Project Diary

Titanic Dataset and COVID-19

Elnaz Dehkharghani
MSc. Big Data and Business Analytics

Prof. Dr. Frank Schulz

SRH University Heidelberg
13th November, 2020

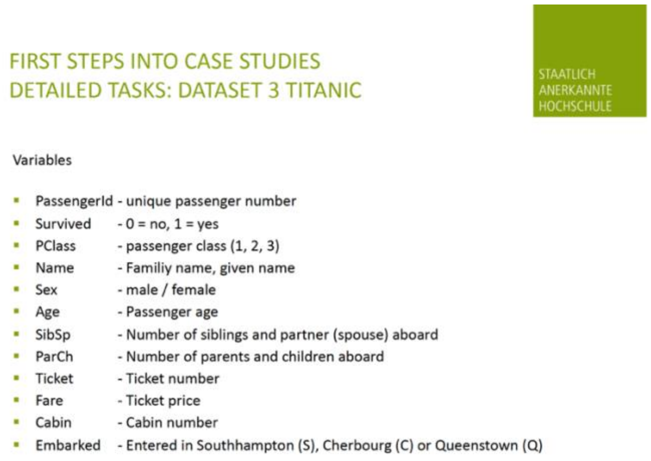
Titanic Data Preparation and Analysis
Cleaning and Integration
By
OpenRefine

Understanding and installation of OpenRefine:

The aim of this project is to work on messy data and try to clean it. For this purpose, I chose OpenRefine as my data cleaning tool. Till now, I did not have experience in OpenRefine and Trifacta for data cleaning process but for this project I tried it and found them as very useful and interactive tool for data cleaning purpose. I usually had to cleaned my data by help of some libraries in Python programming languages. Therefor first of all, I needed to carefully study and search about this OpenRefine. There were 3 videos for watching on the OpenRefine website about Exploring on Data, Cleaning and Transforming Data, Reconciling, and Matching Data which I in addition to reading the OpenRefine documentation referred to them. I also study more in another source provided by “University of Idaho Library Digital Initiatives” as complementary source. After good understanding the tool and its usage, I downlead and installed the latest version of it, 3.4.1 Mac Kit which does not need to install java separately. By clicking the OpenRefine app it will automatically open in: http://127.0.0.1:3333/

Understanding the Titanic Data Set:

The titanic data set has been downloaded directly from the Microsoft Team provided by Prof. Dr. Frank Schulz. The detailed information about data set is as below, having a good knowledge of dataset help to better clean the data:



FIRST STEPS INTO CASE STUDIES
DETAILED TASKS: DATASET 3 TITANIC

STAATLICH
ANERKANNTE
HOCHSCHULE

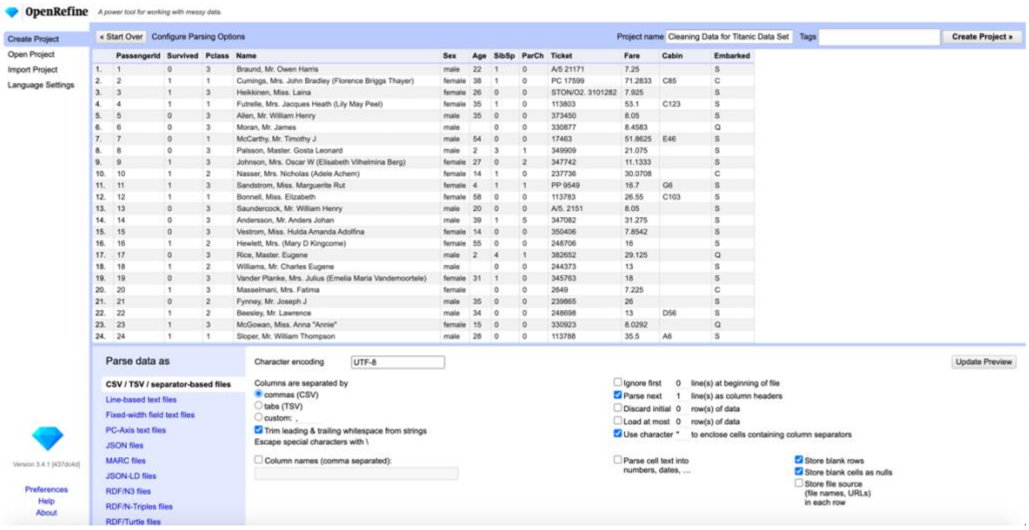
Variables

PassengerId	- unique passenger number
Survived	- 0 = no, 1 = yes
PClass	- passenger class (1, 2, 3)
Name	- Familiy name, given name
Sex	- male / female
Age	- Passenger age
SibSp	- Number of siblings and partner (spouse) aboard
ParCh	- Number of parents and children aboard
Ticket	- Ticket number
Fare	- Ticket price
Cabin	- Cabin number
Embarked	- Entered in Southhampton (S), Cherbourg (C) or Queenstown (Q)

Importing the titanic data set in OpenRefine:

For importing the data, I need to Create Project. Since my data set is already downloaded and placed in my local computer, I chose “This Computer, Choose File (dataset_titanic.csv)” and pressed the Next button. Now data is uploaded in OpenRefine. In case, I wanted to work with a raw data link grabbed from GitHub I could easily do by putting the link in Web addresses (URLs) part.

Now there is a preview of my dataset. Since there are some special characters in the Passenger Names, it is very important to choose Character encoding in a correct way. (I set it as UTF-8)



Project name: Cleaning Data for Titanic Data Set
Tags:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
1.	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2.	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3.	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4.	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5.	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6.	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7.	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8.	8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9.	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10.	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11.	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12.	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13.	13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
14.	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15.	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
16.	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17.	17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18.	18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19.	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
20.	20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
21.	21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
22.	22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
23.	23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292		Q
24.	24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5	A6	S

Parse data as

- CSV / TSV / separator-based files
- Line-based text files
- Fixed-width field text files
- PC-Axis text files
- JSON files
- MARC files
- JSON-LD files
- RDF/N3 files
- RDF/N-Triples files
- RDF/Turtle files

Character encoding: UTF-8

Columns are separated by
● commas (CSV)
○ tabs (TSV)
○ custom: ,
☑ Trim leading & trailing whitespace from strings
Escape special characters with \
□ Column names (comma separated):

□ Ignore first 0 line(s) at beginning of file
☑ Parse next 1 line(s) as column headers
□ Discard initial 0 row(s) of data
□ Load at most 0 row(s) of data
☑ Use character " to enclose cells containing column separators

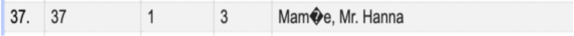
□ Parse cell text into numbers, dates, ...

☑ Store blank rows
☑ Store blank cells as nulls
□ Store file source (file names, URLs) in each row

After checking all columns and inserted data under them, set a Project Name for the project and press Create Project to start cleaning data.

Messy data in Passenger Names:

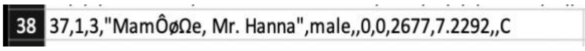
When I imported the data set in OpenRefine, I encountered with a problem in the 37th row and some other rows in the preview step. (a black question marke)



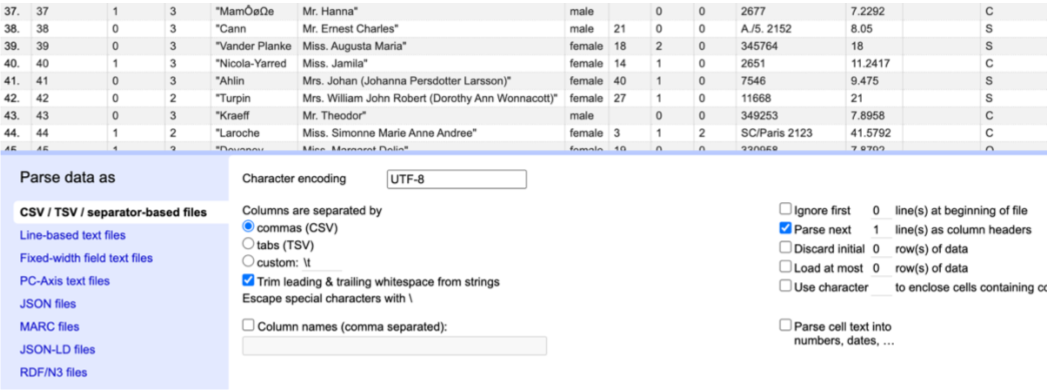
37.	37	1	3	Mam�e, Mr. Hanna

1. I chose UTF-8 as the character encoding which is the most widely used encoding in WWW, but no changes happened and the problem remained as before.
2. I tried to select other Character encoding in both “Common Encodings” and “All Encodings” list but again nothing happened. The ISO-8859 (Latin-1) which also defines 256 characters for western Europe languages did not solve this problem also. I checked these steps for any changes in both preview and complete creation project.

3. I thought maybe problem is with original CSV file’s character encoding. So, I changed the dataset_titanic.csv character encoding to UTF-8 CSV, but this time instead of question mark, some unsense characters appeared.



```
38 37,1,3,"MamÔøΩe, Mr. Hanna",male,,0,0,2677,7.2292,,C
```



37.	37	1	3	"MamÔøΩe	Mr. Hanna"	male		0	0	2677	7.2292		C
38.	38	0	3	"Cann	Mr. Ernest Charles"	male	21	0	0	A./5. 2152	8.05		S
39.	39	0	3	"Vander Planke	Miss. Augusta Maria"	female	18	2	0	345764	18		S
40.	40	1	3	"Nicola-Yarred	Miss. Jamila"	female	14	1	0	2651	11.2417		C
41.	41	0	3	"Ahlin	Mrs. Johan (Johanna Persdotter Larsson)"	female	40	1	0	7546	9.475		S
42.	42	0	2	"Turpin	Mrs. William John Robert (Dorothy Ann Wonnacott)"	female	27	1	0	11668	21		S
43.	43	0	3	"Kraeff	Mr. Theodor"	male		0	0	349253	7.8958		C
44.	44	1	2	"Laroche	Miss. Simonne Marie Anne Andree"	female	3	1	2	SC/Paris 2123	41.5792		C

Parse data as

Character encoding UTF-8

CSV / TSV / separator-based files
Line-based text files
Fixed-width field text files
PC-Axis text files
JSON files
MARC files
JSON-LD files
RDF/N3 files

Columns are separated by
● commas (CSV)
○ tabs (TSV)
○ custom: \t
☑ Trim leading & trailing whitespace from strings
Escape special characters with \

□ Column names (comma separated):

□ Ignore first 0 line(s) at beginning of file
☑ Parse next 1 line(s) as column headers
□ Discard initial 0 row(s) of data
□ Load at most 0 row(s) of data
□ Use character to enclose cells containing co

□ Parse cell text into numbers, dates, ...

4. At the end I understood this problem cannot be solved easily by character encoding settings as there is serious and obvious problem in the CSV file. (Maybe these are Noisy Data!)

   So, I tried to solve this problem on data by one of the OpenRefine functions:

   ```
   Column Name -> Edit Cells -> Transform -> value. replace(/[^\u0020-\u007F]/,"")
   ```

   7 rows have been affected and then by finding their real names in encycolopedia-titanica manually edited them in their cells.

   Row 37: Changed to Māmā, Mr. Hannā.
   Row 131: Changed to Mr Jozef Draženović
   Row 177: Changed to Lefbre, Mr. Henry Forbes
   Row 405: Changed to Oreśković, Miss. Marija
   Row 927: Changed to Katavelos, Mr. Vasilios
   Row 937: Changed to Peltomäki, Mr. Nikolai Johannes
   Row 938: Changed to Chevré, Mr. Paul Romaine

5. If there were web scape characters I could use: `value. unescape("url"),` this function give the original character.

   Also, there were some other issues in the column Name with misspelling like, Master, Sir, Mme, Mlle,… that they replaced with the original ones. A useful function, to replace these words was as example:

   ```
   value.replace("Master", "Mr")

   value.replace("Goldsmith","GoldsmithSTH").Replace("Mr","Mrs"))
   ```

   These functions used in the Expression part of the edit cells, transforms. Also, some duplicated in the names found with the Facet, Customized Facet, Duplicates facets. I checked these names one by one and compare the data with encycolopedia-titanica. If two people were same, I deleted them and if by the mistake their name were same I edited them manually and finally if there were completely two different people I remained them.

   The other good tool for finding the same items in the Name column was clustering. This tool with some algorithms and methods suggested me some names that were a little similar. By browsing each cluster, I decided to whether keep or merge the names together. By merging them no changes in the number of rows happened and only they took their same values. So, I decided to after merging delete them to do not have any duplicates in my rows , to avoid for any bias. Those clusters which were clearly different person kept as before.

After finishing cleaning work with Name column I decided to separate the title of names in the different column named as title. So I used value.split(“,”)[1] on the column name to first split the titles and family names in a new column named Title_Surname. And then by value.split[‘.’][0] separate titles from the family name on the same column. After finishing the job I changed the column name to Title.

In the column Title I have only titles as Mr, Mrs, Miss, Other that are a categorical data and should map to the numbers for further machine learning process. Again with the help of replace function I changed all of them to numbers.

Title: {‘Mr’:0, ‘Mrs’:1, ‘Miss’:2, ‘Other’:3}

Faceting:

- **Survived:** Facet, Text Facet: (with error, only 0 and 1 possible)

  `0:585, 1: 362, 2:1, 3: 2` I have to deal with these 2 and 3!

Options:

1. Remove the rows: Because there are only 3 value with errors it is possible to remove them, and still in the statistical view it can be valid.
2. Ignoring
3. Use mode 0 as it is appearing most of the time.
4. Check with external data source encycolopedia-titanica and change the survived part of them.

Part a)

For the Miss. Margaret only the survived part was incorrect that by checking to the encycolopedia-titanica edited to 1.

Part b)

For the 3s, it seems that the survived part was missed and as a result all of the other information are shifted to the left.

Options:

1. Manually edit them by changing the original csv to shift them to the right.
2. look at the external source in encycolopedia-titanica and edit them in encyclopedia. (I preferred this option and explained it in my project report)
3. Or simply remove these two rows.

But how to remove rows?

```
Put a star -> All-> Facet-> Facet by star ->include the true
All-> Edit rows-> remove matching rows.
```

- **Age:**

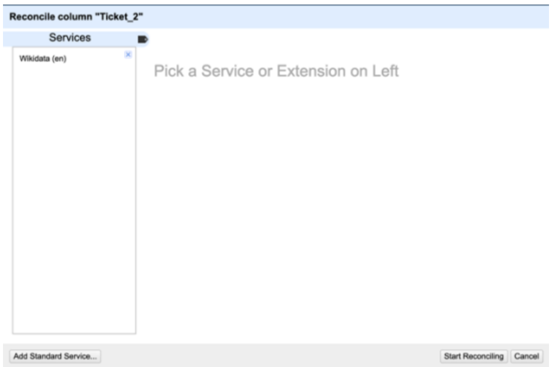
```
Age -> Facet -> Numeric Facet -> (Range: 0-81)
```

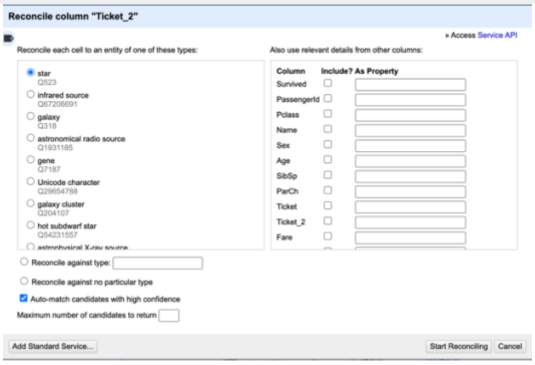
There is not any outlier here but the number of blank cells are too much. Removing the blank cells will not be a good approach. So, there is a need to fill them. The null value can be filled with the help of an external source like wikidata by reconciliation feature in OpenRefine or by mean value of passengers ages. It can be done by making average on whole passengers, or group them in two different groups of survived and not survived or either group them in three different passenger classes. Here I grouped the rows by their survived or not survived and after that found the average ages of two different groups and then replace them with the null values. Grouping the records is done by grouping the same value of rows and this is possible for the only first column.

- **Ticket:**

```
Facet, Text Facet: (with error)
```

There are large number of tickets which are the same or the ticket numbers are in both character and numbers. These cannot be easily handled manually so I needed to use an external source for reconciling.
I created another column just left to the ticket column base on it and named it to Ticket_2. Then chose reconcile, start reconciling. And then chose the wikidata.





Reconcile each cell to an entity of one of these types:

● star Q523
○ infrared source Q67206691
○ galaxy Q318
○ astronomical radio source Q1931185
○ gene Q7187
○ Unicode character Q29654788
○ galaxy cluster Q204107
○ hot subdwarf star Q54231557
○ astrophysical X-ray source

○ Reconcile against type:

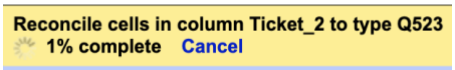
○ Reconcile against no particular type

☑ Auto-match candidates with high confidence

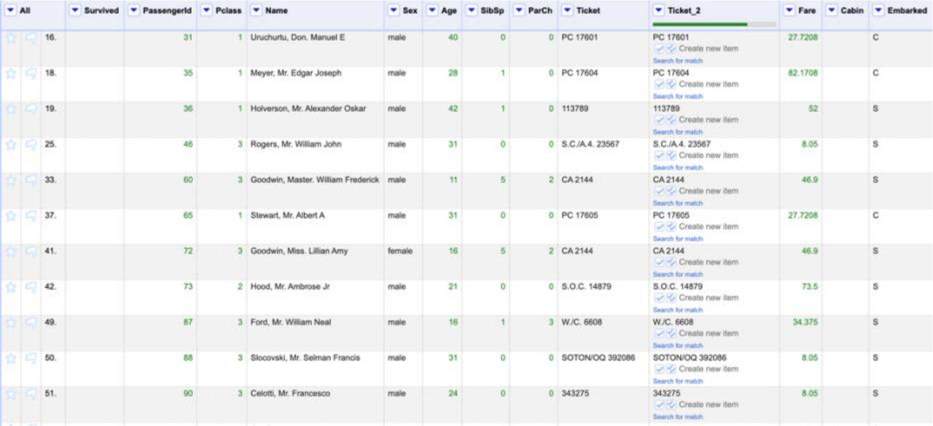
Maximum number of candidates to return

Also use relevant details from other columns:

Column	Include?	As Property
Survived	□	
PassengerId	□	
Pclass	□	
Name	□	
Sex	□	
Age	□	
SibSp	□	
ParCh	□	
Ticket	□	
Ticket_2	□	
Fare	□	


Reconcile cells in column Ticket_2 to type Q523
1% complete Cancel

After that chose Match each cell to the best candidate.



All	Survived	PassengerId	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Ticket_2	Fare	Cabin	Embarked
16.		31	1	Uruchurtu, Don. Manuel E	male	40	0	0	PC 17601	PC 17601 Create new item Search for match	27.7208		C
18.		35	1	Meyer, Mr. Edgar Joseph	male	28	1	0	PC 17604	PC 17604 Create new item Search for match	82.1708		C
19.		36	1	Holverson, Mr. Alexander Oskar	male	42	1	0	113789	113789 Create new item Search for match	52		S
25.		46	3	Rogers, Mr. William John	male	31	0	0	S.C./A.4. 23567	S.C./A.4. 23567 Create new item Search for match	8.05		S
33.		60	3	Goodwin, Master. William Frederick	male	11	5	2	CA 2144	CA 2144 Create new item Search for match	46.9		S
37.		65	1	Stewart, Mr. Albert A	male	31	0	0	PC 17605	PC 17605 Create new item Search for match	27.7208		C
41.		72	3	Goodwin, Miss. Lillian Amy	female	16	5	2	CA 2144	CA 2144 Create new item Search for match	46.9		S
42.		73	2	Hood, Mr. Ambrose Jr	male	21	0	0	S.O.C. 14879	S.O.C. 14879 Create new item Search for match	73.5		S
49.		87	3	Ford, Mr. William Neal	male	16	1	3	W./C. 6608	W./C. 6608 Create new item Search for match	34.375		S
50.		88	3	Slocovski, Mr. Selman Francis	male	31	0	0	SOTON/OQ 392086	SOTON/OQ 392086 Create new item Search for match	8.05		S
51.		90	3	Celotti, Mr. Francesco	male	24	0	0	343275	343275 Create new item Search for match	8.05		S

I can do the same process for cabin column as well. But I think this cabin number and ticket number do not play any important role for further analysis so I just ignore them for now.

But there is a unique character at the beginning of each cabin number which maybe can be useful in future analysis, but by the now I preferred to totally drop both Ticket and Cabin column at the end of my work.

- **Embarked:**
  There were only two missing data here, which by encycolopedia-titanica edited in a correct form. At the end of the project with the replace function I mapped the categorical embarked values to the numerical values.

  Embarked: {‘C’:0, ‘Q’:1, ‘S’:2}

- **Sex:**

  For this column also at the end of cleaning and dealing with null values I mapped these categorical values to numbers.

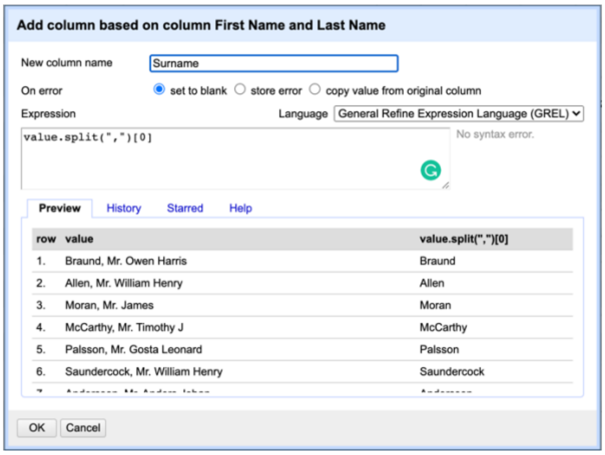
  Sex: {‘male’: 0, ‘female’:1}

- **SibSp and ParCh:**

  There was not any issue here, I only added another column name FamilySize and summed up the value of these two columns for new FamilySize column.

Splitting the column:
Edit column, split into several columns, then in the dialogue box should chose the separator, and chose the split into number.

Creating a new column named:
Surname
`First Name and Last Name Column -> Edit Column -> add column based on column -> in the expression part wrote:  value.split(",")[0]`



Add column based on column First Name and Last Name

New column name: Surname

On error: ● set to blank ○ store error ○ copy value from original column

Expression — Language: General Refine Expression Language (GREL)

```
value.split(",")[0]
```

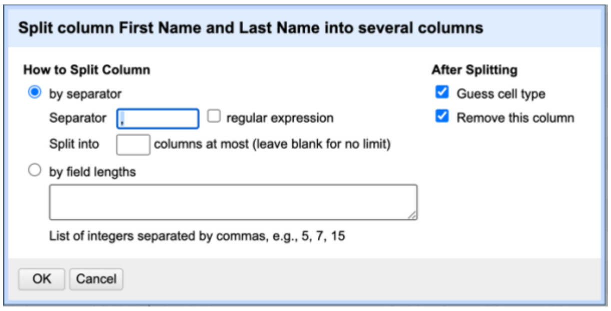
No syntax error.

row	value	value.split(",")[0]
1.	Braund, Mr. Owen Harris	Braund
2.	Allen, Mr. William Henry	Allen
3.	Moran, Mr. James	Moran
4.	McCarthy, Mr. Timothy J	McCarthy
5.	Palsson, Mr. Gosta Leonard	Palsson
6.	Saundercock, Mr. William Henry	Saundercock

Or, creating two separate columns for each:
Name and Surname:

`First Name and Last Name -> Edit Column -> split into several columns`

By choosing separator as, two separated columns one for Name and the other for family name is created.



Split column First Name and Last Name into several columns

How to Split Column

● by separator

Separator: , □ regular expression

Split into: columns at most (leave blank for no limit)

○ by field lengths

List of integers separated by commas, e.g., 5, 7, 15

After Splitting

☑ Guess cell type

☑ Remove this column

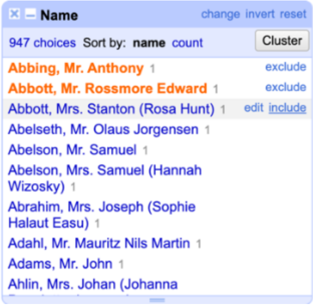
- I used the separator task for splitting the titles from the family names that is completely explained to the final report.

Recombining back the split column:
Edit column, transform: `value + “” + cells [“column_name”].value`

We can check those cells which are blank do not combine with columns:
`Facet, Customized Facets, Facet by blank, just include the false ones.`

Faceting more than one group:

This can easily be done by including them.



947 choices Sort by: name count

Abbing, Mr. Anthony 1
Abbott, Mr. Rossmore Edward 1
Abbott, Mrs. Stanton (Rosa Hunt) 1
Abelseth, Mr. Olaus Jorgensen 1
Abelson, Mr. Samuel 1
Abelson, Mrs. Samuel (Hannah Wizosky) 1
Abrahim, Mrs. Joseph (Sophie Halaut Easu) 1
Adahl, Mr. Mauritz Nils Martin 1
Adams, Mr. John 1
Ahlin, Mrs. Johan (Johanna

Also, only these subsets of data can be exported separately rather than the whole dataset.

All:
If we do anything here it will apply on the whole dataset. Edit columns, re-order/ remove columns with this we easily can change the order of columns or drop them for removing.

For example: How to deal with several blank cells?

```
Column -> Facet -> Customized Facets-> Facet by blank
```

include the true ones, then go to the all column and edit rows, remove matching rows.

Fetching information from web - reconciling:

We can reconcile our values with database, we can call things from the outside into the data set to improve the quality of dataset. For example:

```
Column -> edit column -> add column based on this column-> delete the value in
expression-> copy the URL ->choose a column name.
```

now there is a column with URL link. On this column.

```
edit the column-> add column by fetching URLs-> new column name-> choose delay for
5000 milliseconds.
```

Export:
The project can be export in many different formats like: tab-separated value, comma separated value, HTML Table, Excel, Google Sheets, SQL Exporter, Templating. We can export the whole data set, or just select a special subset of our data by faceting and export it. After data cleaning by OpenRefine we can export it again in csv and use it in other tools for complex analysis in python or R.

The final cleaned data, for further analysis exported in excel to gain more knowledge from the data, and the results are illustrated in final report.

Analysis of COVID-19

For this project I chose Python programming language, because I have experience in it about two years. Python programming language is the best language to deal with data, since it has every essential tool for all steps, data collection and cleaning, data exploration, data modeling and data visualization. I usually for data science tasks use Google Colab or Jupyter Notebook and for other purpose for example for web developing (Python, Django) use Visual Studio Code or PyCharm IDE. Here for this project I chose Google Colab since work with it is really easier than Jupyter Notebook.

Importing data in Google Colab:
I found the data link in Project work PDF as below:
https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data

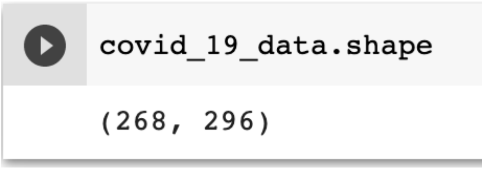
then opened the csse_covid_19_time_series folder, and find the target csv file as:
time_series_covid19_confirmed_global.csv

For working with data we need the raw data so press the raw button to grab the link from there as below:
https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv

After that for easier use, the link placed in variable named, covid_19_url.

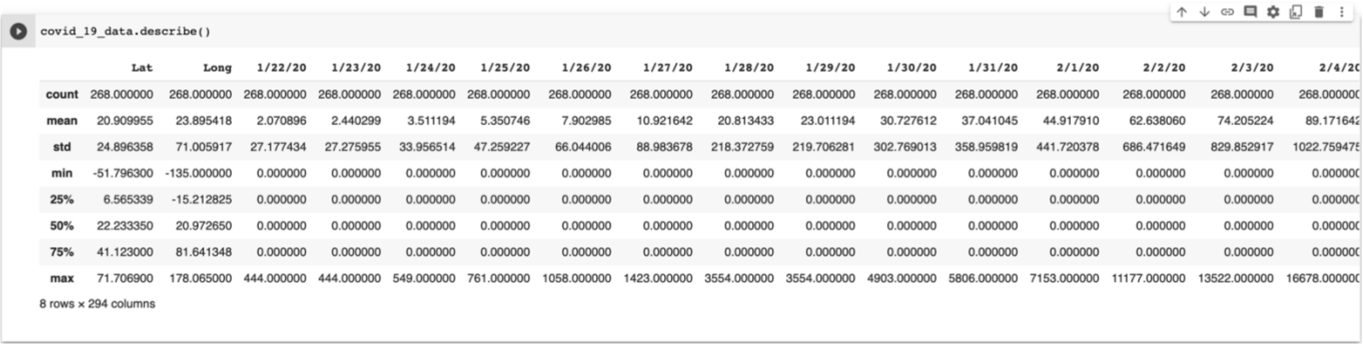
Understanding the data schema:

After importing some libraries like pandas and matplotlib. I fetch data by Pandas as a dataframe in Google Colab. Then by head() function saw the first five rows of it. It is important before working with data have a good understanding of data. Therefore, I used some other property like shape to understand the size of dataframe. There are 268 rows and 296 columns.



```
covid_19_data.shape
```

```
(268, 296)
```



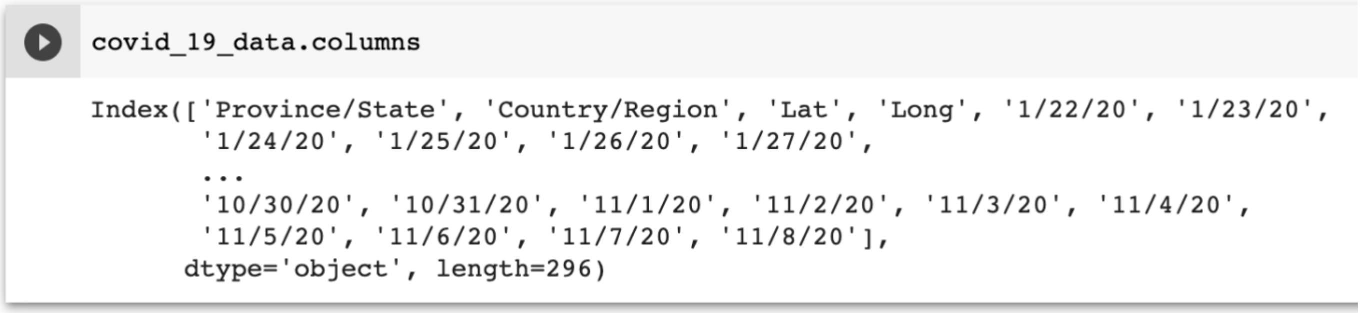
```
covid_19_data.describe()
```

	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/2
count	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	268.00000
mean	20.909955	23.895418	2.070896	2.440299	3.511194	5.350746	7.902985	10.921642	20.813433	23.011194	30.727612	37.041045	44.917910	62.638060	74.205224	89.17164
std	24.896358	71.005917	27.177434	27.275955	33.956514	47.259227	66.044006	88.983678	218.372759	219.706281	302.769013	358.959819	441.720378	686.471649	829.852917	1022.75947
min	-51.796300	-135.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	6.565339	-15.212825	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
50%	22.233350	20.972650	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
75%	41.123000	81.641348	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
max	71.706900	178.065000	444.000000	444.000000	549.000000	761.000000	1058.000000	1423.000000	3554.000000	3554.000000	4903.000000	5806.000000	7153.000000	11177.000000	13522.000000	16678.00000

8 rows × 294 columns

The describe() function is another function which is very useful to know our data, it summarize some descriptive statistics.

We can see name of the columns by this line of code:



```
covid_19_data.columns
```

```
Index(['Province/State', 'Country/Region', 'Lat', 'Long', '1/22/20', '1/23/20',
       '1/24/20', '1/25/20', '1/26/20', '1/27/20',
       ...
       '10/30/20', '10/31/20', '11/1/20', '11/2/20', '11/3/20', '11/4/20',
       '11/5/20', '11/6/20', '11/7/20', '11/8/20'],
      dtype='object', length=296)
```

Showing the whole dataframe:

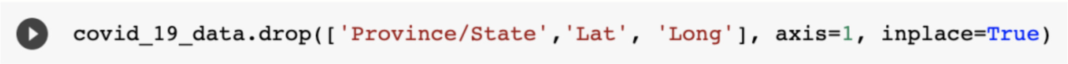
```
covid_19_data[:]
```

Showing a part of dataframe:

```
covid_19_data[1:9]
```

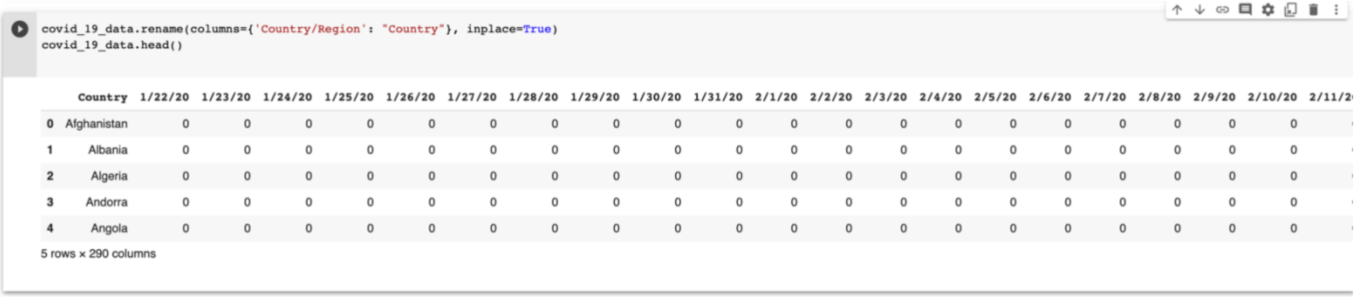
Only rows from 1 to 8 will display and row number 9 is not included.

This data is clean and ready for work, but if I wanted to delete some columns that I do not need them it is easy by dropping them. For example, here Province/State, Lat, Long are not useful to work with them in this project so I can easily drop them:



```
covid_19_data.drop(['Province/State','Lat', 'Long'], axis=1, inplace=True)
```

Also, I can rename the column names:

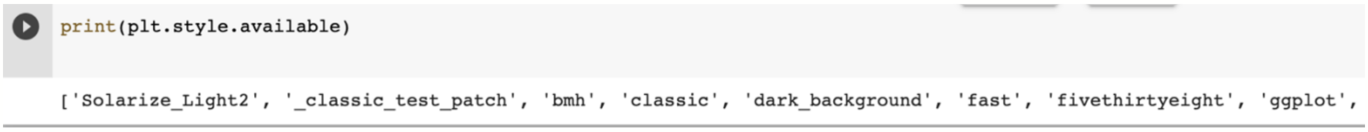


```
covid_19_data.rename(columns={'Country/Region': "Country"}, inplace=True)
covid_19_data.head()
```

	Country	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	2/1/20	2/2/20	2/3/20	2/4/20	2/5/20	2/6/20	2/7/20	2/8/20	2/9/20	2/10/20	2/11/2
0	Afghanistan	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Albania	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	Algeria	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	Andorra	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	Angola	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 290 columns

Using Style Sheets:
With this line of code all the available styles for Matplotlib display to choose between:



```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot',
```

And then with this line apply it to choose a different look for our visualization, here I preferred to use the default one (seaborn- white):

```
    plt.style.use('fivethirtyeight')
```

Choose a country for visualization:

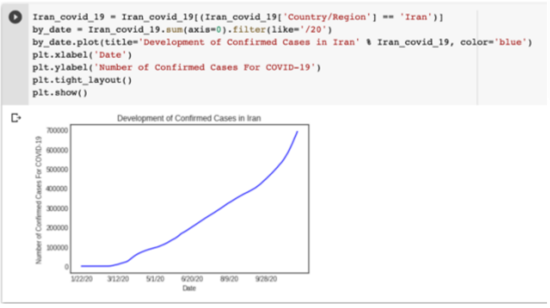
Here I chose Iran (which is my home country) and for the comparison, I chose Germany (which is my current residential place).

By choosing the column covid_19_data [Country/Region] I could see all the country names with their indexes in the dataframe. Some countries were hidden by three dots that with pd.set_option (‘display.max_rows’, 268) opened all of the hidden rows. With iloc[] property I could get the target country row by their indexes. But I preferred to filter countries by their names that is much easier.

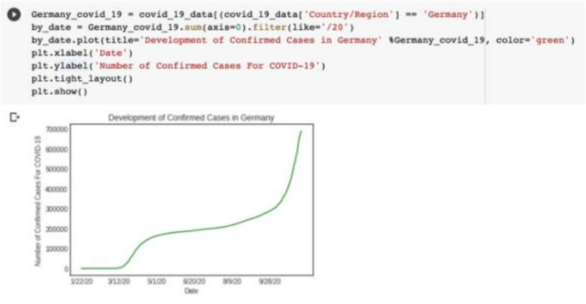
After that by matplotlib library, I plotted two countries of Germany and Iran in two different diagrams for the whole time series. In Data visualization it is very important to visualize your data in a way that your audience completely understand your diagram. Therefor it was important here to choose an appropriate title for my plots and also put a recognizable name for each x-axis and y-axis. The title must be descriptive but at the same time as short as possible. The curve should be also recognizable with good color and thickness also put a legend to understand each curve uses.

For this time-series all the confirmed cases for both filtered countries here as Iran and Germany were plotted in two different diagrams for whole year of 2020 to 10th November as it is shown in x-axis. The number of confirmed cases in y-axis are in a range of the 0 to more than 700,000 for both countries since this number is a bit same for Iran and Germany.

plt. tight_layout () assure that subplots are nicely fit in the figure and plt.show() display the figure.



```
Iran_covid_19 = Iran_covid_19[(Iran_covid_19['Country/Region'] == 'Iran')]
by_date = Iran_covid_19.sum(axis=0).filter(like='/20')
by_date.plot(title='Development of Confirmed Cases in Iran' % Iran_covid_19, color='blue')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases For COVID-19')
plt.tight_layout()
plt.show()
```

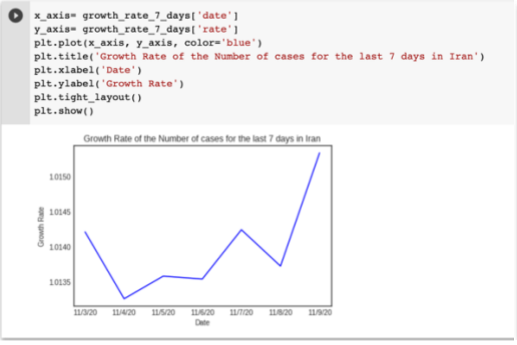


```
Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
by_date = Germany_covid_19.sum(axis=0).filter(like='/20')
by_date.plot(title='Development of Confirmed Cases in Germany' %Germany_covid_19, color='green')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases For COVID-19')
plt.tight_layout()
plt.show()
```

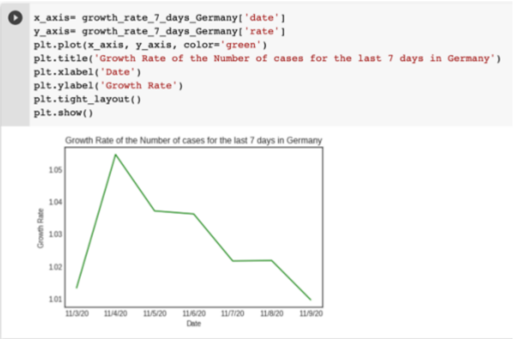
Growth Rate:

For the growth rate I sum up the whole confirmed cases in 2020 with help of (axis=0) which refers to the row and filtered in the year by filter(like='/20') subset of the dataframe rows according to the specified index labels and then change the series to dataframe by to_frame() function and finally with pct_change() + 1 , computes the percentage change from the immediately previous row by default. This is useful in comparing the percentage of change in a time series of elements. With the second row of code I checked the smoothed data.

- First, I calculated the growth rate for the whole time series day to days as explained above. But understood that the time series was too long and x-axis labels were too much to be readable. Therefore, I corrected myself to show only the last 7 days of growth rate for both Germany and Iran, in this way the changes are more visual.

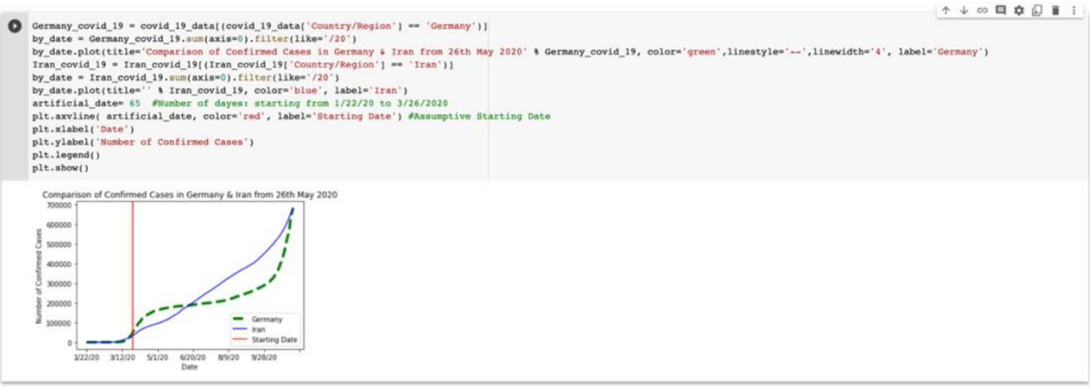


```
x_axis= growth_rate_7_days['date']
y_axis= growth_rate_7_days['rate']
plt.plot(x_axis, y_axis, color='blue')
plt.title('Growth Rate of the Number of cases for the last 7 days in Iran')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.tight_layout()
plt.show()
```



```
x_axis= growth_rate_7_days_Germany['date']
y_axis= growth_rate_7_days_Germany['rate']
plt.plot(x_axis, y_axis, color='green')
plt.title('Growth Rate of the Number of cases for the last 7 days in Germany')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.tight_layout()
plt.show()
```

Choose another country and compare the development of confirmed case:



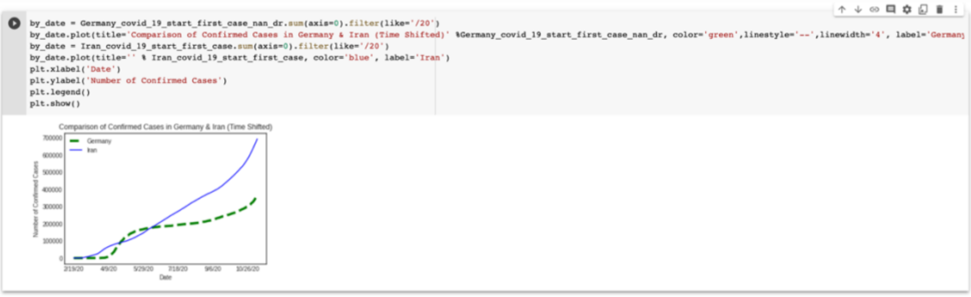
```
Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
by_date = Germany_covid_19.sum(axis=0).filter(like='/20')
by_date.plot(title='Comparison of Confirmed Cases in Germany & Iran from 26th May 2020' % Germany_covid_19, color='green',linestyle='--',linewidth='4', label='Germany')
Iran_covid_19 = Iran_covid_19[(Iran_covid_19['Country/Region'] == 'Iran')]
by_date = Iran_covid_19.sum(axis=0).filter(like='/20')
by_date.plot(title='' % Iran_covid_19, color='blue', label='Iran')
artificial_date= 65  #Number of dayes: starting from 1/22/20 to 3/26/2020
plt.axvline( artificial_date, color='red', label='Starting Date') #Assumptive Starting Date
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases')
plt.legend()
plt.show()
```

- For this problem I had to correct myself, because I understood that only by plotting two countries on the same figure, we cannot have a fair comparison between them. I assumed that since both Germany and Iran have a little same confirmed cases and same population can be plotted in the same figure but after trying the other ways, I understood that I am completely wrong, and it is better to plot the countries base on their first appearing the cases on the same artificial time and as another approach compare them by their population, I also mixed these two approaches together at the end of the work.

So, I tried to check when Iran first confirmed cases appeared, and it was on 2/19/2020 with two people. Then check it for the Germany that it was on 1/27/20 with one person. So, it is obvious that the Iran confirmed cases are started 23 days after Germany. There for having a good comparison I need to shift the Germany curve to the starting point of Iran which is on 2/19/2020.

For this purpose, first I removed some columns of Germany which had 0 on their cells, they indicated not any confirmed cases on those days, after that I shifted the columns 23 days forward. To the first appearing of the confirmed cases of Germany be in the same day with Iran which is 2/19/2020. Whit this all shifted cell values got NaN value instead, I dropped them all by dropna. Then I also needed to drop those columns that are below the days but with string values.

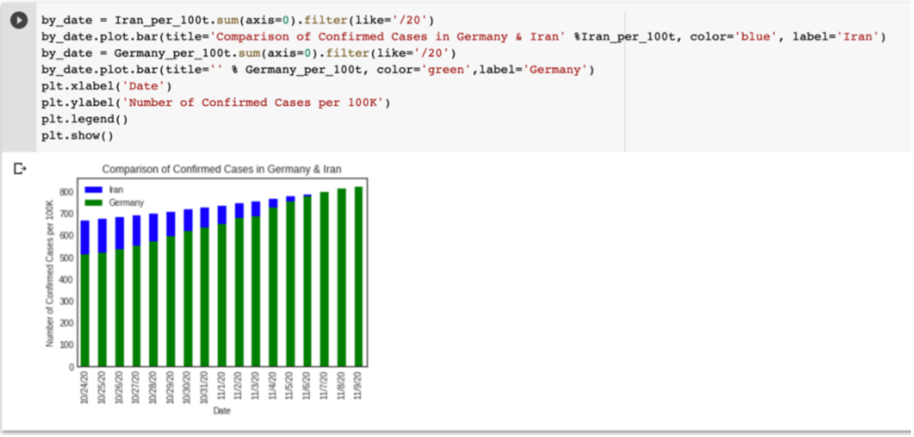
For the Iran I only dropped those columns that had 0 values in their cells. After all of these processes I plotted two curves in the same figure. Iran with blue curve and Germany with green dashed curve. And now yes, the comparison is fair, and it is understandable that although Iran cases appeared 23 days later than Germany but its development is too much faster than Germany which they today reach at the same level.



```
by_date = Germany_covid_19_start_first_case_nan_dr.sum(axis=0).filter(like='/20')
by_date.plot(title='Comparison of Confirmed Cases in Germany & Iran (Time Shifted)' %Germany_covid_19_start_first_case_nan_dr, color='green',linestyle='--',linewidth='4', label='Germany
by_date = Iran_covid_19_start_first_case.sum(axis=0).filter(like='/20')
by_date.plot(title='' % Iran_covid_19_start_first_case, color='blue', label='Iran')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases')
plt.legend()
plt.show()
```

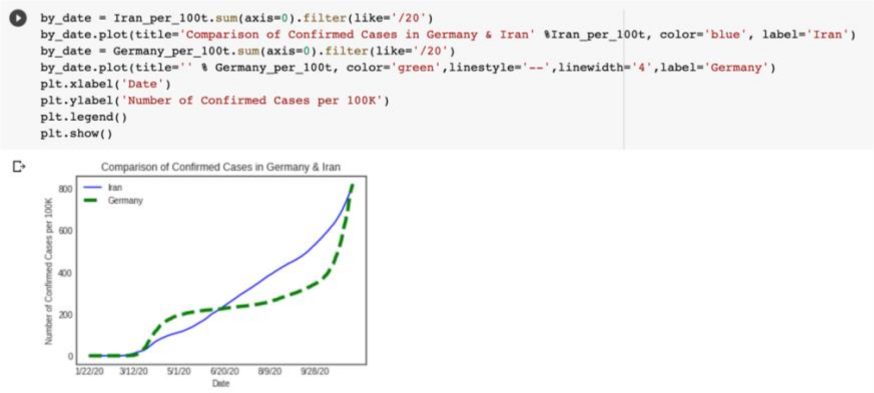
For the comparison between the population of two countries and number of cases, I calculated number of confirmed cases per 100.000 person base on the each total population of the countries. Then I plot them in by barchart.

This figure indicated to the only last 17 days.



```
by_date = Iran_per_100t.sum(axis=0).filter(like='/20')
by_date.plot.bar(title='Comparison of Confirmed Cases in Germany & Iran' %Iran_per_100t, color='blue', label='Iran')
by_date = Germany_per_100t.sum(axis=0).filter(like='/20')
by_date.plot.bar(title='' % Germany_per_100t, color='green',label='Germany')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases per 100K')
plt.legend()
plt.show()
```

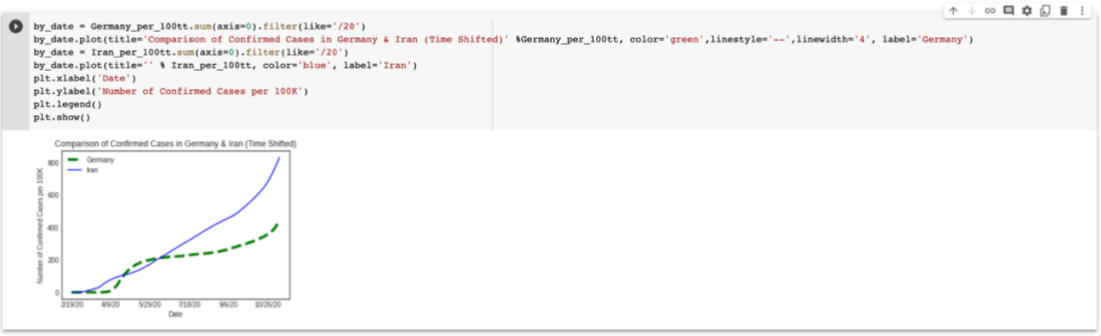
This figure shows it for the whole time series:



```
by_date = Iran_per_100t.sum(axis=0).filter(like='/20')
by_date.plot(title='Comparison of Confirmed Cases in Germany & Iran' %Iran_per_100t, color='blue', label='Iran')
by_date = Germany_per_100t.sum(axis=0).filter(like='/20')
by_date.plot(title='' % Germany_per_100t, color='green',linestyle='--',linewidth='4',label='Germany')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases per 100K')
plt.legend()
plt.show()
```

Comparison of Confirmed Cases in Germany & Iran
Iran
Germany
Number of Confirmed Cases per 100K
0
200
400
600
800
1/22/20
3/12/20
5/1/20
6/20/20
8/9/20
9/28/20
Date

At the end I tried to mixed these two approaches together, shifted time series and compare per 100K.



```
by_date = Germany_per_100tt.sum(axis=0).filter(like='/20')
by_date.plot(title='Comparison of Confirmed Cases in Germany & Iran (Time Shifted)' %Germany_per_100tt, color='green',linestyle='--',linewidth='4', label='Germany')
by_date = Iran_per_100tt.sum(axis=0).filter(like='/20')
by_date.plot(title='' % Iran_per_100tt, color='blue', label='Iran')
plt.xlabel('Date')
plt.ylabel('Number of Confirmed Cases per 100K')
plt.legend()
plt.show()
```

Comparison of Confirmed Cases in Germany & Iran (Time Shifted)
Germany
Iran
Number of Confirmed Cases per 100K
0
200
400
600
800
2/19/20
4/9/20
5/29/20
7/18/20
9/6/20
10/26/20
Date