**Data Analysis Capstone Labs**

print('There are', df\_survey['Country']**.**nunique(), 'unique countries in the survey:')

print('\nCountry Respondants')

**with** pd**.**option\_context('display.max\_rows', **None**, 'display.max\_columns', **None**):

print(df\_survey['Country']**.**value\_counts())

There are 135 unique countries in the survey:

Country Respondants

United States 3173

India 911

United Kingdom 841

Germany 715

Canada 442

France 339

Brazil 328

Australia 287

Netherlands 259

Spain 257

Russian Federation 211

Poland 205

Italy 188

Sweden 162

Switzerland 151

Ukraine 111

Israel 104

South Africa 104

Turkey 98

Mexico 98

Romania 95

Pakistan 93

Austria 89

Belgium 89

Denmark 85

Norway 84

Iran 83

Bulgaria 77

Greece 76

New Zealand 76

Argentina 73

Finland 72

Czech Republic 72

China 69

Portugal 69

Ireland 67

Hungary 64

Bangladesh 58

Serbia 53

Japan 50

Colombia 50

Nigeria 44

Sri Lanka 43

Malaysia 42

Egypt 39

Indonesia 36

Philippines 36

Slovenia 35

Lithuania 35

Hong Kong (S.A.R.) 34

Singapore 33

Croatia 31

Slovakia 28

Estonia 27

Belarus 24

Taiwan 23

Chile 21

Kenya 20

Latvia 19

Thailand 19

United Arab Emirates 19

Other Country (Not Listed Above) 18

Uruguay 16

Dominican Republic 16

South Korea 15

Costa Rica 15

Nepal 15

Guatemala 13

Armenia 12

Peru 12

Albania 12

Lebanon 12

Saudi Arabia 12

Viet Nam 12

Ecuador 11

Luxembourg 11

Bosnia and Herzegovina 10

Republic of Moldova 10

Tunisia 9

Georgia 9

El Salvador 8

Malta 8

Morocco 8

Jordan 7

The former Yugoslav Republic of Macedonia 7

Cyprus 7

Venezuela, Bolivarian Republic of... 7

Ghana 7

Paraguay 6

Uganda 6

Iceland 6

Honduras 5

Zimbabwe 5

Azerbaijan 5

Côte d'Ivoire 5

Kyrgyzstan 4

Cameroon 4

Bolivia 4

Cuba 4

Ethiopia 4

Nicaragua 3

Myanmar 3

Swaziland 3

Algeria 3

Bahrain 3

Syrian Arab Republic 3

Congo, Republic of the... 2

United Republic of Tanzania 2

Kuwait 2

Montenegro 2

Mauritius 2

Panama 2

Afghanistan 2

Senegal 2

Mongolia 2

Burundi 2

Sudan 2

Uzbekistan 2

Monaco 1

Togo 1

Somalia 1

Cambodia 1

Timor-Leste 1

Republic of Korea 1

Cape Verde 1

Jamaica 1

Mozambique 1

Rwanda 1

Liechtenstein 1

Yemen 1

Libyan Arab Jamahiriya 1

Iraq 1

Qatar 1

Turkmenistan 1

Brunei Darussalam 1

Name: Country, dtype: int64

[](https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

**Collecting Job Data Using APIs**

Estimated time needed: **45 to 60** minutes

**Objectives**

After completing this lab, you will be able to:

* Collect job data from Jobs API
* Store the collected data into an excel spreadsheet.

**Instructions**

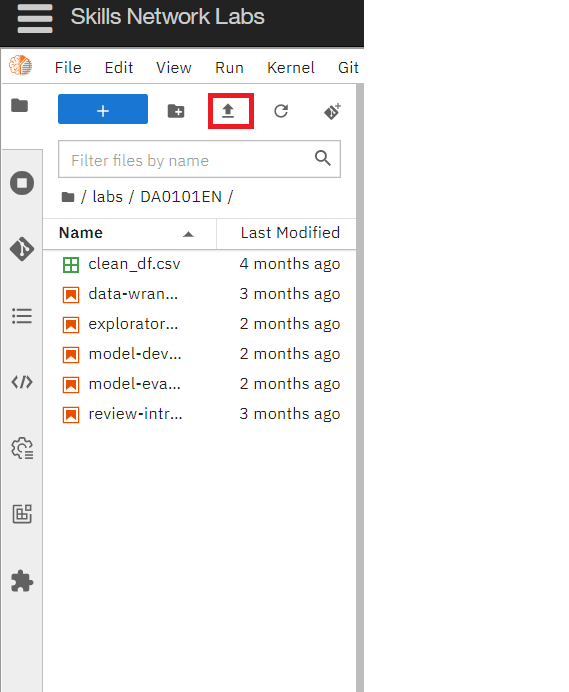
To run the actual lab, firstly you need to click on the [Jobs\_API](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/Jobs_API.ipynb" \t "_blank) notebook link. The file contains flask code which is required to run the Jobs API data.

Now, to run the code in the file that opens up follow the below steps.

Step1: Download the file.

Step2: Upload it on the IBM Watson studio. (If IBM Watson Cloud service does not work in your system, follow the alternate Step 2 below)

Step2(alternate): Upload it in your SN labs environment using the upload button which is highlighted in red in the image below: Remember to upload this Jobs\_API file in the same folder as your current .ipynb file



Step3: Run all the cells of the Jobs\_API file. (Even if you receive an asterik sign after running the last cell, the code works fine.)

If you want to learn more about flask, which is optional, you can click on this link [here](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/FLASK_API.md.html).

Once you run the flask code, you can start with your assignment.

**Dataset Used in this Assignment**

The dataset used in this lab comes from the following source: <https://www.kaggle.com/promptcloud/jobs-on-naukricom> under the under a **Public Domain license**.

Note: We are using a modified subset of that dataset for the lab, so to follow the lab instructions successfully please use the dataset provided with the lab, rather than the dataset from the original source.

The original dataset is a csv. We have converted the csv to json as per the requirement of the lab.

**Warm-Up Exercise**

Before you attempt the actual lab, here is a fully solved warmup exercise that will help you to learn how to access an API.

Using an API, let us find out who currently are on the International Space Station (ISS).  
The API at [http://api.open-notify.org/astros.json](http://api.open-notify.org/astros.json?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork21426264-2021-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBM-DA0321EN-SkillsNetwork-21426264&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newsletter.M12345678&cvo_campaign=000026UJ) gives us the information of astronauts currently on ISS in json format.  
You can read more about this API at [http://open-notify.org/Open-Notify-API/People-In-Space/](http://open-notify.org/Open-Notify-API/People-In-Space?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork21426264-2021-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBM-DA0321EN-SkillsNetwork-21426264&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=email.Newsletter.M12345678&cvo_campaign=000026UJ)

[ ]:

**import** requests *# you need this module to make an API call*

**import** pandas **as** pd

[ ]:

api\_url **=** "http://api.open-notify.org/astros.json" *# this url gives use the astronaut data*

[ ]:

response **=** requests.get(api\_url) *# Call the API using the get method and store the*

*# output of the API call in a variable called response.*

[ ]:

**if** response.ok: *# if all is well() no errors, no network timeouts)*

data **=** response.json() *# store the result in json format in a variable called data*

*# the variable data is of type dictionary.*

[ ]:

​

[ ]:

print(data) *# print the data just to check the output or for debugging*

Print the number of astronauts currently on ISS.

[ ]:

print(data.get('number'))

Print the names of the astronauts currently on ISS.

[ ]:

astronauts **=** data.get('people')

print("There are {} astronauts on ISS".format(len(astronauts)))

print("And their names are :")

**for** astronaut **in** astronauts:

print(astronaut.get('name'))

Hope the warmup was helpful. Good luck with your next lab!

**Lab: Collect Jobs Data using Jobs API**

**Objective: Determine the number of jobs currently open for various technologies and for various locations**

Collect the number of job postings for the following locations using the API:

* Los Angeles
* New York
* San Francisco
* Washington DC
* Seattle
* Austin
* Detroit

[ ]:

*#Import required libraries*

**import** pandas **as** pd

**import** json

**Write a function to get the number of jobs for the Python technology.**

Note: While using the lab you need to pass the **payload** information for the **params** attribute in the form of **key** **value** pairs. Refer the ungraded **rest api lab** in the course **Python for Data Science, AI & Development** [link](https://www.coursera.org/learn/python-for-applied-data-science-ai/ungradedLti/P6sW8/hands-on-lab-access-rest-apis-request-http?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

**The keys in the json are**

* Job Title
* Job Experience Required
* Key Skills
* Role Category
* Location
* Functional Area
* Industry
* Role

You can also view the json file contents from the following [json](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.json" \t "_blank) URL.

[ ]:

api\_url**=**"http://127.0.0.1:5000/data"

**def** get\_number\_of\_jobs\_T(technology):

*#your code goes here*

**return** technology,number\_of\_jobs

Calling the function for Python and checking if it works.

[ ]:

get\_number\_of\_jobs\_T("Python")

**Write a function to find number of jobs in US for a location of your choice**

[ ]:

**def** get\_number\_of\_jobs\_L(location):

*#your coe goes here*

**return** location,number\_of\_jobs

Call the function for Los Angeles and check if it is working.

[ ]:

*#your code goes here*

​

**Store the results in an excel file**

Call the API for all the given technologies above and write the results in an excel spreadsheet.

If you do not know how create excel file using python, double click here for **hints**.

Create a python list of all locations for which you need to find the number of jobs postings.

[ ]:

*#your code goes here*

​

Import libraries required to create excel spreadsheet

[ ]:

*# your code goes here*

Create a workbook and select the active worksheet

[ ]:

*# your code goes here*

Find the number of jobs postings for each of the location in the above list. Write the Location name and the number of jobs postings into the excel spreadsheet.

[ ]:

*#your code goes here*

Save into an excel spreadsheet named 'job-postings.xlsx'.

[ ]:

*#your code goes here*

**In the similar way, you can try for below given technologies and results can be stored in an excel sheet.**

Collect the number of job postings for the following languages using the API:

* C
* C#
* C++
* Java
* JavaScript
* Python
* Scala
* Oracle
* SQL Server
* MySQL Server
* PostgreSQL
* MongoDB

[ ]:

*# your code goes here*

[](https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

# **Web Scraping Lab**

Estimated time needed: **30** minutes

## Objectives

After completing this lab you will be able to:

* Download a webpage using requests module
* Scrape all links from a web page
* Scrape all image urls from a web page
* Scrape data from html tables

## Scrape [www.ibm.com](http://www.ibm.com/)

Import the required modules and functions

[6]:

**from** bs4 **import** BeautifulSoup *# this module helps in web scrapping.*

**import** requests *# this module helps us to download a web page*

Download the contents of the web page

[7]:

url **=** "http://www.ibm.com"

[8]:

*# get the contents of the webpage in text format and store in a variable called data*

data **=** requests.get(url).text

Create a soup object using the class BeautifulSoup

[9]:

soup **=** BeautifulSoup(data,"html5lib") *# create a soup object using the variable 'data'*

---------------------------------------------------------------------------

FeatureNotFound Traceback (most recent call last)

/tmp/ipykernel\_68/1112302729.py in <module>

----> 1 soup = BeautifulSoup(data,"html5lib") # create a soup object using the variable 'data'

~/conda/envs/python/lib/python3.7/site-packages/bs4/\_\_init\_\_.py in \_\_init\_\_(self, markup, features, builder, parse\_only, from\_encoding, exclude\_encodings, element\_classes, \*\*kwargs)

**249** "Couldn't find a tree builder with the features you "

**250** "requested: %s. Do you need to install a parser library?"

--> 251 % ",".join(features))

**252**

**253** # At this point either we have a TreeBuilder instance in

FeatureNotFound: Couldn't find a tree builder with the features you requested: html5lib. Do you need to install a parser library?

Scrape all links

[10]:

**for** link **in** soup.find\_all('a'): *# in html anchor/link is represented by the tag <a>*

print(link.get('href'))

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

/tmp/ipykernel\_68/2914620188.py in <module>

----> 1 for link in soup.find\_all('a'): # in html anchor/link is represented by the tag <a>

**2** print(link.get('href'))

NameError: name 'soup' is not defined

Scrape all images

[11]:

**for** link **in** soup.find\_all('img'):*# in html image is represented by the tag <img>*

print(link.get('src'))

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

/tmp/ipykernel\_68/1129526688.py in <module>

----> 1 for link in soup.find\_all('img'):# in html image is represented by the tag <img>

**2** print(link.get('src'))

NameError: name 'soup' is not defined

## Scrape data from html tables

[ ]:

*#The below url contains a html table with data about colors and color codes.*

[12]:

url **=** "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/HTMLColorCodes.html"

Before proceeding to scrape a web site, you need to examine the contents, and the way data is organized on the website. Open the above url in your browser and check how many rows and columns are there in the color table.

[13]:

*# get the contents of the webpage in text format and store in a variable called data*

data **=** requests.get(url).text

[14]:

soup **=** BeautifulSoup(data,"html5lib")

---------------------------------------------------------------------------

FeatureNotFound Traceback (most recent call last)

/tmp/ipykernel\_68/1869437811.py in <module>

----> 1 soup = BeautifulSoup(data,"html5lib")

~/conda/envs/python/lib/python3.7/site-packages/bs4/\_\_init\_\_.py in \_\_init\_\_(self, markup, features, builder, parse\_only, from\_encoding, exclude\_encodings, element\_classes, \*\*kwargs)

**249** "Couldn't find a tree builder with the features you "

**250** "requested: %s. Do you need to install a parser library?"

--> 251 % ",".join(features))

**252**

**253** # At this point either we have a TreeBuilder instance in

FeatureNotFound: Couldn't find a tree builder with the features you requested: html5lib. Do you need to install a parser library?

[15]:

*#find a html table in the web page*

table **=** soup.find('table') *# in html table is represented by the tag <table>*

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

/tmp/ipykernel\_68/3682931288.py in <module>

**1** #find a html table in the web page

----> 2 table = soup.find('table') # in html table is represented by the tag <table>

NameError: name 'soup' is not defined

[16]:

*#Get all rows from the table*

**for** row **in** table.find\_all('tr'): *# in html table row is represented by the tag <tr>*

*# Get all columns in each row.*

cols **=** row.find\_all('td') *# in html a column is represented by the tag <td>*

color\_name **=** cols[2].getText() *# store the value in column 3 as color\_name*

color\_code **=** cols[3].getText() *# store the value in column 4 as color\_code*

print("{}--->{}".format(color\_name,color\_code))

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

/tmp/ipykernel\_68/2538351921.py in <module>

**1** #Get all rows from the table

----> 2 for row in table.find\_all('tr'): # in html table row is represented by the tag <tr>

**3** # Get all columns in each row.

**4** cols = row.find\_all('td') # in html a column is represented by the tag <td>

**5** color\_name = cols[2].getText() # store the value in column 3 as color\_name

NameError: name 'table' is not defined

[](https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

# **Survey Dataset Exploration Lab**

Estimated time needed: **30** minutes

## Objectives

After completing this lab you will be able to:

* Load the dataset that will used thru the capstone project.
* Explore the dataset.
* Get familier with the data types.

## Load the dataset

Import the required libraries.

[3]:

**import** pandas **as** pd

The dataset is available on the IBM Cloud at the below url.

[ ]:

dataset\_url **=** "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1\_survey\_data.csv"

Load the data available at dataset\_url into a dataframe.

[4]:

*# your code goes here*

## Explore the data set

It is a good idea to print the top 5 rows of the dataset to get a feel of how the dataset will look.

Display the top 5 rows and columns from your dataset.

[14]:

*# your code goes here*

## Find out the number of rows and columns

Start by exploring the numbers of rows and columns of data in the dataset.

Print the number of rows in the dataset.

[9]:

*# your code goes here*

Print the number of columns in the dataset.

[ ]:

*# your code goes here*

## Identify the data types of each column

Explore the dataset and identify the data types of each column.

Print the datatype of all columns.

[10]:

*# your code goes here*

Print the mean age of the survey participants.

[8]:

*# your code goes here*

The dataset is the result of a world wide survey. Print how many unique countries are there in the Country column.

[ ]:

*# your code goes here*

# **Survey Dataset Exploration Lab**

Estimated time needed: **30** minutes

## Objectives

After completing this lab you will be able to:

* Load the dataset that will used thru the capstone project.
* Explore the dataset.
* Get familier with the data types.

## Load the dataset

Import the required libraries.

In [1]:

**import** pandas **as** pd

The dataset is available on the IBM Cloud at the below url.

In [2]:

dataset\_url **=** "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1\_survey\_data.csv"

Load the data available at dataset\_url into a dataframe.

In [4]:

*# your code goes here*

df\_survey **=** pd**.**read\_csv(dataset\_url)

## Explore the data set

It is a good idea to print the top 5 rows of the dataset to get a feel of how the dataset will look.

Display the top 5 rows and columns from your dataset.

In [5]:

*# your code goes here*

df\_survey**.**head()

Out[5]:

|  | **Respondent** | **MainBranch** | **Hobbyist** | **OpenSourcer** | **OpenSource** | **Employment** | **Country** | **Student** | **EdLevel** | **UndergradMajor** | **...** | **WelcomeChange** | **SONewContent** | **Age** | **Gender** | **Trans** | **Sexuality** | **Ethnicity** | **Dependents** | **SurveyLength** | **SurveyEase** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 4 | I am a developer by profession | No | Never | The quality of OSS and closed source software ... | Employed full-time | United States | No | Bachelor’s degree (BA, BS, B.Eng., etc.) | Computer science, computer engineering, or sof... | ... | Just as welcome now as I felt last year | Tech articles written by other developers;Indu... | 22.0 | Man | No | Straight / Heterosexual | White or of European descent | No | Appropriate in length | Easy |
| **1** | 9 | I am a developer by profession | Yes | Once a month or more often | The quality of OSS and closed source software ... | Employed full-time | New Zealand | No | Some college/university study without earning ... | Computer science, computer engineering, or sof... | ... | Just as welcome now as I felt last year | NaN | 23.0 | Man | No | Bisexual | White or of European descent | No | Appropriate in length | Neither easy nor difficult |
| **2** | 13 | I am a developer by profession | Yes | Less than once a month but more than once per ... | OSS is, on average, of HIGHER quality than pro... | Employed full-time | United States | No | Master’s degree (MA, MS, M.Eng., MBA, etc.) | Computer science, computer engineering, or sof... | ... | Somewhat more welcome now than last year | Tech articles written by other developers;Cour... | 28.0 | Man | No | Straight / Heterosexual | White or of European descent | Yes | Appropriate in length | Easy |
| **3** | 16 | I am a developer by profession | Yes | Never | The quality of OSS and closed source software ... | Employed full-time | United Kingdom | No | Master’s degree (MA, MS, M.Eng., MBA, etc.) | NaN | ... | Just as welcome now as I felt last year | Tech articles written by other developers;Indu... | 26.0 | Man | No | Straight / Heterosexual | White or of European descent | No | Appropriate in length | Neither easy nor difficult |
| **4** | 17 | I am a developer by profession | Yes | Less than once a month but more than once per ... | The quality of OSS and closed source software ... | Employed full-time | Australia | No | Bachelor’s degree (BA, BS, B.Eng., etc.) | Computer science, computer engineering, or sof... | ... | Just as welcome now as I felt last year | Tech articles written by other developers;Indu... | 29.0 | Man | No | Straight / Heterosexual | Hispanic or Latino/Latina;Multiracial | No | Appropriate in length | Easy |

5 rows × 85 columns

## Find out the number of rows and columns

Start by exploring the numbers of rows and columns of data in the dataset.

Print the number of rows in the dataset.

In [14]:

*# your code goes here*

print('The number of rows in the dataframe is ' **+** str(df\_survey**.**shape[0]) **+**'.')

The number of rows in the dataframe is 11552.

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

<ipython-input-14-94ead6c633d8> in <module>

**1** # your code goes here

**2** print('The number of rows in the dataframe is ' + str(df\_survey.shape[0]) +'.')

----> 3 rows(df\_survey)

NameError: name 'rows' is not defined

Print the number of columns in the dataset.

In [15]:

*# your code goes here*

print('The number of columns in the dataframe is ' **+** str(df\_survey**.**shape[1]) **+**'.')

The number of columns in the dataframe is 85.

## Identify the data types of each column

Explore the dataset and identify the data types of each column.

Print the datatype of all columns.

In [36]:

*# your code goes here*

*# option 1 (doesn't show all columns):*

*#df\_survey.dtypes*

*# option 2 (prints more info than needed):*

*#df\_survey.info(verbose=True)*

*# option 3 (works, but wordy)*

**with** pd**.**option\_context('display.max\_rows', **None**, 'display.max\_columns', **None**):

print(df\_survey**.**dtypes)

Respondent int64

MainBranch object

Hobbyist object

OpenSourcer object

OpenSource object

Employment object

Country object

Student object

EdLevel object

UndergradMajor object

EduOther object

OrgSize object

DevType object

YearsCode object

Age1stCode object

YearsCodePro object

CareerSat object

JobSat object

MgrIdiot object

MgrMoney object

MgrWant object

JobSeek object

LastHireDate object

LastInt object

FizzBuzz object

JobFactors object

ResumeUpdate object

CurrencySymbol object

CurrencyDesc object

CompTotal float64

CompFreq object

ConvertedComp float64

WorkWeekHrs float64

WorkPlan object

WorkChallenge object

WorkRemote object

WorkLoc object

ImpSyn object

CodeRev object

CodeRevHrs float64

UnitTests object

PurchaseHow object

PurchaseWhat object

LanguageWorkedWith object

LanguageDesireNextYear object

DatabaseWorkedWith object

DatabaseDesireNextYear object

PlatformWorkedWith object

PlatformDesireNextYear object

WebFrameWorkedWith object

WebFrameDesireNextYear object

MiscTechWorkedWith object

MiscTechDesireNextYear object

DevEnviron object

OpSys object

Containers object

BlockchainOrg object

BlockchainIs object

BetterLife object

ITperson object

OffOn object

SocialMedia object

Extraversion object

ScreenName object

SOVisit1st object

SOVisitFreq object

SOVisitTo object

SOFindAnswer object

SOTimeSaved object

SOHowMuchTime object

SOAccount object

SOPartFreq object

SOJobs object

EntTeams object

SOComm object

WelcomeChange object

SONewContent object

Age float64

Gender object

Trans object

Sexuality object

Ethnicity object

Dependents object

SurveyLength object

SurveyEase object

dtype: object

Print the mean age of the survey participants.

In [42]:

*# your code goes here*

print('The mean age of the survey participants is ', df\_survey['Age']**.**mean()**.**round(1))

The mean age of the survey participants is {} 30.8

The dataset is the result of a world wide survey. Print how many unique countries are there in the Country column.

In [54]:

*# your code goes here*

print('There are', df\_survey['Country']**.**nunique(), 'unique countries in the survey:')

print('\nCountry Respondants')

**with** pd**.**option\_context('display.max\_rows', **None**, 'display.max\_columns', **None**):

print(df\_survey['Country']**.**value\_counts())

There are 135 unique countries in the survey:

Country Respondants

United States 3173

India 911

United Kingdom 841

Germany 715

Canada 442

France 339

Brazil 328

Australia 287

Netherlands 259

Spain 257

Russian Federation 211

Poland 205

Italy 188

Sweden 162

Switzerland 151

Ukraine 111

Israel 104

South Africa 104

Turkey 98

Mexico 98

Romania 95

Pakistan 93

Austria 89

Belgium 89

Denmark 85

Norway 84

Iran 83

Bulgaria 77

Greece 76

New Zealand 76

Argentina 73

Finland 72

Czech Republic 72

China 69

Portugal 69

Ireland 67

Hungary 64

Bangladesh 58

Serbia 53

Japan 50

Colombia 50

Nigeria 44

Sri Lanka 43

Malaysia 42

Egypt 39

Indonesia 36

Philippines 36

Slovenia 35

Lithuania 35

Hong Kong (S.A.R.) 34

Singapore 33

Croatia 31

Slovakia 28

Estonia 27

Belarus 24

Taiwan 23

Chile 21

Kenya 20

Latvia 19

Thailand 19

United Arab Emirates 19

Other Country (Not Listed Above) 18

Uruguay 16

Dominican Republic 16

South Korea 15

Costa Rica 15

Nepal 15

Guatemala 13

Armenia 12

Peru 12

Albania 12

Lebanon 12

Saudi Arabia 12

Viet Nam 12

Ecuador 11

Luxembourg 11

Bosnia and Herzegovina 10

Republic of Moldova 10

Tunisia 9

Georgia 9

El Salvador 8

Malta 8

Morocco 8

Jordan 7

The former Yugoslav Republic of Macedonia 7

Cyprus 7

Venezuela, Bolivarian Republic of... 7

Ghana 7

Paraguay 6

Uganda 6

Iceland 6

Honduras 5

Zimbabwe 5

Azerbaijan 5

Côte d'Ivoire 5

Kyrgyzstan 4

Cameroon 4

Bolivia 4

Cuba 4

Ethiopia 4

Nicaragua 3

Myanmar 3

Swaziland 3

Algeria 3

Bahrain 3

Syrian Arab Republic 3

Congo, Republic of the... 2

United Republic of Tanzania 2

Kuwait 2

Montenegro 2

Mauritius 2

Panama 2

Afghanistan 2

Senegal 2

Mongolia 2

Burundi 2

Sudan 2

Uzbekistan 2

Monaco 1

Togo 1

Somalia 1

Cambodia 1

Timor-Leste 1

Republic of Korea 1

Cape Verde 1

Jamaica 1

Mozambique 1

Rwanda 1

Liechtenstein 1

Yemen 1

Libyan Arab Jamahiriya 1

Iraq 1

Qatar 1

Turkmenistan 1

Brunei Darussalam 1

Name: Country, dtype: int64

[**Coursera Capstone Project W2 - Data Wrangling Lab.ipynb**](https://gist.github.com/edownin1/64eb2798eaefc49a03593b1eeb70877c#file-coursera-capstone-project-w2-data-wrangling-lab-ipynb)

[Raw](https://gist.github.com/edownin1/64eb2798eaefc49a03593b1eeb70877c/raw/90db4d00a5af17bebb2780949531daaf44697654/Coursera%2520Capstone%2520Project%2520W2L1%2520-%2520Data%2520Wrangling%2520Lab.ipynb)

[**Coursera Capstone Project W2L1 - Data Wrangling Lab.ipynb**](https://gist.github.com/edownin1/64eb2798eaefc49a03593b1eeb70877c#file-coursera-capstone-project-w2l1-data-wrangling-lab-ipynb)

# **Data Wrangling Lab**

Estimated time needed: **45 to 60** minutes

In this assignment you will be performing data wrangling.

## Objectives

In this lab you will perform the following:

* Identify duplicate values in the dataset.
* Remove duplicate values from the dataset.
* Identify missing values in the dataset.
* Impute the missing values in the dataset.
* Normalize data in the dataset.

## Hands on Lab

Import pandas module.

In [1]:

**import** pandas **as** pd

Load the dataset into a dataframe.

In [2]:

df **=** pd**.**read\_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1\_survey\_data.csv")

## Finding duplicates

In this section you will identify duplicate values in the dataset.

Find how many duplicate rows exist in the dataframe.

In [3]:

*# your code goes here*

print(df**.**duplicated()**.**sum())

154

## Removing duplicates

Remove the duplicate rows from the dataframe.

In [4]:

*# your code goes here*

df**.**drop\_duplicates(inplace **=** **True**)

Verify if duplicates were actually dropped.

In [5]:

*# your code goes here*

print(df**.**duplicated()**.**value\_counts())

False 11398

dtype: int64

## Finding Missing values

Find the missing values for all columns.

In [6]:

*# your code goes here*

df**.**columns[df**.**isnull()**.**any()]

Out[6]:

Index(['OpenSource', 'Student', 'EdLevel', 'UndergradMajor', 'EduOther',

'OrgSize', 'DevType', 'YearsCode', 'Age1stCode', 'YearsCodePro',

'JobSat', 'MgrIdiot', 'MgrMoney', 'MgrWant', 'LastInt', 'FizzBuzz',

'JobFactors', 'ResumeUpdate', 'CompTotal', 'CompFreq', 'ConvertedComp',

'WorkWeekHrs', 'WorkPlan', 'WorkChallenge', 'WorkRemote', 'WorkLoc',

'ImpSyn', 'CodeRev', 'CodeRevHrs', 'UnitTests', 'PurchaseHow',

'PurchaseWhat', 'LanguageWorkedWith', 'LanguageDesireNextYear',

'DatabaseWorkedWith', 'DatabaseDesireNextYear', 'PlatformWorkedWith',

'PlatformDesireNextYear', 'WebFrameWorkedWith',

'WebFrameDesireNextYear', 'MiscTechWorkedWith',

'MiscTechDesireNextYear', 'DevEnviron', 'OpSys', 'Containers',

'BlockchainOrg', 'BlockchainIs', 'BetterLife', 'ITperson', 'OffOn',

'SocialMedia', 'Extraversion', 'ScreenName', 'SOVisit1st',

'SOVisitFreq', 'SOVisitTo', 'SOFindAnswer', 'SOTimeSaved',

'SOHowMuchTime', 'SOAccount', 'SOPartFreq', 'SOJobs', 'EntTeams',

'WelcomeChange', 'SONewContent', 'Age', 'Gender', 'Trans', 'Sexuality',

'Ethnicity', 'Dependents', 'SurveyLength', 'SurveyEase'],

dtype='object')

Find out how many rows are missing in the column 'WorkLoc'

In [7]:

*# your code goes here*

df['WorkLoc']**.**isna()**.**sum()

Out[7]:

32

## Imputing missing values

Find the value counts for the column WorkLoc.

In [8]:

*# your code goes here*

df['WorkLoc']**.**value\_counts()

Out[8]:

Office 6806

Home 3589

Other place, such as a coworking space or cafe 971

Name: WorkLoc, dtype: int64

Identify the value that is most frequent (majority) in the WorkLoc column.

In [9]:

*#make a note of the majority value here, for future reference*

majority **=** df['WorkLoc']**.**value\_counts()**.**idxmax()

print(majority)

Office

Impute (replace) all the empty rows in the column WorkLoc with the value that you have identified as majority.

In [10]:

*# your code goes here*

df['WorkLoc']**.**fillna('Office', inplace**=True**)

After imputation there should ideally not be any empty rows in the WorkLoc column.

Verify if imputing was successful.

In [11]:

*# your code goes here*

df['WorkLoc']**.**isna()**.**sum()

Out[11]:

0

## Normalizing data

There are two columns in the dataset that talk about compensation.

One is "CompFreq". This column shows how often a developer is paid (Yearly, Monthly, Weekly).

The other is "CompTotal". This column talks about how much the developer is paid per Year, Month, or Week depending upon his/her "CompFreq".

This makes it difficult to compare the total compensation of the developers.

In this section you will create a new column called 'NormalizedAnnualCompensation' which contains the 'Annual Compensation' irrespective of the 'CompFreq'.

Once this column is ready, it makes comparison of salaries easy.

List out the various categories in the column 'CompFreq'

In [12]:

*# your code goes here*

Out[12]:

Index(['Yearly', 'Monthly', 'Weekly'], dtype='object')

Create a new column named 'NormalizedAnnualCompensation'. Use the hint given below if needed.

Double click to see the **Hint**.

In [13]:

*# your code goes here*

|  | **NormalizedAnnualCompensation** |
| --- | --- |
| **0** | 61000.0 |
| **1** | 138000.0 |
| **2** | 90000.0 |
| **3** | 348000.0 |
| **4** | 90000.0 |
| **...** | ... |
| **11547** | 130000.0 |
| **11548** | 74400.0 |
| **11549** | 105000.0 |
| **11550** | 80000.0 |
| **11551** | NaN |

11398 rows × 1 columns

# Lab: Explore your dataset using SQL queries using Datasette

**Lab Overview:**  
Now that you have learned how querying techniques can help you to explore and analyze your data, you will run some basic SQL queries on the data you loaded into your database instance.

**Estimated time needed:** 15 minutes

## Objectives

After completing this lab, you will be able to:

* Execute basic SQL queries on the data you loaded into your database.

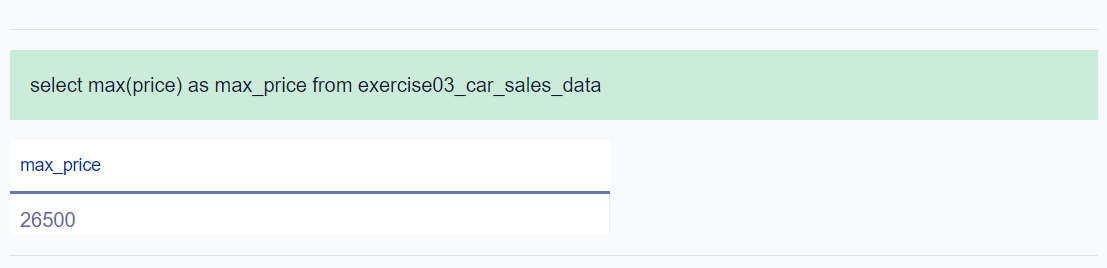
## Database Used in this Lab

Here, we are making use of an internal preloaded **CarSalesdata** database that contains the **exercise03\_car\_sales\_data table**.

## Exercise 1: Execute basic SQL queries on the data you loaded into your database.

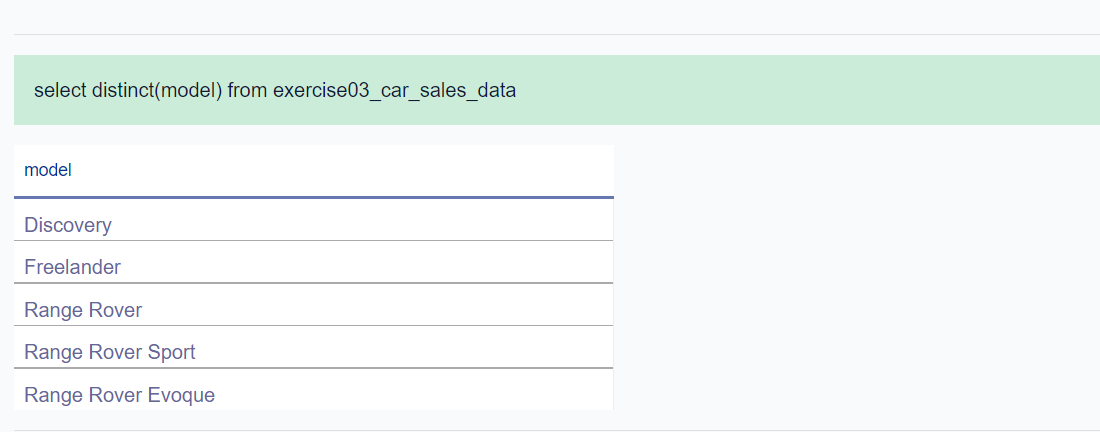
1. Execute the following query to check the maximum price.
   1. 1
   2. select max(price) as max\_price from exercise03\_car\_sales\_data

Copied!



1. Execute the following query to display the distinct models.
   1. 1
   2. select distinct(model) from exercise03\_car\_sales\_data;

Copied!



### Congratulations! You have successfully executed SQL queries on your loaded data