Practical 02: Data Abstraction

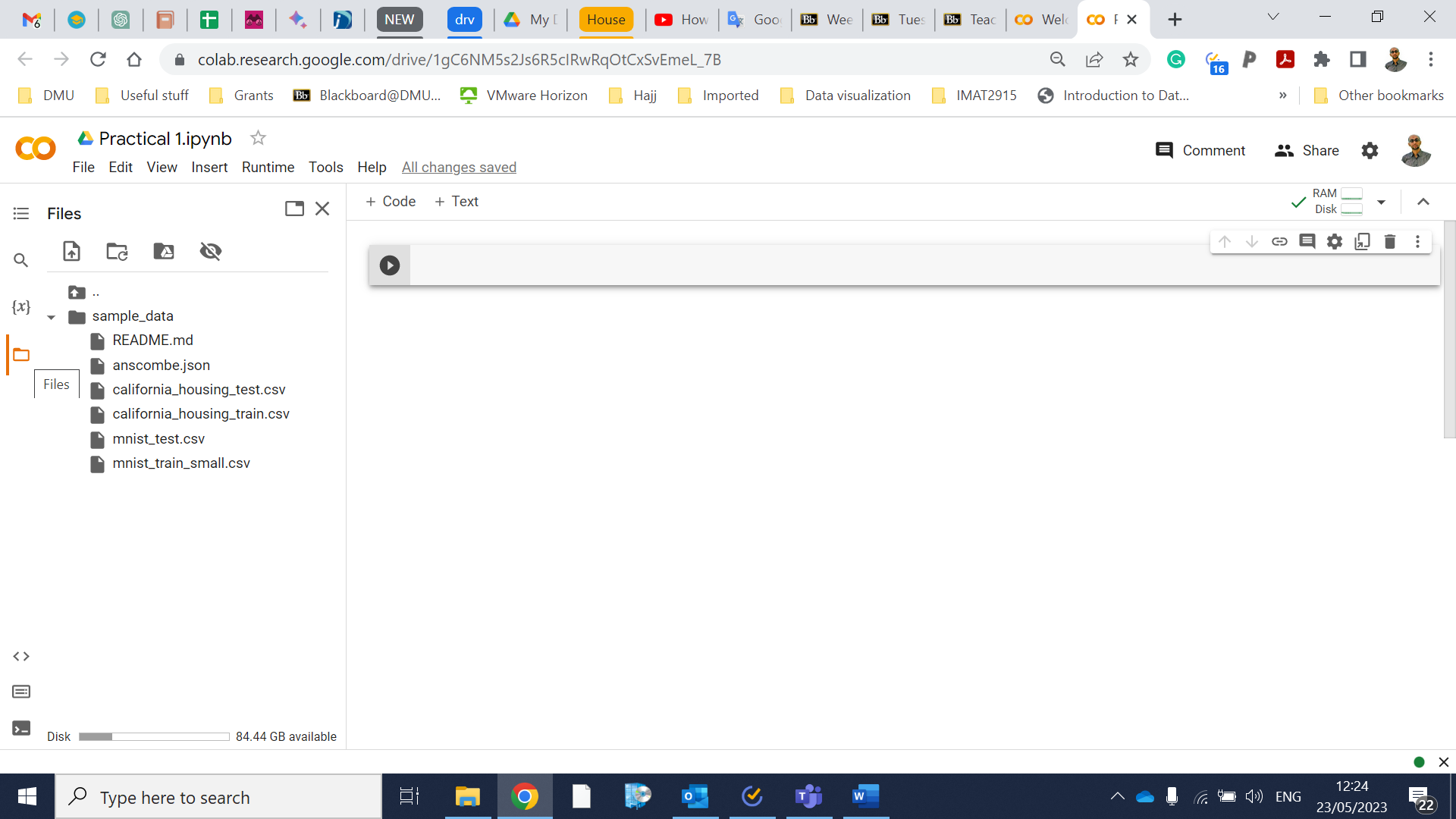
**Learning objectives:**

Visualizing a variable correctly requires understanding its data type. Having completed all of this week’s work activities, you should be able to:

* Print information about the columns (i.e., attribute) of a data frame such as data types, non-null values, and number of rows and columns.
* Exploring the unique values of the columns.
* Change from one data type to another.
* Detect missing values.
* Generate descriptive statistics.
* Access a row, a column, and a scalar value.

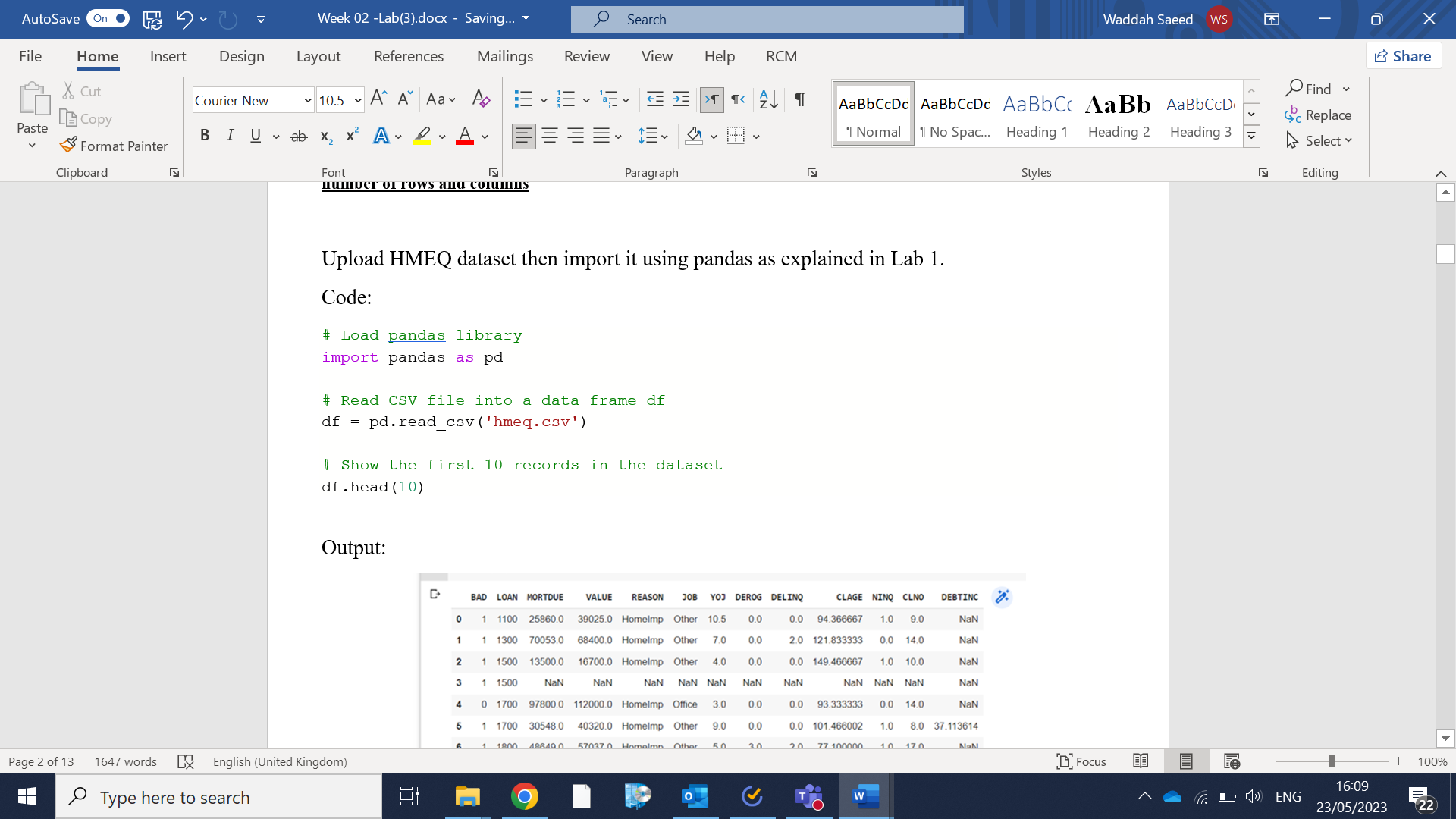
**Print information about the columns (i.e., attribute) of a data frame such as data types, non-null values, and number of rows and columns**

Download HMEQ dataset from the learning zone shell (the dataset is available on the Learning Zone shell, where you downloaded this lab sheet). Then, upload it to Colab then import it using pandas as explained in Lab 1. Follow what is shown below to upload the dataset to Colab.



Click this button

Code:



Output:

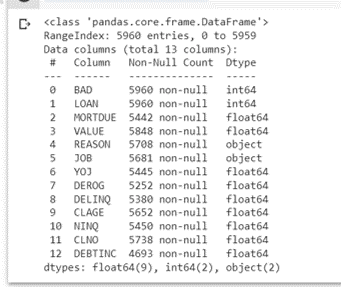


Code:

df.info(memory\_usage=False)

This method prints information about the data frame as shown in the output below. There are 5960 rows and 13 columns. The name of the columns are shown in the ‘Column’. Number of non-null values are shown in ‘Non-Null Count’. The data type for each column is shown in ‘Dtype’.

Output:



More about *info* function can be found in the link below.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.info.html>

Before moving to the next step, you should know about HMEQ dataset. The dataset was compiled by a financial services company offering home equity loans to its clients. Several thousand credit lines have been extended by the company in the past, and many of the borrowers defaulted. The attributes in HMEQ dataset are shown in the table below.

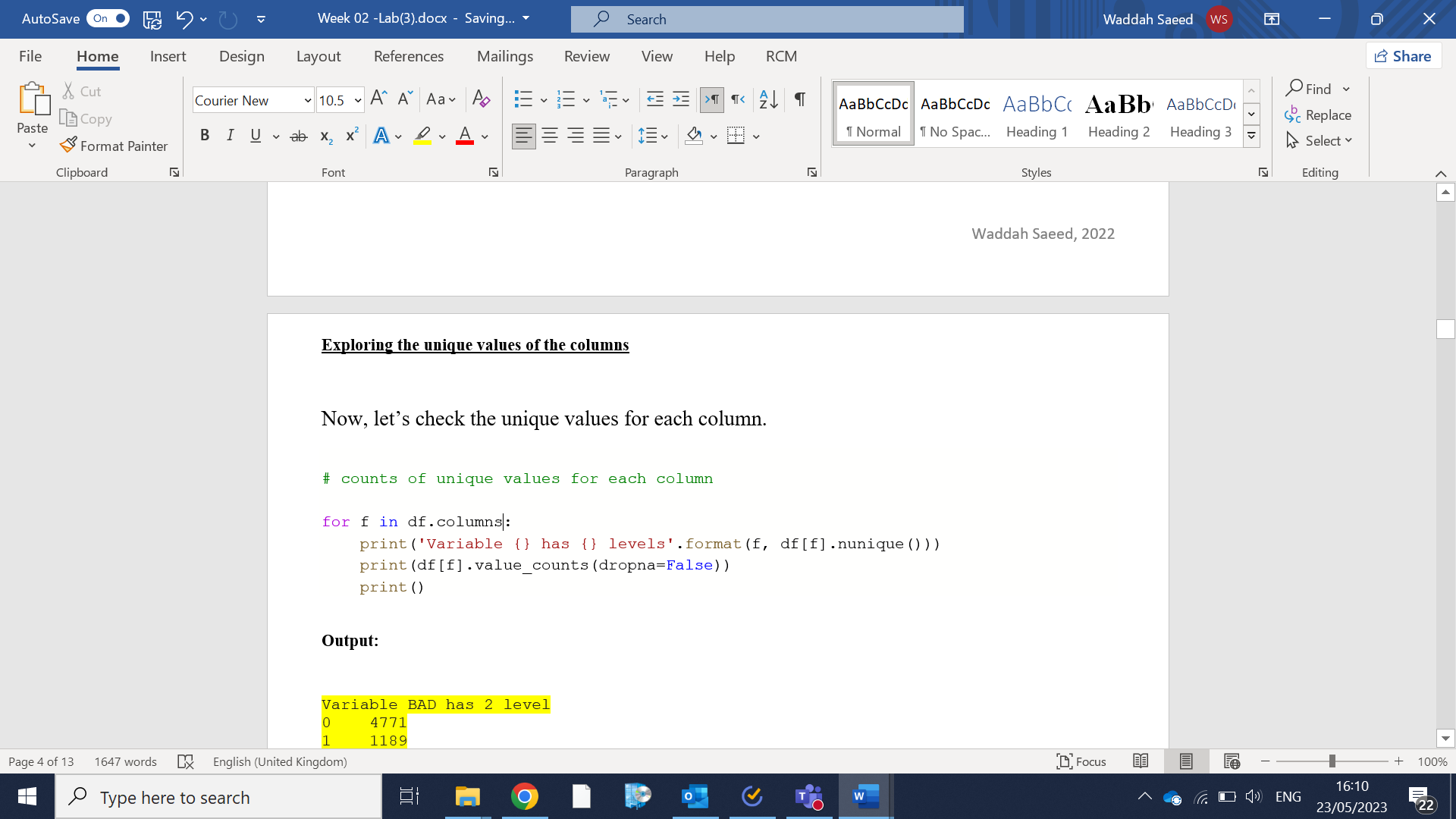
**Table 1** Variables in the HMEQ dataset

|  |  |
| --- | --- |
| Name | Description |
| BAD | 1 = applicant defaulted on loan or seriously delinquent,  0 = applicant paid loan. |
| CLAGE | Age of oldest credit line (months) |
| CLNO | Number of credit lines |
| DEBTINC | Debt-to-income ratio |
| DELINQ | Number of delinquent credit lines |
| DEROG | Number of major derogatory reports |
| JOB | Occupational categories |
| LOAN | Amount of the loan request |
| MORTDUE | Amount due on existing mortgage |
| NINQ | Number of recent credit inquiries |
| REASON | DebtCon = debt consolidation,  HomeImp = home improvement |
| VALUE | Value of current property |
| YOJ | Years at present job |

**Source: https://documentation.sas.com/doc/en/emcs/14.3/n12yb479smsxxsn0zphu0n8a0nz2.htm**

**Exploring the unique values of the columns**

Now, let’s check the unique values for each column.



**Output:**

Variable BAD has 2 level

0 4771

1 1189

Name: BAD, dtype: int64

Variable LOAN has 540 level

15000 105

10000 81

20000 74

25000 73

12000 69

...

47700 1

47100 1

46900 1

46700 1

89900 1

Name: LOAN, Length: 540, dtype: int64

Variable MORTDUE has 5053 level

NaN 518

42000.0 11

47000.0 10

65000.0 9

124000.0 7

...

65372.0 1

15346.0 1

58549.0 1

69195.0 1

48811.0 1

Name: MORTDUE, Length: 5054, dtype: int64

Variable VALUE has 5381 level

NaN 112

60000.0 15

80000.0 14

85000.0 12

65000.0 11

...

116994.0 1

42682.0 1

72175.0 1

70095.0 1

88934.0 1

Name: VALUE, Length: 5382, dtype: int64

Variable REASON has 2 level

DebtCon 3928

HomeImp 1780

NaN 252

Name: REASON, dtype: int64

Variable JOB has 6 level

Other 2388

ProfExe 1276

Office 948

Mgr 767

NaN 279

Self 193

Sales 109

Name: JOB, dtype: int64

Variable YOJ has 99 level

NaN 515

0.00 415

1.00 363

2.00 347

5.00 333

...

29.90 1

12.90 1

13.50 1

0.25 1

8.30 1

Name: YOJ, Length: 100, dtype: int64

Variable DEROG has 11 level

0.0 4527

NaN 708

1.0 435

2.0 160

3.0 58

4.0 23

5.0 15

6.0 15

7.0 8

8.0 6

9.0 3

10.0 2

Name: DEROG, dtype: int64

Variable DELINQ has 14 level

0.0 4179

1.0 654

NaN 580

2.0 250

3.0 129

4.0 78

5.0 38

6.0 27

7.0 13

8.0 5

10.0 2

11.0 2

15.0 1

12.0 1

13.0 1

Name: DELINQ, dtype: int64

Variable CLAGE has 5314 level

NaN 308

102.500000 7

206.966667 7

177.500000 6

123.766667 6

...

240.856017 1

196.241371 1

71.461705 1

184.880011 1

219.601002 1

Name: CLAGE, Length: 5315, dtype: int64

Variable NINQ has 16 level

0.0 2531

1.0 1339

2.0 780

NaN 510

3.0 392

4.0 156

5.0 75

6.0 56

7.0 44

10.0 28

8.0 22

9.0 11

11.0 10

12.0 2

13.0 2

14.0 1

17.0 1

Name: NINQ, dtype: int64

Variable CLNO has 62 level

16.0 316

19.0 307

24.0 264

23.0 259

21.0 235

...

58.0 3

71.0 2

53.0 2

57.0 1

63.0 1

Name: CLNO, Length: 63, dtype: int64

Variable DEBTINC has 4693 level

NaN 1267

34.964141 1

41.576701 1

41.395462 1

20.688715 1

...

39.244669 1

40.943866 1

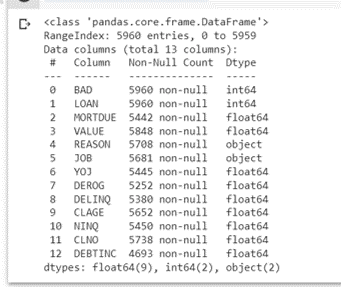
30.444839 1

36.158718 1

34.571519 1

Name: DEBTINC, Length: 4694, dtype: int64

You may be noticed that the data type of BAD, REASON, and JOB attributes were not correctly identified by pandas. The data type of BAD is int64 (it is numeric, but we do not need to do any type of mathematic operations with this attribute), while it is object for REASON and JOB. However, the results from the unique values shows that all these attributes are categorical variables.



Variable BAD has 2 level

0 4771

1 1189

Name: BAD, dtype: int64

Variable REASON has 2 level

DebtCon 3928

HomeImp 1780

NaN 252

Name: REASON, dtype: int64

Variable JOB has 6 level

Other 2388

ProfExe 1276

Office 948

Mgr 767

NaN 279

Self 193

Sales 109

Name: JOB, dtype: int64

More about *value\_counts* function can be found in the link below.

<https://pandas.pydata.org/docs/reference/api/pandas.Series.value_counts.html>

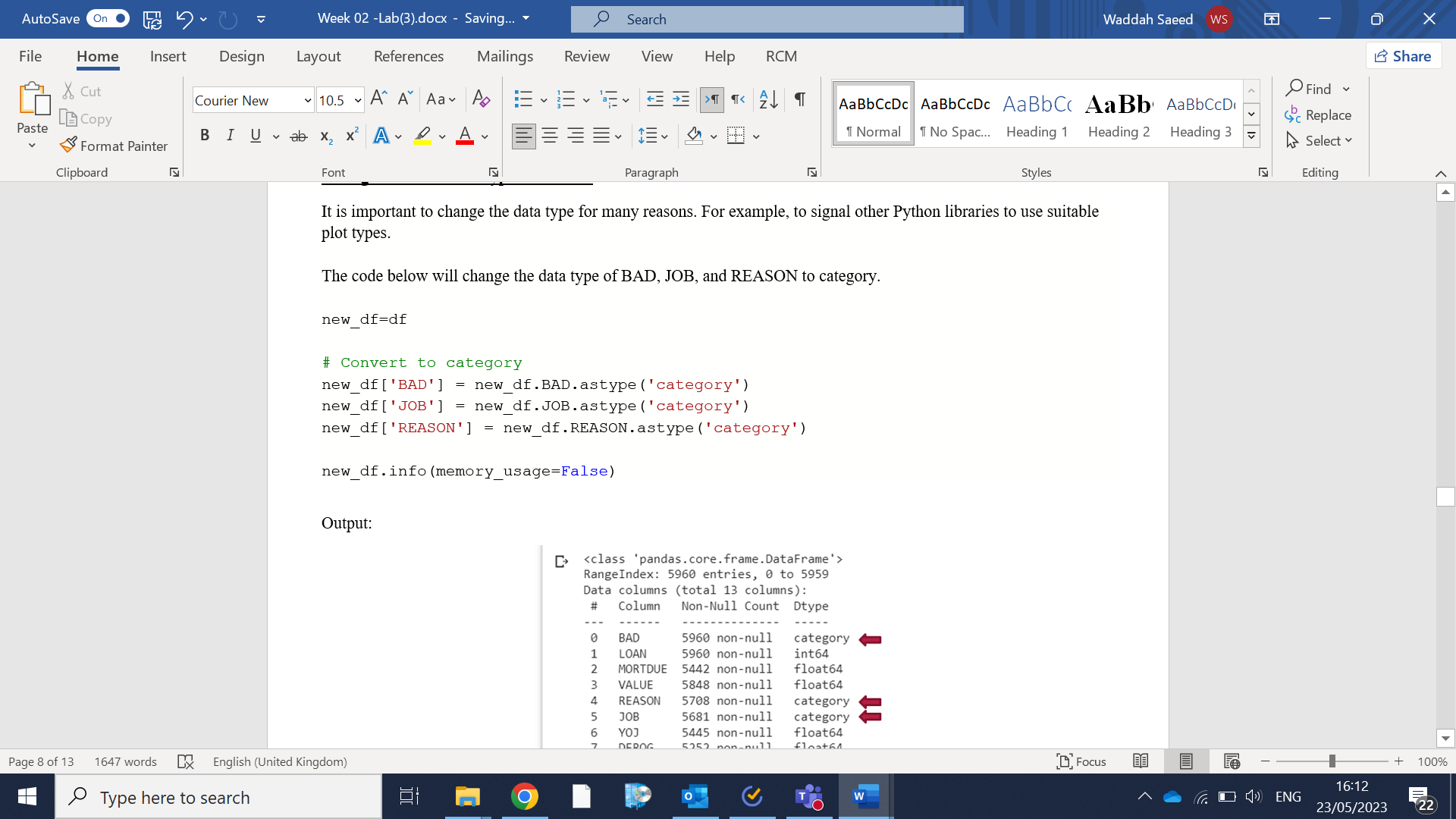
More about *nunique* function can be found in the link below.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.nunique.html>

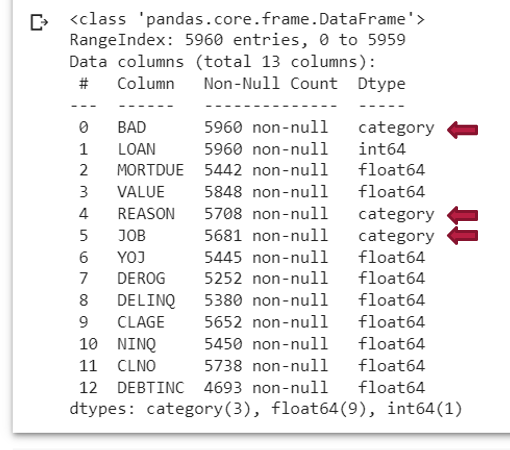
**Change from one data type to another**

It is important to change the data type for many reasons. For example, to signal other Python libraries to use suitable plot types.

The code below will change the data type of BAD, JOB, and REASON to category using *astype*() function.



Output:



More about *astype* function can be found in the link below.

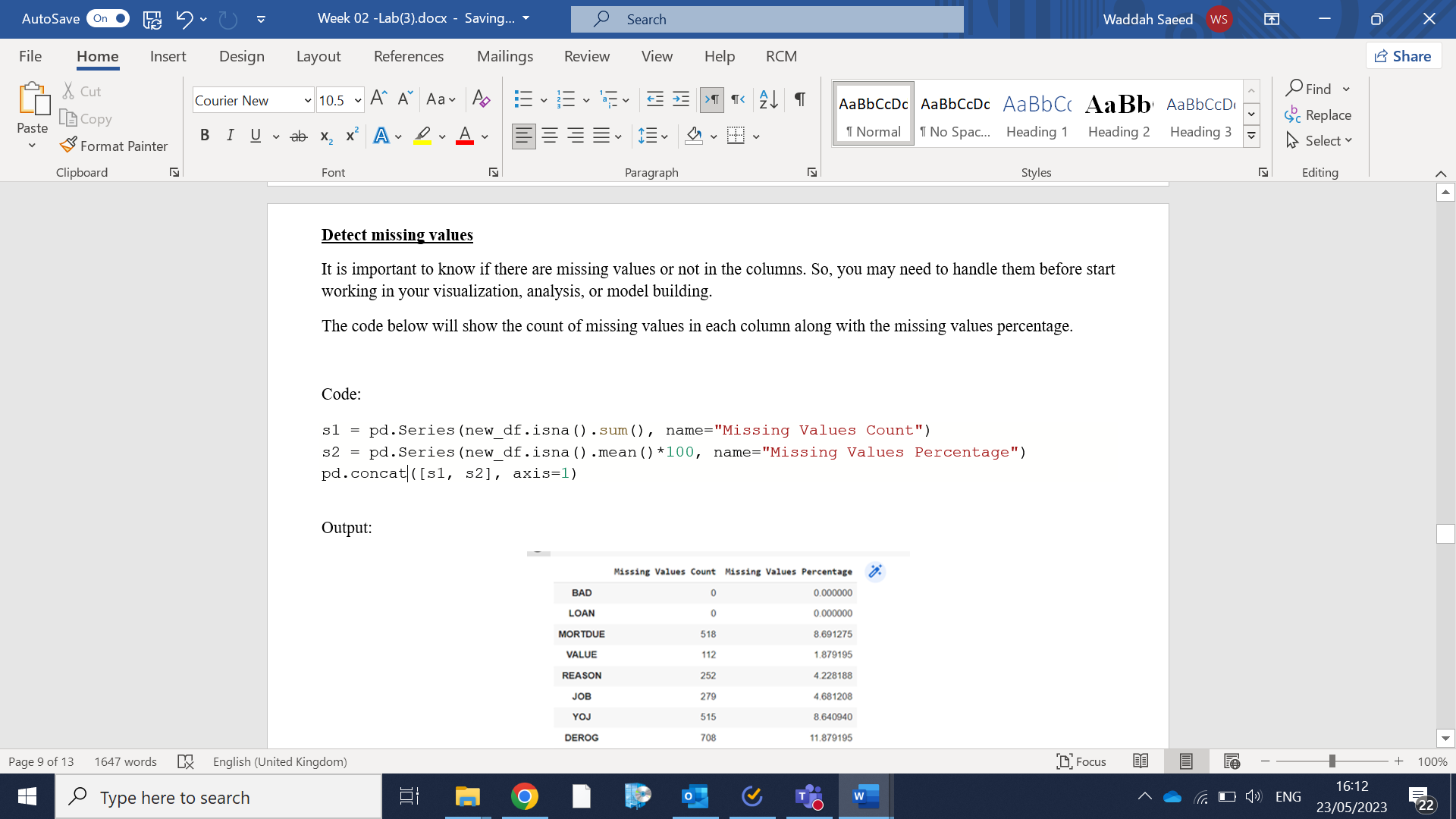
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.astype.html>

**Detect missing values**

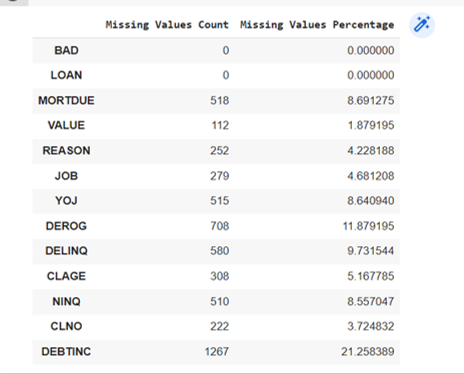
It is important to know if there are missing values or not in the columns. So, you may need to handle them before start working in your visualization, analysis, or model building.

The code below will show the count of missing values in each column along with the missing values percentage.

Code:



Output:

****

More about *Series* function can be found in the link below.

https://pandas.pydata.org/docs/user\_guide/categorical.html

More about *isna* function can be found in the link below.

https://pandas.pydata.org/docs/reference/api/pandas.isna.html

More about *concat* function can be found in the link below.

https://pandas.pydata.org/docs/reference/api/pandas.concat.html

**Generate descriptive statistics**

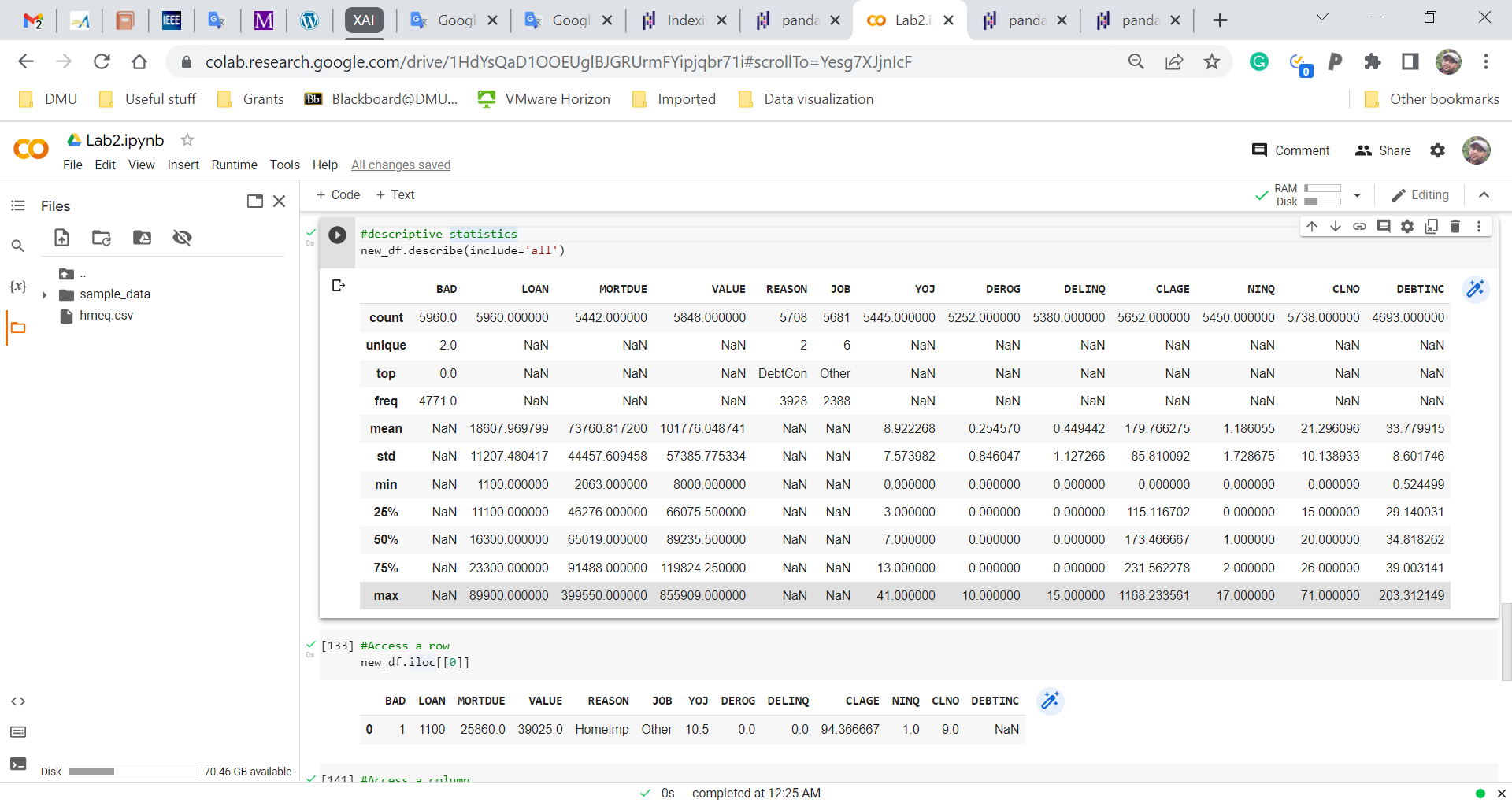
During data exploration stage, it is a good practice to generate the descriptive statistics that summarize the central tendency, dispersion and shape of a dataset’s distribution to have a better idea about the data. To do so, you can use the describe() function to generate the descriptive statistics for the data frame, as shown in the code below.

Code:

#Descriptive statistics

new\_df.describe(include='all')

Output:

****

For numeric data, it shows count, mean, std, min, max as well as lower, 50 and upper percentiles. By default, the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

For category data, it shows count, unique, top, and freq. The top is the most common value. The freq is the most common value’s frequency.

More about *describe* function can be found in the link below.

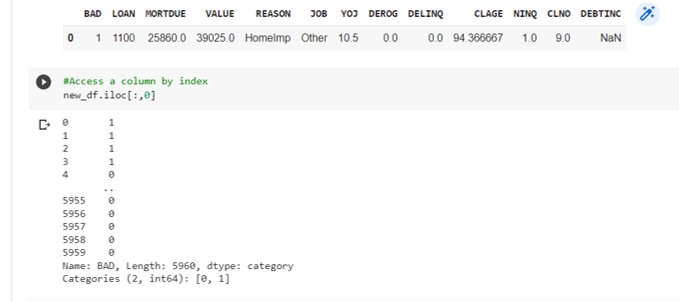
[https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html#](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html)

**Access a row, a column, and a scalar value**

The code below shows how to access a row, a column, and a scalar value.

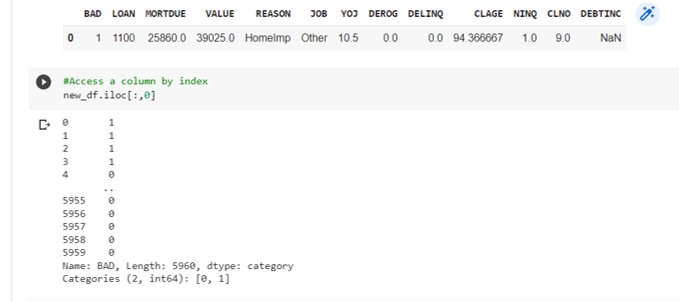
#Access a row

new\_df.iloc[[0]]



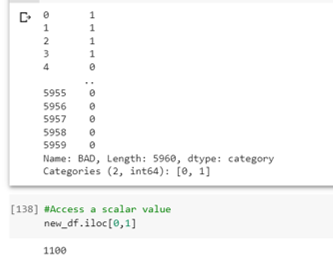
#Access a column by index

new\_df.iloc[:,0]



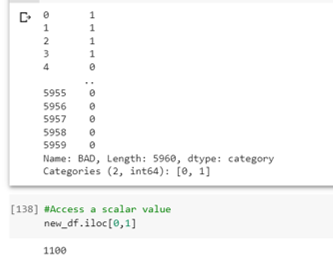
#Access a column by name

new\_df.loc[:,'BAD']



#Access a scalar value

new\_df.iloc[0,1]



More about *iloc* can be found in the link below.

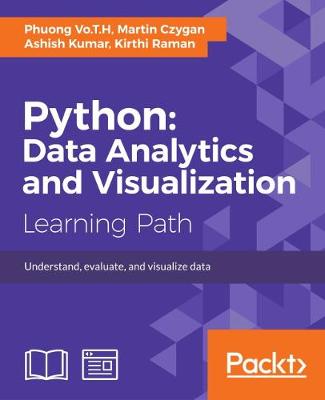
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html>

More about *loc* can be found in the link below.

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html>

Extra Reading (Recommended):

ResourceList on the learning zone -> Python: Data Analytics and Visualization -> Data Analysis with Pandas – Chapter 3.



**Exercises:**

Bagendshire Council currently uses a digital platform called *‘Mobi-Sort-it’*, for collecting reports on services from local residents. Your consulting firm has recently been hired by the *Bagendshire local Council* to support decision making around its basic service delivery operations on Water, Sanitation, Electricity, Roads, and Parks & Recreation. Below are the attributes in the MobiSortIt\_data (the dataset is available on the Learning Zone shell, where you downloaded this lab sheet).

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **TicketID** | The unique number assigned to each service delivery report recorded, as a ticket. |
| **Category** | The type of service issue i.e. Water, Roads, Electricity, Parks & Recreation, and Sanitation |
| **Similar\_problems** | This records whether the resident reporting the issues if aware of the same problem affecting other areas in the neighbourhood. |
| **Duration** | The duration in days, that the issue has existed in the area. |
| **Status** | This indicates the status of the ticket. When a ticket is first recorded, its status is ‘**open**’. When it gets assigned to the relevant council department to be addressed, it is referred to as ‘**assigned**’. When the reported issue has been addressed by the council, the status of the ticket changes to ‘**resolved**’. Once the resident who reported the issue is satisfied that it is addressed, the ticket is closed and its status is recorded as ‘**closed**’. |
| **Issuetype** | Each service delivery category has a type of issue. This attribute records the specific *type* of issue recorded for a particular service delivery category. |
| **Private property** | This indicates whether the issue reported is on private property (e.g. a resident’s house) or an area accessible to the public (e.g. on the street or community hall, etc) |
| **Assignment** | The specific department at the Bagendshire Council that has been assigned the service delivery issue to address. |

Individually:

Using the dataset, perform the following tasks:

1. Print information about the attributes, including data types, non-null values, and the number of rows and columns.
2. Explore the unique values within the attributes.
3. Determine if there is a need to change data types (e.g., convert to the 'category' data type).
4. Identify any missing values within the attributes.
5. Generate descriptive statistics.
6. Access the second row, the first column, and the scalar value in the third row and the second column.

Peer Discussion:

* Discuss with your peer what you did for Question 3 and explain the reasons behind your decision.
* Share your thoughts with your peer regarding any interesting findings in the results from Question 5 and explain why you find them interesting.