. So, you can say this modified csv and give index as true then it will show it as a new file, if I do this, so that is what this modified file is what it gets to just created just now, it got created just now.

(Refer Slide Time: 14:05)




jupyter Python_Part_2 Last Checkpoint: Last Friday at 16:04 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
df.to_csv(modified.csv ,index=True) # saves as a csv file
#df.to_excel('modified.xlsx', index = True) # saves as a xlsx file
#df.to_csv('modified.txt', index = True, sep='\t') #saves as a txt file
Filtering data
In [48]: #Returns those instances only where type 1 is grass and type 2 is poison
df.loc[(df['Type 1'] == 'Grass') & (df['Type 2'] == 'Poison')]
Out[48]:
Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed Generation Legendary
0 1 Bulbasaur Grass Poison 45 49 49 65 65 45 1 False
1 2 Ivysaur Grass Poison 60 62 63 80 80 60 1 False
2 3 Venusaur Grass Poison 80 82 83 100 100 80 1 False
3 3 VenusaurMega Venusaur Grass Poison 80 100 123 122 120 80 1 False
48 43 Oddish Grass Poison 45 50 55 75 65 30 1 False
49 44 Gloom Grass Poison 60 65 70 85 75 40 1 False
50 45 Vileplume Grass Poison 75 80 85 110 90 50 1 False
75 69 Bellsprout Grass Poison 50 75 35 70 30 40 1 False
76 70 Weepinbell Grass Poison 65 90 50 85 45 55 1 False
77 71 Victreebel Grass Poison 80 105 65 100 70 70 1 False
344 315 Roselia Grass Poison 50 60 45 100 80 65 3 False
451 406 Budew Grass Poison 40 30 35 50 70 55 4 False

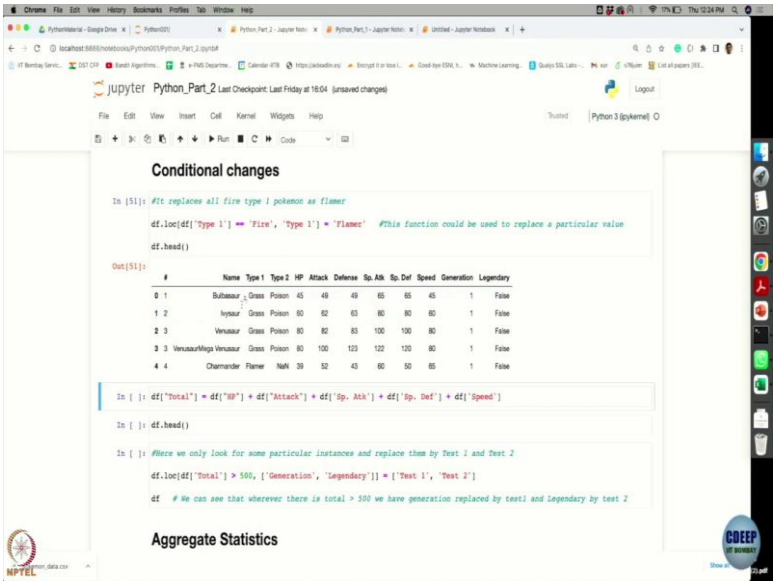

jupyter Python_Part_2 Last Checkpoint: Last Friday at 16:04 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
452 407 Roserade Grass Poison 60 70 65 125 105 90 4 False
651 590 Foongus Grass Poison 69 55 45 55 55 15 5 False
652 591 Amoonguss Grass Poison 114 85 70 85 80 30 5 False
In [49]: #returns those instances where type 1 is grass or type 2 is poison
df.loc[(df['Type 1'] == 'Grass') | (df['Type 2'] == 'Poison')]
Out[49]:
Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed Generation Legendary
0 1 Bulbasaur Grass Poison 45 49 49 65 65 45 1 False
1 2 Ivysaur Grass Poison 60 62 63 80 80 60 1 False
2 3 Venusaur Grass Poison 80 82 83 100 100 80 1 False
3 3 VenusaurMega Venusaur Grass Poison 80 100 123 122 120 80 1 False
16 13 Weedle Bug Poison 40 35 30 20 20 50 1 False
718 650 Chespin Grass NaN 56 61 65 48 45 38 6 False
719 651 Quilladin Grass NaN 61 78 95 56 58 57 6 False
720 652 Chesnaught Grass Fighting 88 107 122 74 75 64 6 False
740 672 Skiddo Grass NaN 66 65 48 62 57 52 6 False
741 673 Gogoat Grass NaN 123 100 62 97 81 68 6 False
89 rows × 12 columns
In [ ]: # We can use the combination of both


jupyter Python_Part_2 Last Checkpoint: Last Friday at 16:04 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
720 652 Chesnaught Grass Fighting 88 107 122 74 75 64 6 False
740 672 Skiddo Grass NaN 66 65 48 62 57 52 6 False
741 673 Gogoat Grass NaN 123 100 62 97 81 68 6 False
89 rows × 12 columns
In [ ]: # We can use the combination of both
df.loc[(df['Type 1'] == 'Grass') & (df['Type 2'] == 'Poison') | (df['HP'] == '45')]
In [50]: #Setting other filter and saving it with different name i.e. a new dataframe
new_df = df.loc[(df['Type 1'] == 'Grass') & (df['Type 2'] == 'Poison') & (df['HP'] > 70)] # It will return
new_df.reset_index(drop = True, inplace = True)
new_df
Out[50]:
Name Type 1 Type 2 HP Attack Defense Sp. Atk Sp. Def Speed Generation Legendary
0 3 Venusaur Grass Poison 80 82 83 100 100 80 1 False
1 3 VenusaurMega Venusaur Grass Poison 80 100 123 122 120 80 1 False
2 45 Vileplume Grass Poison 75 80 85 110 90 50 1 False
3 71 Victreebel Grass Poison 80 105 65 100 70 70 1 False
4 591 Amoonguss Grass Poison 114 85 70 85 80 30 5 False
In [ ]: #Checking if word contains something
df.loc[df['Name'].str.contains('Mega')] # It searchs for string mega in name and returns those instances

So, there are many multiple things you can do like you can filter out your data. For example, in the type 1 you want to only look where the value is grass and only look for that where the type 2 is poison and you can lose them. So, if you look now type 1 is all grass and type 2 is all poison. You can do various permutation and combination of this. You want to list only those columns where the type 1 is grass or type 2 is poison, you can just use this loc function and get them.

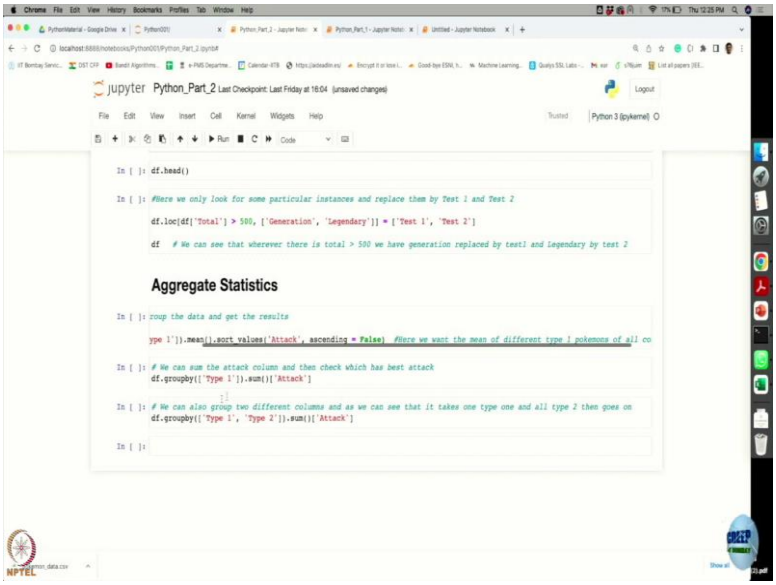
See here if you see that type 1 and type 2. At least one of them is grass or poison, so you can do all the combinations here. And yeah, you can even check these conditions and get like here if what I have done is I will look for type 1 to be grass and type 2 to be poison and when the HP value is greater than 70. So, you will see that in all the things I have there are all HP values are greater than 70.

(Refer Slide Time: 15:09)



I can also do some conditional changes, like if I have, if I want to, let us say type 1, wherever it is value is fire, I want to replace it by let us say flamer I can do that by using this loc function again. So, you will see that wherever that fire was there now it is called replaced by flamer. And you can do the similar things here.

(Refer Slide Time: 15:42)



So, there are other things like you can get some aggregate statistics, I think you can just play around, you just look into this function group by and you will get this. So, we will stop here. And I think with this we have all the basics of how to create variables, manipulate them, not only locally, but also read it from the another files and play around this. So, in the next session, we will see that how we can use some of the statistical tools on data and see how some of the things we have learned in this course can be used, with this we will stop here.