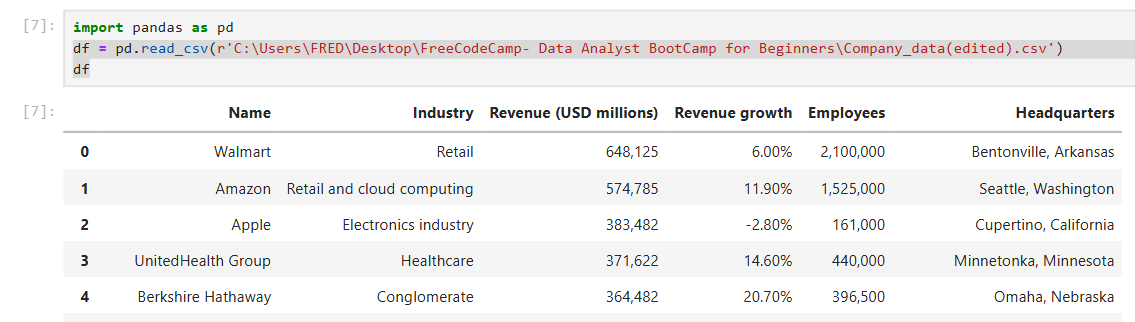


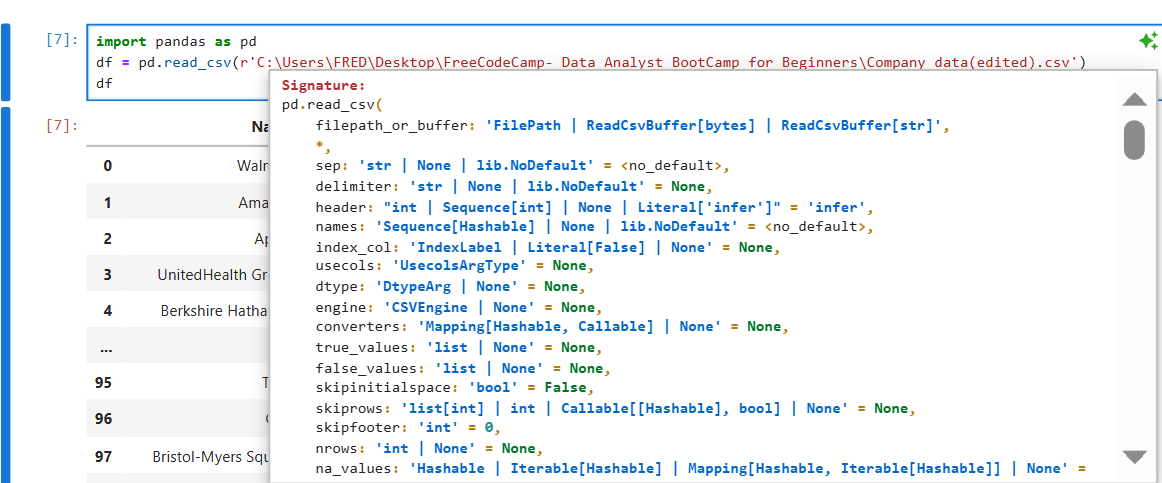
IMPORTING FILES USING PANDAS

1. I’ve edited the previous exported data and removed the ranking numbers.

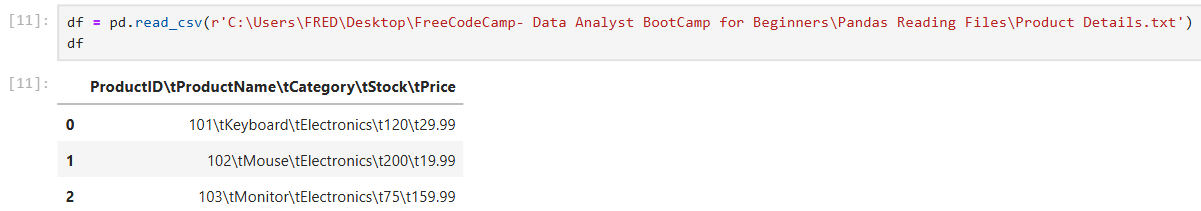
If you import using pandas, it automatically index rows



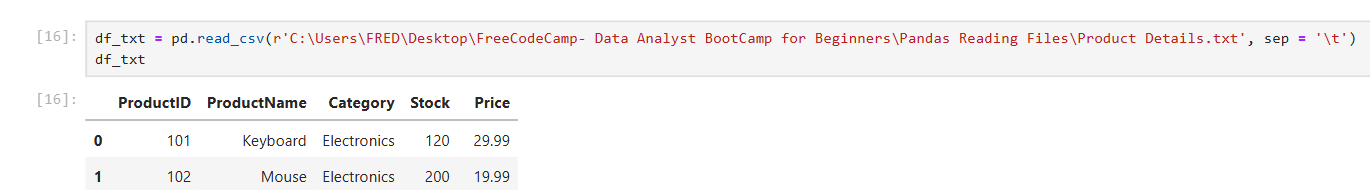
2. Select the code line and press shift + tab and this will display all the available functions that you can use during importing of file



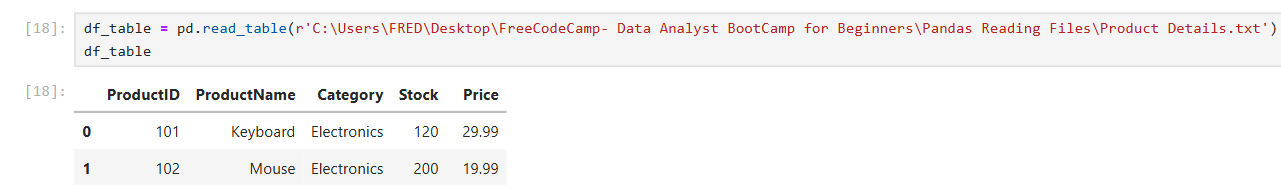
3. you can also use the same function to read into txt file but it will result into like this

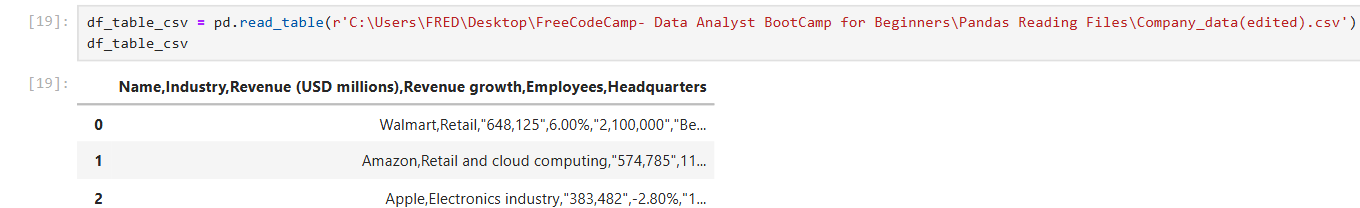


To rectify it use a separator

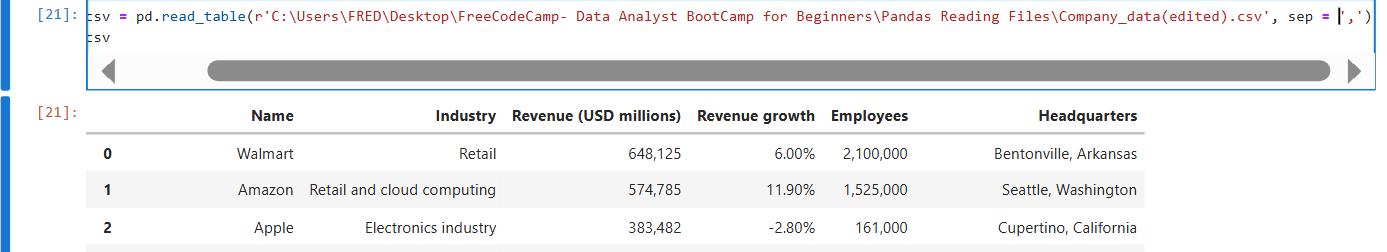


4. The proper way to read into a txt file is to use read\_table method

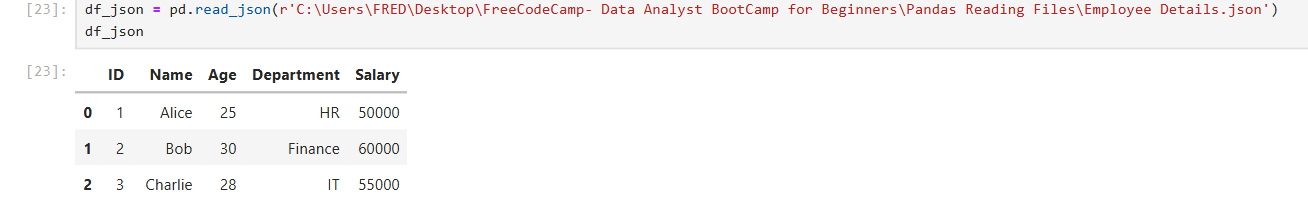


5. If you use read\_table to read csv it will be like this   


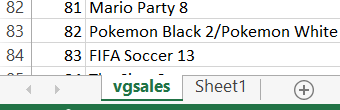
So now you will need a separator

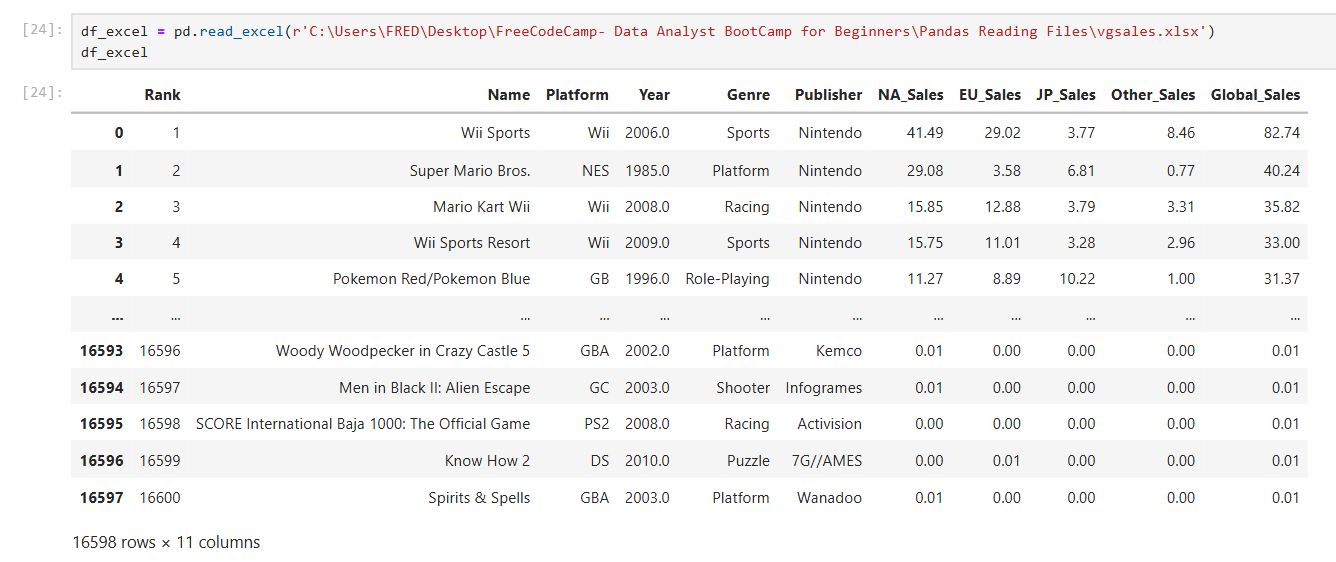


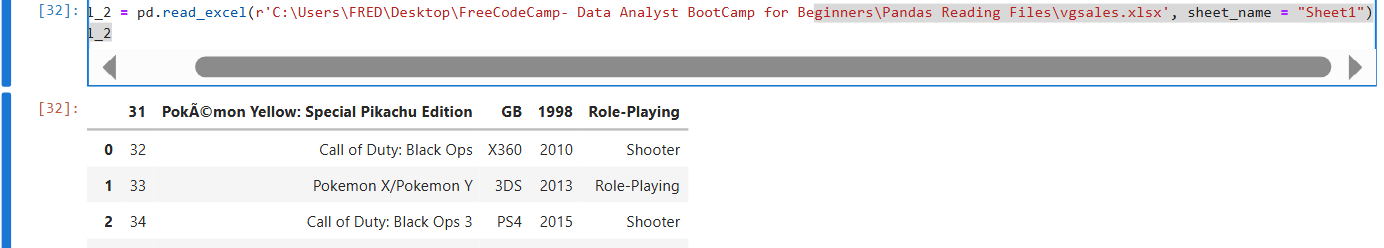
6. Reading into json file



7. If you want to read into excel file, the method read\_excel will default read the first sheet. An additional parameter is needed to read into another sheet by using the sheet name







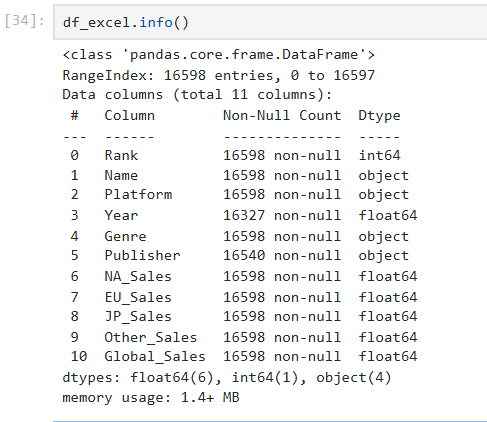
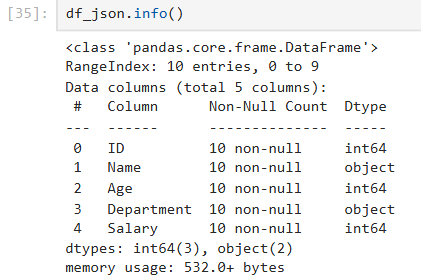
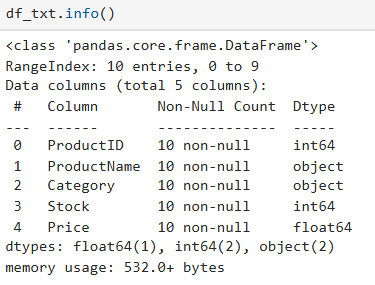
8. Do you want to see all the rows in this table?



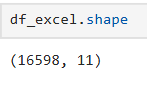
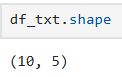
Use this 

To display all columns 

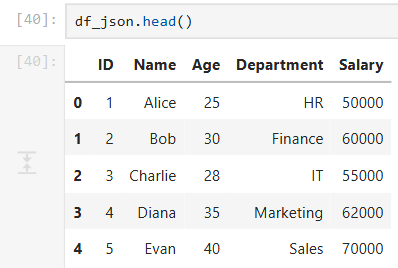
9. If you want to know the info in the imported file , use .info(). It displays the columns and how many non-null values it has and along with the data type of each row

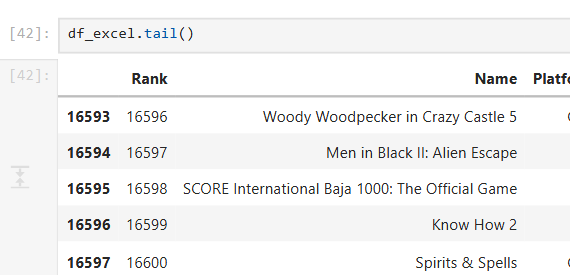
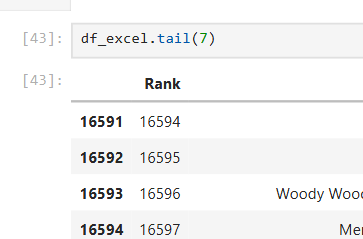
10. ‘shape’ tell the rows and columns

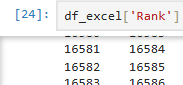
11. head() displays the first 5 values BUT it can use a parameter to display certain number of rows

12. tail() displays the last 5 rows and parameters can be used also

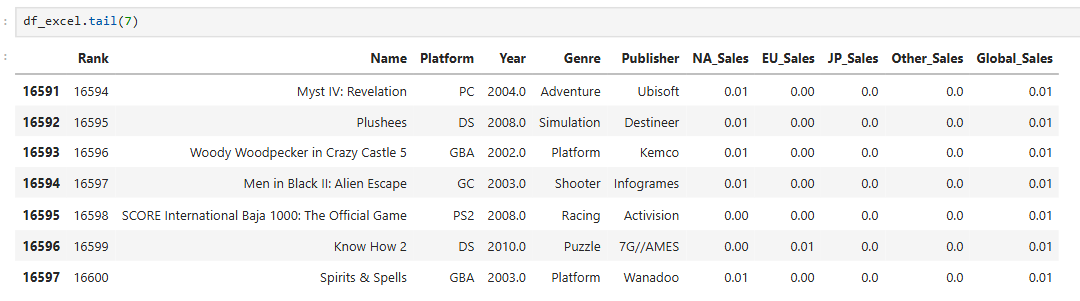
 

13. You can also choose to display/output all data from a specific column by using like a dictionary querying

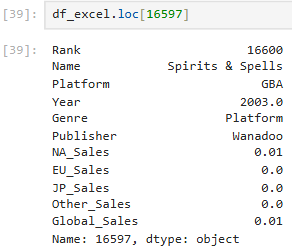
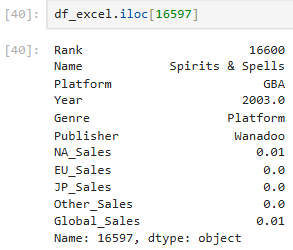


14. loc and iloc

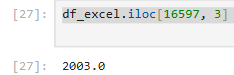
For this 2 methods we will use this chart as a reference for the output



Both loc and iloc can use the row\_index to output all the value in that row

‘iloc’ can output value of a specific row and columns. The column\_index starts from zero



For loc, the column\_index should be the name of the column



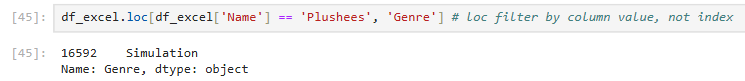
14. If you want to query using a value from a column and display all the value in the corresponding row



Using loc filter by column value



Using iloc to filter specific value by column name

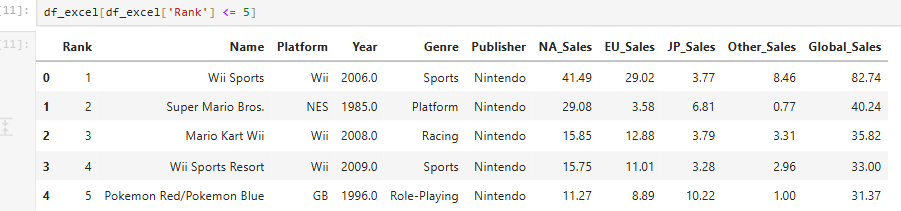


FILTERING AND ORDERING

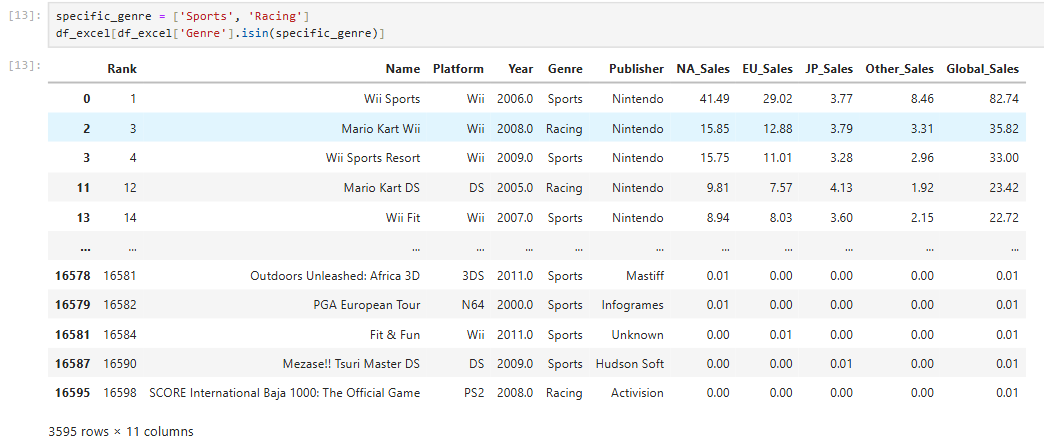
We will use this data



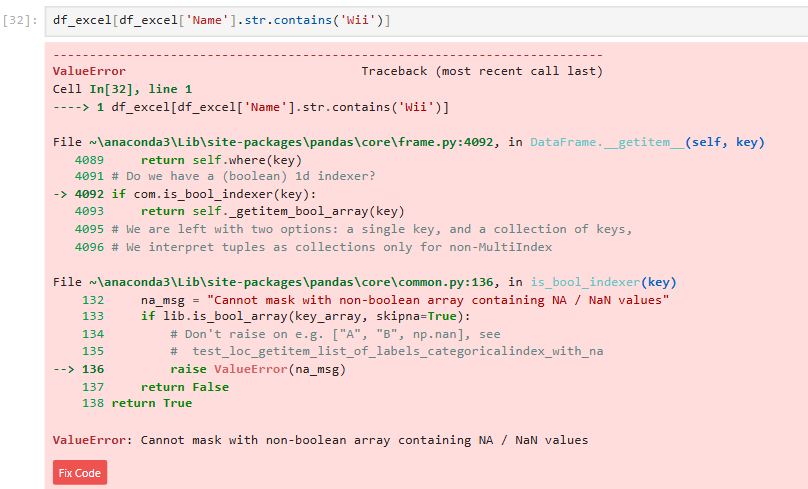
1. filtering using conditional statement



2. ‘.isin()” checks if a particular value is in the column



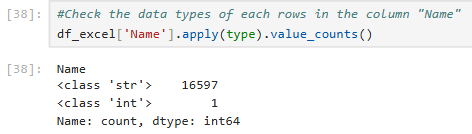
3. ‘.str.contains’ checks if the string exists in the data in the column BUT we have a special scenario. ‘str.contains’ check for data type string only. If the column contains null values or other data types is will return value error



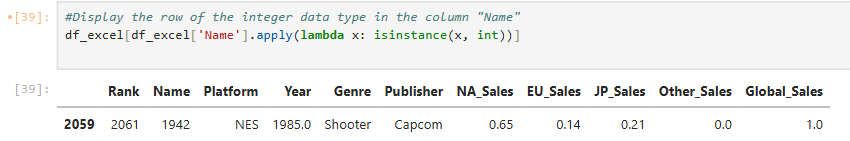
a. check for null values in the column “Name”. Result is there is no NULL values



b. Check for the data types in the column “name”. It reveals one data is int. This causes the error



c. Identify that data that is int





NOTE: “.apply()” — applies a function to each **column or row** in a DataFrame.

“.value\_counts()” — counts unique values in a Series, not a DataFrame. This method belongs to Pandas Series, not apply.

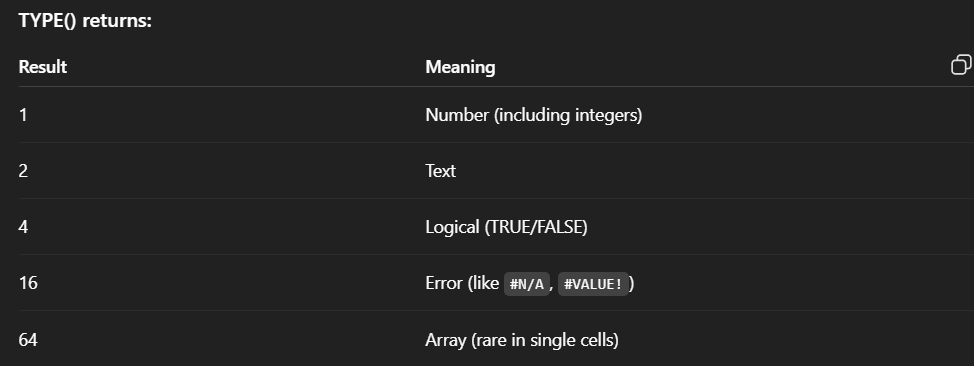
**df\_excel['Name']**

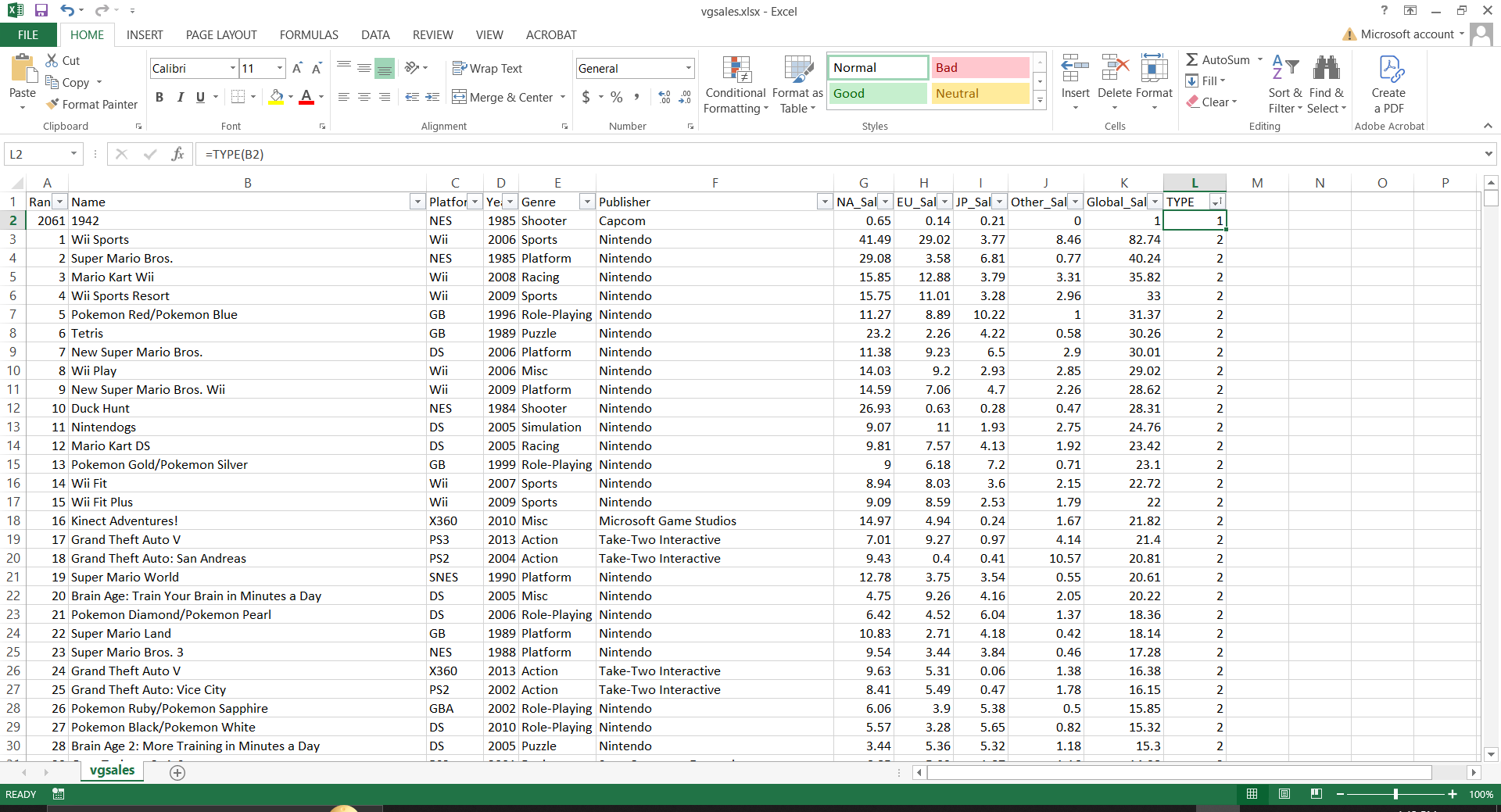
* This selects the **"Name" column** from the DataFrame. It's now a **Series** (i.e., a single column of values).

**.apply(type)**

* This applies the built-in type() function to **each row (i.e., each value)** in that column.
* So you're checking the **data type** of each individual value in the "Name" column.

d. Identifying data types in excel using the function ‘=TYPE()’

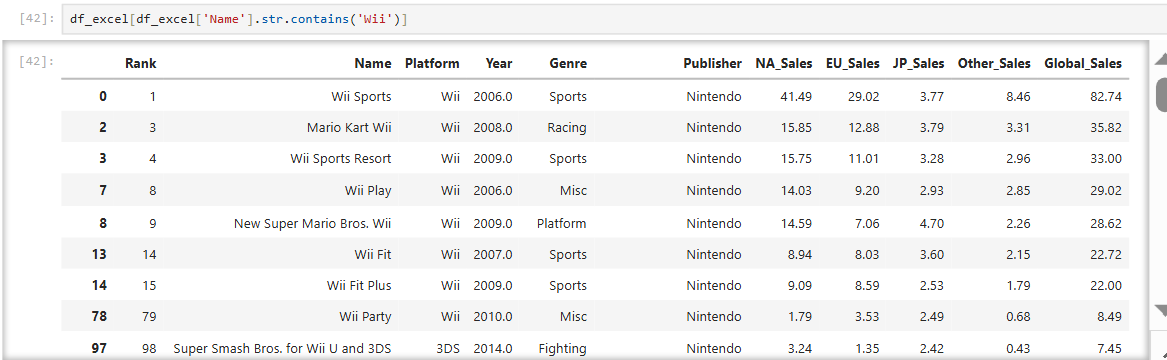




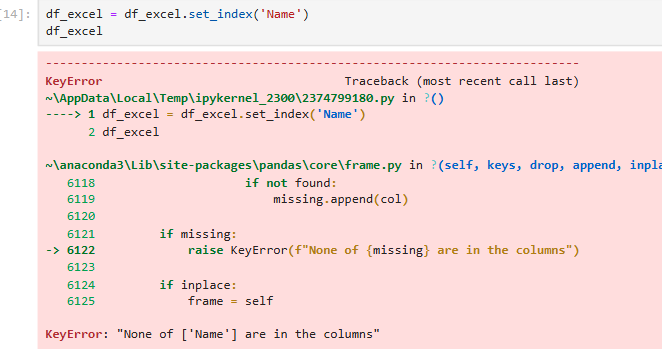
e. converting the column to data type string



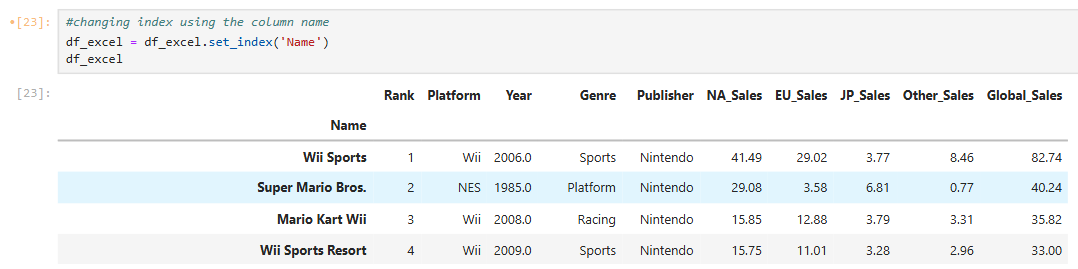
f. Querying again and it returns all names that contains “Wii”



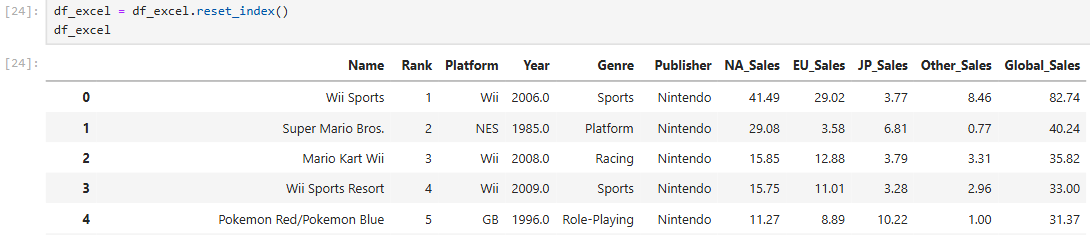
4. You can change the index by using ‘set\_index’but but sometimes it gives error if it has spaces. You can use the strip() method to remove whitespaces from the column name

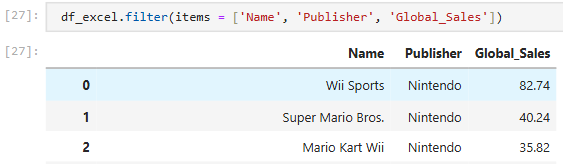
Now the column name is the index

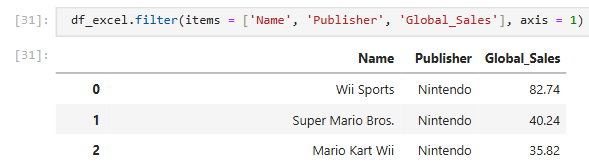


You can also reset the index back to original

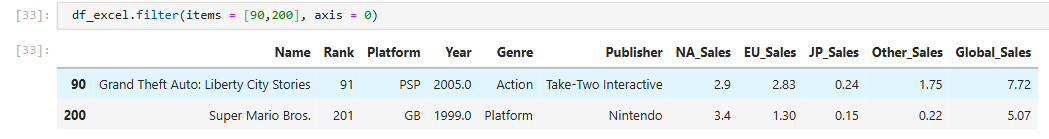


5. ‘.filter()’ lets you query on the columns and rows. If no axis indicated, the default query is the column (axis =1), it will display only the columns in the parameter items



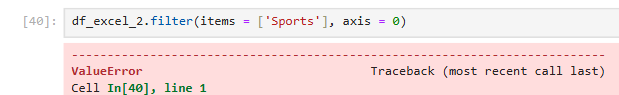


a. You can query on axis = 0 using the index



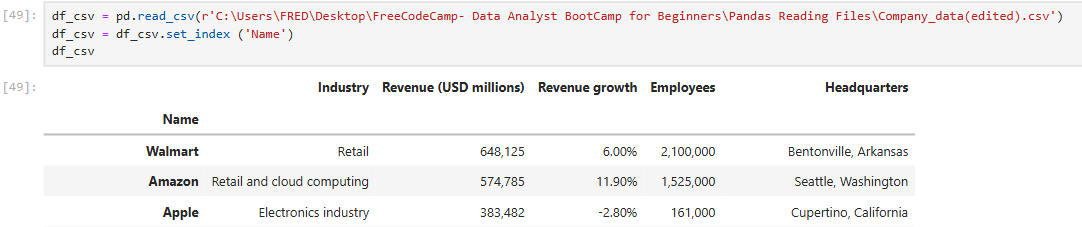
b. We can re-index and use the new index as the parameters in items. BUT be careful when reindexing because the new index should not have duplicates or there will be error using filter(items=)

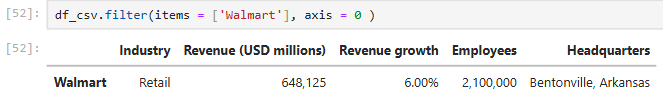




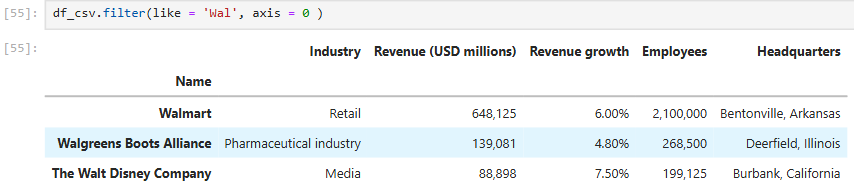


c. so now we will use other files to show filtering in axis =0

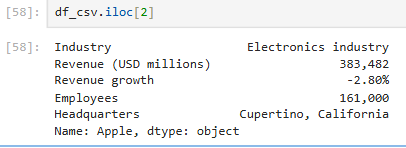
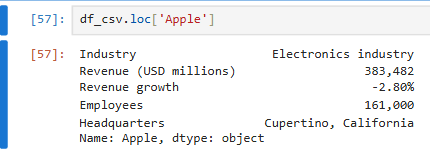




d. we can also use the parameter ‘like’, which is used to search for only the part of a keyword



e. Even though we change the index to “Name” we can still use index to query using ‘iloc’. Index still starts with zero. BUT when it comes to ‘loc’ it will need the actual index as a keyword which is the ‘Name’

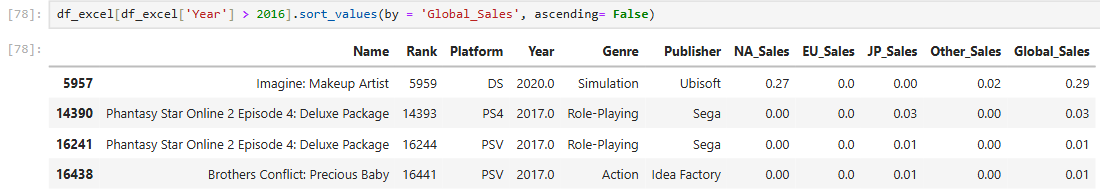
6. ordering using ‘sort\_values(by=)’. By default it is in ascending order



a. Sorting with conditional statement.



b. You can reverse the order by setting 2nd parameter to False



c. Multiple sorting. Rows will be sorted based from the first (Global\_sales) on the list on the first parameter. If there are duplicates then it will be sorted based from the second on the list (‘Genre’) of the first parameter



INDEXING

In pandas, "index" refers to the labels that identify rows (and columns in a DataFrame). It is a fundamental component of pandas Series and DataFrames, providing a way to uniquely identify and access data. The index can consist of various data types, including integers, strings, or even datetime objects.

Key aspects of pandas indexes:

Row Labels:

The primary role of an index is to serve as labels for the rows of a DataFrame or Series. This allows for label-based access to data, where you can select rows based on their specific labels rather than just their numerical position.

Data Alignment:

Indexes play a crucial role in data alignment during operations like merging, joining, or performing arithmetic operations between Series or DataFrames. Pandas uses the index to ensure that operations are performed on corresponding labels, even if the order of elements differs.

Customization:

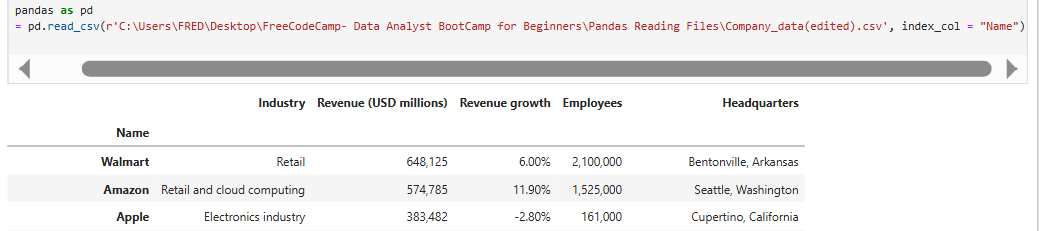
While pandas often assigns a default integer-based index (RangeIndex), you can customize the index by setting one or more columns as the index using methods like set\_index(). This is particularly useful when you have a natural identifier within your data that can serve as a meaningful index.

Accessing the Index:

You can access the index of a DataFrame or Series using the .index attribute. This returns a pandas Index object, providing information about the labels.

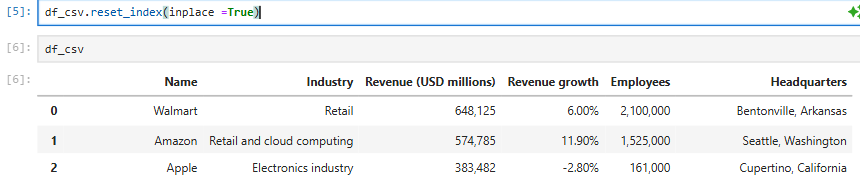
"Indexing" in pandas refers to the process of accessing and selecting data from Series and DataFrames using various methods and techniques.

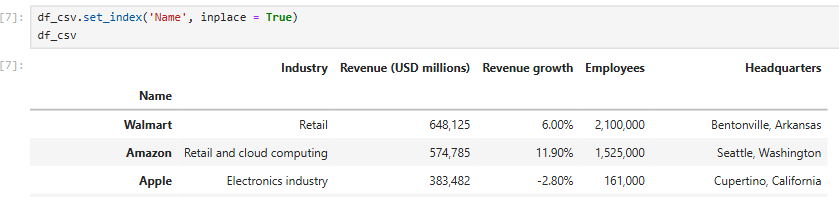
1. During reading of file, you can initially assign an index



2. “inplace()” tells pandas to **modify the DataFrame directly, without returning a new one**

Normally, reset\_index and set\_index need to be assigned to a variable in order to store the new or reset index. If you use inplace(), it directly stores the changes and there is no need to assigned to a variable



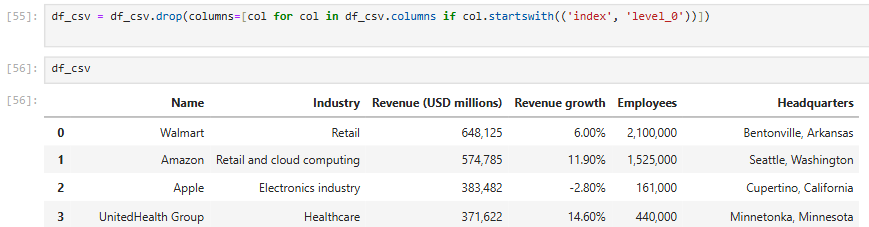


**NOTE:** if you execute ‘reset\_index(inplace = True)’ it will create new column of index and have an error in the fourth attempt. From a modified index, the first reset will reset the index. The second reset will make the ‘index’ column, the third reset will make the ‘level\_0’ column and finally on the fourth reset it will have an error

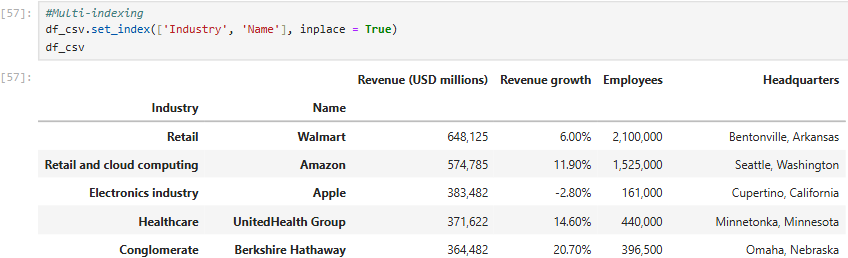




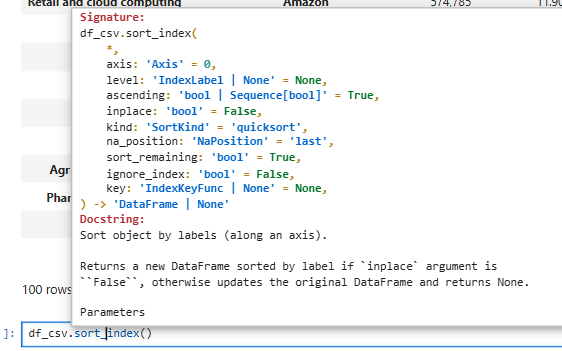
**To fix that we need to loop and remove extra columns.** “startswith()” doesn't take multiple string arguments so arguments need to pass as a **tuple** of strings if matching with multiple prefixes.

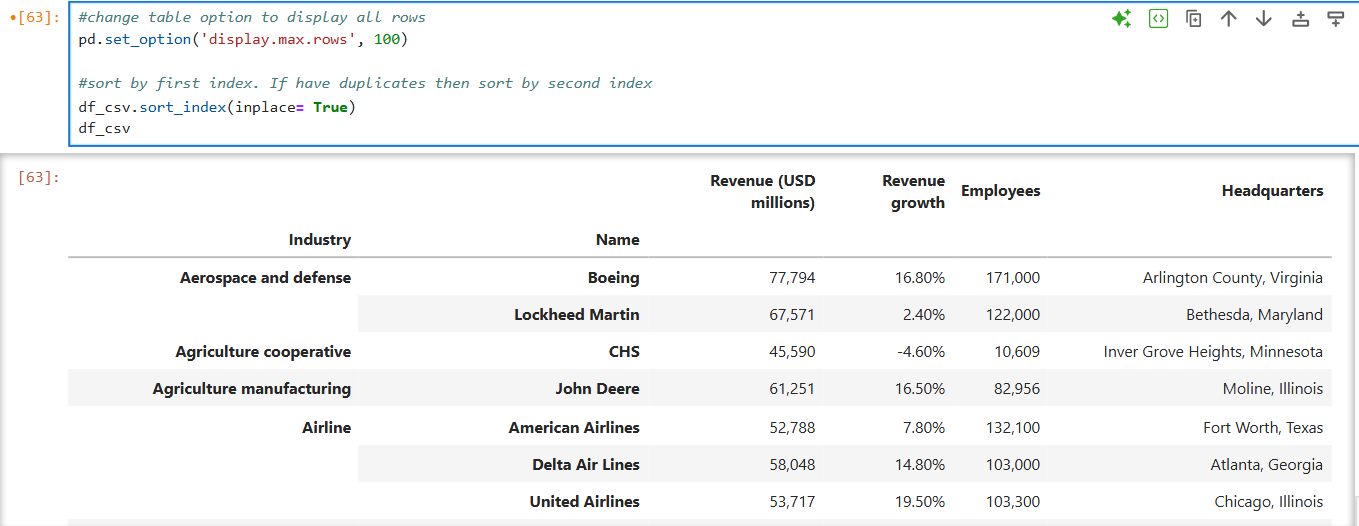


3. Multi-indexing need a list of keywords

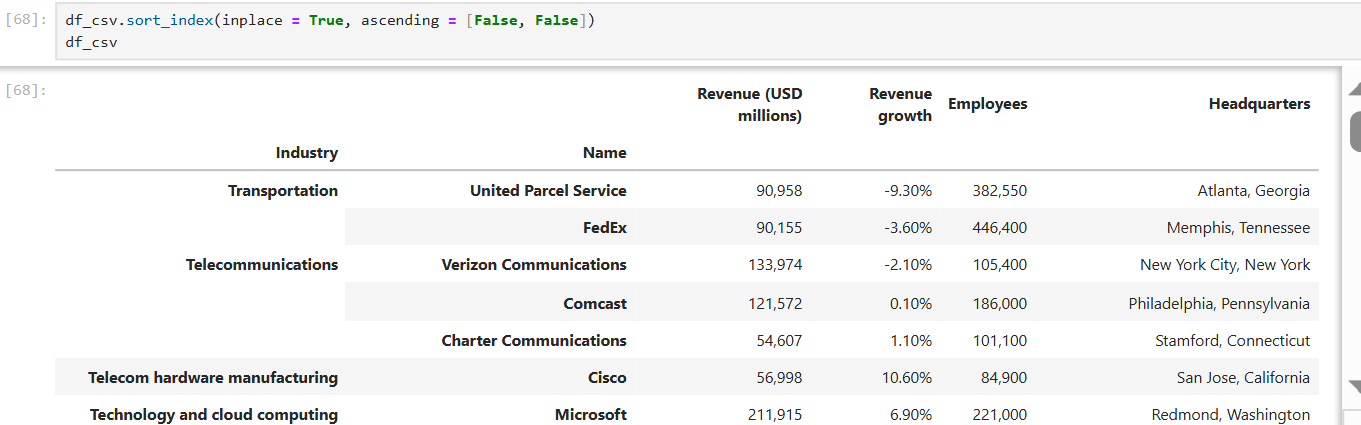


4. Sorting index

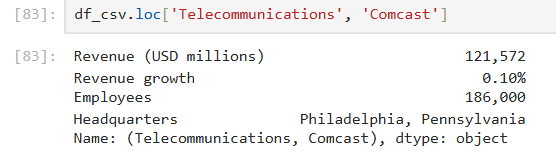




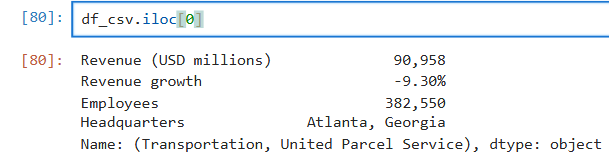
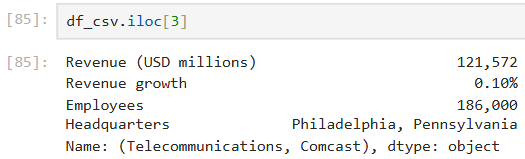
We can also arrange sorting order for each index



5. Looking up using ‘loc’ in multi-index



6. Looking up using ‘iloc’ in multi-index. In using ‘iloc’ it follows the index of the second level index ‘Name’

For specific data in the table (zero based indexing in columns and rows):

