Government College of Technology, Coimbatore – 13

**TamilNadu Marginal Workers Assessment**

**Team Members:**

1. **VenkatesaPerumal. R**
2. **Aravinth M**
3. **Sugash M**

**PROJECT DEVELOPMENT LEVEL – 1**

**Libraries Used:**

* **Pandas** – *Pandas is a fundamental library for data analysis and manipulation. You can use it to load, clean and process data in various formats such as CSV, Excel or databases. This library is essential for data preprocessing.*
* **Numpy** *– Numpy is essential for numerical operations. You can use it for various calculations, such as statistical analysis, averaging values and more.*
* **Matplotlib and Seaborn** – *Matplotlib and Seaborn are great choices for creating visualizations of data. You can generate time series plots, scatter plots, bar plots, heat map to understand the trends and patterns in data.*
* **Scikit-Learn** – *A popular machine learning library in python that provides simple and efficient tools for data mining and data analysis.*

**Loading a Dataset:**

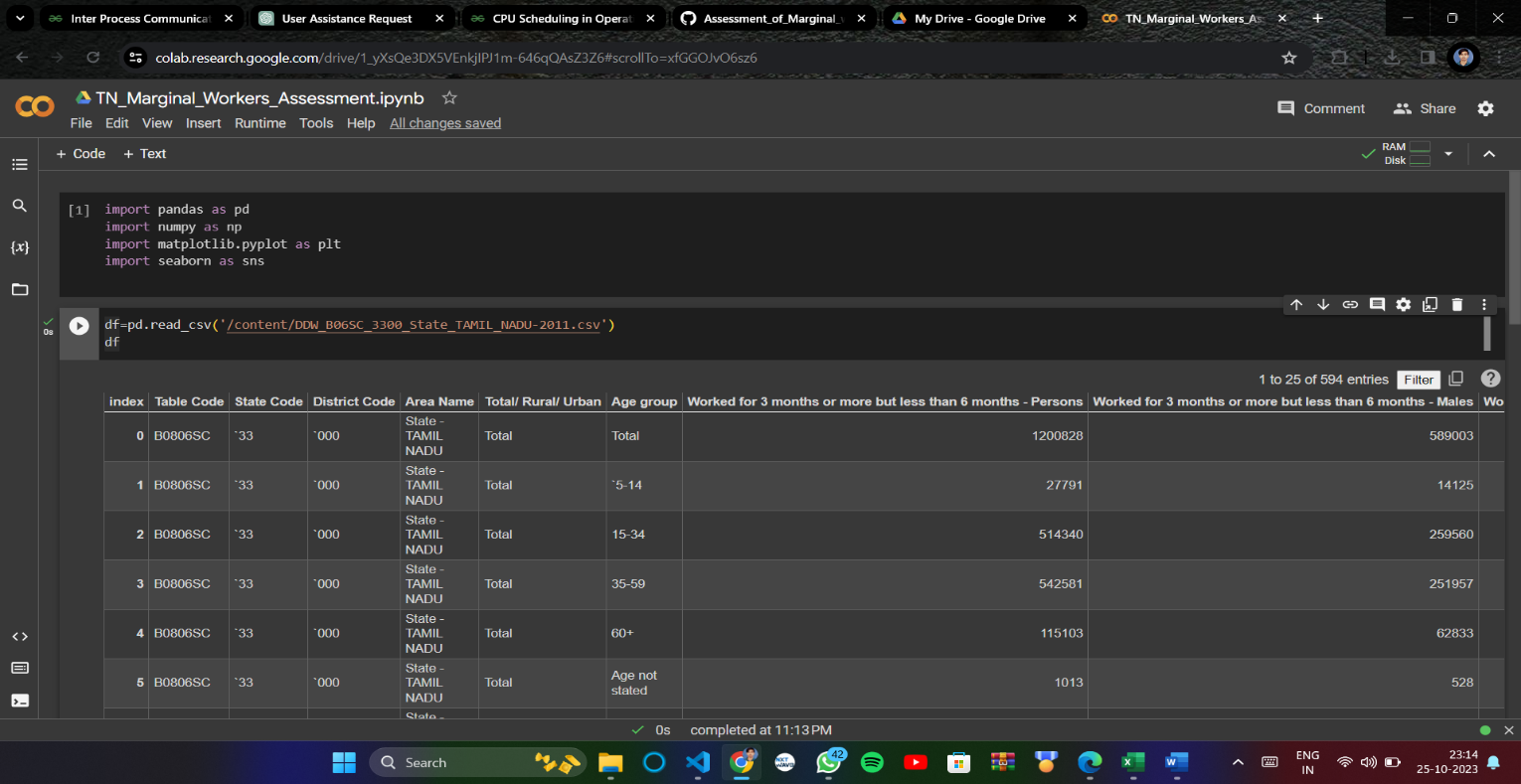
1. Using **pd.read\_csv()** for a CSV file

import pandas as pd

df=pd.read\_csv(‘dataset\_name.csv’)

2. Using **pd.read\_excel()** for a Excel file

import pandas as pd

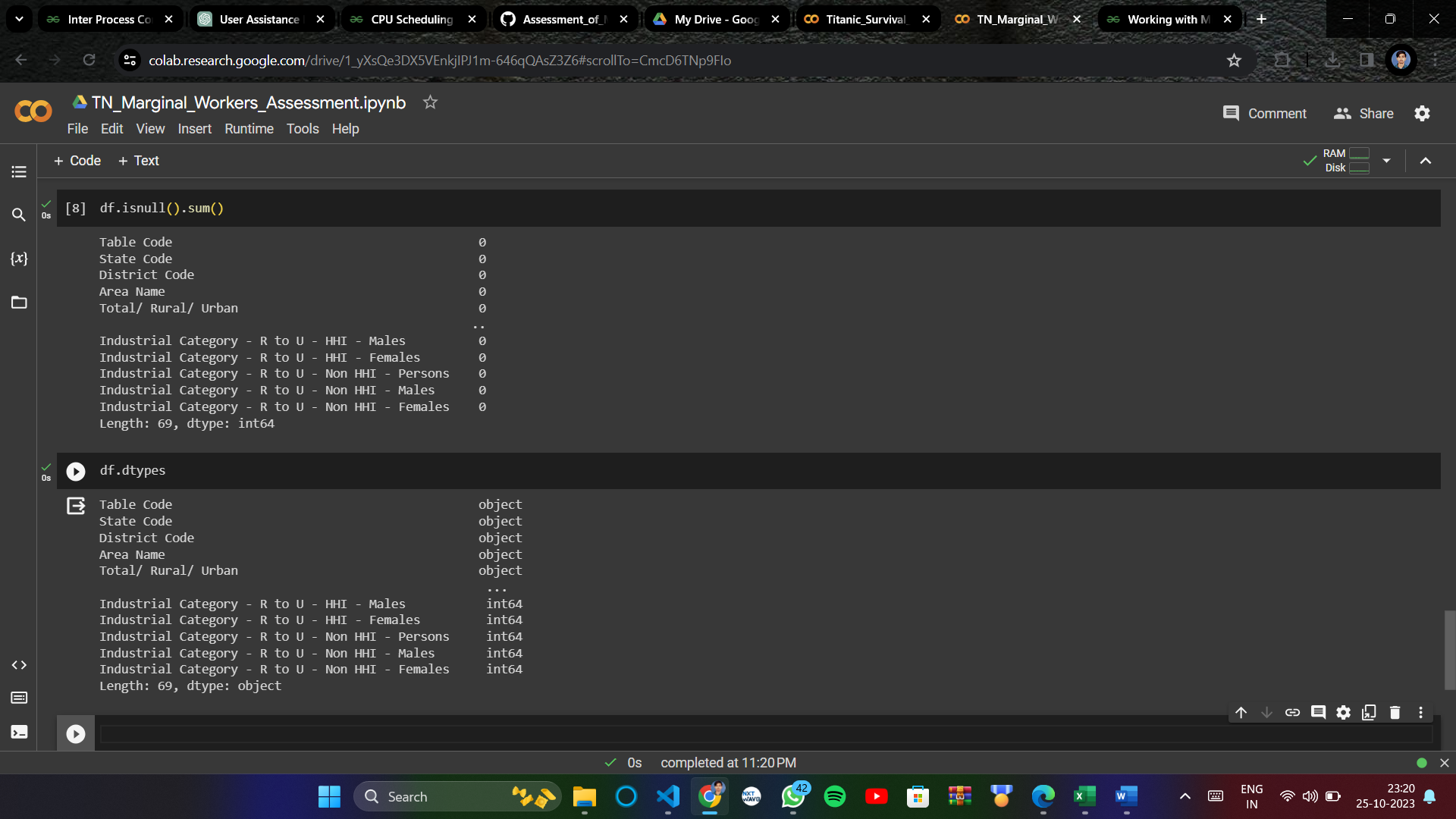
 df=pd.read\_excel(‘dataset\_name.xlsx’)

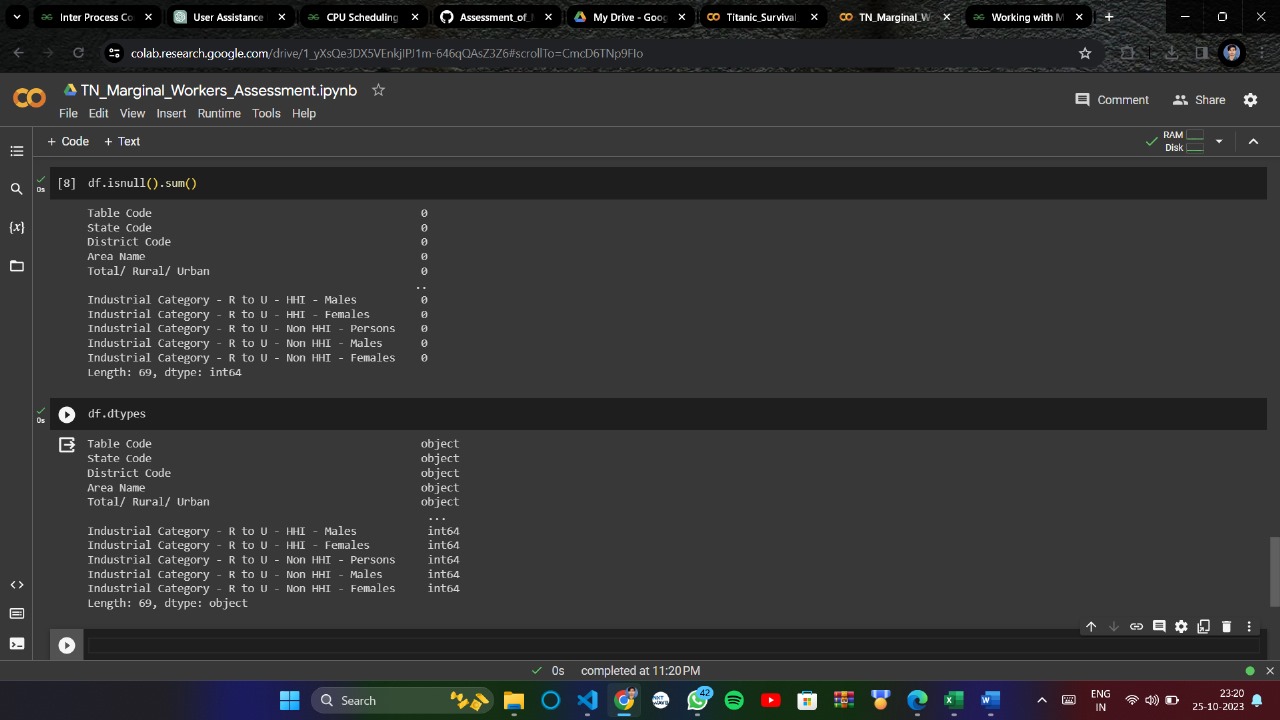
**Data Preprocessing:**

**1. Checking for null values in dataset:**

df.isnull().sum() - returns a pandas Series with the count of missing values in each column of the DataFrame df. Each element in the resulting Series represents the number of missing values in the corresponding column of the original DataFrame. This information is valuable when dealing with data cleaning tasks, as it helps to understand the extent of missing data in the dataset.

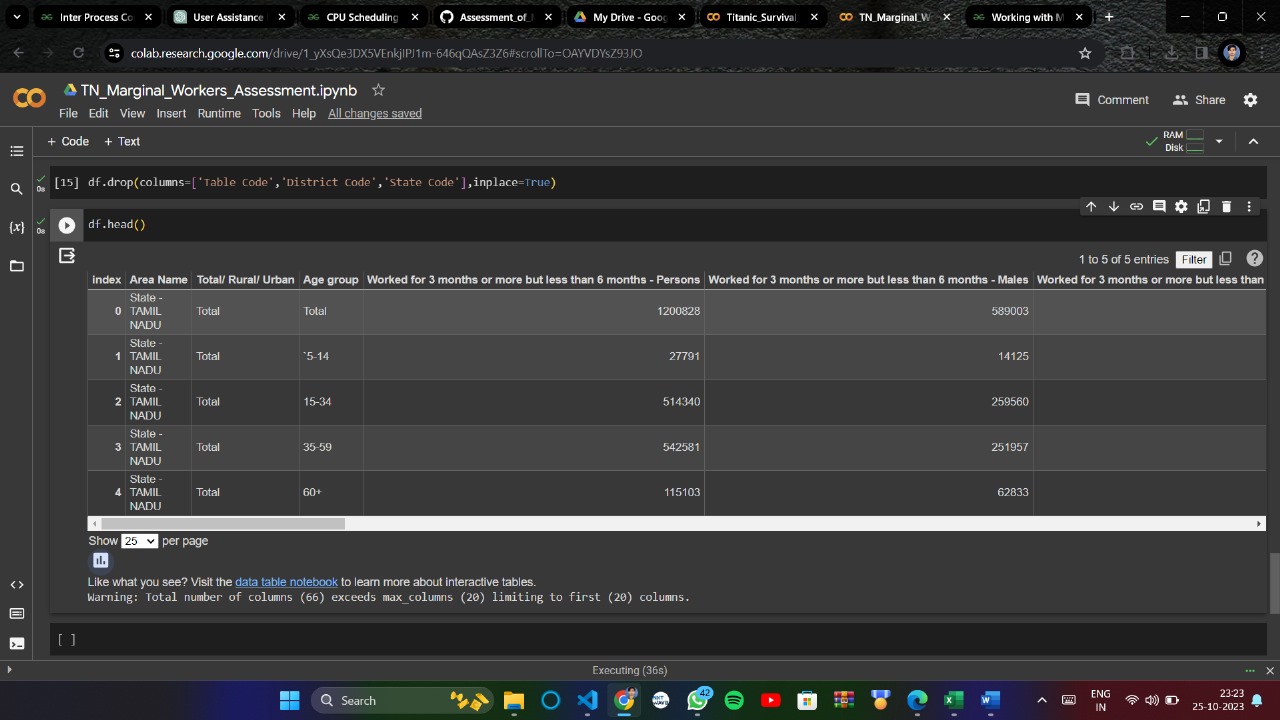
**2. Checking the datatypes of each column:**

df.dtypes - The dtype attribute provides information about the type of data stored in the object's elements (i.e., the data type of the elements within a column of a DataFrame or a Series.



**3. Dropping unwanted Columns in a dataset:**

df.drop(columns=['Table Code','District Code','State Code'],inp.lace=True) - removes the columns named 'Table Code', 'District Code', and 'State Code' from the DataFrame df without creating a new DataFrame, as the modification is done in place. After executing this code, the original DataFrame df will no longer contain these columns.



**4. Converting Categorical values to Numerical Values:**

from sklearn.preprocessing import LabelEncoder

label=LabelEncoder()

df['Area Name']=label.fit\_transform(df['Area Name'])

df['Age group']=label.fit\_transform(df['Age group'])

df['Total/ Rural/ Urban']=label.fit\_transform(df['Total/ Rural/ Urban'])

**import LabelEncoder**:

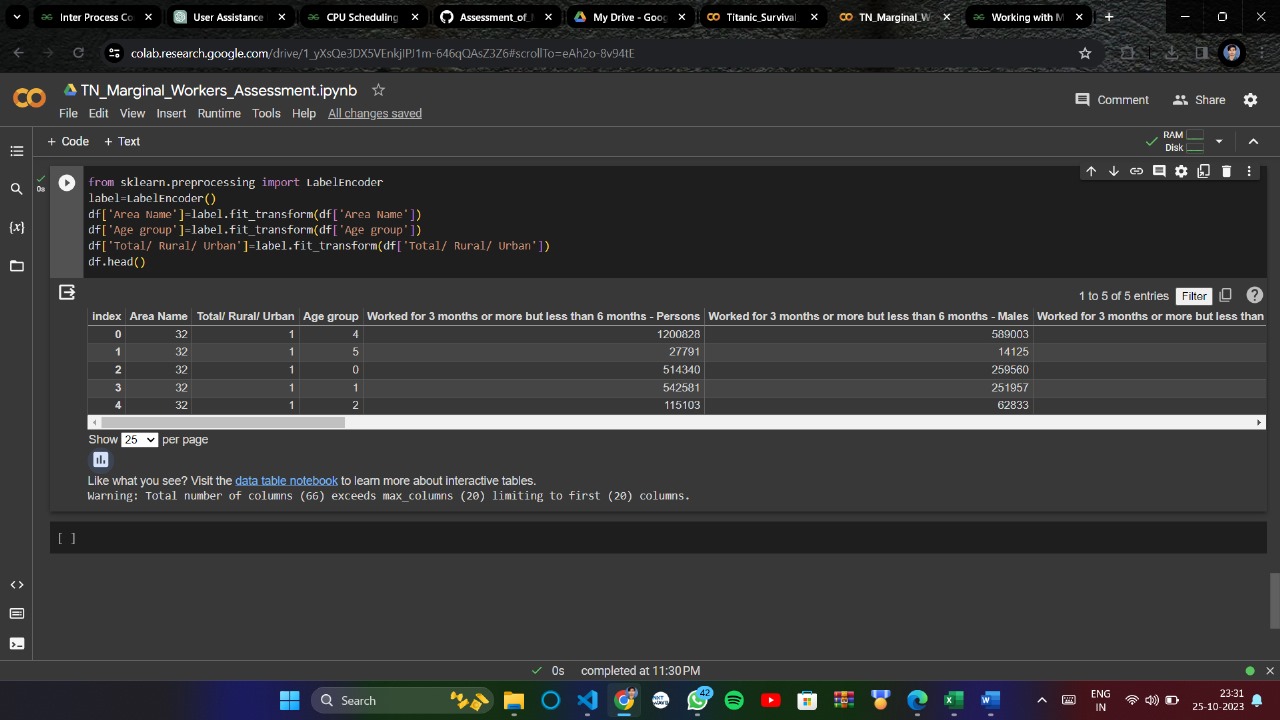
* The LabelEncoder class from sklearn.preprocessing is imported. This class helps convert categorical data into numerical labels.

**Instantiate LabelEncoder Object**:

* A LabelEncoder object named label is created. This object will be used to perform the encoding.

**Encode Categorical Columns**:

* The code applies the fit\_transform method of LabelEncoder to three columns ('Area Name', 'Age group', 'Total/ Rural/ Urban') in the DataFrame df. This transforms categorical strings in these columns into corresponding numerical labels.



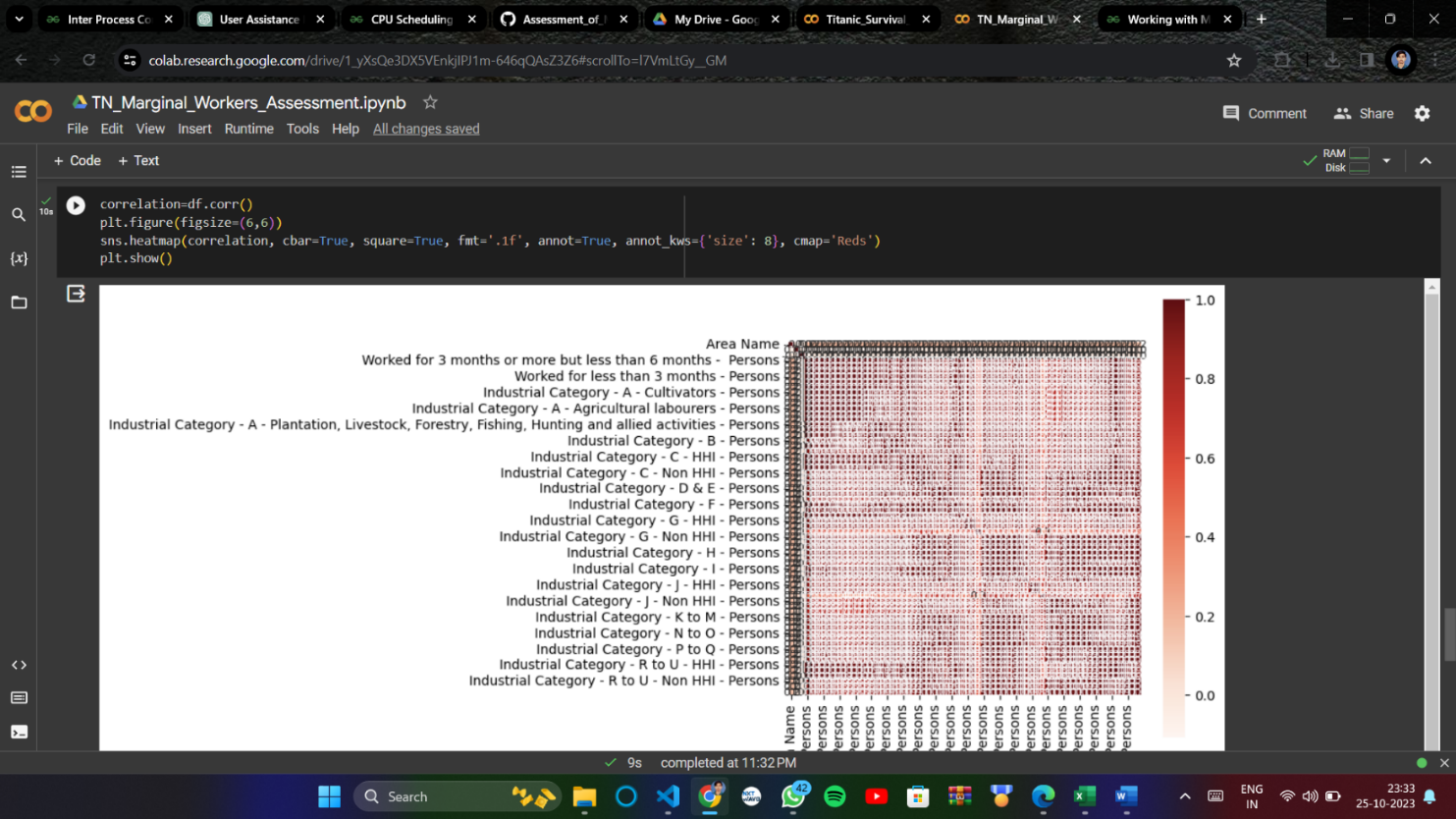
**5. Visualize the dataset:**

correlation=df.corr()

plt.figure(figsize=(6,6))

sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot\_kws={'size': 8}, cmap='Reds')

plt.show()

 The heatmap provides a visual representation of how strongly different columns are correlated. Positive correlations are shown in lighter shades, negative correlations in darker shades, and stronger correlations are indicated by values closer to 1 or -1. The annotations inside the heatmap cells provide the exact correlation values between the corresponding pairs of columns.