

## ICIDJ Investigations

### Scenario:

You are working as a journalist at the minor news outlet. It's a slow day. Suddenly, the Chief Editor calls you to attend a secret emergency meeting!

The International Consortium of Investigative Data Journalists (ICIDJ - made up name) has asked your organisation to check a [PDF document](#). It's a part of a large leak of documents which consists of records of the monetary transaction to secret offshore private accounts of corrupt politicians. ICIDJ suspects that the politicians from your country are also involved. The public does not know about the leak, and the newspapers around the world will jointly release their stories the week of 5th Nov 2020, in order to get the involved politicians caught off guard!

Since you are currently taking the Intro to Data Investigation & Storytelling course at UCD, you are the most qualified person in the newsroom for this task. Your task is to use all the skills you got so far to find the story in the data!

ICIDJ has informed you that the document that they have provided contains the following data: Name, Surname, Bank Account, Amount of Transaction, Code of the Transaction.

Your Chief Editor suggests that you should compare the names from the PDF documents with the public database of parliament members: <https://datathrills.github.io/dis/exercise/index.html>

Three controversial votes took place in the parliament during the last year. While communicating with the ICIDJ you have mutually concluded that it is highly likely that some members of parliament were bribed to vote a certain way by powerful lobbies!

Those highly controversial votes are:

1. Foreign electoral interference and disinformation in national and European democratic processes
2. Adjustments to the amounts mobilised from the Flexibility Instrument for 2019 to be used for migration, refugee inflows and security threats
3. Nuclear Power Strategy

Maybe this data can show which powerful lobby has used the money to influence the politics of your country!

### Assignment:

Analyse the PDF file and the online database to come up with the frontpage headline and subheading. We will discuss your findings in the class after reading week.

**Tips:**

1. Start by using Tabula to extract the pdf data into the table (download the pdf file from [here](#))
2. Use OpenRefine to clean this data:
  - a. Check for blanks
  - b. Check for duplicates
  - c. Remove unnecessary data
3. Use ParseHub to scrape the data from <https://datathrills.github.io/>
  - a. You will need the politician's name column and the data about their votes during the five previously mentioned controversial votes
4. Use OpenRefine to join names and surnames from PDF data, and transform them so they have the structure of your scraped data.
5. Now you can try to join the two datasets based on the "name, surname" cell (look for tutorials online on how to join two datasets by column in OpenRefine).
6. Find out who are the politicians who took bribes.
7. Use OpenRefine to check if the politicians voted the same during the five controversial votes, and thus trace the source of the bribe!

This document contains a step-to-step tutorial which will help you if you get stuck. If have any problems with the investigation, feel free to contact me for help: [marko.bralic@ucdconnect.ie](mailto:marko.bralic@ucdconnect.ie)

Note: All the data used in this exercise is purely fictional and was created solely for the purpose of this exercise

## Step-By-Step Tutorial

1. Use **Tabula** to extract the pdf data into the table.

1.1. Download and install Tabula from <https://tabula.technology/>  
(Windows & Linux users will need a copy of Java installed, you can find the instructions on how to do that on Tabula website)

## Tabula



Tabula is a tool for liberating data tables locked inside PDF files.

View the Project on GitHub  
tabulapdf/tabula

Download for  
**Windows**

Download for  
**Mac**

View source on  
**GitHub**

Current Version: [1.2.1](#)

Other Versions: [pre-releases & archives](#)

1.2. Once inside Tabula, import the PDF file (you can download the PDF from [here](#))

Import one or more PDFs

Browse...

Secret\_Transactions\_2020.pdf

Import

1.3 Press Autodetect Tables and Tabula will figure where are the tables in the PDF file. Then press Preview & Export Extracted Data.

[illegible]

## 1.4. Finally export the data as a CVS file

Export

Copy to Clipboard

Preview of Extracted Tabular Data

Mcbride	Echavarria	STSPAT2G133	758255¥   300446
Cinar	Hefty	VLRTATWW529	6223297¥   375327
Mathias	Haskew	AGRXATWW416	4185996\$   309262
Kaiden	Wee	VBOEATWWIN644	7501917€   927544
Flyn	Pulford	GIBAAATWWLA5728	7935510€   350981
Johnson	Gravenstein	STSPAT2G459	9560476¥   323570

## 2. Use OpenRefine to clean this data.

2.1. Follow the instruction and download OpenRefine from here:

<https://openrefine.org/download.html>

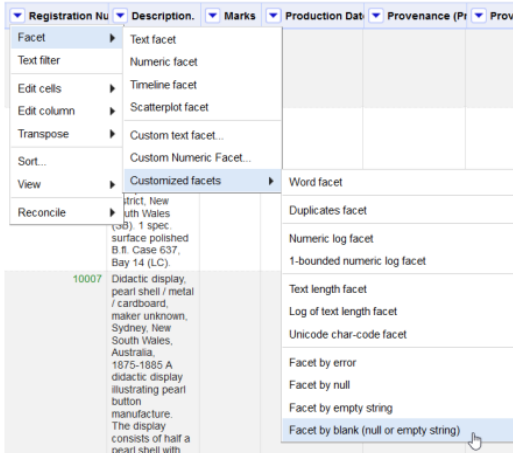
2.2. Import the CVS file we created. You can check the “Column names box” and write the column names in the form bellow like this. Click Crate project.

☒ Column names (comma separated):

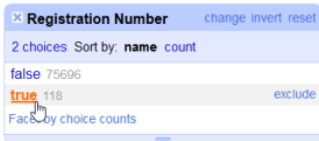
name, surname, account, amount\_code

2.3. First, let us check for any blanks (1). After you have found blanks, select them and remove them (2 & 3).

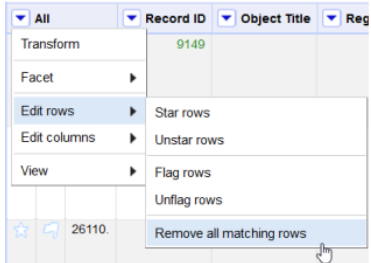
1.



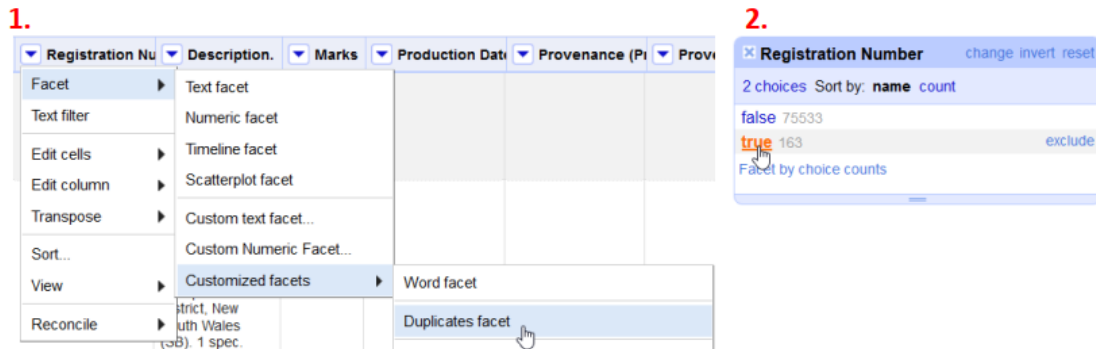
2.



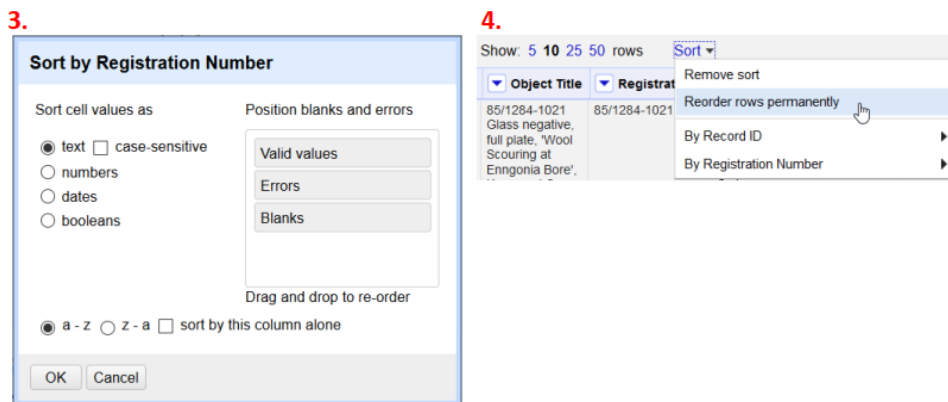
3.



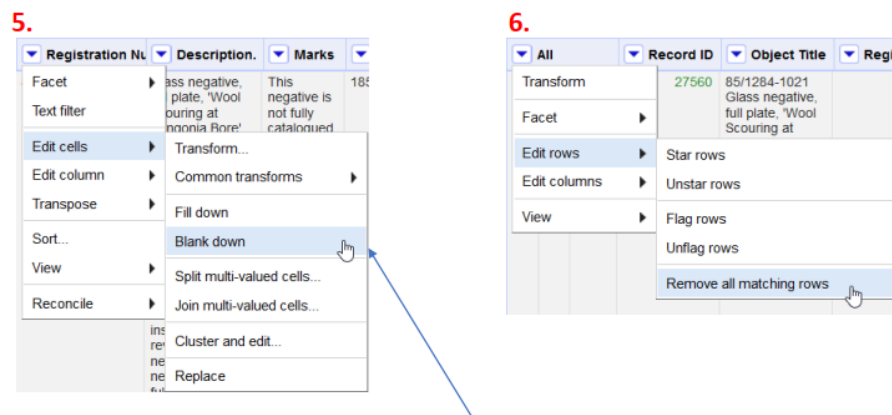
2.4. Removing duplicates is a bit trickier. First, let us check for any duplicates (1 & 2).



Then, click "Sort" and press OK (3). The duplicates are now sorted one above each other. Now select little "Sort" select menu that have appeared and choose "Reorder rows permanently" (4).



Finally, select "Edit cells" and click "Blank down" (5). Rows with the same content will be changed to blanks now. Same as before, you can now remove all blanks (6).



Blank down: if two rows that follow each other have the same content -> change the content of second row into *blank*

2.5. Now let us remove that “|” in front of the surnames. We will need this gone in order to connect this CVS with the data that we will scrape.

Select “Edit cells” and click “Transform”. In the expression box, write this simple formula: `value.replace("|", "")`

The screenshot shows the OpenRefine interface. On the left, the 'Edit cells' menu is open, and 'Transform...' is selected. The 'Custom text transform on column Categories 3' dialog is open. The 'Expression' box contains the formula `value.replace("|", "")`. The 'Language' dropdown is set to 'General Refine Expression Language (GREL)'. Below the expression box, there are tabs for 'Preview', 'History', 'Starred', and 'Help'. The 'Preview' tab is active, showing a table with two columns: 'row' and 'value'. The table has 6 rows. The first row is '1. null'. The second row is '2. Mineral Samples-Geological'. The third row is '3. Buttons|Didactic Displays'. The fourth row is '4. null'. The fifth row is '5. null'. The sixth row is '6. Botanical Specimens|Didactic Displays|Models'. The 'value' column shows the result of the transformation: 'null', 'Mineral Samples-Geological', 'Buttons,Didactic Displays', 'null', 'null', and 'Botanical Specimens,Didactic Displays,Models'. Below the table, there are options for 'On error': 'keep original' (selected), 'set to blank', and 'store error'. There is also a checkbox for 'Re-transform up to 10 times until no change'. At the bottom are 'OK' and 'Cancel' buttons.

2.6. The last column has multiple values. OpenRefine is simple to use for splitting columns if you have a symbol that can act as a separator. If you want you can try to split the last column into two, one for the “amount” and one for the “transaction code”. This will help you:

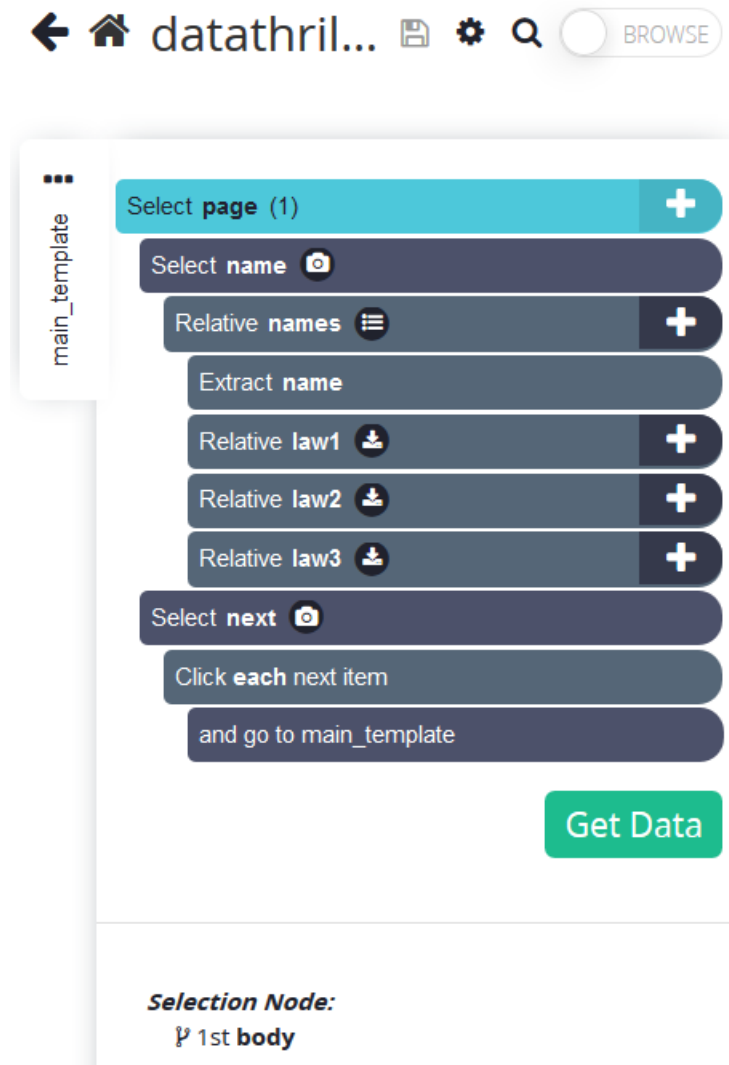
The screenshot shows the OpenRefine interface. On the left, the 'Edit column' menu is open, and 'Split into several columns...' is selected. The 'Split column Categories into several columns' dialog is open. The 'How to Split Column' section has two options: 'by separator' (selected) and 'by field lengths'. The 'by separator' option has a 'Separator' input box with a vertical bar '|' and a checkbox for 'regular expression'. The 'Split into' input box has the value '3' and the text 'columns at most (leave blank for no limit)'. The 'by field lengths' option has an empty input box and the text 'List of integers separated by commas, e.g., 5, 7, 15'. The 'After Splitting' section has two checkboxes: 'Guess cell type' and 'Remove this column', both of which are checked. At the bottom are 'OK' and 'Cancel' buttons.

3. Use **ParseHub** to scrape the data from <https://datathrills.github.io/dis/exercise/index.html>

3.1. Download ParseHub and Sign up (<https://www.parsehub.com/>)

3.2. ParseHub is a very powerful tool. I suggest that you go through the tutorials first before trying to scrape the data the “SEARCH VOTES BY MEMBERS OF PARLIAMENT” table.

3.3. This is the solution for scraping the table:



If you go through the tutorials on ParseHub it should make sense.

### 3.4. Download the scraped data from ParseHub as a CSV file.

Your data is ready! Click on the green buttons to download.

#### Download Data

CSV/Excel

JSON

API

Template Name

main\_template

Pages Scraped

30

### 4. Use **OpenRefine** to join names and surnames from PDF data, and transform them so they have the structure of your scraped data.

4.1. We have two datasets, one from the PDF and one that we scraped from the website. What we need to find out is which names appear on both datasets. We can do this by joining the two datasets in OpenRefine.

However, we can only join the datasets if the structure of the cells by which we will join them is identical. Currently, it is not. Our PDF dataset has two columns “name” and “surname”, and our scraped dataset has a single column with both name and a surname which structured like this: “name, surname”.

We will transform the PDF dataset so it matches the structure from the scraped dataset.

Open the PDF data set in OpenRefine.

#### 4.2. Select “Edit column” and click “Add column based on this column”.

Show as: <b>rows</b> records		Show: 5 10 25 50 rows		
All	name	surname	account	amount_code
1.	Facet		BTVAAT22LAN217	7822705â,~   134144
2.	Text filter		ASPKAT2LENN384	392922\$   263076
3.	Edit cells		HSEEAT2K844	75123\$   303554
4.	Edit column		ASPKAT2LFRE213	2668695Â¥   175083
5.	Transpose		Split into several columns...	631
6.	Sort...		Add column based on this column	986
7.	View		Add column by fetching URLs...	82
8.	Reconcile		Add columns from reconciled values...	116
9.			Rename this column	32
10.			Remove this column	327
11.			Move column to beginning	148
12.			Move column to end	272
13.			Move column left	9
14.			Move column right	023
15.				12
16.				130
17.				720



Then use this formula to set the values of the new column:

`value + "," + cells['surname'].value`

### Add column based on column name

New column name

On error
☒ set to blank
☐ store error
☐ copy value from original column

Expression

No syntax error.

Language

Preview

History

Starred

Help

row	value	value + \",\" + cells['surname'] ...
1.	Aaron	Aaron, Shugars
2.	Abubakar	Abubakar, Mackinaw
3.	Allesandro	Allesandro, Fredicks
4.	Andrejs	Andrejs, Baeza
5.	Aydon	Aydon, Hardegree
6.	Brandan	Brandan, Lasseigne
7.	Butali	Butali, Maltosa

OK

Cancel

4.3. Final step: select the new column and click “Add column based on this column”.

▼ All	▼ name	▼ name_surname	▼ surname	▼ account	▼ amount_code
1. Aaron	Facet	Shugars	BTVAAT22LAN217	7822705â~	134144
2. Abubakar	Text filter	Mackinaw	ASPKAT2LENN384	392922\$	263076
3. Allesandro	Edit cells	Fredicks	HSEET2K844	75123\$	303554
4. Andrejs	Edit column	Baeza	ASPKAT2LFRE213	2668695Â#	175083
5. Aydon	Split into several columns...			7â~	244631
6. Brandan	Transpose			2â~	797986
7. Butali	Add column based on this column...			7\$	827882
8. Butchi	Add column by fetching URLs...			6Â#	467116
9. Caidan	Add columns from reconciled values...			8\$	548432
10. Cinar	Reconcile			7Â#	375327
11. Cobi	Rename this column			1Â#	639148
12. Dan	Remove this column			3â~	520272
13. Daniele	Move column to beginning			5\$	182069
14. Daryl	Move column to end			4â~	594023
15. Dimitri	Move column left			8\$	836812
16. Drakeo	Move column right			8Â#	375130
17. Dylan-James				3â~	343720
18. Dylan-Patrick					

You now need to write the formula for for joining two datasets in OpenRefine. You can use this tutorial for help: <https://guides.library.illinois.edu/openrefine/joiningprojects>

The main part is understanding this part:

5. In the pop-up window, give the new column a name and then enter this expression in the GREL expression box:

```
cell.cross('arg1', 'arg2').cells['arg3'].value[arg4]
```

- arg1 = name of project you are exporting data from
- arg2 = name of the key column
- arg3 = name of the column you are importing
- arg4 = indicate which value to import in the array (if multiple matches for the key) (recommended to use 0)

For me, the implementation of this formula looks like this (but yours will depend on what names have you given to your columns and projects):

**Add column based on column name\_surname**

New column name

On error ☒ set to blank ☐ store error ☐ copy value from original column

Expression

cell.cross('run\_results  
csv', 'names\_name').cells['names\_name'].value[0]

Language General Refine Expression Language (GREL) ▼

No syntax error.

PreviewHistoryStarredHelp

row	value	cell.cross('run_results csv', ' ...
1.	Aaron, Shugars	null
2.	Abubakar, Mackinaw	null
3.	Allesandro, Fredicks	null
4.	Andrejs, Baeza	null
5.	Aydon, Hardegree	null
6.	Brandan, Lasseigne	null
7.	Ritali, Maltosa	Ritali, Maltosa

OK

Cancel

4.4. You are now able to see which politicians appear on both lists.

4.5. Go back to the scraped datasets and see on which law did they all voted the same – this suggests that the lobby interested in this law has bribed them 😊