Project Report

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

Abstract. Data Preprocessing is one of the most important tasks for any data scientist. This phase consumes time more than any other steps, usually, more than 50, 60, or even more percent of the whole process because it is a manual task and the decisions depend on the different cases. It depends on the data, and the goal of the analysis. The importance of it could be understandable by this phrase: “Garbage in, garbage out”. There are some tools either as an example OpenRefine and Trifacta at a high level which is more visual and interactive or NumPy, Pandas which are libraries in Python at a low level that help in data preprocessing. The aim of this project is to deal with the titanic dataset to make it clean from any messy or heterogeneity data by OpenRefine.

OpenRefine, “Previously Google Refine” is a free, open-source, and extensible powerful tool for working with messy data that runs offline in a web browser. OpenRefine can handle all sorts of data. Rows, columns, and cells as each individual part are the dealing sections in it. Importing is possible in any kind of formats such as TSV, CSV, *SV, Excel, XML, JSON, Google Data documents, RDF and sources can be a local file, URLs, clipboard, Database, or Google Data. OpenRefine use cases are: **Cleaning**: discovering and fixing inconsistency with faceting, clustering, cell transforms, GREL expressions. **Transforming**: changing formats or reshape with split/join multi-valued cells, splitting columns, transposing columns/row. **Extending**: enriching data by combining files, merging projects, fetching URLs, reconciliation with online databases. **Automating**: reusing your processing routine by exporting operation history in JSON could be named as the use cases of Openrefine. One of the great features about OpenRefine is that it never changes the original data, it means when we import a file in, OpenRefine, it just makes a copy of it.

Task 1

Importing Data in OpenRefine:

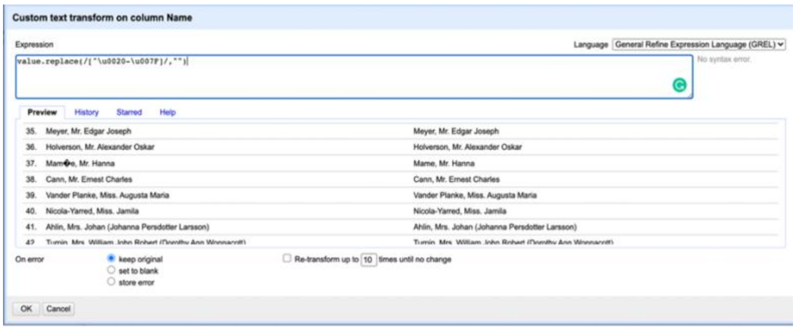
The dataset was downloaded directly from Microsoft Teams. OpenRefine is opened in the browser at the port of 3333. Data is uploaded in OpenRefine and shows a preview of it. Since we have column names then choose Parse next 1 line as columns header and Character Encoding as UTF-8. By having a look at the column names and inserted data below them, it is understandable that data is imported correctly. At a glance, there are a total of 950 Rows and 12 Columns related to passengers details named: (PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, ParCh, Ticket, Fare, Cabin, Embarked). Understanding the data set and each of its column is very important.

1. Messy/Noisy Data in Passenger Names (A Black Question Mark):

- Since some passengers are from different countries, there are some special characters between their names. Usually, by setting character encoding to UTF-8 all the special characters will show properly but here the problem is different or maybe problem is with original CSV file. So, by trying a special function and catching that black question mark between the passenger names and replacing it with a white space this problem solved. Explaining the method:

Using General Refine Expression Language (GREL):

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



By doing this operation 7 rows affected, after that all seven names searched and found in between encyclopedia-Titanica then edited manually in a correct form.

2. Column Data Types:

When data set import into OpenRefine all the values appear as strings or text. Therefore, by having a look at the data set details, columns data types should be changed to the proper types such as Numbers, Texts or Dates.

`Edit Cells -> Common transforms -> (to Number/ to Text/ to date)`

All	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

As it is shown in the above picture, PassengerId, Survived, Pclass, Age, SibSp, ParCh, Fare changed to numbers and the others changed to text. Ticket number and Cabin number changed to text also since there is combination of characters and numbers.

Compare to Trifacta: The trifecta data wrangling is a little more advanced because we can also change data types to phone number, credit card numbers and many more but In OpenRefine we have only text in black, number in green that are right justified and date.

3. **When we are working with text it is better to do:**

   ```
   Edit Cells -> Common Transformation -> Unescape HTML entities AND Trim
   Leading and  trailing white space.
   ```

   This applied to all columns but nothing changed. Usually white spacing comes from data wrangling existence so it is better to solve them in this way.

4. **Undo/ Redo Tab:**
   For undoing one step, just need to go to the Undo/ Redo part and press the last step of work to continue from there. Here, there are all records of operations that is done so far. Also, there is an option to extract this information as an JSON file. This can be useful when there is a demand for explaining task to others, or maybe some operations in certain type of databases are same so they can be reusable for multiple times. By pressing the extract button and copying all of the information in a Notepad++ , can have a copy of them and then apply it to the other projects under the undo/redo button by pressing the apply button, and insert all those operations there then press the perform operations.

5. **Faceting:**
   Faceting is the one of the most important and powerful features of OpenRefine, it goes to the rows of the selected column and gives the combine of same items.

- **PassengerId:**

  ```
  PassengerId -> Facet -> Numeric Facet -> (No Error)
  PassengerId -> Facet -> Text Facet -> (No Error)
  ```

  Each PassengerId should be unique from 1 to 950 as it is.

- **Survived:**
  This column is the target variable, which we will predict once the preprocessing of data is complete. So this column is very important to be retained and cleaned. In this column only 0 (Not survived) and 1 for (Survived) is possible but errors found as below:

  ```
  Servived -> Facet -> Text Facet -> (Error: 0: 585, 1: 362, 2:1, 3: 2)
  ```

  Error (2:1) – The Passenger name found in encycolopedia-titanica under the name of Hays, Miss. Margaret Bechstein. Then all the information for her checked but the problem was only with survived part that changed to the 1 as rescued.

All	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
311.	311	1	1	Hays, Miss. Margaret Bechstein	female	24	0	0	11767	83.1583	C54	C

  Error (3:2) - There were 2 persons (Daher, Mr. Shedid and Samaan, Mr. Elias) here that seems their inserted information is one step shifted back. Since there is only two rows with this problem, these two rows can easily be removed without making any significantly statistical problem. But here to get a more precise information for end result by checking their names in encycolopedia-titanica the correct values inserted one by one, and again the Pclass datatype changed to number.

Before:

All	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
919.	919	3	Daher, Mr. Shedid	male	22.5	0	0	2698	7.225		C	
921.	921	3	Samaan, Mr. Elias	male		2	0	2662	21.6792		C	

After:

All	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
919.	919	0	3	Dāhir, Mr. Shadīd	male	19	0	0	2698	7.225		C
921.	921	0	3	Samaan, Mr. Elias	male	17	2	0	2662	21.6792		C

- **Pclass:**
`Pclass -> Facet -> text Facet -> (No Error)`

Only 1, 2, 3 is possible. Since in the last step two 919 and 921 rows are modified correctly now as a result here also under the Pclass part there is not any problem.

- **Name:**
`Name -> Facet -> text Facet -> (No Error BUT needs to check for duplicates)`

Only names are possible. Since in the last step two row number 919 and 921 are modified correctly now as a result here also under the name part there is not any problem.

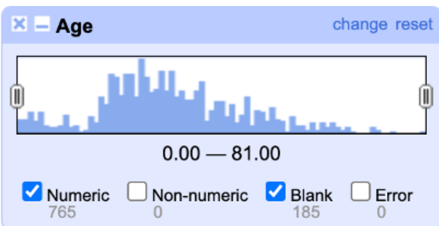
Also, as explained before the problem with question mark in the middle of the names has been solved. But now there are two names that are appeared for two times. Both Connolly, Miss. Kate and Kelly, Mr. James that remained, because they are completely two different persons, only there was a bit difference in their ages which edited manually same to the encyclopedia. Still with this column there are some other problems which in next steps will be explained.

- **Sex**:
`Sex -> Facet ->Text Facet -> (Female: 336, male: 614)`

Only age is possible. Since in the last step two rows 919 and 921 are modified correctly now as a result here also under the sex part there is not any problem. Before there were one blank and one 22.5.

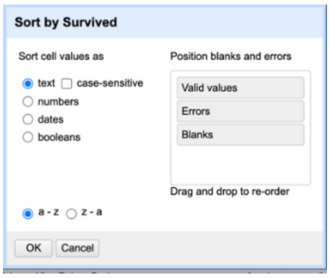
For further analysis and doing Machin learning these female and male should map to the numbers.

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



There is not any outlier in age range but there are 185 blank cells that is too much. As a statistical view it may affect dramatically in results. So, it is better to replace these null values by the average or mean values of two groups of the passengers as survived or not survived.

Making Records:
Grouping can be done in the first column. In the survived column, sorting first 0 and the other 1, then making it as permanent.



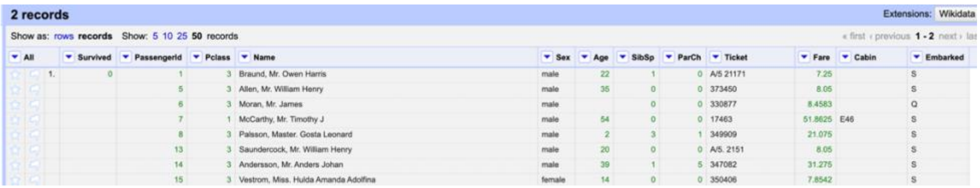
Sorting:
By this we have first zeros and then all other ones. This will happen on UI surface, If I want to make it permanent should go to the tab bar, sort, reorder rows permanently. Then sort as permanent to index values appear in a correct way. Now the survived column needs to move at the beginning, I did this by:

```
Edit Column -> Move column to beginning
```

Then there is a need to blank down all the values after the first 0 and 1.

```
Edit cells -> blank Down
```

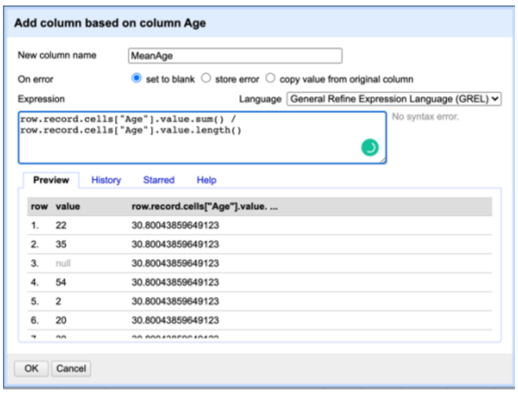
Now it is easy to switch from rows to records.


2 records
Show as: rows records Show: 5 10 25 50 records

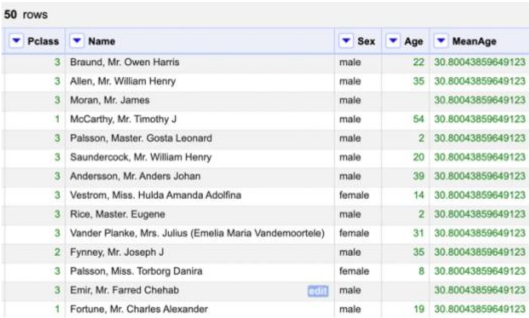
All	Survived	PassengerId	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
1.	0	1	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
		5	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
		6	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
		7	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
		8	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
		13	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
		14	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
		15	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S

Then, I need to calculate the mean value. For this purpose, added a new column.

```
Age -> add column based on column Age -> Expression -> row.
record.cells["age"].value.sum()/row.record.cells["age"].value.length()
```



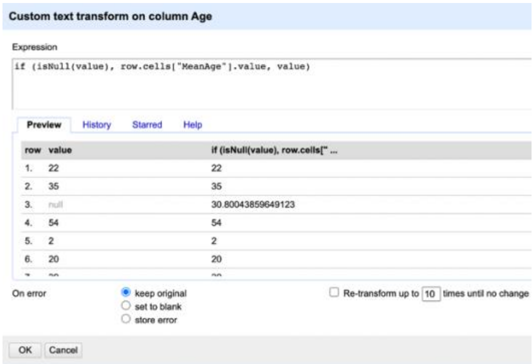
By pressing the ok, an artificial column is inserted named as MeanAge.

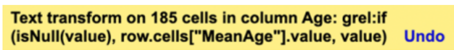


50 rows

Pclass	Name	Sex	Age	MeanAge
3	Braund, Mr. Owen Harris	male	22	30.80043859649123
3	Allen, Mr. William Henry	male	35	30.80043859649123
3	Moran, Mr. James	male		30.80043859649123
1	McCarthy, Mr. Timothy J	male	54	30.80043859649123
3	Palsson, Master. Gosta Leonard	male	2	30.80043859649123
3	Saundercock, Mr. William Henry	male	20	30.80043859649123
3	Andersson, Mr. Anders Johan	male	39	30.80043859649123
3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	30.80043859649123
3	Rice, Master. Eugene	male	2	30.80043859649123
3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	30.80043859649123
2	Fynney, Mr. Joseph J	male	35	30.80043859649123
3	Palsson, Miss. Torborg Danira	female	8	30.80043859649123
3	Emir, Mr. Farred Chehab edit	male		30.80043859649123
1	Fortune, Mr. Charles Alexander	male	19	30.80043859649123

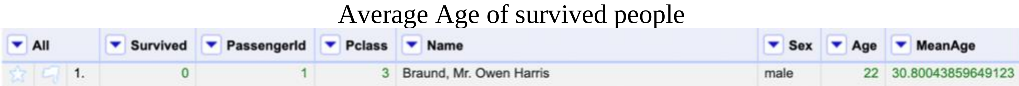
The null values of Age column can be filled by these average ages.
`Age -> edit cells -> transform -> custom text transform on column Age -> if isNull(value), row.cells[“MeanAge”].value, value)`





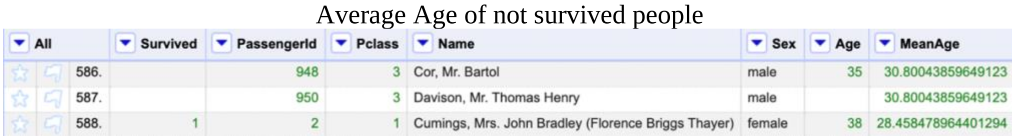
For the not survived passengers the average is calculated as 30, and for the survived passengers the average is calculated as 28 with long decimals.

Average Age of survived people



	All	Survived	PassengerId	Pclass	Name	Sex	Age	MeanAge
	1.	0	1	3	Braund, Mr. Owen Harris	male	22	30.80043859649123

Average Age of not survived people



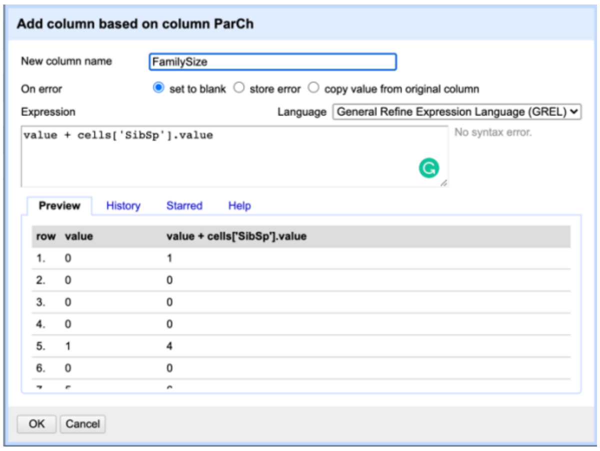
	All	Survived	PassengerId	Pclass	Name	Sex	Age	MeanAge
	586.		948	3	Cor, Mr. Bartol	male	35	30.80043859649123
	587.		950	3	Davison, Mr. Thomas Henry	male		30.80043859649123
	588.	1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	28.458478964401294

Then replacing the null values in column Age with these average ages and drop the column MeanAge.

by using the `round(value),` all floating numbers in Age column rounded to the nearest integer.

- **SibSp** (Siblings/ Spouse):
  `Facet, Text Facet: (No error)`
  0: 643, 1:227, 2: 32, 3:17, 4:19, 5:5, 8:7
  `Facet, Numeric Facet: (Range: 0-8)`

- **ParCh** (Parent/Children):
  `Facet, Text Facet: (No error)`
  0: 724, 1:126, 2: 84, 3:6, 4:4, 5:5, 6:1
  `Facet, Numeric Facet: (Range: 0-6)`
  - For further analysis by summing up the two ParCh and SibSp columns together the number of family members is discovered in a new column named FamilySize.



- **Ticket:**
  There are some ticket numbers that are the same (some family members have the same ticket number), and the formats are weird. Tickets must be in numbers but they appeared in both characters and numbers. As the number of these ambiguities is too much it cannot be easily handled manually and there is a need to connect to other external sources. For this purpose, there is an option in OpenRefine named reconcile that by checking the external source like "wikidata" will clear all these ambiguities. (Steps are explained in diary report)

All	Survived	PassengerId	Pclass	First Name and Last Na	Sex	Age	SibSp	ParCh	Ticket	Ticket_2	Fare	Cabin	Embarked
1.	0	1	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	HD 21171 Choose new match	7.25		S
2.		5	3	Allen, Mr. William Henry	male	35	0	0	373450	SDSS J093017.95+373450.1 Choose new match	8.05		S
3.		6	3	Moran, Mr. James	male	31	0	0	330877	HD 330877 Choose new match	8.4583		Q
4.		7	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	HD 17463 Choose new match	51.8625	E46	S
5.		8	3	Palsson, Mr. Gosta Leonard	male	2	3	1	349909	HD 349909 Choose new match	21.075		S

- **Fare:**
  All the values are in numbers started from 0 to 512.32. Those who did not paid their ticket price are the ship crews. And by checking the encyclopedia-Titanica, the unexpected values like 512 for three persons found that are not outlier.

- **Cabin:**
  `Facet, Text Facet`
  There were some columns that had more than one cabin number inserted into it. As it is discovered in the encyclopedia, there are some persons which have more than one cabin number so instead of a few of them, others remained as before. But there are 731 cells which are blanks. As this is too much, access to the external source is essential or if this column does not play any important role in our future analysis can completely remove (drop) or just ignore it. There is a unique character

before each cabin number that maybe can be useful in future analysis, but for this project I just ignore this column number and delete it.

- **Embarked:**
  `Facet, Text Facet`
  C:185, Q:83, S:680, blank:2

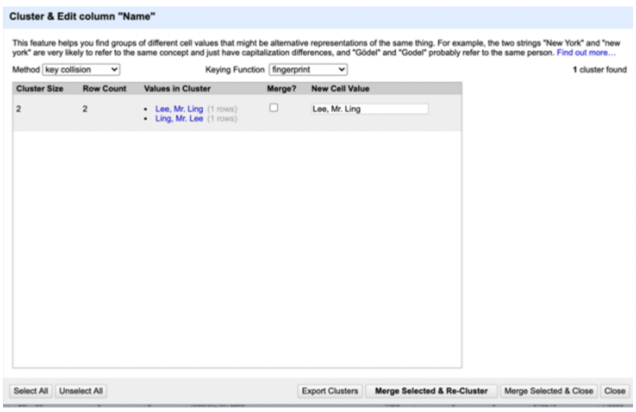
  There were only two women with missed embarked values which manually edited by having a look at the encyclopedia.
  For further prediction and Machin learning process we should map these categorical values to numbers that will be explained in next steps.

6. Clustering:

`Name, Edit Cells, Cluster and Edit`

By choosing different Methods and different Keying Functions can discover different types of clusters. Clusters suggest things that are the same base on the algorithm. Potential duplicates can be identified with this option. Merging the values in a cluster means that all cluster records will receive the same value in this column and records will be kept and the number of rows is unchanged.

Method: Key collision and Keying Function: fingerprint, which is one of the most precise algorithms, showed a name (Lee, Mr. Ling and Ling, Mr. Lee) which can be for the same person, so browsed this cluster:



All	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Fare	Cabin	Embarked
170.	170	0	3	Ling, Mr. Lee	male	28	0	0	1601	56.4958		S
931.	931	0	3	Lee, Mr. Ling	male	28	0	0	1601	56.4958		S

Since all the data were the same so these two rows indicated to one person. Therefore, ticked the merge box and chose the merge selected & re-cluster button. Then to do not have duplicates in rows, star it and facet it by star then delete the matched row. The passenger 170 is totally removed.

With metaphone3 mostly family members listed, for example, ticket number 347082 were the same for all 4 persons since they were family members, a couple with two children, they remained.

Also, ticket number: 363291 was same for 3 other family members in the same way. These family members without any change remained since they had different values except their ticket numbers.

Thayer, Mr. John Borland their name was same but they were totally two different people so remained as before.

All	Survived	PassengerId	Pclass	First Name and L	Sex	Age	SibSp	ParCh	Ticket	Ticket_2	Fare	Cabin	Embarked
45.		77	3	Mineff, Mr. Ivan	male	31	0	0	349208	HD 349208 Choose new match	7.8958		S
188.		295	3	Mineff, Mr. Ivan	male	24	0	0	349233	HD 349233 Choose new match	7.8958		S
224.		365	3	O'Brien, Mr. Thomas	male	31	1	0	370365	OGLE BLG-ELL-14637 Choose new match	15.5		Q
335.		553	3	O'Brien, Mr. Thomas	male	31	0	0	330979	HD 330979 Choose new match	7.8292		Q
425.		697	3	Kelly, Mr. James	male	44	0	0	363592	UCAC3 152-363592 Choose new match	8.05		S
548.		892	3	Kelly, Mr. James	male	20	0	0	330911	HD 330911 Choose new match	7.8292		Q
551.		898	3	Connolly, Miss. Kate	female	42	0	0	330972	HD 330972 Choose new match	7.6292		Q
689.		290	3	Connolly, Miss. Kate	female	24	0	0	370373	HD 83063 Choose new match	7.75		Q

For O'Brien, Mr. Thomas there was only one person by these names in encyclopedia with the ticket number 370365 so they merged together, and delete one of them.

Cluster & Edit column "Name"

This feature helps you find groups of different cell values that might be alternative representations of the same thing. For ex
york" are very likely to refer to the same concept and just have capitalization differences, and "Gödel" and "Godel" probably

Method nearest neighbor | levenshtein | Radius 1.0 | Block Chars 6

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	2	• Youseff, Mr. Gerious (1 rows) • Youssef, Mr. Gerious (1 rows) Browse this cluster	☐	Youseff, Mr. Gerious

With the Method of nearest neighbor and levenshtein two names: Youseff, Mr. Gerious, and Youssef, Mr. Gerious appeared: Only one of them found in an encyclopedia with the age of 45 but the ticket number was different, so the ticket number for the 45 years old edited and then they merged together. But preferred to delete the row for the 31 years old one since this 31 years old Youseff was calculated from the last step by mean age and the name was not found in eccyclopedia so to do not have any duplicate this was another reason to remove this row.

Method nearest neighbor | ppm | Radius 1.0 | Block Chars 6

Cluster Size	Row Count	Values in Cluster	Merge?	New Cell Value
2	2	• Youseff, Mr. Gerious (1 rows) • Youssef, Mr. Gerious (1 rows)	☐	Youseff, Mr. Gerious
2	2	• Mineff, Mr. Ivan (1 rows) • Staneff, Mr. Ivan (1 rows)	☐	Mineff, Mr. Ivan
2	2	• Thayer, Mr. John Borland (1 rows) • Thayer, Mr. John Borland Jr (1 rows)	☐	Thayer, Mr. John Borland

All	Survived	PassengerId	Pclass	Name	Sex	Age	SibSp	ParCh	Ticket	Ticket_2	Fare	Cabin	Embarked
45.		77	3	Staneff, Mr. Ivan	male	31	0	0	349208	HD 349208 Choose new match	7.8958		S
189.		295	3	Mineff, Mr. Ivan	male	24	0	0	349233	HD 349233 Choose new match	7.8958		S

For the above picture, only the name under the Mr Ivan Staneff with the age of 23 was found in the encyclopedia so they merged together, and delete one of them.

There is also a visualization in the right hand when we have many clusters, we can navigate between them that how many clusters are and how many rows are there. The fingerprint is the most accurate clustering method and the cologne-phonetic is the least accurate one.

7. **Removing Duplicates:**

```
Name -> Facet -> customized facet -> Duplicates Facet
```

All			Survived	PassengerId	Pclass	First Name and L	Sex	Age	SibSp	ParCh	Ticket	Ticket_2	Fare	Cabin	Embarked
		45.		77	3	Mineff, Mr. Ivan	male	31	0	0	349208	HD 349208 Choose new match	7.8958		S
		188.		295	3	Mineff, Mr. Ivan	male	24	0	0	349233	HD 349233 Choose new match	7.8958		S
		224.		365	3	O'Brien, Mr. Thomas	male	31	1	0	370365	OGLE BLG-ELL-14637 Choose new match	15.5		Q
		335.		553	3	O'Brien, Mr. Thomas	male	31	0	0	330979	HD 330979 Choose new match	7.8292		Q
		425.		697	3	Kelly, Mr. James	male	44	0	0	363592	UCAC3 152-363592 Choose new match	8.05		S
		548.		892	3	Kelly, Mr. James	male	20	0	0	330911	HD 330911 Choose new match	7.8292		Q
		551.		898	3	Connolly, Miss. Kate	female	42	0	0	330972	HD 330972 Choose new match	7.6292		Q
		689.		290	3	Connolly, Miss. Kate	female	24	0	0	370373	HD 83063 Choose new match	7.75		Q

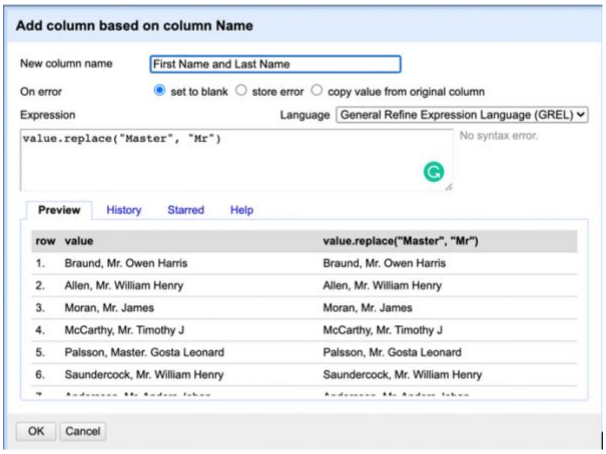
1. Mr. Ivan 31 years old did not find in encyclopedia, so removed.
2. Mr. O'Brien with ticket number 330979 also did not find in encyclopedia and removed.
3. For Ticket number 370373 there was a mistake in her name so manually edited to Miss Catherine Connolly.

8. **Filtering / Replacing:**

In the column Name, there were some words with misspelling. By using filter on the Name column 41 rows found with misspelling of Master. This problem solved by:

```
Name -> Edit Column -> add column based on column name -> in the expression
part wrote: value.replace("Master", "Mr")
```

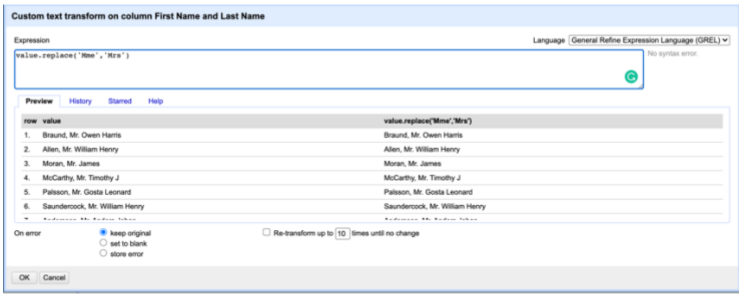
The new column named as First Name and Last Name and the column Name removed.



After exploring more other misspellings also found which this time replaced with the new version with this method on the same column:

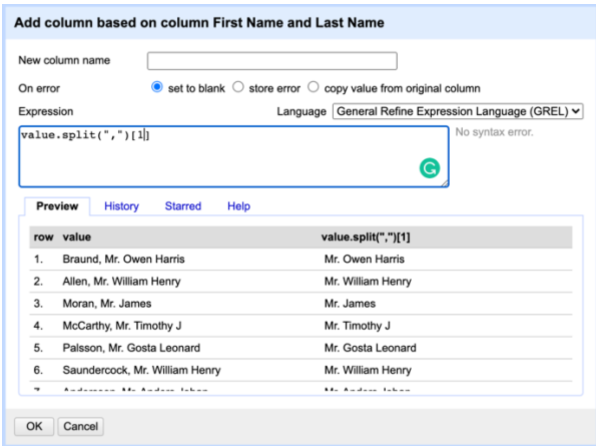
```
 First Name and Last Name -> Edit Cells -> Transform ->

value.replace("Master", "Mr")
value.replace("Sir ", "Mr")
value.replace("Mme", "Mrs")
value.replace("Mlle", "Miss")
value.replace("Ms", "Miss")
value.replace("Rev", "other")
value.replace("Major ", "other")
value.replace("Col ", "other")
value.replace("Capt ", "other")
```

```
value.replace("Jonkheer ", "other")
value.replace("Countess ", "other")
value.replace("Lady ", "other")
value.replace("Don ", "other")
value.replace("Dr ", "other")
value.replace("the other ", "other")
```

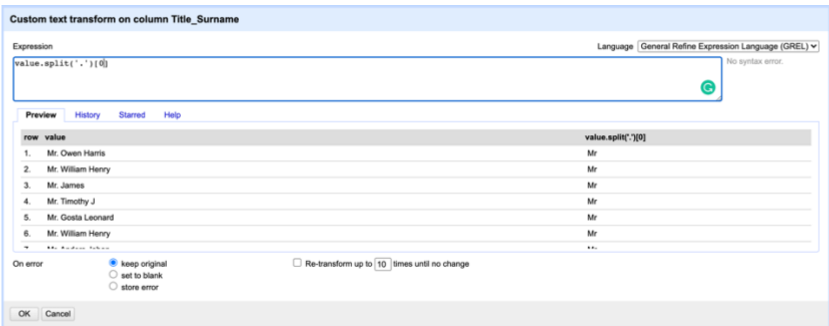


9. **Splitting**:
For further analysis should change categorical data to numerical. For this purpose, there is possibility to separate the title of people (Mr, Mrs, Miss, other) and put them in a new column named title.

First in the column First Name and Last Name , split the title.family name.



And in the second step, separate the titles from the family name.



After that if we wish, we can completely drop the column First Name and Family Name.

By faceting on the column title we have this:

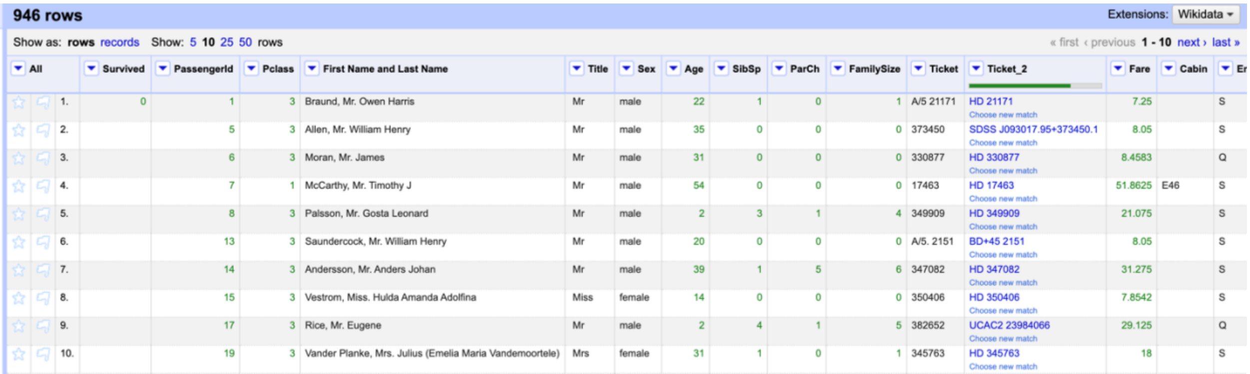
Now each of these titles should map to the specified number.

10. Some Data Analysis by faceting:

As an example, we can facet sex, include male then facet survived, include 1. Now we can understand how many men is survived.

11. Final Results:

In the following pictures the end results are shown one by one, here we have ticket column which is reconciled and matched with the external source. The Name column after cleaning (black question mark, misspellings, duplicates, separation) is totally removed and replaced with the new column named Title. Age column is fully filled with Mean age of the two group of survived and not survived people.



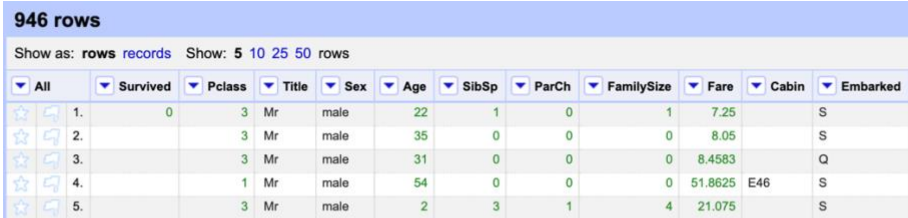
946 rows   Extensions: Wikidata

Show as: rows records   Show: 5 10 25 50 rows   « first ‹ previous 1 - 10 next › last »

All	Survived	PassengerId	Pclass	First Name and Last Name	Title	Sex	Age	SibSp	ParCh	FamilySize	Ticket	Ticket_2	Fare	Cabin	Er
1.	0	1	3	Braund, Mr. Owen Harris	Mr	male	22	1	0	1	A/5 21171	HD 21171 Choose new match	7.25		S
2.		5	3	Allen, Mr. William Henry	Mr	male	35	0	0	0	373450	SDSS J093017.95+373450.1 Choose new match	8.05		S
3.		6	3	Moran, Mr. James	Mr	male	31	0	0	0	330877	HD 330877 Choose new match	8.4583		Q
4.		7	1	McCarthy, Mr. Timothy J	Mr	male	54	0	0	0	17463	HD 17463 Choose new match	51.8625	E46	S
5.		8	3	Palsson, Mr. Gosta Leonard	Mr	male	2	3	1	4	349909	HD 349909 Choose new match	21.075		S
6.		13	3	Saundercock, Mr. William Henry	Mr	male	20	0	0	0	A/5. 2151	BD+45 2151 Choose new match	8.05		S
7.		14	3	Andersson, Mr. Anders Johan	Mr	male	39	1	5	6	347082	HD 347082 Choose new match	31.275		S
8.		15	3	Vestrom, Miss. Hulda Amanda Adolfina	Miss	female	14	0	0	0	350406	HD 350406 Choose new match	7.8542		S
9.		17	3	Rice, Mr. Eugene	Mr	male	2	4	1	5	382652	UCAC2 23984066 Choose new match	29.125		Q
10.		19	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	Mrs	female	31	1	0	1	345763	HD 345763 Choose new match	18		S

All	Survived	PassengerId	Pclass	Title	Sex	Age	SibSp	ParCh	FamilySize	Ticket	Ticket_2	Fare	Cabin	Embarked
1.	0	1	3	Mr	male	22	1	0	1	A/5 21171	HD 21171 Choose new match	7.25		S
2.		5	3	Mr	male	35	0	0	0	373450	SDSS J093017.95+373450.1 Choose new match	8.05		S
3.		6	3	Mr	male	31	0	0	0	330877	HD 330877 Choose new match	8.4583		Q
4.		7	1	Mr	male	54	0	0	0	17463	HD 17463 Choose new match	51.8625	E46	S
5.		8	3	Mr	male	2	3	1	4	349909	HD 349909 Choose new match	21.075		S

Since this ticket column do not play any important role in future analysis, preferred to completely remove it. Instead of it created a new column named FamilySize which has the important effect on the survival of the passengers.



946 rows

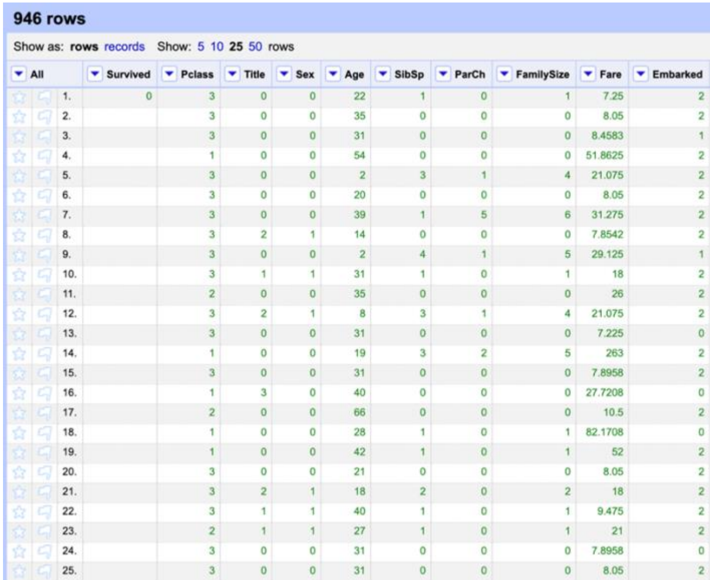
Show as: rows records Show: 5 10 25 50 rows

All	Survived	Pclass	Title	Sex	Age	SibSp	ParCh	FamilySize	Fare	Cabin	Embarked
1.	0	3	Mr	male	22	1	0	1	7.25		S
2.		3	Mr	male	35	0	0	0	8.05		S
3.		3	Mr	male	31	0	0	0	8.4583		Q
4.		1	Mr	male	54	0	0	0	51.8625	E46	S
5.		3	Mr	male	2	3	1	4	21.075		S

For any Machin learning process there is need to convert categorical values to numbers, so values of three columns Embarked, Title, and Sex mapped to the proper numbers as below:

```
value.replace('C', '0')
value.replace('Q', '1')
value.replace('S', '2')
value.replace('female', '1')
value.replace('male', '0')
value.replace('Mrs', '1')
value.replace('Mr', '0')
value.replace(‘Miss’, '2')
value.replace(‘other’, '3')
```

The final cleaned data set is as below:

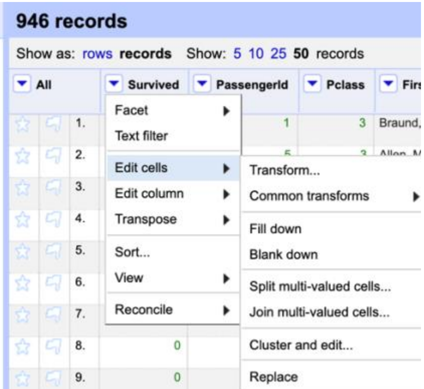


946 rows

Show as: rows records Show: 5 10 25 50 rows

All	Survived	Pclass	Title	Sex	Age	SibSp	ParCh	FamilySize	Fare	Embarked
1.	0	3	0	0	22	1	0	1	7.25	2
2.		3	0	0	35	0	0	0	8.05	2
3.		3	0	0	31	0	0	0	8.4583	1
4.		1	0	0	54	0	0	0	51.8625	2
5.		3	0	0	2	3	1	4	21.075	2
6.		3	0	0	20	0	0	0	8.05	2
7.		3	0	0	39	1	5	6	31.275	2
8.		3	2	1	14	0	0	0	7.8542	2
9.		3	0	0	2	4	1	5	29.125	1
10.		3	1	1	31	1	0	1	18	2
11.		2	0	0	35	0	0	0	26	2
12.		3	2	1	8	3	1	4	21.075	2
13.		3	0	0	31	0	0	0	7.225	0
14.		1	0	0	19	3	2	5	263	2
15.		3	0	0	31	0	0	0	7.8958	2
16.		1	3	0	40	0	0	0	27.7208	0
17.		2	0	0	66	0	0	0	10.5	2
18.		1	0	0	28	1	0	1	82.1708	0
19.		1	0	0	42	1	0	1	52	2
20.		3	0	0	21	0	0	0	8.05	2
21.		3	2	1	18	2	0	2	18	2
22.		3	1	1	40	1	0	1	9.475	2
23.		2	1	1	27	1	0	1	21	2
24.		3	0	0	31	0	0	0	7.8958	0
25.		3	0	0	31	0	0	0	8.05	2

I could fill the number of survived and victim in the column above by: (Fill down)



Task 2

a) How would you describe the overall data quality?

Data quality is one of the core components of the data management process. It is a measure of the condition of data based on some factors like validity, accuracy, completeness, consistency, uniformity, reliability, timeliness.

In titanic data set since most of the important columns such as Survived, Name/Title, Pclass, Sex, Age, FamilySize, Fare, Embarked that play a vital role in further analysis, for example what factors make people more survived, are complete and precise we can measure data quality as good.

-validity, column data types that should be in number like (Age, Fare, …) or text like (Name) is correctly specified and value range for example for column Age (0-81), SibSp (0-8), ParCh (0-6), FamilySize (0-10), Fare (0-520), Pclass(1-3), Survived(0,1) were true and without any outlier.

There were some misspelling between names such as Master rather Mr that changed correctly (Norming). There were some noisy data like a black question mark between names that they also removed and replaced by a white space and edited by an external source. Also, there was some duplicated between names which merged and removed those duplicates that refers exactly two one person to avoid any bias.

-Accuracy measurement errors are known and precise. Data values are in a right value and are represented in a consistent and unambiguous form.

- Completeness there were some null and missing values which are filled with the appropriated values such as the mean of the age or for the ticket number filled with the help of reconciliation method and fetch the data from wikidata, or for other filled with missing values filled them manually by data given from encyclopedia-Titanica.

- Consistency, there is not any contradictions between values, and any duplication solved.
- Uniformity all the values under a column have same unit and format.
- Timeliness, this is a historical data that can be correctly match with good sources like wikidata.

b) Is the interpretation of each variable clearly defined?

Interpretation means what the numbers express in the words or express it verbally. For example, some tools like python can give us the results that what is the median of the numbers. But here we should understand each column and variable and what exactly they represent to us as a their correct value.

- **Survived:** is a target variable for future prediction. This column should be in only 1 as survived and 0 as victim and the column data type should be in numerical. So, this column is well defined and without null values.

- **Age:** It is number so the column datatype changed to number, and as it had missing values, they filled with mean of the age of the two group of survived and victim people. This column shows the age of the passengers started from 0 to 81. The youngest passenger is 5-month baby and the oldest one is 81 years old man.

- **PassengerId**, **Ticket** does not much value in predicting. PassengerId is a number that refers to the index or id of each passengers, so the column datatype changed to number, and ticket is the ticket number of each passenger which is in both character and numbers remained in text and reconciliation was done with wikidata. Although for this project some data preprocessing was done on the ticket column but these two columns for further analysis and predictions can easily be dropped or ignored.

  Name: The data type of this column is in text and refers to the first name, last name and title of each passengers. Some noisy characters (black question marked) removed, some misspelling replaced with their correct forms, duplication removed and separation of the titles in a new column named as title is done. These titles are categorical data which mapped to the numbers for the Machin learning process.

- **ParCh, SibSp** ParCh number of parents and children, SibSp number of siblings and spouse are well defined and they are numbers so their columns data type changed to number. With summing up the two columns together and creating a new column named family size, the number of family members is measured.

- **Sex, Cabin and Embarked:** are text and the categorical data so they encoded and mapped to the numerical values. Sex refers to the female or male of the passengers, and embarked is referring where the passengers mount in the ship. These two in the end mapped to the numbers to be understandable for Machin learning process. Cabin had lots of null values with less value for future prediction so dropped the column, but there are some unique characters at the beginning the cabin number which denotes to the deck number. With creating a new column named Deck, this information can be extract for further prediction. But for now as it had lots of missing value and the reconciliations take too much time I only ignore and at the end drop this column.

- **Fare and Pclass:** column datatype also changed to number and they correctly refers to price and class numbers of the passengers.

c) How to deal with missing data?

For dealing with missing values it is important to understand their value for future prediction so we need to recognize them. Sometimes a few of records have null value and cannot fill with any method, so in such cases that numbers of records do not affect too much in statistical view we can easily remove those rows, or even in such cases that the whole column is not important drop the total column or ignore them.

But most of the times records are important for us. we can do imputation by inferring to other fields, from a random record, the previous record, or interpolation of values before and after, also from a regression analysis.

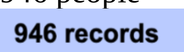
Here for titanic data set Age filled with the mean value of age of two group of passenger list as survived and victim. With this some key value like median did not affect. Another work that is done here is referred to external source such as encycolopedia-titanica and add those null, missing values manually or with the ability of OpenRefine reconcile them to fetch these missing values from wikidata.

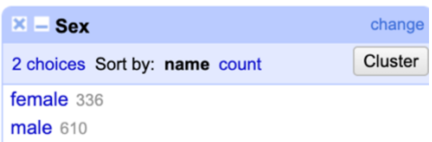
d) How to correct wrong data or interpret data with unclear semantics?

Having a good knowledge of the dataset can help us. Also those columns with wrong data types can be detected and change to the right one. Also, in some cases some external sources like wikidata or encycolopedia-titanica can be used. In OpenRefine there is an option for reconciliation that connect to wikidata or another external source to remove any ambiguities. Also, it is easy to compare the duplicates and same clusters with some algorithms. We can delete them or If we want to keep them, we can replace (imputation) those duplicates that seemed are same but they were different.

In titanic dataset there were some persons with same name which with some algorithms founded. They were checked if there were same as a one person then one of them removed. Some of them had same name but they were completely different persons so they kept, and the other that had the same name by mistake checked in the encycolopedia-titanica to change the name to the correct form that was inserted incorrectly. In titanic dataset also there were some misspelling data which could be easily understand that they are mistake and replaced with correct forms for example master instead of Mr. But in some cases that we cannot understand the wrong data in an easy way we have to get help from the domain knowledge experts.

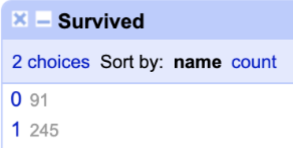
e) What are potential questions that can be answered with the passenger list? Please identify at least 3 questions and give answers from the data.

1. How many people totally were in titanic?
   946 people
   

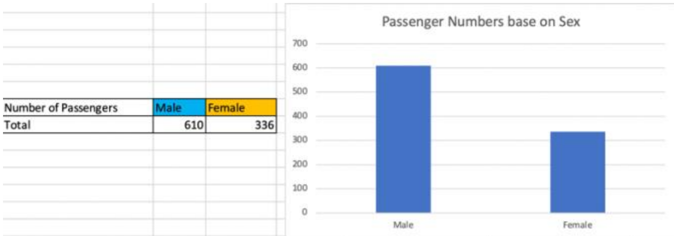
2. How many people were female and male separately?
   336 Females and 610 Male
   

3. How many males did survive or not survived?
   With text faceting on sex column and then survived column then including male the answer is: 492 males died and 118 males survived.
   

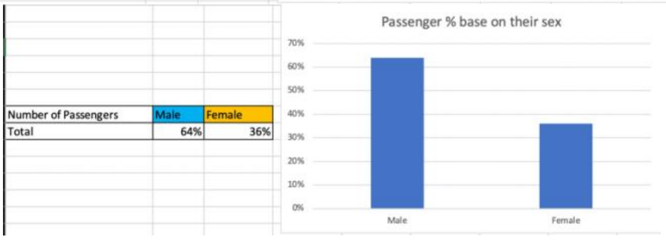
4. How many females did survive or not survived?
With text faceting on sex column and then survived column then including female the answer is: females 91 died and 245 females survived.



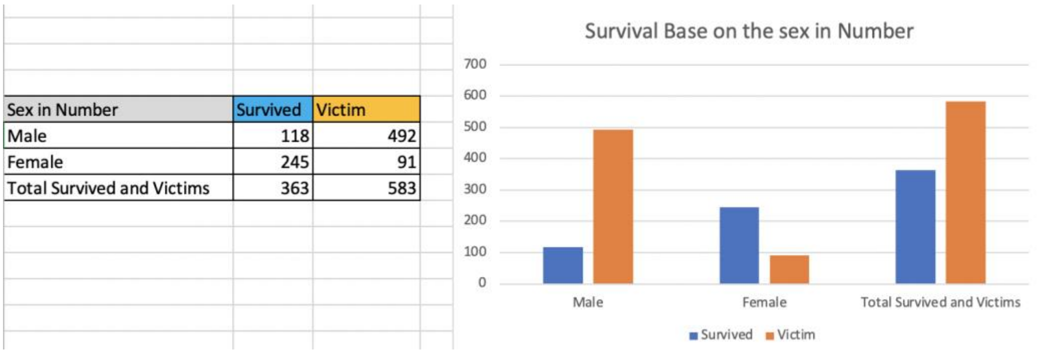
5. Which group of sex mostly survived?
By having a look at data, it is understandable that the number of male’s passenger were more than females but in contrast to the survival more female survived, and also number of victim people in total is more than survived passengers, and number of victims in total is more than survival.



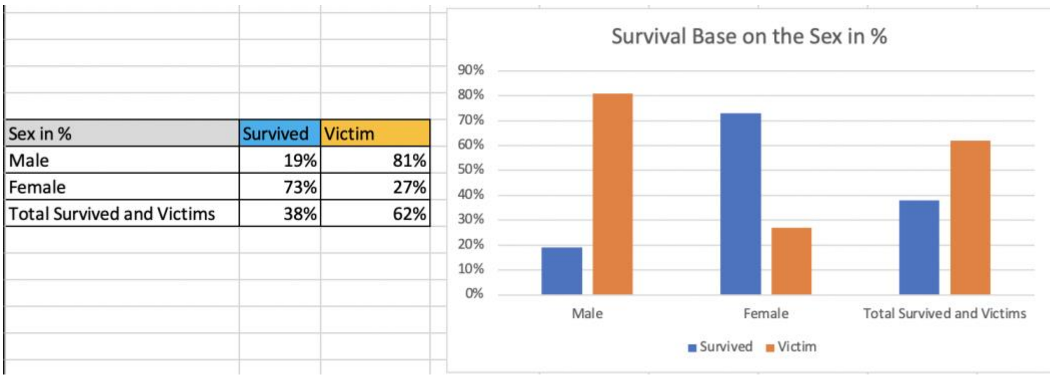
Number of Passengers	Male	Female
Total	610	336



Number of Passengers	Male	Female
Total	64%	36%



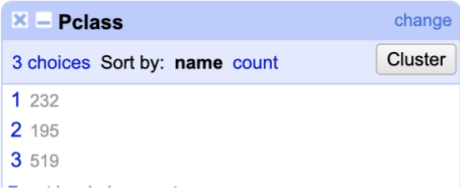
Sex in Number	Survived	Victim
Male	118	492
Female	245	91
Total Survived and Victims	363	583



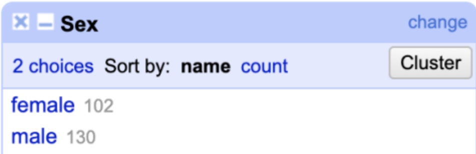
Sex in %	Survived	Victim
Male	19%	81%
Female	73%	27%
Total Survived and Victims	38%	62%

6. How many people were in different classes?

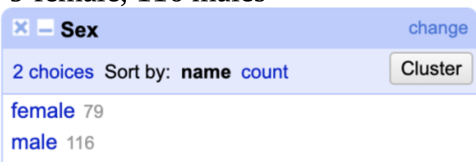
First class: 232 people, Second class: 195 people, Third class: 519 people
It is understandable that almost half of the passengers are in third class and it is reasonable since this type of the transportation was expensive.



7. A) How many female and male were in first class?
102 females, 130 males

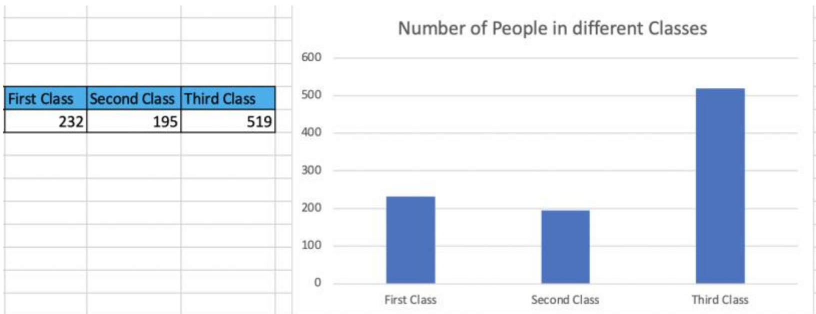


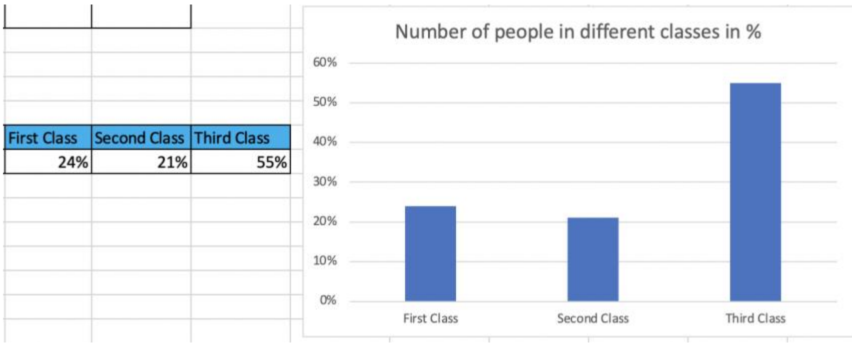
B) How many female and male were in second class?
79 female, 116 males



C) How many female and male were in third class?
155 female and 364 males

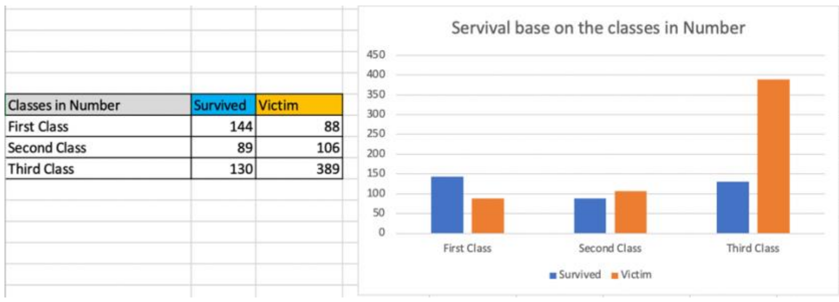
We can see that the number of males in the third class is two times more than females. Also People mostly are in third class.



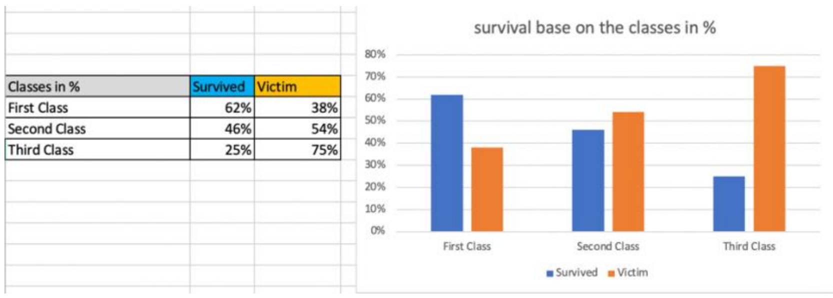


First Class	Second Class	Third Class
24%	21%	55%

People in the first class had a greater chance to survive and people in the Third class had a lower chance to survive.

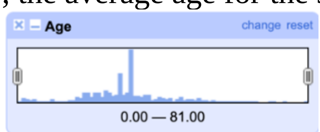


Classes in Number	Survived	Victim
First Class	144	88
Second Class	89	106
Third Class	130	389

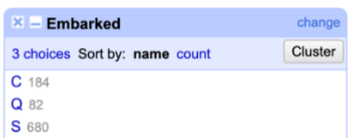


Classes in %	Survived	Victim
First Class	62%	38%
Second Class	46%	54%
Third Class	25%	75%

8. What is the range of the ages?
   It is between 0-81. The youngest passenger is a 5-month baby and the oldest is 81 years old man , the average age for the survived people was 28 and for not survived was 31.



9. Where people mounted in ship?
   680 passengers from Southampton, 184 passengers from Cherbourg, 82 passengers from Queenstown

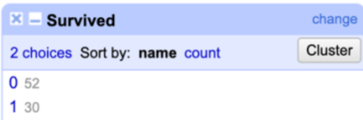


Total: 946,

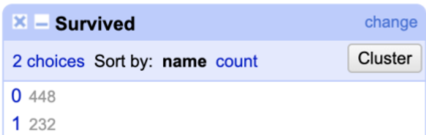
C: 19.45%
Q: 8.66%
S: 71.88%

For the C:

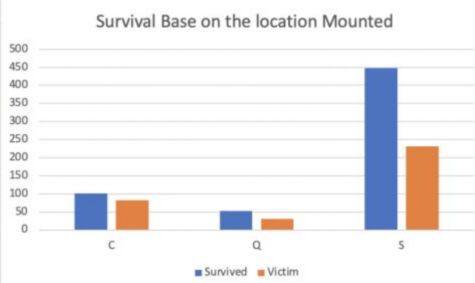
For the Q:



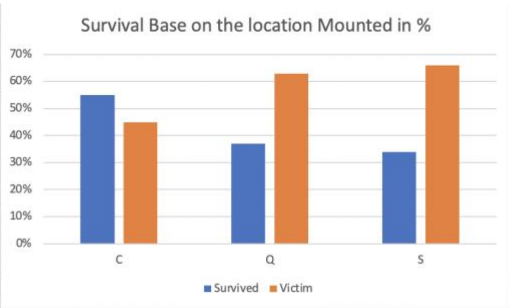
For the S:



Mounted in Number	Survived	Victim
C	101	83
Q	52	30
S	448	232



Mounted in %	Survived	Victim
C	55%	45%
Q	37%	63%
S	34%	66%



From the above result it is understandable that, passengers from the S had the lower chance to survive, and passenger from the C had the most chance to survive, and the number of people were mounted in the ship in the S was the largest.

f) Identify at least 2 additional questions on the Titanic passengers that cannot be answered from the given data, and specify which additional variables and data sources could help to find answers.

Since OpenRefine works for cleaning data it cannot be used for any analytical or prediction model. So, drive lots of questions that here cannot be answered like:

1. Which class has the greater number of the children? And how many of them survived?
   we need to calculate the number of children in the ship, it can be extractable from the Age column with some associations from Sex and survival column.

2. Is the number of family important for passenger survival?
   We need to add an additional column named family size.

3. What features are more related to the survival of the passengers?

It seems there is strong association between (the charge of the ticket, size of the family, the place that they boarded in the ship) with their survival. The higher a tourist paid had more chance to survive. Also, the chance of survival will dropp dramatically if someone traveled with more than 2 siblings or spouse. Also, those who board from Cherbourg had a higher chance to survive.

4. What is the nationality of the most passengers?

We need a separate column for the nationality of passengers, the information should be gain from the external source.

5. How many of Female passengers were married? And how many of them survived?

This question can be answered from the separated column named Title, and with some association with survival column.

Analysis of COVID-19

Abstract. these days whole the world is involved with pandemic covid-19. It is very important to gather all data from different countries throughout the world to control better this problem. Data in single do not have any value, we need to learn from data to get information then knowledge or even more wisdom. On the other hand, data visualization is very important, sometimes one picture values to 1000 words. The aim of this project is to visualize the confirmed case of this virus for two different countries and compare them together. Since here the data is as a time series and it is too much the best data analysis tool for this purpose could be Python programming language with its data manipulation and analysis library Pandas and its visualization library Matplotlib.

Task 1:

Data is provided by John Hopkins university. It is easy to put the link in Google Colab and start the work. But before that there is a need to import some libraries like Pandas and Matplotlib. It was a bit important to see all the rows so set the number of displaying rows to 300.

```
[186] %matplotlib inline
      import pandas as pd
      import matplotlib.pyplot as plt
      pd.set_option('display.max_rows', 300)
```

To see which styles for the matplotlib is available, and use the appropriate one for this project:

```
[187] print(plt.style.available)
      plt.style.use('seaborn-white')

      ['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind',
```

Now put the raw data link in covid_19_url variable to read it with Pandas. With the help of head() function, we can the the first 5 row of the dataframe. To see x rows, we can set the desire number of rows in the head(x) function.

```
[188] covid_19_url = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv'
      covid_19_data= pd.read_csv(covid_19_url)
      covid_19_data.head()
```

	Province/State	Country/Region	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	2/4/20	2/5/20	2/6/20
0	NaN	Afghanistan	33.93911	67.709953	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	NaN	Albania	41.15330	20.168300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	NaN	Algeria	28.03390	1.659600	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	NaN	Andorra	42.50630	1.521800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	NaN	Angola	-11.20270	17.873900	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5 rows × 297 columns

This timeseries shows the daily numbers of confirmed cases for each country.

Task 2:

a) Choose a country and visualize the development of confirmed cases.

Here data are clean and there is no need for any data cleaning. But there is a possibility to change some column name such as Country/Region to only Country or drop some useless columns such as Lat, long, Province/State.

First only by grabbing Country/Region column, get the full insight that which countries are involved and what is their index number in this dataframe. Totally there are 268 countries.

With this line of code `pd.set_option('display.max_rows', 268)` three dots will remove and the whole countries will display.

```
covid_19_data['Country/Region']

0              Afghanistan
1                  Albania
2                  Algeria
3                  Andorra
4                   Angola
                  ...
263     West Bank and Gaza
264         Western Sahara
265                  Yemen
266                 Zambia
267               Zimbabwe
Name: Country/Region, Length: 268, dtype: object
```

The iloc[] property that is integer-location based, help to grab target country row, in this case Iran and Germany.

```
covid_19_data.iloc[145]

Province/State         NaN
Country/Region        Iran
Lat                32.4279
Long                53.688
1/22/20                  0
                    ...
11/4/20             646164
11/5/20             654936
11/6/20             663800
11/7/20             673250
11/8/20             682486
Name: 145, Length: 296, dtype: object
```

```
covid_19_data.iloc[130]

Province/State         NaN
Country/Region     Germany
Lat                51.1657
Long               10.4515
1/22/20                  0
                    ...
11/4/20             608611
11/5/20             631172
11/6/20             653992
11/7/20             668114
11/8/20             682624
Name: 130, Length: 296, dtype: object
```

But sometimes because of the large amount of data, it is not easy to find the index of the rows. So it is better to filter rows with the name of the countries, in this case Iran and Germany.

```
Iran_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
Iran_covid_19.head()
```

	Province/State	Country/Region	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	2/4/20	2/5/20	2/6/
145	NaN	Iran	32.427908	53.688046	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

1 rows × 297 columns

```
Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
Germany_covid_19.head()
```

	Province/State	Country/Region	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	2/4/20	2/5/20	2/6/
130	NaN	Germany	51.165691	10.451526	0	0	0	0	0	1	4	4	4	5	8	10	12	12	12	

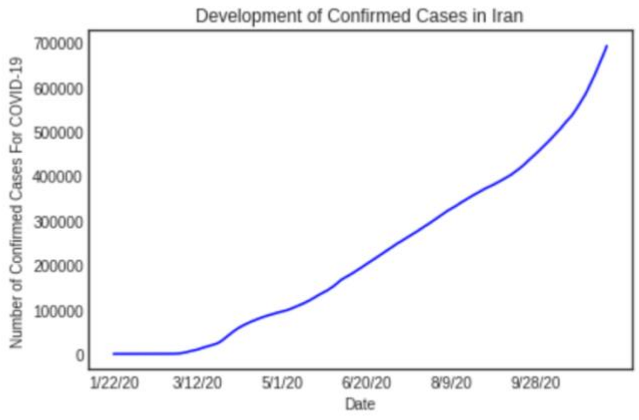
1 rows × 297 columns

Now, the number of confirmed cases in both countries of Iran and Germany in 2020 are going to be plotted as below.

Iran:

First catch the only Iran row in the whole dataframe, then for the x-axis filter the columns by having a '/20' in their names. After that plot the data on the figure by setting a title and choosing the colour for the curve. Finally, for each axis label their names, and show 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()
```



In the above picture we can see all the confirmed cases from the beginning of the series to today 10th November 2020, Iran confirmed cases has almost steady increased in numbers. The confirmed cases in Iran started from 2/19/2020 with two people.

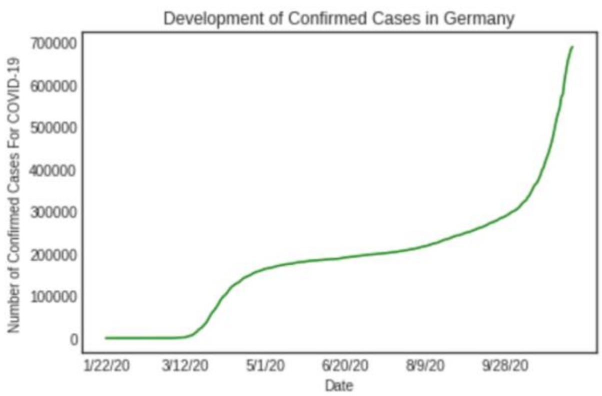
```
[211] Iran_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
      Iran_covid_19.head()
```

2/19/20	2/20/20	2/21/20	2/22/20	2/23/20	2/24/20	2/25/20	2/26/20	...
2	5	18	28	43	61	95	139	...

Germany:

Same coding is done but this time for the Germany country.

```
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()
```



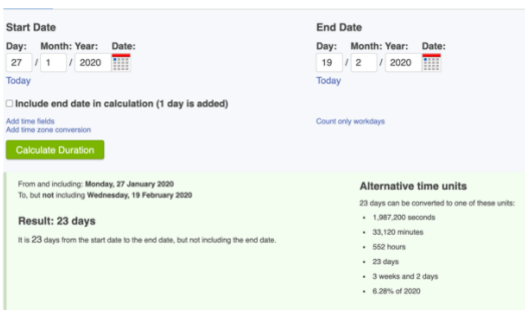
The Germany starting point is from 1/27/20 with one case, from the figure we can understand that the Germany until 9/28/20 controlled the disease in the better way than Iran but after that time number of confirmed cases increased significantly, and is same in the number of the confirmed cases with Iran.

```
[212] Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
      Germany_covid_19.head()
```

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1
130	NaN	Germany	51.165691	10.451526	0	0	0	0	0	1	

1 rows × 297 columns

It is obvious that Iran confirmed case started 23 days later than Germany.



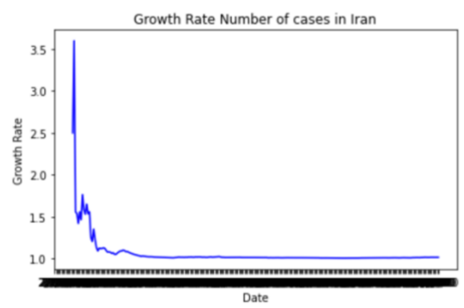
b) Think how you can measure the growth rate of the number of cases. Calculate the measure and visualize it as well.

For the growth rate we have to compare the case increase, this calculation for increase must be in percentages. By below formula the whole growth rate for the day by day on the whole time series is calculated. After calculation there is need to convert the series to dataframe format. As it is shown in the picture the time interval is too long that cannot be represented in the x-axis. So then, I decided to cut the time interval in only last 7 days.

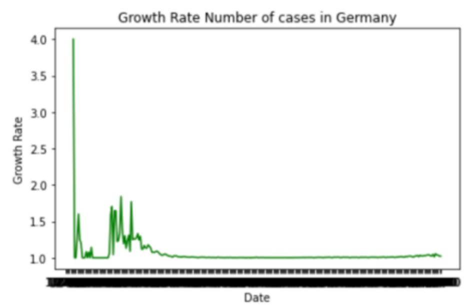
```
[34] growth_rate_Iran = Iran_covid_19.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
     growth_rate_Iran.ewm(span=5).mean()
     growth_rate_Iran.shape

     (292, 1)
```

```
growth_rate_Germany = Germany_covid_19.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
growth_rate_Germany.ewm(span=5).mean()
growth_rate_Germany.shape

(292, 1)
```

```
x_axis= growth_rate_Iran.index
y_axis= growth_rate_Iran[0]
plt.plot(x_axis, y_axis, color='blue')
plt.title('Growth Rate Number of cases in Iran')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.tight_layout()
plt.show()
```




```
x_axis= growth_rate_Germany.index
y_axis= growth_rate_Germany[0]
plt.plot(x_axis, y_axis, color='green')
plt.title('Growth Rate Number of cases in Germany')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.tight_layout()
plt.show()
```



Before calculating the time interval for the only last 7 days. Rename the columns in the appropriate way and make the indexing of the dataframe correctly. Then cut the first rows to 29^{th} rows since they were null values. It is important to set the inplace= true to make the changes permanently.

```
growth_rate_Iran = Iran_covid_19.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
growth_rate_Iran.ewm(span=5).mean()
growth_rate_Iran.reset_index(inplace=True)
growth_rate_Iran.rename(columns={'index':'date', 0: 'rate'}, inplace=True)
growth_rate_Iran = growth_rate_Iran.iloc[29:,:]
growth_rate_Iran
```

	date	rate
29	2/20/20	2.500000
30	2/21/20	3.600000
31	2/22/20	1.555556
32	2/23/20	1.535714
33	2/24/20	1.418605
34	2/25/20	1.557377
35	2/26/20	1.463158
36	2/27/20	1.762590
37	2/28/20	1.583673
38	2/29/20	1.528351
39	3/1/20	1.649241
40	3/2/20	1.534765
41	3/3/20	1.556296
42	3/4/20	1.250856

286	11/3/20	1.014205
287	11/4/20	1.013254
288	11/5/20	1.013576
289	11/6/20	1.013534
290	11/7/20	1.014236
291	11/8/20	1.013719
292	11/9/20	1.015331

Growth rate for the last 7 days of Iran

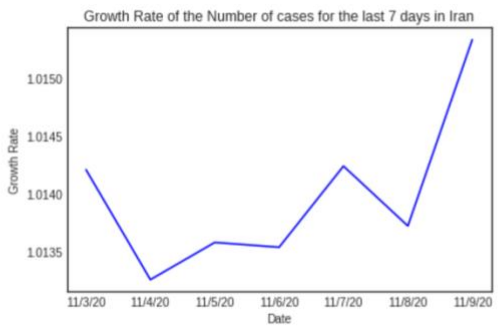
Then slice the data frame only for these last 7 days.

```
last_7_days = Iran_covid_19[['11/2/20','11/3/20','11/4/20','11/5/20','11/6/20','11/7/20','11/8/20','11/9/20']]
growth_rate_7_days = last_7_days.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
growth_rate_7_days.ewm(span=5).mean()
growth_rate_7_days.reset_index(inplace=True)
growth_rate_7_days.rename(columns={'index':'date', 0: 'rate'}, inplace=True)
growth_rate_7_days
```

	date	rate
0	11/2/20	NaN
1	11/3/20	1.014205
2	11/4/20	1.013254
3	11/5/20	1.013576
4	11/6/20	1.013534
5	11/7/20	1.014236
6	11/8/20	1.013719
7	11/9/20	1.015331

And tried to plot it:

```
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()
```



It is obvious that in last 7 days, Iran tried to control disease hardly but at the end it was not very successful and as a result it increased to 1.015.

Also the same process is done for the Germany:

Data for the last 7 days:

```
growth_rate_Germany = Germany_covid_19.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
growth_rate_Germany.ewm(span=5).mean()
growth_rate_Germany.reset_index(inplace=True)
growth_rate_Germany.rename(columns={'index':'date', 0: 'rate'}, inplace=True)
growth_rate_Germany.iloc[6:,:]
```

	date	rate
6	1/28/20	4.000000
7	1/29/20	1.000000
8	1/30/20	1.000000
9	1/31/20	1.250000
10	2/1/20	1.600000
11	2/2/20	1.250000
12	2/3/20	1.200000

285	11/2/20	1.046390
286	11/3/20	1.013225
287	11/4/20	1.054546
288	11/5/20	1.037070
289	11/6/20	1.036155
290	11/7/20	1.021594
291	11/8/20	1.021718
292	11/9/20	1.009554

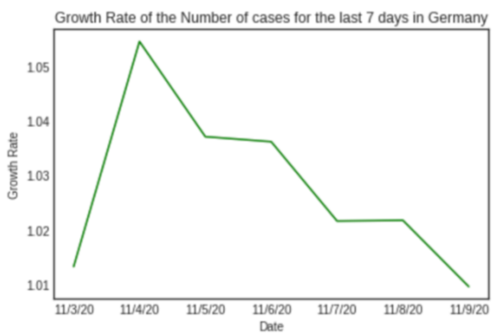
Then slice the data frame only for these last 7 days:

```
last_7_days = Germany_covid_19[['11/2/20','11/3/20','11/4/20','11/5/20','11/6/20','11/7/20','11/8/20','11/9/20']]
growth_rate_7_days_Germany = last_7_days.sum(axis=0).filter(like='/20').to_frame().pct_change() + 1
growth_rate_7_days_Germany.ewm(span=5).mean()
growth_rate_7_days_Germany.reset_index(inplace=True)
growth_rate_7_days_Germany.rename(columns={'index':'date', 0: 'rate'}, inplace=True)
growth_rate_7_days_Germany
```

	date	rate
0	11/2/20	NaN
1	11/3/20	1.013225
2	11/4/20	1.054546
3	11/5/20	1.037070
4	11/6/20	1.036155
5	11/7/20	1.021594
6	11/8/20	1.021718
7	11/9/20	1.009554

And tried to plot it:

```
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()
```



It is obvious that in last 7 days, at first Germany had raised significantly from the 1.013 and reached a pick of 1.05 but from then was successful to control disease and decreased it in a steady way to 1.009.

c) **Choose another country and compare the development of confirmed cases in the two countries. How do you adjust the timeseries so that the countries can be compared? For example:**

   a) How to shift the series on the time axis?

   b) Taking the countries' total population size into account?

For this question first I plotted the curves of the two Germany and Iran in the same figure and explained for it as below:

```
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()
```

Information from worldometers.info (9th November 2020):

#	Country, Other	Total Cases	New Cases	Total Deaths	New Deaths	Total Recovered	Active Cases	Serious, Critical	Tot Cases/ 1M pop	Deaths/ 1M pop	Total Tests	Tests/ 1M pop	Population
14	Iran	682,486		38,291		520,329	123,866	5,523	8,089	454	5,224,252	61,918	84,373,726
15	Germany	672,507		11,505		419,200	241,802	2,904	8,018	137	23,393,311	278,891	83,879,886

Information from worldometers.info (10th November 2020):

14	Germany	705,341	+16,369	11,853	+196	441,200	252,288	3,005	8,409	141	23,393,311	278,888	83,880,616
15	Iran	703,288	+10,339	39,202	+453	530,694	133,392	5,584	8,335	465	5,302,200	62,840	84,376,663

As it is shown in above picture two selected countries have almost same population with almost same confirmed cases. For comparison of development of confirmed cases in two countries, their curves are plotted in the same diagram. The Germany curve chose to be dashed and a little ticker than Iran curve to visualize differences in the better way. By selecting an artificial starting point in x-axis in 26th May 2020 gives this confidence that these two countries have already collected their confirmed cases on that time since. So, compare these two countries from the common point, and also it is obvious that these two curves in this time cross exactly each other, that shows they have exactly same confirmed cases.

- **But when more deeply think about the problem, it is obvious that this cannot be a fair comparison since Iran confirmed cases are starting later than Germany, so try in a different way:**

a) It would be interesting to think about the time line because in Iran confirmed cases is started 23 days later than Germany. So, when we want to compare the development over the time, we can adjust these two time lines, to compare them in a meaningful way. We can decide on the artificial starting point for example the day when the first case was reported and then move one of the curves that both of them have a same start in this artificial day.

b) Also, For this question there are other dimensions that can be consider, for example comparison to scale it down to the number of cases per 100,000 people or per 1 million people that it is regarding the numbers.

Steps for the First dimension (Timeseries):

As it is mentioned before confirmed case in Iran started 23 days after Germany. So, there is a need to shift firstly confirmed cases of Germany to the starting point of the Iran that is 2/19/2020.

First select the only Germany row from the whole dataframe. Then drop those columns in the timeseries which had not any amount of confirmed cases to reach to the first case of Germany. After that shifted the row 23 steps or days further to reach to in the starting point of Iran on 2/19/20, set the axis=1 to apply it for columns.
Some NaN values was appeared in the 23 last cells. So by dropna delete all of them. Also drop other columns which have string values on the date columns to easily plot only numbers, the result is as below:

```
[202] Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
      Germany_covid_19_start_first_case = Germany_covid_19.drop(columns=['1/22/20', '1/23/20', '1/24/20', '1/25/20', '1/26/20'])
      Germany_covid_19_start_first_case = Germany_covid_19_start_first_case.shift(23, axis = 1)
      Germany_covid_19_start_first_case_nan = Germany_covid_19_start_first_case.dropna(axis=1)
      Germany_covid_19_start_first_case_nan_dr = Germany_covid_19_start_first_case_nan.drop(columns=['2/16/20', '2/17/20', '2/18/20'])
      Germany_covid_19_start_first_case_nan_dr
```

	2/19/20	2/20/20	2/21/20	2/22/20	2/23/20	2/24/20	2/25/20	2/26/20	2/27/20	2/28/20	2/29/20	3/1/20	3/2/20	3/3/20	3/4/20	3/5/20	3/6/20	3/7/20	3/8/20	3/9/20	3/10/20	3/1
130	1	4	4	4	5	8	10	12	12	12	12	13	13	14	14	16	16	16	16	16	16	

1 rows × 265 columns

For the Iran also drop those columns with 0 values to start our curve only by started confirmed cases.

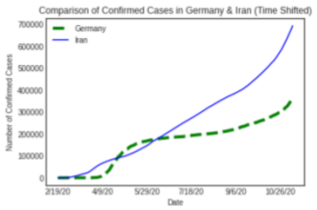
```
[ ] ran_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
    ran_covid_19_start_first_case = Iran_covid_19.drop(columns=['1/22/20', '1/23/20', '1/24/20', '1/25/20', '1/26/20', '1/27/20', '1/28/20', '1/29/20', '1/30/20', '1/31/20', '2/1/20', '2/2.
    ran_covid_19_start_first_case
```

	Province/State	Country/Region	Lat	Long	2/19/20	2/20/20	2/21/20	2/22/20	2/23/20	2/24/20	2/25/20	2/26/20	2/27/20	2/28/20	2/29/20	3/1/20	3/2/20	3/3/20	3/4/20	3/5
145	NaN	Iran	32.427908	53.688046	2	5	18	28	43	61	95	139	245	388	593	978	1501	2336	2922	

1 rows × 269 columns

Then like befor two curves are plotted in the same figure:

```
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()
```



Now we can see that the comparison regarding the time is more fair. In the last attemep it was assumed that two countries are same in increasing the numbers but now by shifting the Germany curve to the at the beginning of the start cases of Iran, it is obvious that Iran confiremd case are increaseng more rapdlily than germany since beginning, that today they reach almost at the same level. For example for the 26th october before changing the axis we had:

```
= 'Iran')]
```

10/26/20	1
574856	

```
= 'Germany')]
```

20	10/26/20	:
98	450258	

That looked a bit similar amount, but now we have:

```
== 'Iran')]
/22/20', '1/
```

10/26/20	1
574856	

```
Germany')]
22/20', '1/23/20',
shift(23, axis = 1)
ase.dropna(axis=1)
t_case_nan.drop(col
```

25/20	10/26/20	10
298374	300027	

Steps for the Second dimension:

In the above picture (today 10th November 2020) number of cases for Iran per 100.000 person is 833 persons, and for the Germany per 100.000 person is calculated as 840 persons.

```
Iran_populatin = 84376663
Germany_population = 83880616

Iran_confirmed_cases=703288
Germany_confirmed_case= 705341

Iran_per_100t = (100000 * Iran_confirmed_cases)/Iran_populatin
print(Iran_per_100t)

Germany_per_100t = (100000 * Germany_confirmed_case) /Germany_population
print(Germany_per_100t)
```

```
833.5100903433453
840.8867669736712
```

Since the timeseries was too big and the x-axis could not be visualize in a correct way sliced the data for the last 17 days. Then for the each cells wrote the number of confirmed cases per 100.000 people, based on the total population of that country.

```
[27] Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
     cell_Germany = Germany_covid_19.iloc[0:, 280:]
     cell_Germany
     Germany_population = 83880616
     Germany_per_100t = (100000 * cell_Germany )/Germany_population
     Germany_per_100t
```

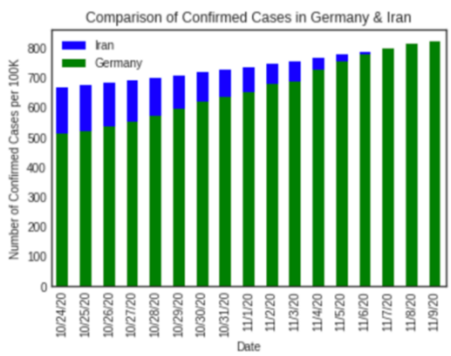
	10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	10/30/20	10/31/20	11/1/20	11/2/20	11/3/20	11/4/20	11/5/20	11/6/20	11/7/20	11/8/20	11/9/20
130	510.020098	521.810665	536.784327	552.474483	571.790031	594.122962	617.229611	633.984376	648.953269	679.05796	688.038581	725.56811	752.46467	779.670001	796.505834	813.804229	821.579565

```
Iran_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
cell_Iran = Iran_covid_19.iloc[0:, 280:]
cell_Iran
Iran_populatin = 84376663
Iran_per_100t = (100000 * cell_Iran )/Iran_populatin
Iran_per_100t
```

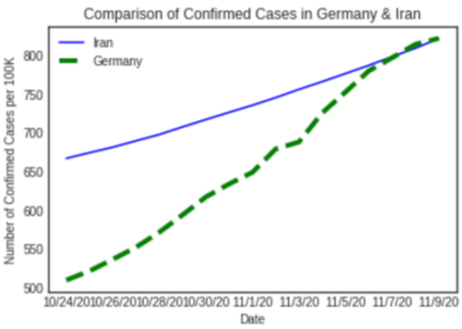
	10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	10/30/20	10/31/20	11/1/20	11/2/20	11/3/20	11/4/20	11/5/20	11/6/20	11/7/20	11/8/20	11/9/20
145	666.896485	674.233822	681.297387	689.555594	697.643139	707.471686	716.966017	726.233982	735.382247	745.206053	755.791918	765.808906	776.205146	786.71042	797.910199	808.856354	821.256702

In the last 17 days, germany confirmed cases was abit lower but now at the last day, both reach to the same level.

```
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()
```



```python
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()
```



Also, For whole the time series per 100K:

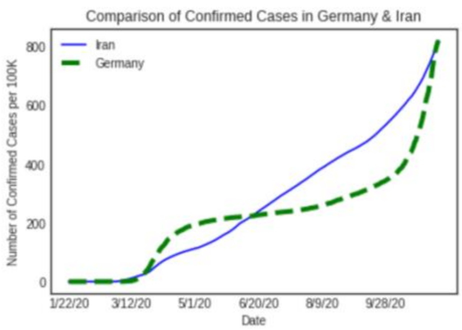
```python
Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
cell_Germany = Germany_covid_19.iloc[:, 2:]
cell_Germany
Germany_population = 83880616
Germany_per_100t = (100000 * cell_Germany )/Germany_population
Germany_per_100t
```

10/23/20	10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	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	11/9/20
497.552378	510.020098	521.810665	536.784327	552.474483	571.790031	594.122962	617.229611	633.984376	648.953269	679.05796	688.038581	725.56811	752.46467	779.670001	796.505834	813.804229	821.579565

```python
[38] Iran_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
cell_Iran = Iran_covid_19.iloc[:, 2:]
cell_Iran
Iran_populatin = 84376663
Iran_per_100t = (100000 * cell_Iran )/Iran_populatin
Iran_per_100t
```

10/23/20	10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	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	11/9/20
60.005954	666.896485	674.233822	681.297387	689.555594	697.643139	707.471686	716.966017	726.233982	735.382247	745.206053	755.791918	765.808906	776.205146	786.71042	797.910199	808.856354	821.256702

```python
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 for Both shifted curve and per 100K: (Extra Approach)

In the following codes, used both shifted Germany time series and confirmed cases per 100k:

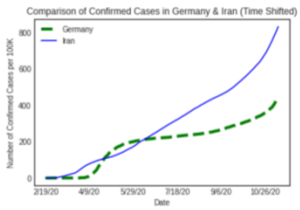
```
[90] Germany_covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Germany')]
     Germany_covid_19_start_first_case = Germany_covid_19.drop(columns=['1/22/20', '1/23/20', '1/24/20', '1/25/20', '1/26/20'])
     Germany_covid_19_start_first_case = Germany_covid_19_start_first_case.shift(23, axis = 1)
     Germany_covid_19_start_first_case_nan = Germany_covid_19_start_first_case.dropna(axis=1)
     cell_germany_time = Germany_covid_19_start_first_case_nan.drop(columns=['2/16/20', '2/17/20', '2/18/20'])
     Germany_population = 83880616
     Germany_per_100tt = (100000 * (cell_germany_time.filter(like='/20')) )/Germany_population
     Germany_per_100tt
```

10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	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	11/9/20	11/10/20
352.332892	355.712695	357.683353	359.526449	363.222178	366.147764	370.928368	376.655555	382.0847	385.623062	389.01598	396.813967	402.135816	410.687256	419.771595	428.945348	434.741681	439.518708

```
[91] _covid_19 = covid_19_data[(covid_19_data['Country/Region'] == 'Iran')]
     _Iran_time = Iran_covid_19.drop(columns=['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.
     _populatin = 84376663
     _per_100tt = (100000 * (cell_Iran_time.filter(like='/20')) )/Iran_populatin
     _per_100tt
```

10/24/20	10/25/20	10/26/20	10/27/20	10/28/20	10/29/20	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	11/9/20	11/10/20
666.896485	674.233822	681.297387	689.555594	697.643139	707.471686	716.966017	726.233982	735.382247	745.206053	755.791918	765.808906	776.205146	786.71042	797.910199	808.856354	821.256702	833.51009

```
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()
```



Bibliography:

https://pandas.pydata.org/pandas-docs/stable/index.html
https://matplotlib.org/3.1.0/gallery/style_sheets/style_sheets_reference.html
https://matplotlib.org/3.1.1/tutorials/introductory/customizing.html
https://www.worldometers.info/coronavirus/
https://joachim-gassen.github.io/2020/03/tidying-the-new-johns-hopkins-covid-19-datasests/
https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data
https://medium.com/@lindagcai/plot-a-covid-19-time-series-per-country-e43927dda6fa
https://openrefine.org/
https://openrefine.org/documentation.html
https://www.wikidata.org/w/index.php?search=Q21027972&title=Special%3ASearch&profile=advanced&fulltext=1&advancedSearch-current=%7B%7D&ns0=1&ns120=1.
https://libjohn.github.io/openrefine/hands-on-reconciliation.html
https://www.terena.org/activities/multiling/ml-docs/iso-8859.html
https://www.encyclopedia-titanica.org/titanic-survivor/hanna-mamee.html (As a Sample)