



One University. One World. Yours.

MCDA 5560 BUSSINESS INTELLIGENCE AND DATA VISUALIZATION

Assignment 1 - Data Preparation

Team Members

Names	A#
Raghav Sharma	A00445168
Timur	A00439340
Kuzhippallil, Maria A.	A00442283

Contents

Data Preparation using OpenRefine	1
Data Preparation using Google Cloud.....	20
Comparison of all tools	39

Data Preparation using OpenRefine

1) Install OpenRefine as shown in the tutorial

2) Create Project question1 in OpenRefine

Steps –

- Creating project by loading the question1.csv file (*question 3*).
- Provide name for project by
- Check parse headers option to consider first row as header Parse next 1 line(s) as column headers
- Columns are separated by csv commas (CSV)
- Create project by clicking on

The screenshot shows the OpenRefine interface. At the top, there's a navigation bar with 'Create Project', 'Start Over', 'Configure Parsing Options', 'Project name: question1', 'Tags', and 'Create Project'. Below this is a table view of 11 rows of data from 'question1.csv'. The columns are: File, id, full_name, email, sex, birth_date, credit_card, lat, and lng. The data includes various names and their corresponding details like email addresses and coordinates. Below the table is a 'Parse data as' section with tabs for 'CSV / TSV / separator-based files', 'Line-based text files', 'Fixed-width field files', 'PC-Axis text files', 'JSON files', 'MARC files', 'JSON-LD files', 'RDF/N3 files', and 'RDF/N-Triples files'. It also contains settings for character encoding (US-ASCII), ignore first line(s), parse next line(s), discard initial row(s), load at most row(s), use character as separator, and store blank rows, blank cells as nulls, and file source in each row. On the left side, there are icons for 'Open Project', 'Import Project', 'Language Settings', and 'Version 3.4.1 [437dc4d]'. At the bottom left, there are links for 'Preferences', 'Help', and 'About'.

Fig – Create question1 project

This screenshot shows the '1000 rows' view of the 'question1' project. At the top, it says '1000 rows' and 'Show as: rows records Show: 5 10 25 50 rows'. Below is a table with 10 rows of data, identical to the one in the previous screenshot. The columns are: All, File, id, full_name, email, sex, birth_date, credit_card, lat, and lng. The data includes names like Arnie Burdis, Piotr Burwhistle, Elizabeth Ferrillo, Wesley Gathwaite, Malvina Coonhan, Barnie Dulany, Dacey Floris, Marten Edwirthie, Valentine Couling, and Kriste Beardsdale, along with their respective details. On the left, there's a sidebar titled 'Using facets and filters' with instructions on how to use facets and filters to select subsets of data. There are also links to 'Watch these screencasts' and 'Extensions: Wikidata'.

Fig – Create question1 project

OpenRefine question1 Permalink

Remove column File Undo

Facet / Filter Undo / Redo 1 / 1 Extract... Apply... Filter: 0. Create project 1. Remove column File

1000 rows Show as: rows records Show: 5 10 25 50 rows Extensions: Wikidata ▾

« first < previous 1 - 10 next > last »

All	id	full_name	email	sex	birth_date	credit_card	lat	lng
1.	1	Arnie Burdis	aburdis0@army.mil	Male	31.01.1992	5428362425731160	38.0230869	23.8724272
2.	2	Piotr Burwhistle	pburwhistle1@ftc.gov	Male	07.11.1989	5100141984484140	40.0039328	-8.5808436
3.	3	Elizabeth Ferrillo	eferrillo2@latimes.com	Female	19.03.1982	3537771455193920	10.3307399	123.8822107
4.		Wesley Gathwaite	wgathwaite3@eepurl.com	Male	02.12.1990	3583253989463300	45.6409009	14.8633128
5.	5	Malvina Coonihan	mcoonihan4@constantcontact.com	Female	28.08.2005	3528833562374970	43.2	82.63333
6.	6	Barnie Dulany	bdulany5@abc.net.au	Male	15.09.1982	3552732418082180	18.159131	-77.7652346
7.	7	Dacey Floris	dfloris6@columbia.edu	Female	06.02.1996	67634849985878000	54.0058465	17.7763729
8.	8	Marten Edworfie	medworfie7@bloglovin.com	Male	07.02.1990	3541393658076420	16.3112941	104.8221147
9.	9	Valentine Couling	vcouling8@lhs.gov	Male	14.05.1999	3580471384929910	-7.2538172	-79.129784
10.	10	Kriste Beardsdale	kbeardsdale9@imgur.com	Female	13.08.2005	670692415427828000	-25.99566	28.1971031

Fig – Deleting File column

Row count after loading data → 1000

3) Load question1.csv

Steps –

- Click on choose files question1.csv and load question1.csv.
 - Then go ahead by clicking next

 **OpenRefine** A power tool for working with messy data.

[Create Project](#)
[Open Project](#)
[Import Project](#)
[Language Settings](#)

Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported. Support for other formats can be added with OpenRefine extensions.

Get data from question1.csv

This Computer

[Web Addresses \(URLs\)](#)
[Clipboard](#)
[Database](#)
[Google Data](#)

Fig – Load question1.csv

4) Remove all rows in which id is empty

Steps –

- Use drop down option next to id .
 - Choose facet, followed by customized facet then facet by blank to find empty records for id.



- Select true **true** which satisfies the criteria of finding the blank records.
 - Remove matching records by selecting dropdown of all followed by edit rows and remove matching rows. 

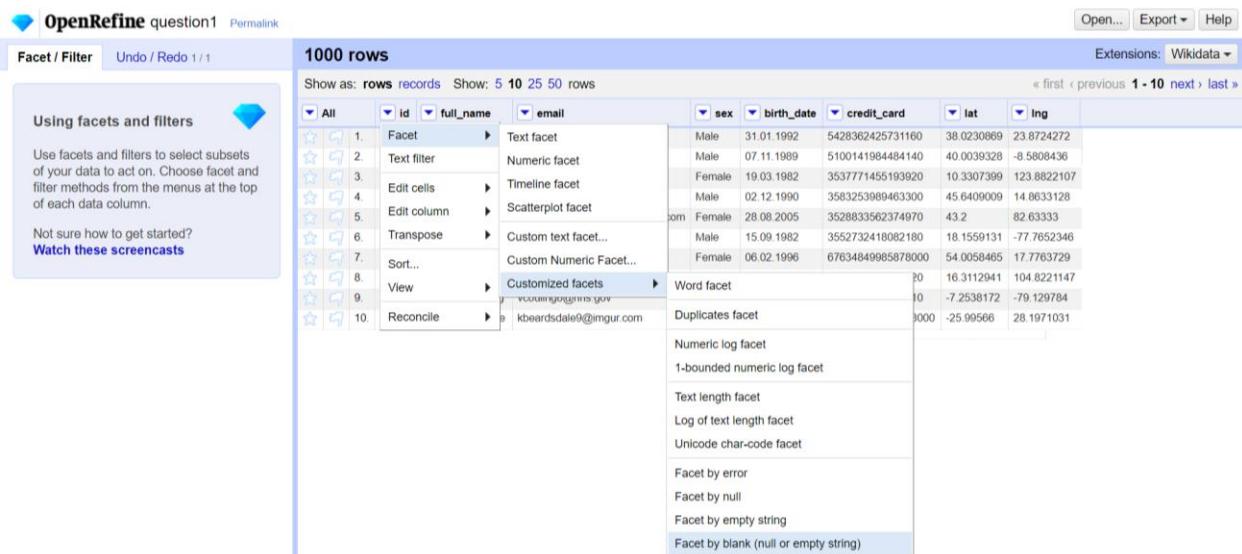


Fig – Facet by blank

All	id	full_name	email	sex	birth_date	credit_card	lat	lng
4.		Wesley Gathwaite	wgathwaite3@eepurl.com	Male	02.12.1990	3583253989463300	45.6409009	14.8633128
15.		Marena Passie	mpassiee@wikimedia.org	Female	24.09.1983	4844308271417770	4.625907	-75.761885
156.		Burton Daniels	bdanielsfb@google.ru	Male	18.04.1981	3537937508555800	36.368888	25.4633333
213.		Roze Abbate	rabbate5w@desdev.cn	Female	04.01.1989	374622337871882	19.4779768	-99.1771001
221.		Lee Firebrace	lfirebrace64@mapquest.com	Male	11.04.1995	3543971471660520	35.6958419	139.5530816
262.		Costa Viste	cviste79@turtl.net	Male	16.10.1990	3589608517437690	-6.5881066	105.6823124
290.		Dilly Duckels	dduckels1@nifty.com	Male	08.12.1999	3537055498242610	33.86	115.484454
307.		Ursa Krolman	ukrolman8@hao123.com	Female	26.12.1981	3577215916588750	38.7121599	-9.4356652
418.		Billy Crumby	bcrumbyb@rucoz.ru	Female	02.05.1997	5100136196986450	36.4126577	28.1554071
429.		Shaun Orchard	sorchartbw@slideshare.net	Female	03.10.1986	30288372702238	43.9336277	42.5107159

Fig – 27 records with null value for id

Fig – Removing id with null value

Fig – 0 records with id null value

Fig – Removing id with null value

Row count after deleting id with null/blank values → 973

5) Remove all rows in which email is empty

Steps –

- Use drop down option next to email .
- Choose facet, followed by customized facet then facet by blank to find empty records for email. → →
- Select true which satisfies the criteria of finding the blank records.
- Remove matching records by selecting dropdown of all followed by edit rows and remove matching rows. → →

The screenshot shows the OpenRefine interface with a table containing 973 rows. The 'email' column has a context menu open, leading to 'Facet' > 'Customized facets' > 'Facet by blank (null or empty string)'. This highlights the process of identifying rows where the email field is empty.

Fig – Facet by blank

The screenshot shows the OpenRefine interface after applying the 'Facet by blank' filter. It displays 33 matching rows out of a total of 973. The 'email' column is highlighted, showing that 33 rows have an empty value.

Fig – 33 blank records for email id field

The screenshot shows the OpenRefine interface after removing the 33 blank email records. The total row count is now 940. The 'email' column is highlighted, showing that all rows now have a valid email address.

Fig – Removing email id with null values

Row count after deleting id with null/blank values → 940

6) Split Full Name in to First name and Last Name. Store first name into f_name and last name into l_name (you will create these two columns).

Steps –

- Select split column option **Split column full_name into several columns**.
- Provide separator i.e “ ” and provide the number of columns to split into.
- Check remove this column Remove this column
- Rename the columns to f_name and l_name.

Fig – Splitting full name column with space separator

Fig – Full name column split to full_name2 and full_name2

Fig – Renaming to f_name

The screenshot shows the OpenRefine interface with a project titled "OpenRefine question1". The main pane displays 940 rows of data. A context menu is open over the first few rows, specifically targeting the "full_name" column. The menu path "Rename column full_name 2 to l_name" is highlighted. Other options in the menu include "Text filter", "Edit cells", "Edit column", "Transpose", "Sort...", "View", and "Reconcile". The right pane shows a preview of the data with columns: th_date, credit_card, lat, and lng.

Fig – Renaming to l_name

The screenshot shows the OpenRefine interface with a project titled "OpenRefine question1". The main pane displays 940 rows of data. The "full_name" column has been renamed to "l_name" and split into "f_name" and "l_name". The data now includes columns: id, f_name, l_name, email, sex, birth_date, credit_card, lat, and lng. The right pane shows a preview of the data with the same columns as before: th_date, credit_card, lat, and lng.

Fig – Removed full_name column and split to f_name and l_name

Row count after creating f_name and l_name columns from full_name → 940

7) Code Gender (sex column) into M and F instead of Male and Female (Do not create a new column).

Steps –

- Choose replace **Replace** for sex column.
- Find the value to be replaced **Find:** and provide new value to replace with **Replace with:** for both Male and Female with M and F.
- Then click ok **OK** to apply changes.

Fig – Replacing Male to M

Fig – Transformed Male to M

Fig – Replacing Female to F

OpenRefine question1 [Permalink](#)

Facet / Filter Undo / Redo 8 / 8 Extract... Apply...

Text transform on 460 cells in column sex:
value.replace("Female","F") [Undo](#)

940 rows Show as: rows records Show: 5 10 25 50 rows « first < previous 1 • 10 next > last »

All	ID	f_name	l_name	email	sex	birth_date	credit_card	lat	lng
1.	1	Arnie	Burdis	aburdis0@army.mil	M	31.01.1992	5428362425731160	38.0230869	23.8724272
2.	2	Piotr	Burwhistle	pburwhistle1@fc.gov	M	07.11.1989	5100141984484140	40.0039328	-8.5808436
3.	3	Elizabeth	Ferrillio	eferrillio2@latimes.com	F	19.03.1982	3537771455193920	10.3307399	123.8822107
4.	5	Malvina	Coonihan	mcoonihan4@constantcontact.com	F	28.08.2005	3528833562374970	43.2	82.63333
5.	6	Barnie	Dulany	bdulany5@abc.net.au	M	15.09.1982	3552732418082180	18.1559131	-77.7652348
6.	7	Dacey	Floris	dfloris6@columbia.edu	F	06.02.1998	67634849985878000	54.0058465	17.7763729
7.	8	Marten	Edworfthe	medworfthe7@bloglovin.com	M	07.02.1990	3541393658076420	16.3112941	104.8221147
8.	9	Valentine	Couling	vcouling8@hhs.gov	M	14.06.1999	3580471384929910	-7.2538172	-79.129784
9.	10	Kriste	Beardsdale	kbeardsdale9@imgur.com	F	13.08.2005	670692415427828000	-25.99566	28.1971031
10.	11	Cordy	Joselevitz	cjoselevitz@icio.us	M	15.03.1982	6383212459918100	5.9779631	-0.0886948

Filter:

0. Create project
1. Remove column File
2. Remove 27 rows
3. Remove 33 rows
4. Split 940 cell(s) in column full_name into several columns by separator
5. Rename column full_name 1 to f_name
6. Rename column full_name 2 to l_name
7. Text transform on 480 cells in column sex:
value.replace("Male","M")
8. Text transform on 480 cells in column sex:
value.replace("Female","F")

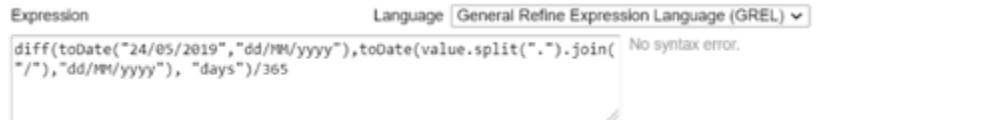
Fig – Transformed Female to F

Row count after transforming Male and Female to “M” and “F” respectively → 940

8) Calculate age of the person on 24th May 2019 from the birth_date column. Store calculated age in age column (You will create this column).

Steps –

- Choose Add column based on this column.
- Provide new column name “age”
- Use GREL to determine the age.



- Click on ok to create the new age column.

The dialog box is titled 'Add column based on column birth_date'. It shows the following settings:

- New column name:
- On error: set to blank
- Expression:
- Language: General Refine Expression Language (GREL)
- Preview section shows the first 7 rows of data with their corresponding ages.

Fig – Adding age column based on birth_date

The data table has a yellow header bar stating: 'Create new column age based on column birth_date by filling 940 rows with gret:diff(toDate(\"24/05/2019\", \"dd/MM/yyyy\"), toDate(value.split(\".\").join(\"/\"), \"dd/MM/yyyy\"), \"days\")/365'.

	id	f_name	l_name	email	sex	birth_date	age	credit_card	lat	lng
1	1	Arnie	Burdis	aburdis0@army.mil	M	31.01.1992	27	5428362425731160	38.0230869	23.8724272
2	2	Piotr	Burtwhistle	pburwhistle@flic.gov	M	07.11.1989	29	510014984484140	40.0093228	-8.5808436
3	3	Elizabeth	Ferrillio	eferrillo2@latimes.com	F	19.03.1982	37	353777145193920	10.3307399	123.8822107
4	5	Malvina	Coonihan	mcoonian4@constantcontact.com	F	28.08.2005	13	352833562374970	43.2	82.83333
5	6	Barrie	Dulany	bduley9@abc.net.au	M	15.09.1982	36	3552732418082180	18.1559131	-77.7652346
6	7	Dacey	Floris	dfloris6@columbia.edu	F	06.02.1996	23	67634849965878000	54.0058465	17.7763729
7	8	Marten	Edworthie	medworthie7@bloglovin.com	M	07.02.1990	29	3541393658078420	16.3112941	104.8221147
8	9	Valentine	Couling	vcouling8@hhs.gov	M	14.05.1999	20	3580471384929910	-7.2538172	-79.129784
9	10	Kriste	Beardsdale	kbeardsdale9@imgur.com	F	13.08.2005	13	670692415427828000	-25.99566	28.1971031
10	11	Cordy	Joselevitz	cjoselevitz@cio.us	M	15.03.1982	37	6383212459918100	5.9779631	-0.0886948

Fig – Age column added

Row count after adding age column → 940

9) Only keep last 4 digits of credit card in the credit_card column (Do not create a new column).

Steps –

- Choose custom text transform.
- Provide the expression `value.slice(-4)`. (Slice -4 provides the last four digits)
- Then click ok  to apply changes.

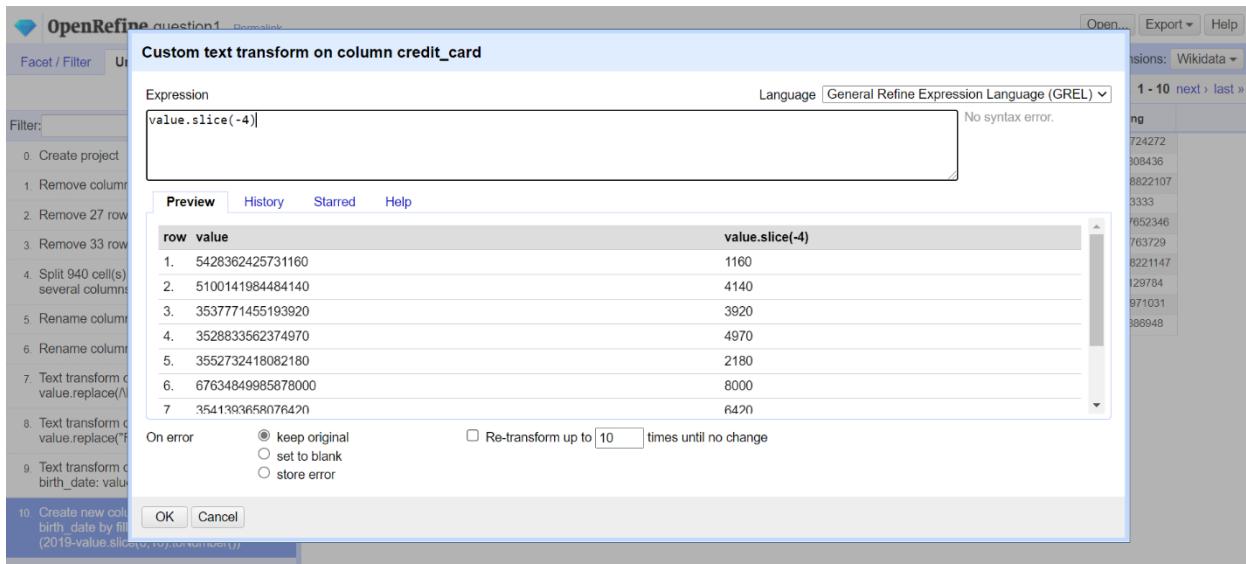


Fig – Slicing last 4 digits of credit card

Text transform on 940 cells in column credit_card: grel:value.slice(-4) Undo													
940 rows													
Show as: rows records Show: 5 10 25 50 rows													
All	#	id	f_name	i_name	email	sex	birth_date	age	credit_card	lat	long		
1.	1	Annie	Burdis	aburdis0@army.mil	M	31.01.1992	27	1160	38.0230869	23.8724272			
2.	2	Piotr	Burtwistle	pburtwistle1@ftc.gov	M	07.11.1989	29	4140	40.0039328	-8.5808436			
3.	3	Elizabeth	Ferrillo	eferrillo2@latimes.com	F	19.03.1982	37	3920	10.3307399	123.8822107			
4.	5	Malvina	Coonihan	mcoonihan4@constantcontact.com	F	28.08.2005	13	4970	43.2	82.63333			
5.	6	Barnie	Dulany	bdulany5@abc.net.au	M	15.09.1992	36	2180	18.1559131	-77.7652346			
6.	7	Dacey	Floris	dfloris6@columbia.edu	F	06.02.1998	23	8000	54.0059465	17.7783729			
7.	8	Marten	Edworthie	medworthie7@bloglovin.com	M	07.02.1990	29	6420	16.3112941	104.8221147			
8.	9	Valentine	Couling	vcouling8@hhs.gov	M	14.05.1999	20	9910	-7.2538172	-79.129784			
9.	10	Krista	Beardsdale	kbeardsdale9@imgur.com	F	13.08.2005	13	8000	-25.99956	28.1971031			
10.	11	Cordy	Josellevitz	cjosellevitz10@icio.us	M	15.03.1982	37	8100	5.9779631	-0.0886948			

Fig – Transformed credit card containing last 4 values

Row count after transforming credit card values → 940

10) Delete full_name column.

Note - For solving question 6 while splitting the full name column is removed by checking this option Remove this column

The screenshot shows the OpenRefine interface with a project titled "question1". The main pane displays 940 rows of data with columns: All, Id, full_name, email, sex, birth_date, credit_card, lat, and lng. A context menu is open over the "full_name" column, with the option "3. Remove 33 rows" selected. A modal dialog box titled "Split column full_name into several columns" is displayed. It contains two sections: "How to Split Column" (radio button selected for "by separator", separator field is empty, split into 2 columns at most) and "After Splitting" (checkboxes for "Guess cell type" and "Remove this column" are checked). Below the dialog is a list of integers separated by commas: 5, 7, 15. At the bottom are "OK" and "Cancel" buttons.

Row count after deleting full_name → 940

11) Delete birth_date column.

Steps –

- Delete birth_date by selecting the column birth_date, then select followed by

The screenshot shows the OpenRefine interface with a table containing 940 rows. The columns are labeled: id, f_name, l_name, email, sex, birth_date, age, credit_card, lat, and lng. The 'birth_date' column is currently selected, as indicated by the blue border around its header. A context menu is open over this column, listing various options for editing and managing it. The 'Edit column' option is highlighted in blue.

Fig – Removing birth_date column

The screenshot shows the OpenRefine interface after the 'birth_date' column has been deleted. The 'Remove column birth_date' button is highlighted in yellow at the top of the screen. The table now displays 940 rows. The columns present are id, f_name, l_name, email, sex, age, credit_card, lat, and lng. The data in the table remains the same as in the previous screenshot, but the column structure has been modified.

Fig – Data after deleting birth_date

Row count after deleting birth_date → 940

12) Signup for OpenCage Reverse Geocoding API (<https://opencagedata.com/api>) and get API Key.

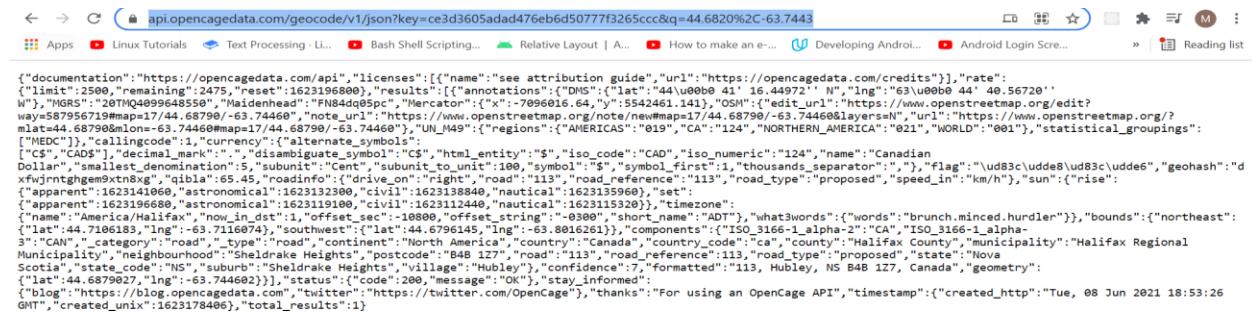
13)URL format to retrieve location

(JSON Data): <https://api.opencagedata.com/geocode/v1/json?q=LAT+LNG&key=YOUR-API>

14)You can replace the text highlighted in red using expression and retrieve location detail (JSON data). You can store this JSON data into raw_data column (You will create this column).

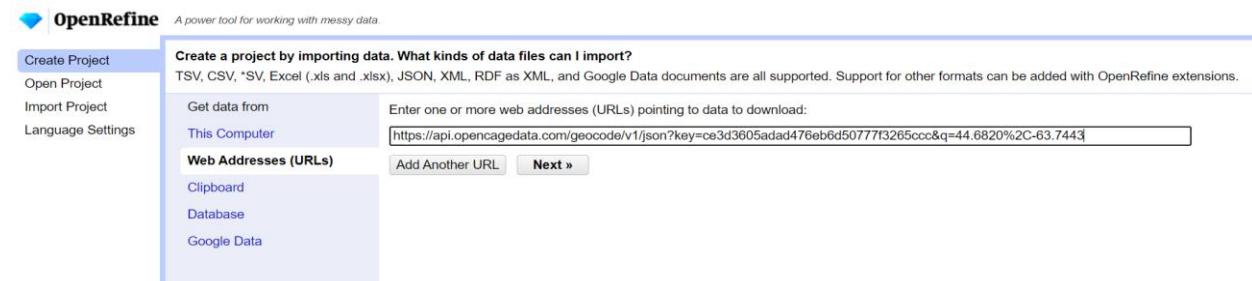
Steps –

- Replacing the latitude and longitude of Halifax in the URL -
<https://api.opencagedata.com/geocode/v1/json?key=ce3d3605adad476eb6d50777f3265ccc&q=44.6820%2C-63.7443>
- Create a project with the above web addresses URL.
- Parse data as CSV/TSV/separator-based files by selecting the option CSV / TSV / separator-based files
- Change column name to raw data by selecting



```
{"documentation": "https://opencagedata.com/api", "licenses": [{"name": "See attribution guide", "url": "https://opencagedata.com/credits"}], "rate": {"limit": 2500, "remaining": 2475, "reset": 1623196800}, "results": [{"annotations": {"DMS": {"lat": "44\u00b008'41\" N", "lng": "\u00b33\u00b008\u2032 44' 40.56720' W"}, "IGRS": "20TMQ4099648550", "Maidenhead": "FN84dq05pc", "Mercator": "X:-70960164,Y:-5542461141}, "OSM": {"edit_url": "https://www.openstreetmap.org/edit?way=587956719#map17/44.68790/-63.74460"}, "note_ur1": "https://www.openstreetmap.org/note/new#map=17/44.68790/-63.74460&layers=N", "ur1": "https://www.openstreetmap.org/?lat=44.68790&lon=-63.74460"}, "UN_M49": {"regions": {"AMERICAS": "019", "CA": "124", "NORTHERN_AMERICA": "021", "WORLD": "001"}, "statistical_groupings": [{"CA": "019"}, {"callingcode": "+1", "country": "Canada", "symbol": "\u262e"}, {"sym": "\u262e"}], "disambiguate": "true"}, {"set": [{"apparent": "1623141860", "astronomical": "1623132300", "civil": "1623138840", "nautical": "1623135960"}, {"set": [{"apparent": "1623196680", "astronomical": "1623119100", "civil": "1623112440", "nautical": "1623115320"}]}, {"timezone": {"name": "America/Halifax", "now_in_dst": 1, "offset_sec": "-10800", "short_name": "ADT"}, "what3words": {"words": "brunch.minced.hundler"}, "bounds": {"northeast": {"lat": 44.7106183, "lng": -63.7116074}, "southwest": {"lat": 44.6796145, "lng": -63.8086261}}, "components": {"ISO_3166-1_alpha-2": "CA", "ISO_3166-1_alpha-3": "CAN", "category": "road", "continent": "North America", "country": "Canada", "country_code": "ca", "county": "Halifax County", "municipality": "Halifax Regional Municipality", "neighborhood": "Sheldrake Heights", "postcode": "B4B 1Z7", "precise": "113", "road_reference": "113", "road_type": "proposed", "state": "Nova Scotia", "state_code": "NS", "street": "High Street", "village": "High Street"}, "geometry": {"lat": 44.688267, "lng": -63.744602}], "status": {"code": 200, "message": "OK"}, "stay_informed": {"blog": "https://blog.opencagedata.com", "twitter": "https://twitter.com/OpenCage"}, "thanks": "For using an OpenCage API", "timestamp": {"created_http": "Tue, 08 Jun 2021 18:53:26 GMT", "created_unix": 1623178406}, "total_results": 1}
```

Fig – JSON data



OpenRefine A power tool for working with messy data.

Create Project Open Project Import Project Language Settings

Create a project by importing data. What kinds of data files can I import? TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are all supported. Support for other formats can be added with OpenRefine extensions.

Get data from This Computer Enter one or more web addresses (URLs) pointing to data to download:
<https://api.opencagedata.com/geocode/v1/json?key=ce3d3605adad476eb6d50777f3265ccc&q=44.6820%2C-63.7443>

Web Addresses (URLs) Add Another URL Next »

Clipboard Database Google Data

Fig – Creating project from URL

Project name: jsonquestion1 Tags Create Project

raw_data

```
1. {"documentation": "https://opencagedata.com/api/v1/licenses", "licenses": [{"name": "see attribution guide", "url": "https://opencagedata.com/credits"}], "rate": {"limit": 2500, "remaining": 2473, "reset": 1623196800}, "results": [{"lat": 44.0008041, "lon": -63.44972, "name": "63u00b0 44\u00b0 56'72\"W"}, "MGRS": "20TMQ4099648550", "Maidenhead": "FN84dg05pc", "Mercator": {"x": -7096016.64, "y": 5542461.141}, "OSM": {"edit_url": "https://www.openstreetmap.org/edit?way=587956719#map=17/44.68790,-63.74460&layers=N"}, "note_ur": "https://www.openstreetmap.org/notes?map=17/44.68790,-63.74460", "note": "note", "url": "https://www.openstreetmap.org/notes?map=17/44.68790,-63.74460#0011", "URL": "https://www.openstreetmap.org/regions?region_id=124", "NAME": "NORTHERN AMERICA", "COUNTRY": "021", "WORLD": "001", "statistical_groupings": "MEDC", "callingcode": "+1", "currency": "CAD", "alternate_symbols": "CAD\u20ac", "html_entity": "\u20ac", "iso_code": "CAD", "iso_numeric": "124", "name": "Canadian Dollar", "smallest_denomination": 5, "subunit": "Cent", "subunit_to_unit": 100, "symbol": "$", "symbol_first": 1, "thousands_separator": ","}, "flag": "ud83cudde6", "geohash": "dxwjmgtghemjtn8xg", "drive_on": "right", "road": "113", "road_reference": "113", "road_type": "proposed", "speed": "in", "km/h": "sun", "rise": {"apparent": 1623141060, "astronomical": 1623119100, "civil": 1623112440, "nautical": 1623115320}, "timezone": "America/Halifax", "now_in_dst": 1, "offset_sec": -10800, "offset_string": "-03:00", "short_name": "ADT"}, "what3words": {"words": "brunch.minced.hurdler"}, "bounds": {"northeast": {"lat": 44.7106183, "lng": -63.7116074}, "southwest": {"lat": 44.6796145, "lng": -63.8016261}}, "components": [{"ISO": "3166-1_alpha-2": "CA", "ISO": "3166-1_alpha-3": "CAN", "category": "road", "type": "road", "cont_America": "country", "Canada": "country_code", "ca": "county", "Halifax County": "municipality", "neighbourhood": "Sheldrake Heights", "postcode": "B4B 1Z7", "road": "113", "road_reference": "113", "road_type": "proposed"}, {"state": "Nova Scotia", "state_code": "NS", "suburb": "Sheldrake Heights", "village": "Hubley"}, "confidence": 7, "formatted": "113, Hubley, NS B4B 1Z7", "status": "code: 200", "message": "OK"}], "timestamp": "2021-06-08T19:00:03Z", "created_http": "https://blog.opencagedata.com/tutorials/json-questionnaire.html", "created_unix": 1623178803, "total_results": 1}
```

Parse data as Character encoding US-ASCII Update Preview

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

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

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

Parse cell text into numbers, dates, ... Store blank rows Store blank cells as nulls Store file source (file names, URLs) in each row

Fig – Stored json data in raw_data column

OpenRefine jsonquestion1 Permalink

Facet / Filter Undo / Redo 0 / 0

Using facets and filters

Show as: rows records Show: 5 10 25 50 rows Extensions: Wikidata ▾

1 rows

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

All raw_data

1. {"documentation": "https://opencagedata.com/api/v1/licenses", "licenses": [{"name": "see attribution guide", "url": "https://opencagedata.com/credits"}], "rate": {"limit": 2500, "remaining": 2473, "reset": 1623196800}, "results": [{"lat": 44.0008041, "lon": -63.44972, "name": "63u00b0 44\u00b0 56'72\"W"}, "MGRS": "20TMQ4099648550", "Maidenhead": "FN84dg05pc", "Mercator": {"x": -7096016.64, "y": 5542461.141}, "OSM": {"edit_url": "https://www.openstreetmap.org/edit?way=587956719#map=17/44.68790,-63.74460&layers=N"}, "note_ur": "https://www.openstreetmap.org/notes?map=17/44.68790,-63.74460#0011", "note": "note", "url": "https://www.openstreetmap.org/notes?map=17/44.68790,-63.74460", "note": "note", "url": "https://www.openstreetmap.org/regions?region_id=124", "NAME": "NORTHERN AMERICA", "COUNTRY": "021", "WORLD": "001", "statistical_groupings": "MEDC", "callingcode": "+1", "currency": "CAD", "alternate_symbols": "CAD\u20ac", "html_entity": "\u20ac", "iso_code": "CAD", "iso_numeric": "124", "name": "Canadian Dollar", "smallest_denomination": 5, "subunit": "Cent", "subunit_to_unit": 100, "symbol": "\$", "symbol_first": 1, "thousands_separator": ","}, "flag": "ud83cudde6", "geohash": "dxwjmgtghemjtn8xg", "drive_on": "right", "road": "113", "road_reference": "113", "road_type": "proposed", "speed": "in", "km/h": "sun", "rise": {"apparent": 1623141060, "astronomical": 1623119100, "civil": 1623112440, "nautical": 1623115320}, "timezone": "America/Halifax", "now_in_dst": 1, "offset_sec": -10800, "offset_string": "-03:00", "short_name": "ADT"}, "what3words": {"words": "brunch.minced.hurdler"}, "bounds": {"northeast": {"lat": 44.7106183, "lng": -63.7116074}, "southwest": {"lat": 44.6796145, "lng": -63.8016261}}, "components": [{"ISO": "3166-1_alpha-2": "CA", "ISO": "3166-1_alpha-3": "CAN", "category": "road", "type": "road", "cont_America": "country", "Canada": "country_code", "ca": "county", "Halifax County": "municipality", "neighbourhood": "Sheldrake Heights", "postcode": "B4B 1Z7", "road": "113", "road_reference": "113", "road_type": "proposed"}, {"state": "Nova Scotia", "state_code": "NS", "suburb": "Sheldrake Heights", "village": "Hubley"}, "confidence": 7, "formatted": "113, Hubley, NS B4B 1Z7", "status": "code: 200", "message": "OK"}], "timestamp": "2021-06-08T19:00:03Z", "created_http": "https://blog.opencagedata.com/tutorials/json-questionnaire.html", "created_unix": 1623178803, "total_results": 1}

Fig – Data

15) Filter continent and country from the JSON data and store into continent and country columns (You will create these two columns).

Steps –

- Add column based on column raw_data
- Choose new column name (country and continent)
- Choose an expression `forEach(value.parseJson().results,v,v.components.country).join(",")` to obtain the country and continent.
- New column country and continent will be added to the data.

The screenshot shows the 'Add column based on column raw_data' dialog in OpenRefine. The 'New column name' is set to 'country'. The 'Expression' field contains the GREL expression: 'forEach(value.parseJson().results,v,v.components.country).join(",")'. The preview pane shows the resulting country names for the first few rows of the data.

Fig – Adding country column

The screenshot shows the OpenRefine interface after adding the 'country' column. The message 'Create new column country based on column raw_data by filling 1 rows with grel:forEach(value.parseJson().results,v,v.components.country).join(',')' is displayed. The data table now includes a 'country' column with values like 'Canada' and 'United States'.

Fig – Added country column



Fig – Country column

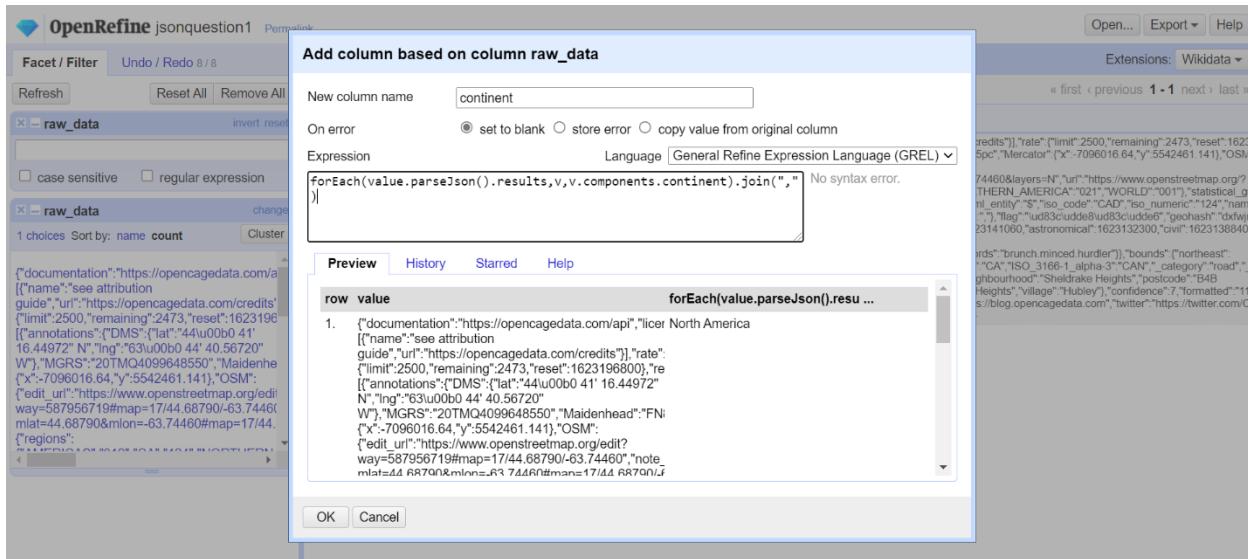


Fig – Adding continent column

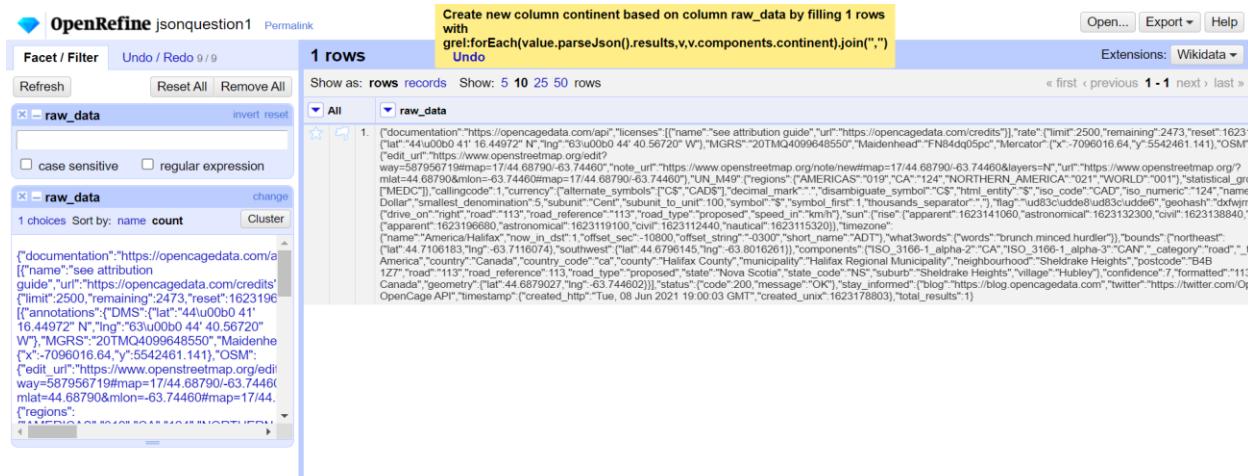


Fig – Added continent column

The screenshot shows the OpenRefine interface with the 'raw_data' column selected. The left sidebar shows facets for 'raw_data'. The main area displays a single row of JSON data. The data includes various geographical and administrative details, such as coordinates, MGRS codes, and place names like 'Maidenhead'. The right sidebar shows filters for continent ('North America') and country ('Canada').

```

{
  "name": "see attribution guide",
  "url": "https://opencagedata.com/credits",
  "rate": {
    "limit": 2500,
    "remaining": 2473,
    "reset": 1623196800
  },
  "results": [
    {
      "annotations": {
        "DMS": {
          "x": -7096016.64,
          "y": 5542461.141
        },
        "OSM": {
          "lat": 44.68790,
          "lon": -63.7446
        }
      },
      "geometry": {
        "bbox": [
          16.44972,
          -63.7446,
          44.68790,
          -16.44972
        ],
        "coordinates": [
          [
            [
              [
                [
                  [
                    [
                      [
                        [
                          [
                            [
                              [
                                [
                                  [
                                    [
                                      [
                                        [
                                          [
                                            [
                                              [
                                                [
                                                  [
                                                    [
                                                      [
                                                        [
                                                          [
                                                            [
                                                              [
                                                                [
                                                                  [
                                                                    [
                                                                      [
                                                                        [
                                                                          [
                                                                            [
                                                                              [
                                                                                [
                                                                                  [
                                                                                    [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
                                                                                      [
................................................................

```

Fig – Continent column

16)Delete raw_data column.

Steps –

- Delete raw_data by selecting the column raw_data, then select followed by

.

The screenshot shows the OpenRefine interface after the 'raw_data' column has been deleted. The left sidebar no longer lists 'raw_data'. The main area shows a single row of data with the 'continent' filter applied, showing 'North America' and 'Canada'. The right sidebar remains the same.

Fig – Deleted raw_data column

17)Export project as well as CSV file. Rename project file as solution1.tar.gz and rename CSV file as solution1.csv

The screenshot shows the OpenRefine interface with the following details:

- Navigation Bar:** Facet / Filter, Undo / Redo 11 / 11, Extract..., Apply....
- Filter Panel:** Shows a list of 8 steps:
 - Create project
 - Remove column File
 - Remove 27 rows
 - Remove 33 rows
 - Split 940 cell(s) in column full_name into several columns by separator
 - Rename column full_name 1 to f_name
 - Text transform on 480 cells in column sex: value.replace("Male","M")
 - Text transform on 480 cells in column sex:
- Main Data Grid:** Titled "940 rows", showing columns: id, f_name, l_name, email, sex, age, credit_card, lat, lng. The data consists of 11 rows (1-11).
- Sidebar (Right):** Shows "OpenRefine project archive to file" dropdown with options: Tab-separated value, Comma-separated value, HTML table, Excel (xls), Excel 2007+ (xlsx), ODF spreadsheet, Custom tabular exporter..., SQL Exporter..., Templating..., OpenRefine project archive to Google Drive, Google Sheets, Wikidata edits, QuickStatements file, Wikidata schema.

Fig – Export project archive file

Data Preparation using Google Cloud

Task - Loading data

Steps -

- Click on Import data and add to data flow
- Load the data set by choosing file [Choose a file](#).
- Import data set question2_1.csv, question2_2.csv and question2_3.csv by clicking on [Import & Add to Flow](#)

(Note – use Edit settings [Edit settings](#) to consider row 1 as header)

The screenshot shows the Google Data Studio interface for importing data. On the left, there's a sidebar with icons for various data sources like GCS, Google Sheets, and BigQuery. The main area is titled 'Import Data and Add to Flow'. It has sections for 'Upload from your computer' (with 'Upload' and 'Choose a file' buttons) and 'Upload location' (set to gs://dataprep-staging-0d7c2aba-2568-435c-a1ee-27e61e...). A table lists three new datasets:

RBC stock_symbol	RBC stock_market
AAV	\$1.15B
ABG	\$1.15B
ABIL	\$31.94M
ABR	\$515.96M
ACTX	n/a

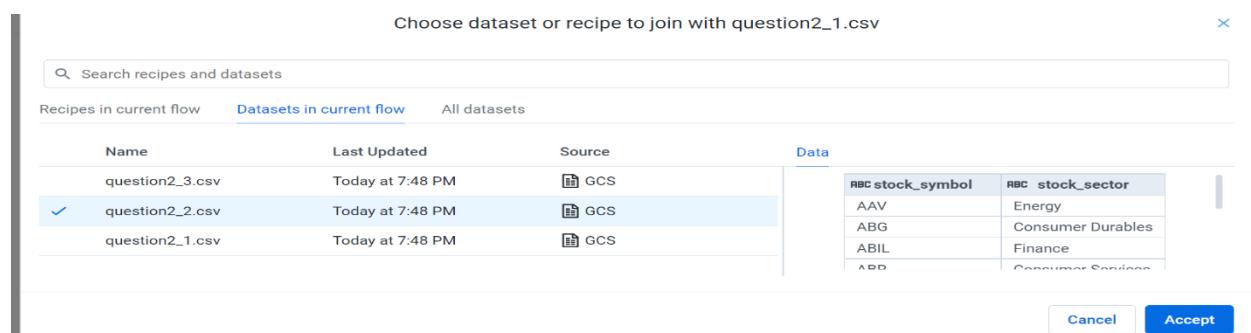
At the bottom right, there are 'Import & Add to Flow' and 'Cancel' buttons.

Fig – Adding dataset to data flow

Task – Joining dataset question2_1, question2_2, question2_3

Steps –

- Add Recipe for joining data and select the dataset and click on accept **Accept**.
- Performing inner join  operation based on key stock_symbol of question2_1.csv and question2_2.csv
- Retaining only the required columns such as stock_symbol, stock_market and stock_sector columns join operation and click on review **Review**.
- On the same recipe, join question2_3.csv along with already joined question2_1.csv and question2_2.csv.
- Performing inner join  operation based on key stock_symbol of question2_1.csv, question2_2.csv and question2_3.csv
- Retaining only the required columns such as stock_symbol, stock_market, stock_sector and stock_market_cap columns join operation and click on review **Review**.

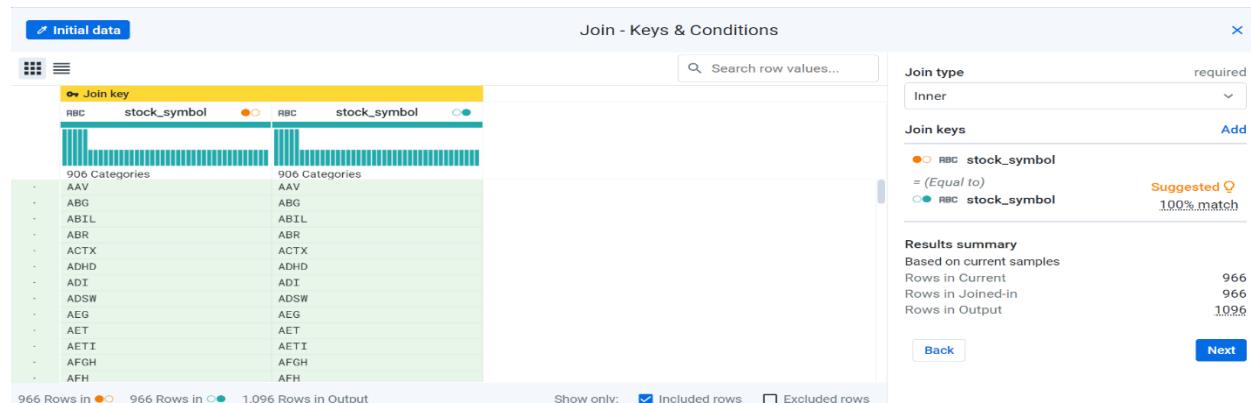


The screenshot shows a dialog titled "Choose dataset or recipe to join with question2_1.csv". It lists three datasets: "question2_3.csv", "question2_2.csv" (selected with a checkmark), and "question2_1.csv". The "Datasets in current flow" tab is active. To the right, a preview table shows the joined data with columns "RBC_stock_symbol" and "RBC_stock_sector". The preview table contains the following data:

RBC_stock_symbol	RBC_stock_sector
AAV	Energy
ABG	Consumer Durables
ABIL	Finance
ABD	Consumer Services

At the bottom right are "Cancel" and "Accept" buttons.

Fig – Joining data with stock_symbol (question2_1.csv and question2_2.csv)



The screenshot shows a "Join - Keys & Conditions" interface. It displays two tables for joining on the "RBC_stock_symbol" key. The left table has 906 categories and the right table also has 906 categories. The sidebar includes a "Join type" dropdown set to "Inner", a "Join keys" section with a "Suggested" condition for equality between the two keys, and a "Results summary" section showing 966 rows in the current sample, 966 rows in the joined-in sample, and 1096 rows in the output. At the bottom are "Back" and "Next" buttons.

Fig – Performing Inner Join

The screenshot shows the 'Join - Output Columns' interface. On the left, there is a preview of the joined data with three columns: 'stock_symbol', 'stock_market', and 'stock_sector'. The 'stock_symbol' column has 906 categories, 'stock_market' has 2 categories (NYSE and NASDAQ), and 'stock_sector' has 13 categories (Energy, Consumer Durables, Finance, Consumer Services, n/a, Health Care, Technology, Public Utilities, Finance, Energy, Finance, Finance). A search bar at the top right says 'Search row values...'. On the right, there is a 'Columns' section with a search bar 'Search columns' and a list of columns under 'All (4)'. The selected columns are 'stock_symbol' (Current 2), 'stock_market' (Joined-In 2), and 'stock_sector' (Joined-In 2). There are also 'Back' and 'Review' buttons.

Fig – Selecting the required columns

The screenshot shows the 'Choose dataset or recipe to join' interface. It lists datasets in current flow: 'question2_3.csv', 'question2_2.csv', and 'question2_1.csv'. 'question2_3.csv' is selected. To the right, a preview of the joined data is shown with columns 'RBC stock_symbol' and 'RBC stock_market_cap'. The data includes rows for AAV, ABG, ABIL, and others. At the bottom right are 'Cancel' and 'Accept' buttons.

Fig – Merging question2_3.csv with the data with key as stock_symbol

The screenshot shows the 'Join - Output Columns' interface again. This time, all columns ('stock_symbol', 'stock_market', 'stock_sector', and 'stock_market_cap') are selected. The joined data preview shows 1,386 rows in output. The 'Columns' section on the right shows 'All (5)' with 'stock_symbol' (Current 3) and 'stock_market_cap' (Joined-In 2) selected. There are also 'Back' and 'Review' buttons.

Fig – Joining data with stock_symbol

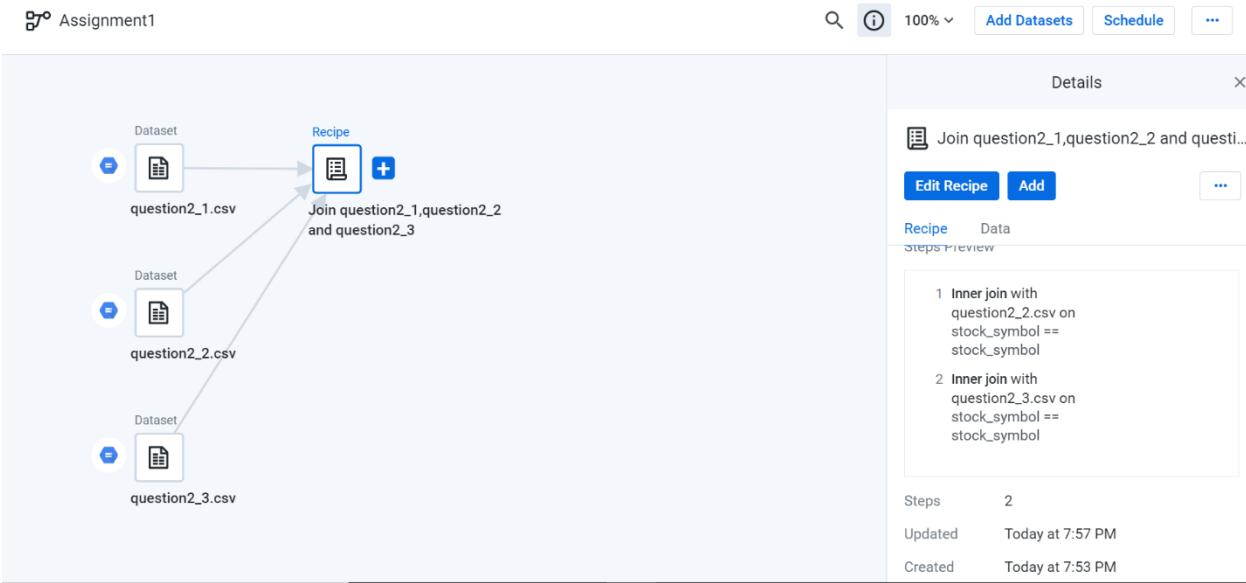


Fig – Joined question2_1, question2_2 and question2_3

The screenshot shows a data preview interface with the following columns:

RBC	stock_symbol	RBC	stock_market	RBC	stock_sector	RBC	stock_market_cap
	906 Categories		2 Categories		13 Categories		693 Categories
AAV	NYSE	ABG	NASDAQ	ABIL	Finance	ABR	Consumer Durables
ABG	NYSE	ABIL	NASDAQ	ACTX	Consumer Services	ADHD	n/a
ABIL	NASDAQ	ACTX	NASDAQ	ADHD	Health Care	ADI	\$39.87M
ACTX	NASDAQ	ADHD	NASDAQ	ADI	Technology	ADSW	\$29.38B
ADHD	NASDAQ	ADI	NYSE	ADSW	Public Utilities	AEG	\$2.1B
ADI	NYSE	ADSW	NYSE	AEG	Finance	AET	\$10.51B
ADSW	NYSE	AEG	NYSE	AET	Health Care	AETI	\$49.95B
AEG	NYSE	AET	NYSE	AETI	Energy	AFGH	\$12.31M
AET	NYSE	AETI	NASDAQ	AFGH	Finance	AFH	n/a
AETI	NASDAQ	AFGH	NASDAQ	AFH	n/a	AFSI^A	n/a
AFGH	NASDAQ	AFH	NASDAQ	AFSI^A	n/a	AFSI^A	n/a
AFH	NASDAQ	AFSI^A	NYSE	AFSI^A	n/a	AFSI^A	n/a
AFSI^A	NYSE	AFSI^A	NYSE	AFSI^A	n/a	AFSI^A	n/a

Below the table, the interface shows:

- 4 Columns
- 1,386 Rows
- 1 Data Type

Fig – Joined Data

Row count after join operation → 1386

Task – Converting Billion to Million

Steps –

- Add a recipe to perform conversion from billion to million.
- Selecting on convert pattern option
- Provide condition and new value to be replaced the condition.
- Using regular expression to convert stock_market_cap that has data in billion to million (capturing groups).

$/(\d*)\.\.(\d)B/$ (condition) → \$1\$200M (new value)

$/(\d*)\.\.(\d)(\d)B/$ (condition) → \$1\$2\$30M (new value)

- Then click on add to apply new value to stock_market_cap column.

RBC	stock_symbol	RBC	stock_market	RBC	stock_sector	RBC	stock_market_cap
	906 Categories		2 Categories		13 Categories		693 Categories
-	AAV	-	NYSE	-	Energy	-	\$1.15B
-	ABG	-	NYSE	-	Consumer Durables	-	\$1.15B
-	ABIL	-	NASDAQ	-	Finance	-	\$31.94M
-	ABR	-	NYSE	-	Consumer Services	-	\$515.96M
-	ACTX	-	NASDAQ	-	n/a	-	n/a
-	ADHD	-	NASDAQ	-	Health Care	-	\$30.87M
-	ADI	-	NASDAQ	-	Technology	-	\$29.30B
-	ADSW	-	NYSE	-	Public Utilities	-	\$2.1B
-	AEG	-	NYSE	-	Finance	-	\$10.51B
-	AET	-	NYSE	-	Health Care	-	\$49.95B
-	AETI	-	NASDAQ	-	Energy	-	\$12.31M
-	AFGH	-	NYSE	-	Finance	-	n/a
-	AFH	-	NASDAQ	-	Finance	-	\$188.28M
-	AFSI^A	-	NYSE	-	n/a	-	n/a
-	AFSI^A	-	NYSE	-	n/a	-	n/a

required
RBC stock_market_cap
Conditions (2)
Condition
 $/(\d*)\.\.(\d)B/$
New Value
\$1\$200M
Condition
 $/(\d*)\.\.(\d)(\d)B/$
New Value
\$1\$2\$30M
Cancel Add

Fig – Regex to convert Billion to Million

RBC	stock_symbol	RBC	stock_market	RBC	stock_sector	RBC	stock_market_cap
	906 Categories		2 Categories		13 Categories		693 Categories
-	AAV	-	NYSE	-	Energy	-	\$1150M
-	ABG	-	NYSE	-	Consumer Durables	-	\$1150M
-	ABIL	-	NASDAQ	-	Finance	-	\$31.94M
-	ABR	-	NYSE	-	Consumer Services	-	\$515.96M
-	ACTX	-	NASDAQ	-	n/a	-	n/a
-	ADHD	-	NASDAQ	-	Health Care	-	\$30.87M
-	ADI	-	NASDAQ	-	Technology	-	\$2930M
-	ADSW	-	NYSE	-	Public Utilities	-	\$2100M
-	AEG	-	NYSE	-	Finance	-	\$10510M
-	AET	-	NYSE	-	Health Care	-	\$49950M
-	AETI	-	NASDAQ	-	Energy	-	\$12.31M
-	AFGH	-	NYSE	-	Finance	-	n/a
-	AFH	-	NASDAQ	-	Finance	-	\$188.28M
-	AFSI^A	-	NYSE	-	n/a	-	n/a
-	AFSI^A	-	NYSE	-	n/a	-	n/a
-	AFSI^A	-	NYSE	-	n/a	-	n/a

Fig – stock_market_cap billion converted to million

Row count after converting billion to million → 1386

Task – Eliminating “n/a” values

Steps –

- Add recipe for deleting n/a or blank values.
- On details section, we can see the number of “n/a” 249 present in stock_market_cap column (249).
- Select filter rows , choose condition (is exactly) , select the column (stock_market_cap) , supply value (“n/a”) and action (delete the rows that contains “n/a”) .
- After selections are made then click on add to delete.
- Similarly for stock_sector column which has n/a values.

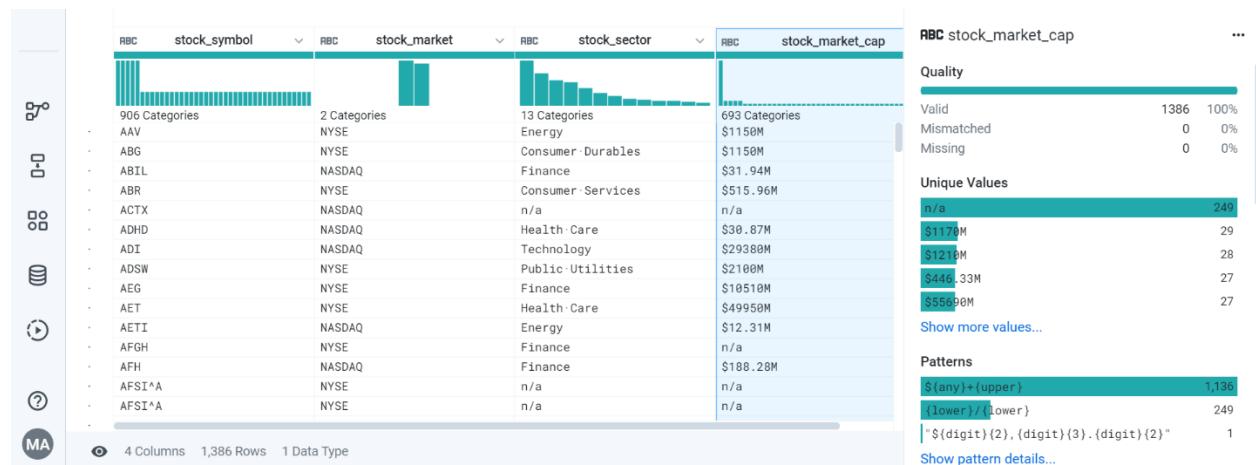


Fig – n/a values present in stock_market_cap



Fig – Deleting rows with stock_market_cap having “n/a”

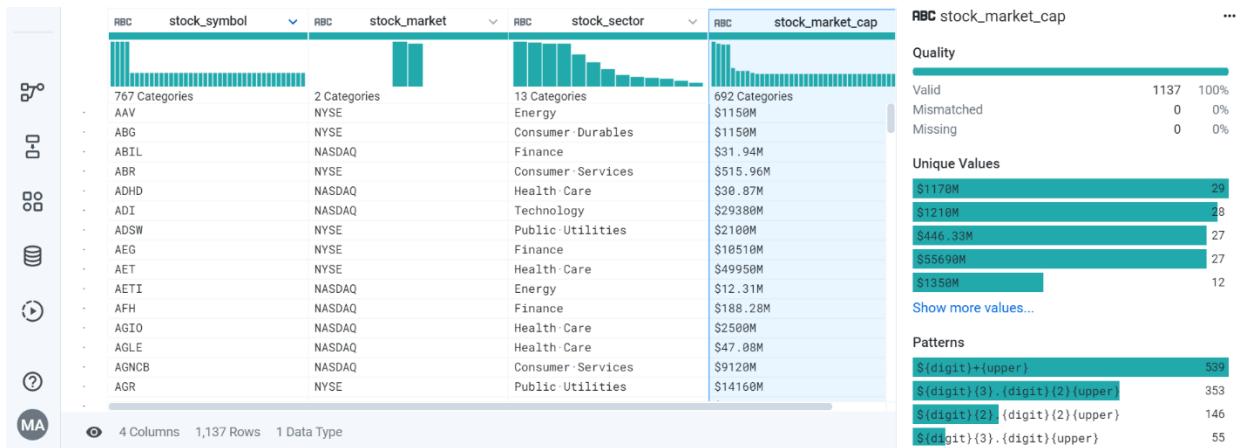


Fig – 0 n/a values present in stock_market_cap

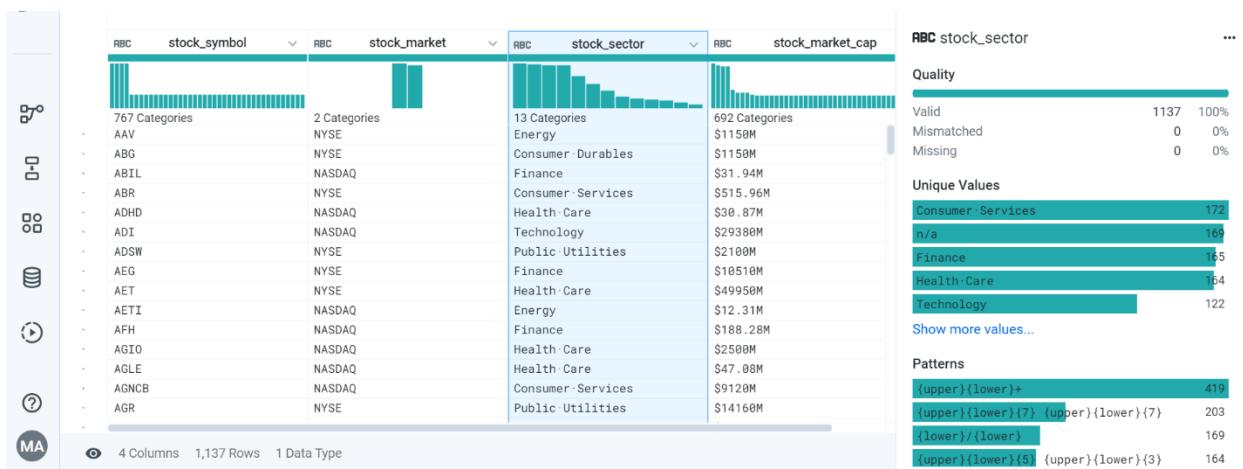


Fig – 169 n/a values present in stock_sector

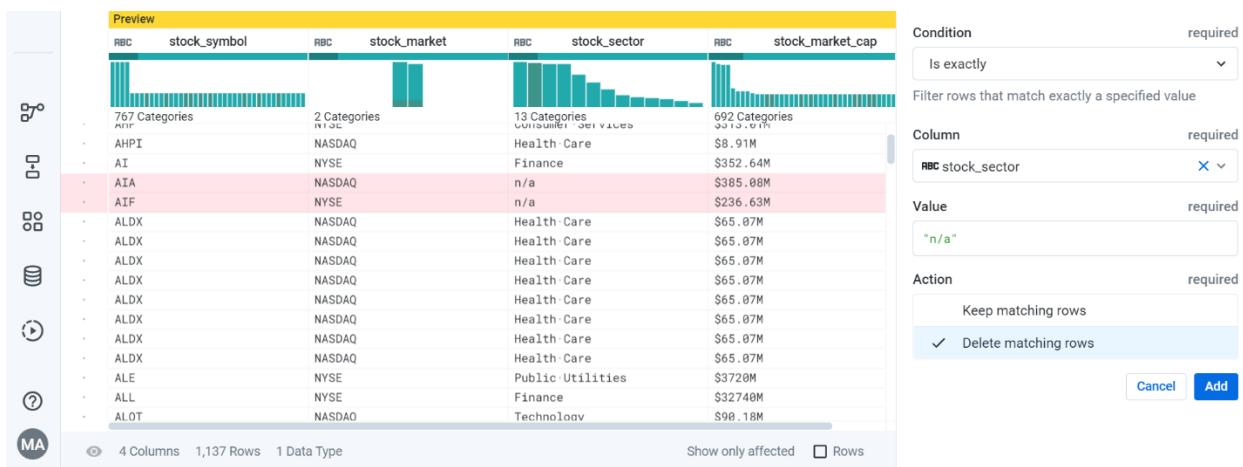


Fig – Eliminating stock_sector with n/a values

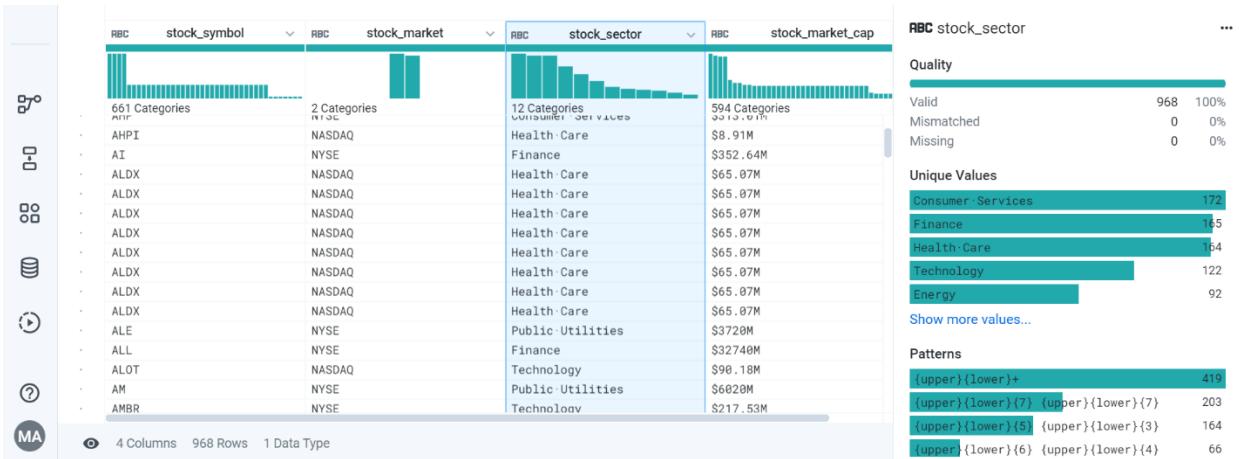


Fig – 0 n/a for stock_sector

Row count after eliminating “n/a” values → **968**

Task – Selecting records greater than 30M

Step –

- Stock_market_cap has text data hence it must be converted to decimal value.
 - Filter rows containing mismatched values by selecting condition (is mismatched), select column (stock_market_cap), delete matching rows with mismatched values.
 - Again, filtering rows based on stock_market_cap with condition (is greater than), selecting value greater than (30) and keep the rows which match the condition.
 - Click add  to make required changes.



Fig – Converting stock_market_cap to decimal



Fig – Decimal value for stock_market_cap

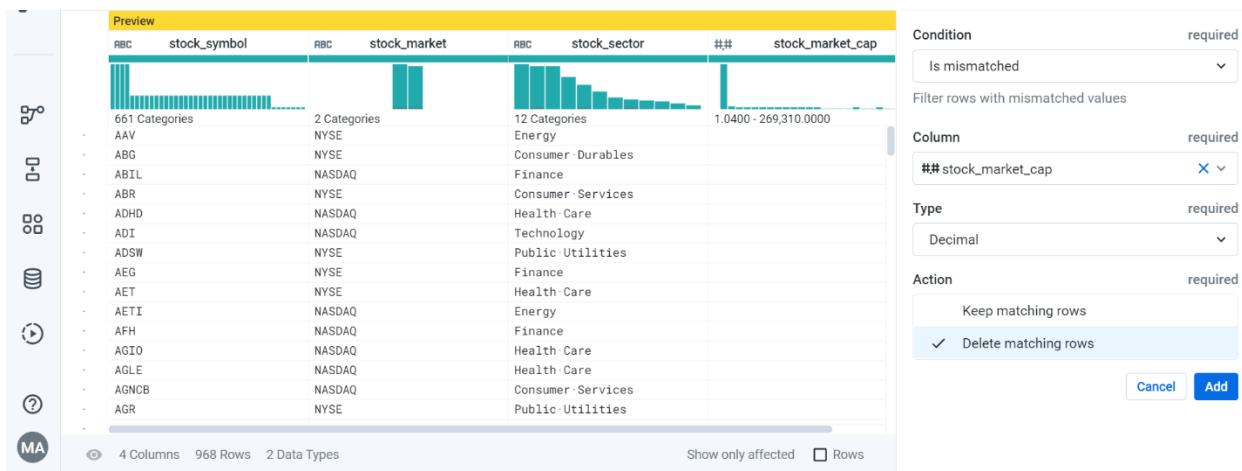


Fig – Deleting mismatched rows



Fig – Deleted mismatched rows

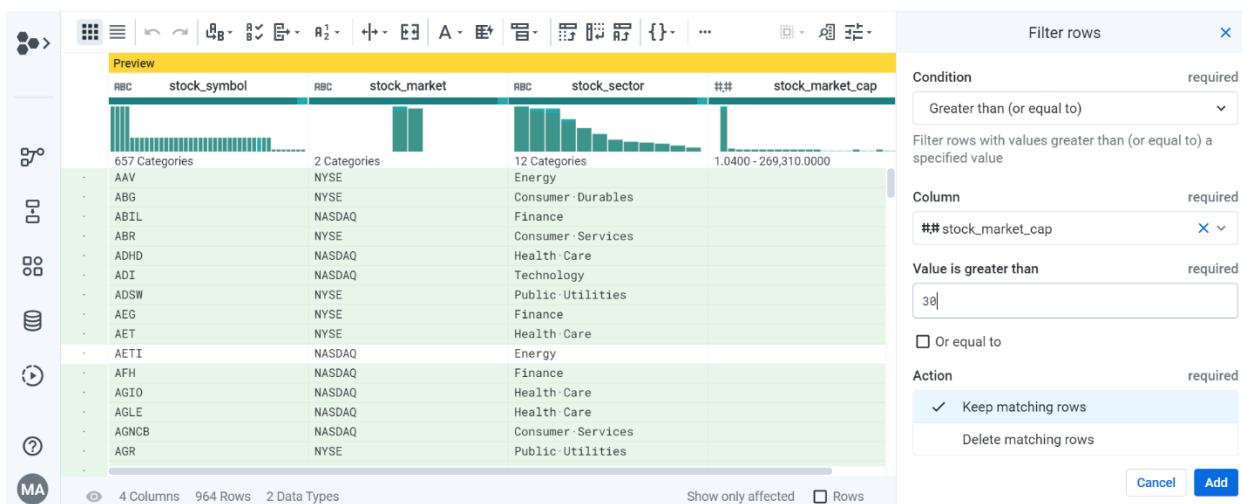


Fig – Selecting stock_market_cap greater than 30 million



Fig – Stock_market_cap greater than 30

Row count after retaining stock_market_cap greater than 30 → **915**

Task – Splitting the data based on stock_sector for Finance, Technology and Health Care

Steps –

- Add recipe for splitting the data into Finance, Technology and Health Care Sector data.
- Filtering rows for splitting the data based on condition (is exactly) , value matching (“Finance”) and action (keep matching rows).
- Delete stock_sector column
- Similarly create filter for Technology and Health Care.
- Click add **Add** to make required changes.

The screenshot shows a data preview and a filter configuration panel. The preview table has columns: stock_symbol, stock_market, stock_sector, and stock_market_cap. The filter configuration panel on the right shows a condition "Is exactly" set to "Finance", an action "Keep matching rows" selected, and a "Value" field containing "Finance".

stock_symbol	stock_market	stock_sector	stock_market_cap
RBC	NYSE	Energy	1150
AEG	NYSE	Consumer Durables	1150
AFH	NASDAQ	Finance	31.94
AI	NYSE	Consumer Services	515.96
ALL	NASDAQ	Health Care	30.87
ARL	NASDAQ	Technology	29388
BBVA	NYSE	Public Utilities	2100
BLMT	NASDAQ	Finance	18518
BLVD	NASDAQ	Health Care	49958
BMA	NYSE	Health Care	188.28
BOKF	NASDAQ	Health Care	2500
BRO	NYSE	Health Care	47.08
BSMX	NYSE	Consumer Services	9128
BUSE	NASDAQ	Public Utilities	14160
CB	NYSE	Consumer Non-Durables	1210
CFR	NYSE		

Fig – Finance stock sector

The screenshot shows a data preview and a filter configuration panel. The preview table has columns: stock_symbol, stock_market, and stock_market_cap. The filter configuration panel on the right shows a condition "Is exactly" set to "Finance", an action "Delete matching rows" selected, and a "Value" field containing "Finance".

stock_symbol	stock_market	stock_market_cap
RBC	NYSE	31.94
ABIL	NASDAQ	18518
AEG	NYSE	188.28
AFH	NASDAQ	352.64
AI	NYSE	32748
ALL	NYSE	137.3
ARL	NYSE	53988
BBVA	NASDAQ	292.59
BLMT	NASDAQ	461.58
BLVD	NASDAQ	5070
BMA	NYSE	5540
BOKF	NASDAQ	6220
BRO	NYSE	13148
BSMX	NYSE	1150
BUSE	NASDAQ	70810
CB	NYSE	6198
CFR	NYSE	

Fig – Deleting stock_sector column

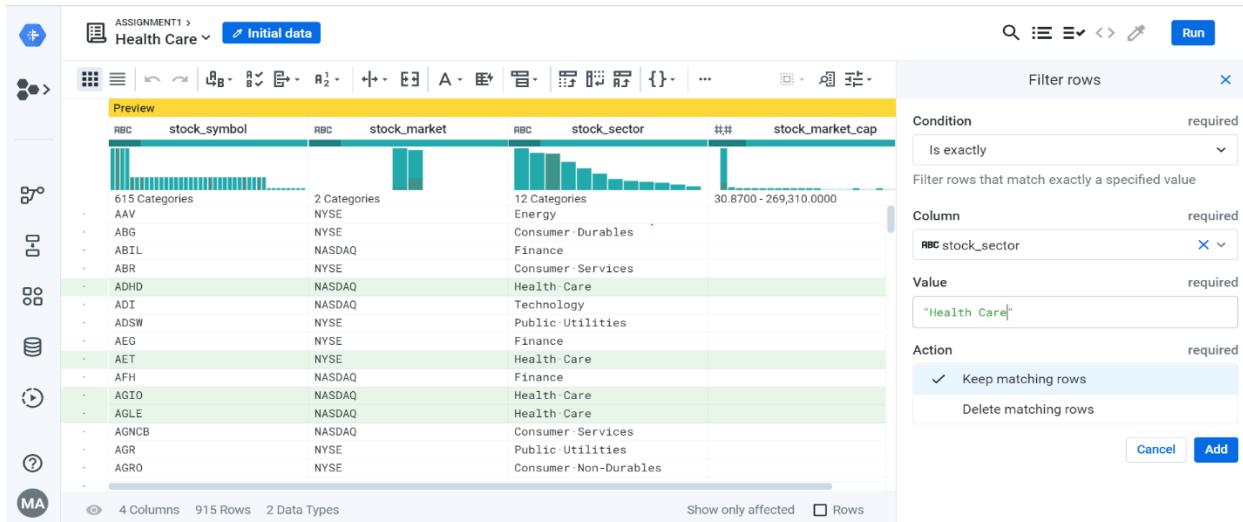


Fig –Health care stock sector

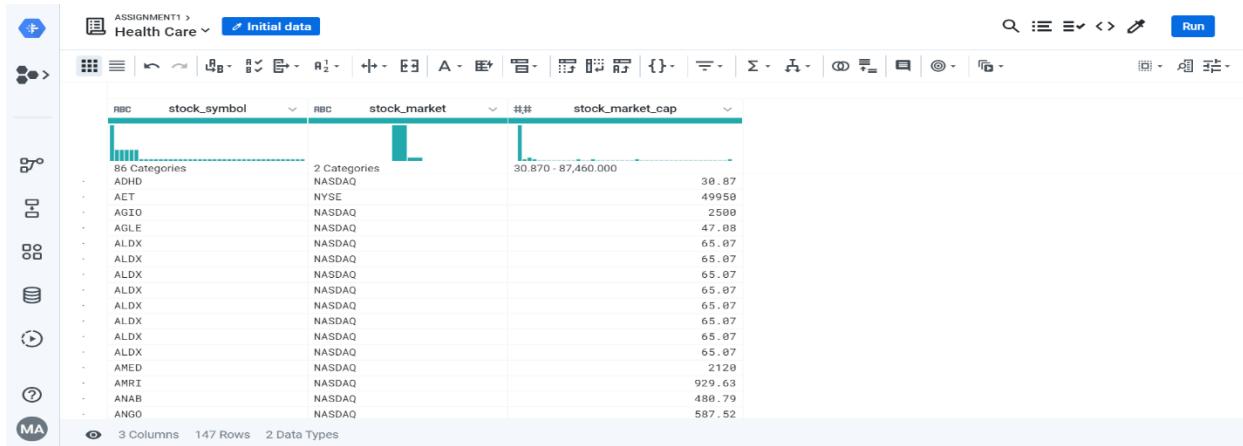


Fig – Deleting stock_sector column

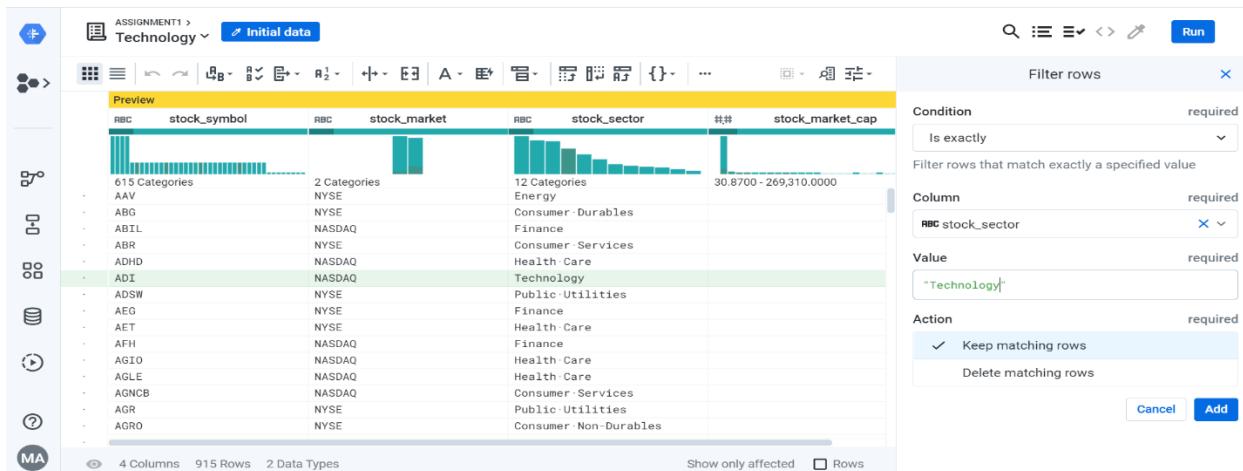


Fig – Technology stock sector

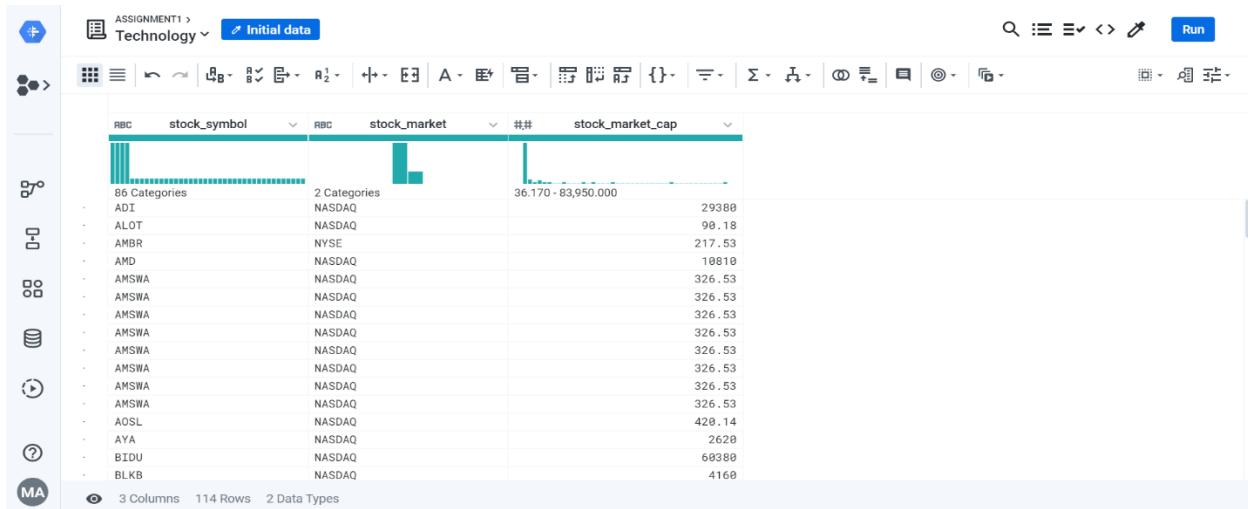


Fig – Deleting stock sector column

Row count for Finance data → **151**

Row count for Technology data → **114**

Row count for Health Care data → **147**

Task –Removing duplicate values present in Finance, Technology and Health Care data

Steps –

- Select stock_symbol, stock_market and stock_market_cap and remove duplicate rows
[Remove duplicate rows](#)
 - Rename the stock_market_cap column to stock_market_cap(Million) .



Fig –Eliminating duplicate values



Fig – Data after eliminating duplicate values

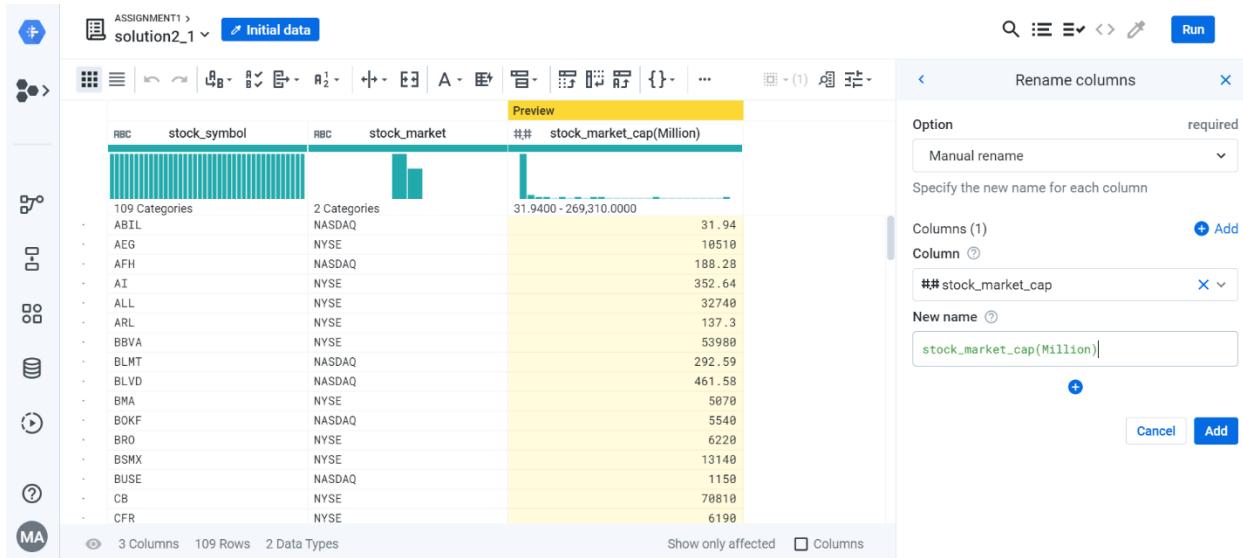


Fig – Renaming the column stock_market_cap(Million)



Fig – Eliminating duplicate values



Fig – Data after eliminating duplicate values

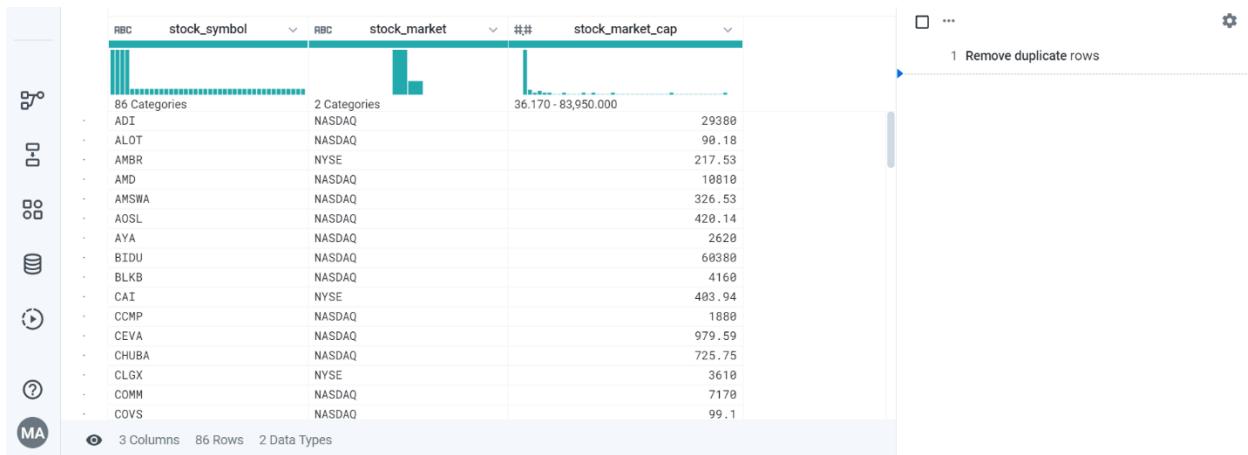


Fig – Eliminating duplicate values

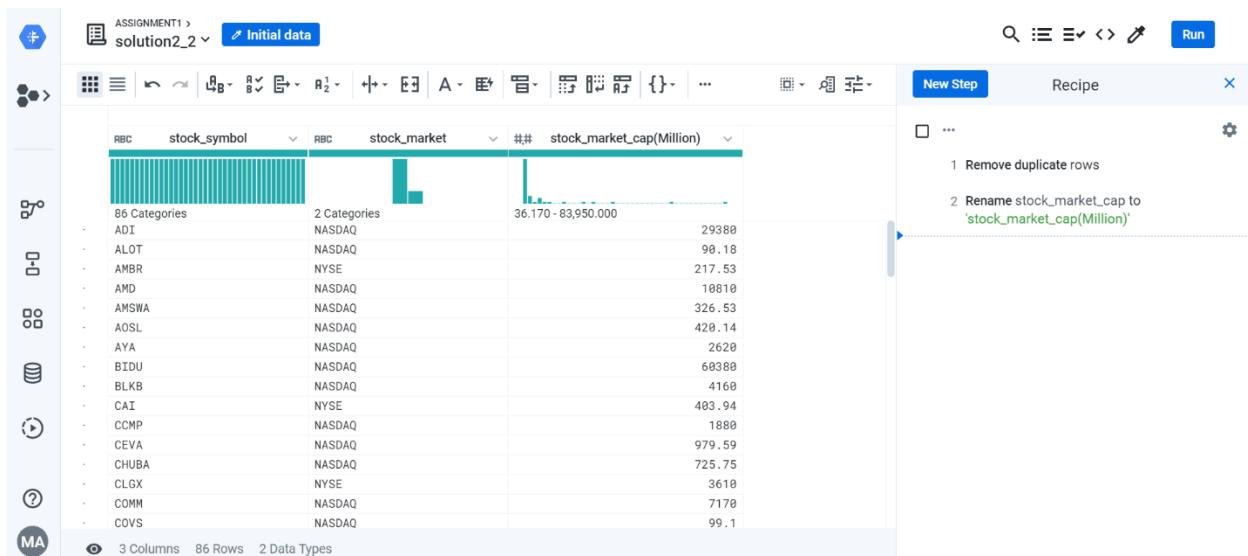


Fig – Data after eliminating duplicate values

Row count for Finance data → **109**

Row count for Technology data → **86**

Row count for Health Care data → **86**

Data Flow diagram containing all the operations –

1. Join all the data
2. Converting billion to million
3. Eliminating n/a
4. Retaining the data greater than or equal to \$30 M
5. Splitting the data Finance, Health Care and Technology data
6. Renaming the column stock_market_cap(Million) for clarity

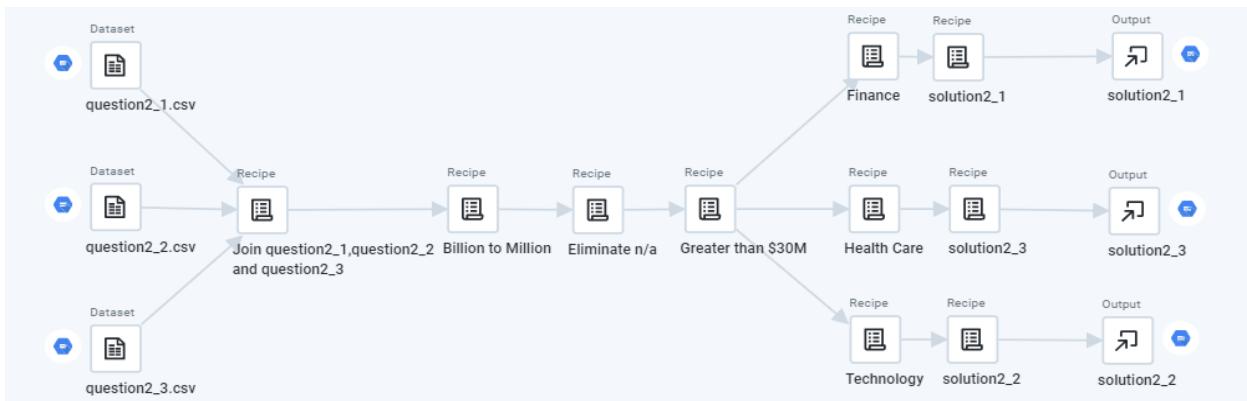


Fig – Data Flow

Task – Running job to create output csv files

Steps –

- Click on Run job
- Click on Edit to include headers

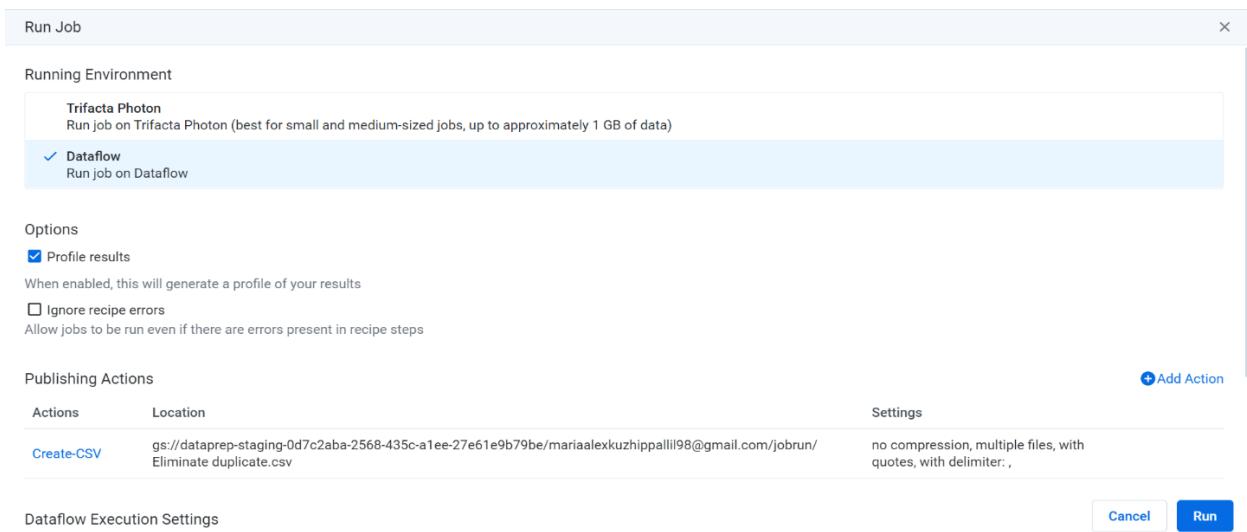


Fig – Run Job to create csv output files

Google Cloud Platform My First Project Search products and resources

The screenshot shows the Google Cloud Platform Dataflow interface. On the left, there's a sidebar with 'Dataflow' selected, showing 'Jobs', 'Schemas', 'Schemas', and 'Notebooks'. The main area is titled 'Jobs' and has a 'Running' filter applied. It lists three jobs:

Name	Type	End time	Elapsed time	Start time	Status	SDK version	ID	Region
cloud-dataprep-assignment1-9549558-by-mariaalexkuzhipallii98	Batch		35 sec	Jun 12, 2021, 10:20:40 PM	Running	2.28.0	2021-06-12_18_20_37-14008092684346029317	us-central1
cloud-dataprep-assignment1-9549549-by-mariaalexkuzhipallii98	Batch		1 min 8 sec	Jun 12, 2021, 10:20:07 PM	Running	2.28.0	2021-06-12_18_20_05-17916751919644578397	us-central1
cloud-dataprep-assignment1-9549542-by-mariaalexkuzhipallii98	Batch		1 min 57 sec	Jun 12, 2021, 10:19:18 PM	Running	2.28.0	2021-06-12_18_19_15-11623716274129565581	us-central1

Fig – Snapshot of running jobs.

Comparison of all tools

Google DataPrep

- Great visualization compared to openrefine.



Fig - Beautiful histograms of columns showing distribution and highlights blank/null records in red (Google DataPrep).

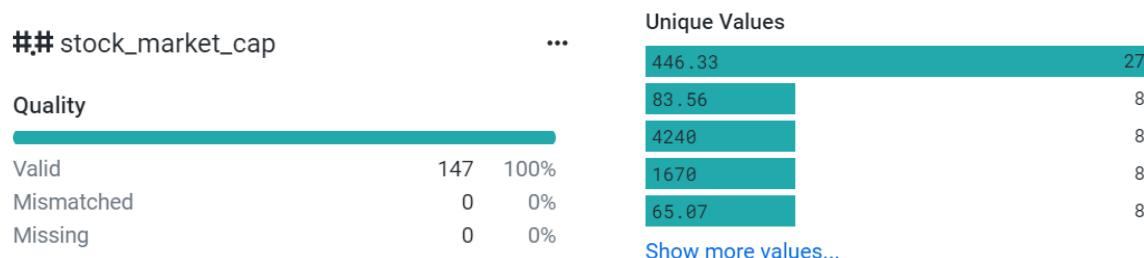


Fig - Great indicators displaying matched, missing, mismatched and unique values (Google DataPrep).

- Data flow concept makes it easier to keep track of the data manipulations performed and helps in making change in particular recipe if needed with greater ease compared to open refine which is comparatively more difficult.
- Great clarity – Clearly states all the steps carried out in details section.

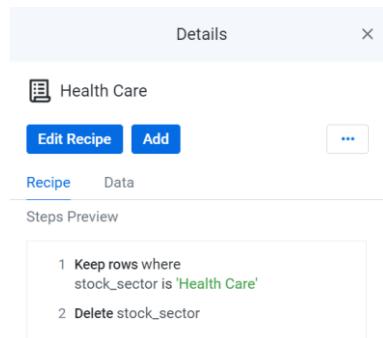
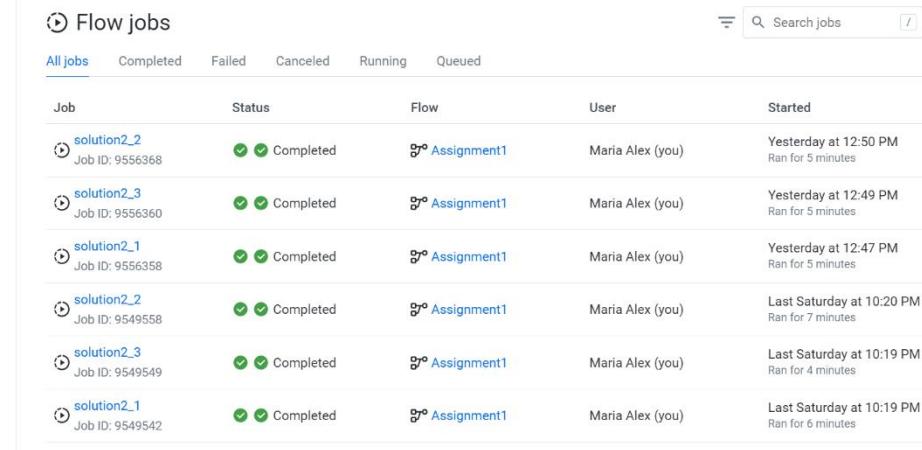


Fig - Sample representation of details section with steps preview

- Great ease of use even for novice users and interactive.

- Excellent tool for data preparation and exploratory analysis in terms of removing duplicate, deleting null/blank values, converting to different data types, renaming, and deleting columns etc.
- Quick in executing the ETL and easy to find completed, failed, and queued jobs.



The screenshot shows a 'Flow jobs' section within a 'Job history' interface. The table has columns: Job, Status, Flow, User, and Started. The data is as follows:

Job	Status	Flow	User	Started
solution2_2 Job ID: 9556368	Completed	Assignment1	Maria Alex (you)	Yesterday at 12:50 PM Ran for 5 minutes
solution2_3 Job ID: 9556360	Completed	Assignment1	Maria Alex (you)	Yesterday at 12:49 PM Ran for 5 minutes
solution2_1 Job ID: 9556358	Completed	Assignment1	Maria Alex (you)	Yesterday at 12:47 PM Ran for 5 minutes
solution2_2 Job ID: 9549558	Completed	Assignment1	Maria Alex (you)	Last Saturday at 10:20 PM Ran for 7 minutes
solution2_3 Job ID: 9549549	Completed	Assignment1	Maria Alex (you)	Last Saturday at 10:19 PM Ran for 4 minutes
solution2_1 Job ID: 9549542	Completed	Assignment1	Maria Alex (you)	Last Saturday at 10:19 PM Ran for 6 minutes

Fig - Sample representation of execution time

- Expensive after trial expires and need to supply credit card details for using the trial.

OpenRefine

- From history of already performed actions, it is possible to undo or redo.

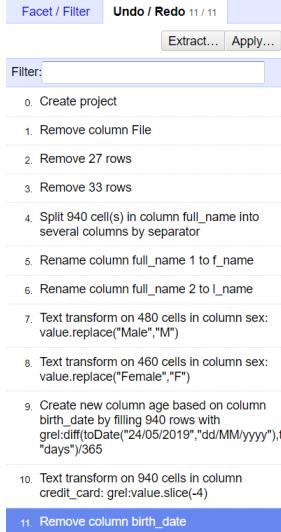
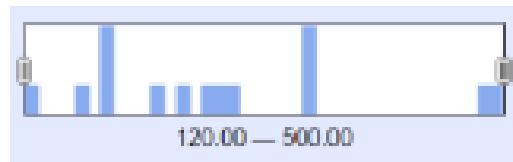


Fig – Undo / Redo for history of operations

- Faceting is an excellent feature that allows to group values. Along with it the slider, check box makes its convenient to filter date without using GREL.



Sample of filtering data using slider

- Works well with csv, json etc input and output.
- Query language GREL can be used to make.
- Free

Microsoft SSIS compared to OpenRefine and Google Dataprep

- Difficult for novice users to grasp compared to openrefine and dataprep.

(Note - From our experience we found it less convenient and lots of installation procedure hence we attempted question1 and question2).

- SSIS memory usage is high.

Submitted Files

1. assignment1.pdf – Project Report
2. Question 1
 - solution1.tar – Open refine solutions from subsection of question 1 1 - 11
 - solution1.csv - Open refine csv file from subsection of question 1 1 - 11
 - solution1_json.tar - Open refine solutions from subsection of question 1 12 - 17
 - solution1_json.csv - Open refine csv file from subsection of question 1 1 - 17
3. Question 2
 - solution2_1.csv – Finance data
 - solution2_2.csv – Technology data
 - solution2_2.csv – Health Care data
 - Dataflow_Assignment1