



SAINT MARY'S
UNIVERSITY SINCE 1802

One University. One World. Yours.

Master of Science in Computing and Data Analytics

Business Intelligence and Data Visualization

MCDA 5560

Assignment 1

Submitted To:

Ms. Trishla Shah

Submitted By:

Haozhou Wang (A00431268)

Hemanchal Joshi (A00433394)

Rishi Karki (A00432524)

Table of Contents

1. Executive Summary	1
2. Objective	2
3. Problem Solving	3
3.1 Data Preparation using OpenRefine	3
3.2 Data Preparation using Google Cloud Dataprep	19
3.3 Data Preparation (ETL Pipeline) using Microsoft SSIS	59
3.3.1. Pre-processing task: Schema Design	59
3.3.2. ETL pipeline: Generating target dataset	61
3.3.3. General Workflow	66
4. Comparison	67
5. Conclusion	68
6. References	69
7. Appendix [SQL Script]	70

1. Executive Summary

Data preparation as a data manipulating process is one of the important steps of data mining process where the activities like data cleaning and transformation is done on raw data prior to any processing and business analysis. The report is solely based on the the data preparation process using different tools like **OpenRefine**, **Google Cloud Dataprep** and **Microsoft SSIS** where the data preparation process is done on provided datasets as of given scenario and finally to derive the conclusion of comparison about the usage of these tools. It clearly mentions all the underlying important steps with necessary screenshots in the process of data preparation using these tools.

Firstly, OpenRefine was used as data preparation tool to work with given ‘question1.csv’ dataset, where we initially loaded the dataset in newly created project. The key steps for data preparation here were **removing empty records**, **splitting** of column, **consistent coding of transformation** for column, **column transforming** to extract particular length values and **deletion of duplicate columns**. Similarly, web scrapping was done to retrieve location detail JSON data from OpenCage Reverse Geocoding API and hence doing **filtering and storing** of only required data which are all the sum of our data preparation steps.

Secondly, Google Cloud Dataprep was another tool for data preparation on cloud platform where we first configure the signup process and loaded three datasets (‘question2_1.csv’, ‘question2_2.csv’ and ‘question2_3.csv’) relating to stock market. The key steps for data preparation here were **merge-split operation**, **transformation conversion** of column values, **removal of inconsistent and duplicate records** from datasets and **filtering** of datasets by specific column. In addition to this, **join operation** was done to combine all three datasets into one and lastly splitting of datasets by filtering against one of the column (‘stock_sector’) was done to derive three output datasets which sums up our data preparation steps.

For the final tasks i.e. the building the ETL pipeline and testing it we used Visual Studio 2017 as it gives access to the integrated SSIS services which is necessary to build an ETL pipeline. The most challenging part of the task is to find a dataset and divide it into dimension and fact tables. For us, we got a dataset online after thorough research but since the dataset was very large (about 550,000 rows) and the system could barely handle the load of extracting, transforming and loading it, we decided to shorten the data after many attempts to run the ETL process. The experiment in various bigger dataset and the wait is shown in the screen recordings. Our finalized chopped small dataset for the final work contained 20,610 rows. We divided our dataset into 4 dimension tables and a fact table in order to design it in form of the star schema. Then, we created SQL tables for the fact and dimension table and then started working on SSIS. **Since our dataset didn't contain any derived attribute, we just filtered the redundant data in the dataset and thoroughly followed the process of transforming the data which was finally loaded in the SQL database named accident. All the functionalities including splitting and merging and joining was understood and tried on the other datasets so as to learn about it.** The final results were loaded in the SQL tables and updated as a CSV file.

2. Objective

We are presented with three challenging tasks which includes thorough understanding of three different tools in order to prep and work with the data. For this task, we have three major objectives:

- Data preparation and analysis i.e. data cleaning and extraction using openrefine.
- Generating the CSV file containing specific fields using Google dataprep.
- Designing and performing the Extract, Load and Transform(ETL) mechanism using Microsoft SSIS.

3. Problem Solving

For the purpose of data preparation, three tools were used: OpenRefine, Google Cloud Dataprep and Microsoft SSIS. The data preparation steps for each of different scenarios using above these three tools are shortly described below:

3.1 Data Preparation using OpenRefine

Question 1: Data Preparation using OpenRefine

After successful installation of OpenRefine, “**question1.csv**” file is loaded, and hence new project is created and named as “**question1**” as per the provided instructions.

Tasks:

1. Install OpenRefine as shown in the tutorial.
2. Create Project **question1** in OpenRefine.
3. Load **question1.csv**.

The screenshot shows the OpenRefine interface with a table titled "973 records". The table has columns: id, full_name, email, sex, birth_date, credit_card, lat, and lng. The data consists of 973 rows, each containing a unique identifier, a full name, an email address, gender, birth date, a credit card number, latitude, and longitude. The interface includes a header bar with "OpenRefine question1 Permalink", "Facet / Filter", "Undo / Redo 0 / 0", "Extensions", "Wikidata", and "Help". Below the table, there's a sidebar with "Using facets and filters" and a note about getting started.

All	id	full_name	email	sex	birth_date	credit_card	lat	lng
1	1	Arnie Burdis	aburdis@army.mil	Male	31.01.1992	5428352425731160	38.023069	23.8724272
2	2	Piotr Burwinski	pburwinski@finc.gov	Male	07.11.1969	5100141904481440	40.0039320	-8.5008436
3	3	Elizabeth Fermilo	efermilo2@latimes.com	Female	19.03.1982	3537771455193920	10.3307399	123.8622107
4	5	Malvina Coonihau	mcloonihau@constantcontact.com	Female	28.08.2000	3528835362374970	43.2	82.63333
5	6	Barme Dutany	bduelany@abc.net.au	Male	15.09.1982	3592732418082180	18.159131	-77.7652346
6	7	Dacey Floris	dfloris6@coumbia.edu	Female	06.02.1996	67634349965878000	54.0056465	17.7763729
7	8	Martien Edeworth	medeworth7@plegiovin.com	Male	07.02.1990	3541930658076420	16.3112941	104.8221147
8	9	Valentine Couling	vcouling@hhs.gov	Male	14.05.1999	3590471384929910	-7.2538172	-79.129784
9	10	Kriste Beardsdale	kbeardsdale@qimngur.com	Female	13.08.2005	670692415427828000	-25.99566	28.1971031
10	11	Cordy Josevitz	cjosevitz@cio.us	Male	15.03.1982	6363212459918100	5.9779631	-0.0866948
11	12	Philly Fallon	pfallon6@ripadvisor.com	Female	06.06.1982	20174995730896	-1.4493473	-79.4634825
12	13	Kevin Jiroc	kjrochic@army.mil	Female	21.02.1993	30540666683981	-31.2657738	-64.460387
13	14	Eolande Kirinen	eikinen02@amazonaws.com	Female	20.04.1987	5602232924039690000	9.9213559	124.4318122
14	15	Marena Passie	mpassie@wikimedia.org	Female	24.09.1983	4044308271417770	4.625907	-75.761965
15	16	Koren Newart	knewart@freewebs.com	Female	27.06.1987	5007662317476660	36.182571	116.768358
16	17	Cletus Deeny	cddeeny@ripadvisor.com	Male	22.07.2001	4844612194756730	57.7244811	12.9309989
17	18	Normie Sanja	nsanjah@mozilla.com	Male	06.10.1988	3545288348299640	-8.1708603	112.6258244
18	19	Sandy Van der Kruif	svan@last.fm	Male	22.08.1999	201667886739119	50.6375507	17.8149641
19	20	Honor Kirkman	hkirkman1@amazon.co.uk	Female	28.06.1984	3574857937455130	52.054395	20.0873091
20	21	Genny McIvor	gmciivor@ripadvisor.com	Female	15.03.1991	52619296757666000	-8.7176937	115.5625434
21	22	Jess Piddletown	jpiddletown@nps.gov	Male	25.09.1996	49560157691940000	44.151742	125.6395731
22	23	Reider Romel	rromeli@roseek.co.jp	Male	07.02.1999	3589992505059640	18.3051999	-77.3593774
23	24	Chrissie Cade	ccade9@zmet.com	Male	04.09.2002	3529959198426440	-7.9780918	112.8007246
24	25	Marlin Mitchell	mmitchell@lifeng.com	Female	13.12.1993	3546483698321210	28.322781	-81.3912772
25	26	Natali Aspy	naspyp@megardan.com	Female	02.10.1997	6397289989653720	53.905048	27.5393261
26	27	Urbanus Chums	uchumsq@weibo.com	Male	20.05.1982	3540773355318780	60.1982825	29.6753924

Here is the snapshot after loading ‘question1.csv’ in ‘question1’ project.

4. Remove all rows in which **id** is empty.

Ans:

The ‘**id**’ column is first transformed to ‘**number**’ and then ‘**Text Facet**’ is done. Then, ‘**(blank)**’ option is selected and then under the ‘**All**’ dropdown menu, edit rows is selected and finally the option of “**Remove all matching rows**” is selected to ensure that all the empty id rows are deleted from the dataset. Hence, 24 total empty id records have been successfully deleted.

The screenshot shows the OpenRefine interface with the following details:

- Header:** OpenRefine question1 Permalink
- Facet / Filter:** Undo / Redo ↺ ↻
- Extensions:** Wikidata ▾
- Record Count:** 973 records
- Show As:** rows records Show: 5 10 25 50 records
- Table Headers:** All, id, null_name, email, sex, birth_date, credit_card, lat, lng
- Table Data:** A list of 973 records. The 'id' column contains various values including empty strings and numbers. The 'email' column contains email addresses. The 'sex' column contains Male/Female. The 'birth_date' column contains dates. The 'credit_card' column contains card numbers. The 'lat' and 'lng' columns contain geographical coordinates.
- Left Panel:**
 - Using facets and filters:** Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.
 - Not sure how to get started?** Watch these screencasts.
- Transform Menu:** The 'Transform...' option under 'Edit cells' is highlighted.
- Sub-menu:** The 'Common transforms' option is selected under 'Transform...'. The 'To number' option is highlighted.
- Bottom:** javascript:0

Here is the snapshot that shows the steps to transform the ‘id’ column into number.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 1 / 1 Extract... Apply...

Filter: Create project Text Transform on 973 cells in column id. value.toIntNumber() 973 records Show as: rows records Show: 5 10 25 50 records

Extensions Wikidata ▾

first < previous 1 - 25 next > last ▾

All	id	full_name	email	sex	birth_date	credit_card	lat	long
1	Facet	Text facet		Male	31.01.1992	5428362425731160	38.023069	23.8724272
2	Text filter	Numeric facet		Male	07.11.1969	5100141964484140	40.0039528	-8.5806436
3	Edit cells	Timeline facet		Female	19.03.1982	355771455193920	10.3307399	123.8822107
4	Edit column	Scatterplot facet		Male	02.12.1990	3583233969463500	45.6409009	14.8653128
5	Transpose	Custom text facet...	t.com	Female	28.08.2005	352853552374970	43.2	82.63333
6	Sort...	Custom Numeric Facet...		Male	15.09.1982	3552732418082180	18.1559131	-77.7652346
7	View	Customized facets		Female	06.02.1996	67634849985878000	54.0058465	17.7763729
8	Reconcile			Male	07.02.1990	354193658076420	16.3112541	104.6221147
9	Catroy Joseph	Customized facets		Male	14.05.1999	358047184529910	-7.2538172	79.129784
10	Correy Josefine	Customized facets		Female	13.08.2005	67069241542782800	-25.95656	28.1917031
11	Cletus Deeney	Customized facets		Male	15.03.1982	6383212459918100	5.9779631	-0.0866548
12	Philly Fallon	Customized facets		Female	06.06.1982	201749957308982	-1.4493473	-79.4634825
13	Kevina Jiruch	Customized facets		Female	21.02.1993	30540666683981	-31.2657738	-64.460387
14	Ecklande Kinen	Customized facets		Female	20.04.1987	560223292403969000	9.9213559	124.4318122
15	Marena Passie	Customized facets		Female	24.09.1983	4844398271417770	4.625907	-75.761965
16	Koren Newell	Customized facets		Female	27.06.1987	5007682317476660	36.182571	116.768358
17	Cletus Deeney	Customized facets		Male	22.07.2001	4844612194756730	57.7244811	12.9309869
18	Norne Sanja	Customized facets		Male	06.10.1986	35452983482996540	-8.170603	112.6253244
19	Sandy Van der Kru	Customized facets		Male	22.08.1999	201667885759113	50.6375507	17.8149641
20	Honor Kirkman	Customized facets		Female	28.06.1984	3574857934755130	52.0564395	20.0873091
21	Genny McIvor	Customized facets		Female	15.03.1991	5261959679566000	-8.7176937	115.5625434
22	Jess Piddetown	Customized facets		Male	25.09.1996	49560157269194000	44.151742	125.8395731
23	Reider Romeo	Customized facets		Male	07.02.1999	3589952505659840	18.3051999	-77.3539774
24	Christie Cade	Customized facets		Male	04.09.2002	3529939196842640	-7.9780918	112.8007246
25	Marlin Mitchell	Customized facets		Female	13.12.1993	3546453698321210	28.322781	-61.3912772
26	Natalia Aspy	Customized facets		Female	02.10.1997	6397298998653720	53.905048	27.5393261
27	Urbanus Churms	Customized facets		Male	20.05.1982	3540773355318780	60.1982625	29.6753924

javascript:()

Here is the snapshot that shows the step to do text facet operation.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 1 / 1 Refresh Reset All Remove All change, invert, reset

0 choices. Sort by: name count Cluster (number) 973 (blank) 24 Facet by choice counts exclude

24 matching records (973 total)

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

Extensions Wikidata ▾

first < previous 1 - 24 next > last ▾

All	id	full_name	email	sex	birth_date	credit_card	lat	long
Transform	Elizabeth Ferilio	eferilio2@attimes.com	Female	19.03.1982	353771458193920	10.3307399	123.8822107	
Facet	Wesley Gathorne	wgathorne@geepurnet.com	Male	02.12.1990	3585253998463300	45.6409009	14.8653128	
Exclude	Ecklande Kinen	ekinendi@amazonaws.com	Female	20.04.1987	560223292403969000	9.9213559	124.4318122	
Facet by choice counts	Star rows	kimedia.org	Female	24.09.1983	4844306271417770	4.625907	-75.761865	
Unstar rows	Unstar rows	si.edu	Male	11.05.2004	3573093118820160	-8.0656040	113.4418908	
Flag rows	Flag rows	google.ru	Male	18.04.1981	3537937058555800	36.3688889	25.4633333	
Unflag rows	Unflag rows	ketwatch.com	Male	26.09.1985	3555478274556370	23.276324	113.57676	
Remove all matching rows	Remove all matching rows	esdev.cn	Female	04.01.1969	37462233781182	19.4779768	-99.1717001	
216	220	Remove all matching rows	Mapquest.com	Male	22.11.1980	4913277461319240	-31.646394	-63.760259
256	261	Megan Hele	mhele78@newyorker.com	Female	11.04.1995	3543971471660520	35.6958419	139.5530916
Costa Vista	criste79@fpt.net	Female	02.05.1990	3561896104020020	-6.032666	26.9263671		
283	289	Cyrillus Slyde	cslyde80@loc.gov	Male	16.10.1990	358960258517437690	-6.5881066	105.6823124
Dilly Duckets	dduckets1@nify.com	Male	08.12.1999	3537055496242610	33.86	115.464454		
299	306	Terrye Basaggio	terryebasaggio@umich.edu	Female	02.01.1991	35887771749037630	46.4722761	-64.724645
Ursula Kroilan	ukrolan7c@hao123.com	Female	25.12.1981	357721591658875	38.712159	-9.4356652		
409	417	Jethi Sticks	jethisticks@laslasdot.org	Male	01.06.1995	5602210000324400	19.307033	-99.1675842
Billy Crumby	bcrumby@uoco.ru	Female	02.05.1997	5100136196996459	36.4126577	28.1554071		
419	428	Eward Cawood	shaun.orchart@slideshare.net	Male	12.03.1988	6304300591541600	-8.7223957	115.1767095
Shaun Orchart	sorchart@slideshare.net	Female	03.10.1966	30288372072238	43.9336277	42.5107159		
430	440	Stephanie Lendon	slondon7@epais.com	Female	17.03.1998	4041596040540730	4.6064435	-61.1053691
Walden Guyonet	wguyonet@ctrapul.org	Male	05.09.1999	3566911469916920	20.2506149	105.9744356		
472	483	Annelinda Matthesen	amathesende@mpg.org	Female	02.06.2002	63853529612584230	55.21664	-119.43605
Lucina Simml	lismild@who.int	Female	03.01.1984	5602235828009640	35.8496818	139.8969749		
478	490	Nicoline Prestho	npreshodt@umn.edu	Female	23.04.1986	374283168523983	3.6540491	-76.1088001
Faulkner Whittand	fwhittandm@independent.co.uk	Male	31.05.1999	3528076865967670	19.0487026	-72.4701945		
Damian Gleeson	dggleford@shutterstock.com	Male	21.12.2000	677147374019390000	-21.0800303	-44.2039901		
654	668	Janezka Passmore	jpassmore@merriam-webster.com	Female	15.12.1994	561087683434704000	62.216473	135.8549859
Sigrid Bodice	sbodiceik@list-manage.com	Male	09.09.1984	201677483155265	-7.01771	109.5463954		
I	gbemardiki@wutco.com	Male	13.06.1992	5312159436585200	44.5394324	3.5333133		

Here is the snapshot after text facet operation, where we remove 'blank' facet values to ensure that all rows with empty id is removed. By this, we are successfully able to delete 24 records.

OpenRefine question1 Permalink Remove 51 rows Undo Open... Export Help

Facet / Filter Undo / Redo 2 / 2 Refresh Reset All Remove All

0 choices. Sort by: name count Cluster (number) 949 exclude

Facet by choice counts

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

All	id	full_name	email	sex	birth_date	credit_card	lat	lng
1	1	Amie Burdis	aburdig0@army.mil	Male	31.01.1992	542836245731160	38.0230669	23.8724272
2	2	Piotr Burwistle	pburwistle1@itc.gov	Male	07.11.1969	510014196448140	40.0093928	-8.5806436
3	5	Malina Connan	mconnan1@constantcontact.com	Female	28.08.2005	3528635562374970	43.2	82.63333
4	6	Bame Dulany	bdulany1@abc.net.au	Male	15.09.1982	3592732418082180	18.1599131	-77.7652346
5	7	Dacey Florts	dflorts6@columbia.edu	Female	06.02.1996	67834594985678000	54.0058465	17.7763729
6	8	Marten Edowthe	medowther1@bloglovin.com	Male	07.02.1990	354193658076420	16.3112941	104.0221147
7	9	Valentine Couling	vcouling1@hhs.gov	Male	14.05.1999	3580471984929910	-7.2538172	-79.129784
8	10	Kriste Beardsdale	kbeardsdale9@engur.com	Female	13.08.2005	6706241542728000	-25.99566	28.1971031
9	11	Cordy Joselevitz	cjoselevitz@cio.us	Male	15.03.1982	6363212459918100	5.9779631	-0.0866948
10	12	Philly Fallon	pfallonb@ripadvisor.com	Female	06.06.1982	201740957308968	-1.4493473	-79.4634825
11	13	Keving Jiruch	kjiruchg@army.mil	Female	21.02.1993	30540666683981	-31.2657738	-64.460387
12	16	Koren Newait	knewart@reewebs.com	Female	27.06.1987	5007652317476660	36.182571	116.768598
13	17	Cletus Deepy	cooerry@ripadvisor.com	Male	22.07.2001	4844612194756730	57.7244811	12.9309989
14	18	Normie Sanja	nsanjah@mozilla.com	Male	06.10.1986	3545288346298640	-8.1708603	112.6256244
15	19	Sandy Van der Kruil	svan@last.fm	Male	22.08.1999	201667786759113	50.6375907	17.8149641
16	20	Honor Kirkman	hkirkman@amazon.co.uk	Female	28.06.1984	3574857934755130	52.0564393	20.0873091
17	21	Genny McIvor	gmciwork@ripadvisor.com	Female	15.03.1991	526192967566000	-8.7176937	115.5625434
18	22	Jess Piddetown	jpiddetown1@psgov	Male	25.09.1996	49360157769194000	44.151742	125.3595731
19	23	Reider Romei	rromei@rioseek.co.jp	Male	07.02.1999	3588992505059840	18.3051999	-77.3539774
20	24	Chrissie Cade	ccaden@zmet.com	Male	04.09.2002	35299039198042840	-7.9780918	112.8007246
21	25	Marin Michali	mmichalio@feng.com	Female	13.12.1993	3546463698321210	28.322781	-81.3912772
22	26	Natali Aspy	nasppg@nguardian.com	Female	02.10.1997	639728999653720	53.905048	27.5393261
23	27	Urbanus Churm	uchurms@webo.com	Male	20.05.1982	3540773355318780	60.198285	29.6753924
24	28	Ibby Manketell	imanketell@blogger.com	Female	11.08.2003	3553795108045130	57.3579277	61.6905815
25	29	Sunny Southworth	ssouthworths@ning.com	Female	11.05.1983	374283534479480	50.7254996	-113.9749472

Here is the snapshot after deletion of empty id records where we are then left with 949 records.

5. Remove all rows in which email is empty.

Ans:

The ‘email’ column is first transformed to ‘text’ and then Customized facet according to ‘Facet by empty string’ is done. Then, ‘(true)’ option is selected and then under the ‘All’ dropdown menu, edit rows is selected and finally the option of “Remove all matching rows” is selected to ensure that all the empty email rows are deleted from the dataset. Hence, 30 total empty email records have been successfully deleted.

OpenRefine question1 Permalink Open... Export Help

Facet / Filter Undo / Redo 3 / 3 Refresh Reset All Remove All

2 choices. Sort by: name count false 919 true 30 exclude

Facet by choice counts

30 matching records (949 total) Show as: rows records Show: 5 10 25 50 records

All	email	id	full_name	email	sex	birth_date	credit_card	lat	lng
259	271	291	Denise Dumigos	Denise.Dumigos@army.mil	Female	04.02.1995	337941295149941	38.6382751	-8.9059561
287	301	311	Tate Gerren	Tate.Gerren@army.mil	Male	25.12.1993	30158068053190	29.1696666	48.1175757
340	356	362	Isidoro MacRory	Isidoro.MacRory@army.mil	Male	18.11.1995	510014485498740	50.119993	12.3484445
343	359	375	Mariam Knights	Mariam.Knights@army.mil	Female	30.04.2005	561005627394040	35.2285451	128.8893517
344	360	376	Connie Stanton	Connie.Stanton@army.mil	Male	22.08.1996	377089120293178	13.7832269	120.9891643
373	389	395	Emeni Murish	Emeni.Murish@army.mil	Male	20.02.1997	201511683227246	57.1630313	13.4153897
383	399	405	Ansel Danell	Ansel.Danell@army.mil	Male	02.06.1985	3567663419505380	24.9909255	51.5493483
404	422	430	Tarrant Gabreith	Tarrant.Gabreith@army.mil	Male	22.09.1988	30159812326909	61.2698079	17.09914
405	423	431	Henrik Knights	Henrik.Knights@army.mil	Male	06.04.2001	49172647456889370	3.191408	113.086634
412	432	438	Leesa Vennuer	Leesa.Vennuer@army.mil	Female	21.04.1990	374283137981817	38.5691789	-9.0814271
428	450	456	Dom Hellmer	Dom.Hellmer@army.mil	Male	26.11.1995	355119961488567	-34.511017	-58.4973265
466	492	508	Osborn Heleker	Osborn.Heleker@army.mil	Male	19.03.2006	3538779458075970	59.1639426	9.6358001
470	496	512	Shawn Roggerone	Shawn.Roggerone@army.mil	Male	25.04.1987	357053355475970	35.066619	116.311532
522	548	564	Zerk Kauffman	Zerk.Kauffman@army.mil	Male	17.05.1983	5602233440183650000	40.1858788	22.002802
560	586	602	Tobie Abbes	Tobie.Abbes@army.mil	Male	27.10.1989	3561434847376560	-31.3964765	-52.678752
671	701	727	Cully Garette	Cully.Garette@army.mil	Male	11.01.1993	639813557121860	29.204661	116.512037
673	703	729	Gillian Nistath	Gillian.Nistath@army.mil	Female	07.10.1986	3554426206374039	25.1434098	-12.3714463
724	756	784	Sandor Tregea	Sandor.Tregea@army.mil	Male	23.04.1997	3554941042911960	35.2024947	127.4626534

Here is the snapshot after text transformation followed by customized facet of ‘Facet by empty string’ on ‘email’ column where 30 records are found to be empty and we perform the delete operation on these records.

#	id	full_name	email	sex	birth_date	credit_card	lat	lng
1	1	Amrie Burdis	aburdis@army.mil	Male	31.01.1992	5408362425731160	38.0230869	23.8724272
2	2	Piotr Burwistle	pburwhistle1@flic.gov	Male	07.11.1989	5100141984434140	40.0039328	-8.5808436
3	3	Malvina Coon	mcoonthan@constantcontact.com	Female	26.08.2005	352883562374970	43.2	82.63333
4	6	Bianie Dulany	bduany5@abc.net.au	Male	15.09.1982	3552732418082180	18.1559131	-77.7652346
5	7	Dacey Flions	dflionsdg@cornelia.edu	Female	06.02.1996	67634894985878000	54.0058465	17.7763729
6	8	Marten Edwortho	medwortho1@ologovin.com	Male	07.02.1990	354193958076420	16.1112941	104.8221147
7	9	Valentine Couling	vcouling@nhs.gov	Male	14.05.1999	3580471384929910	-7.2538172	-75.129784
8	10	Kriste Beardsdale	kbeardsdale9@imgur.com	Female	13.08.2000	670692415427828000	-25.99566	28.1971031
9	11	Cordy Joselevitz	cjoselevitz2@cio.us	Male	15.03.1982	6383212459918100	5.9779631	-0.0886948
10	12	Philly Fallon	pfallont@ripadvisor.com	Female	06.06.1982	201740957308982	-1.4493473	-79.4634825
11	13	Kevinia Jnrich	kjnrich2@army.mil	Female	21.02.1993	30540666683861	-31.2637738	-64.4603867
12	16	Karen Newall	knewarth@reevebs.com	Female	27.06.1987	5007652317476960	36.182571	116.768358
13	17	Cletus Deeney	cedeeryg@ripadvisor.com	Male	22.07.2001	4844612194756730	57.7244811	12.9309869
14	18	Norne Sanja	nsanjah@mozilla.com	Male	06.10.1986	3545283348299640	-8.1706603	112.6258244
15	19	Sandy Van der Kruif	svan@last.fm	Male	22.08.1999	201667865739913	50.6375507	17.8149641
16	20	Honor Kirkman	hkirkmanj@amazon.co.uk	Female	28.06.1984	3574857934755130	52.0564395	20.0873091
17	21	Genny McIvor	gmciwork@ripadvisor.com	Female	15.03.1991	52619296757566000	-8.7179937	115.5625434
18	22	Jess Piddetown	jpiddetown1@cps.gov	Male	25.09.1996	49360157769194000	44.151742	125.8395731
19	23	Reider Romeo	rromeim@fooseek.co.jp	Male	07.02.1999	3588992505659840	18.3051999	-77.3539774
20	24	Chrissie Cade	ccaden9@zmet.com	Male	04.09.2002	3529939196842840	-7.9780918	112.8007246
21	25	Marlin Mitchell	mmitchallo@lifeng.com	Female	13.12.1993	3546453698321210	28.322781	-81.3912772
22	26	Nalani Aspy	naspoy@reguardian.com	Female	02.10.1997	6397238998653720	53.905048	27.5393261
23	27	Urbanus Chums	uchums@weibo.com	Male	20.05.1982	3540773355318780	60.1982625	29.6753924
24	28	Iby Marketell	imarketell@blogger.com	Female	11.08.2003	3553795108045130	57.3579277	61.6905815
25	29	Sunny Southworth	ssouthworths@ning.com	Female	11.05.1983	374283534479480	50.7254936	-113.9749472

Here is the snapshot after deletion of 30 empty email records and then we are left with only 919 records.

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).

Ans:

The ‘**full_name**’ column is first transformed to ‘text’ and then under the ‘**Edit Column**’, ‘**Add column based on this column...**’ option is selected. Then, new column names: ‘**f_name**’ and ‘**l_name**’ are defined with the GREL expression of ‘**value.split("")[0]**’ and ‘**value.split("")[1]**’ in order to split full name into first name and last name respectively. Hence, splitting of full name into first name and last name is successfully done for all the dataset rows.

New column name: f_name

core-views/addasdasd set to blank store error copy value from original column

Expression: value.split('')[0]

No syntax error.

row	value	f_name
1.	Annie Burdis	Annie
2.	Piotr Burwhistle	Piotr
3.	Malvina Coonihan	Malvina
4.	Barnie Dulany	Barnie
5.	Dacey Floris	Dacey
6.	Marten Edworthie	Marten
7.	Valentina Coulina	Valentina

OK Cancel

Here is the snapshot that illustrates the derivation of new column 'f_name' derived from 'full_name' column with the usage of GREL expression.

New column name: l_name

core-views/addasdasd set to blank store error copy value from original column

Expression: value.split('')[1]

No syntax error.

row	value	l_name
1.	Annie Burdis	Burdis
2.	Piotr Burwhistle	Burwhistle
3.	Malvina Coonihan	Coonihan
4.	Barnie Dulany	Dulany
5.	Dacey Floris	Floris
6.	Marten Edworthie	Edworthie
7.	Valentina Coulina	Coulina

OK Cancel

Here is the snapshot that illustrates the derivation of new column 'l_name' derived from 'full_name' column with the usage of GREL expression.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 7 / 7 Extract... Apply...

Create new column `l_name` based on column `full_name` by filling 919 rows with `grel:value.split("")[1]` Undo

Extensions Wikidata ▾

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

Filter:

1. Create project
1. Text transform on 973 cells in column `id`: value.toNumber()
2. Remove 51 rows
3. Text transform on 30 cells in column `email`: value.toString()
4. Remove 30 rows
5. Text transform on 0 cells in column `full_name`: value.toString()
6. Create new column `f_name` based on column `full_name` by filling 919 rows with `grel:value.split("")[0]`
7. Create new column `l_name` based on column `full_name` by filling 919 rows with `grel:value.split("")[1]`

All	id	full_name	l_name	f_name	email	sex	birth_date	credit_card	lat	lng
1.	1	Amie Burds	Burds	Amie	aburdiso@army.mil	Male	31.01.1992	5428362425731160	38.0230869	23.8724272
2.	2	Piotr Burwistle	Burwistle	Piotr	pburwistle1@ftc.gov	Male	07.11.1969	510014964464140	40.0039328	-8.5808436
3.	5	Malina Coonihan	Coonihan	Malina	mcooniha1@constantcontact.com	Female	28.08.2005	3528833962374970	43.2	82.63333
4.	6	Dame Rutanya	Rutanya	Barnie	bdutany6@abc.net.au	Male	15.09.1962	3552732418082180	18.1559131	-77.7652346
5.	7	Dacey Florts	Florts	Dacey	dflorts6@columbia.edu	Female	06.02.1992	6763489985878000	54.0058465	17.7763729
6.	8	Marten Edwirthie	Edwirthie	Marten	medwirthie1@boglovin.com	Male	07.02.1990	354139365076420	16.3112941	104.8221147
7.	9	Valentine Couling	Couling	Valentine	vcouling6@jhs.gov	Male	14.05.1999	35804713949209910	-7.2538172	-79.129784
8.	10	Kriste Beardsdale	Beardsdale	Kriste	kbeardsdale9@imgur.com	Female	13.08.2005	670692415427828000	-25.99566	28.1971031
9.	11	Cordy Joselevitz	Joselevitz	Cordy	cjoselevitz2@cio.us	Male	15.03.1982	6383212459918100	5.9779631	-0.0886948
10.	12	Philly Fallon	Fallon	Philly	pfallon8@ipadvisor.com	Female	06.06.1982	20174095730896	-1.4493473	-79.4634825
11.	13	Keivina Jiruch	Jiruch	Keivina	kjiruch12@army.mil	Female	21.02.1993	30540666685986	-31.2657738	-64.460387
12.	16	Koren Newart	Newart	Koren	knewart1@freewebs.com	Female	27.06.1987	5007662317476660	36.162571	116.768358
13.	17	Cletus Deery	Deery	Cletus	cdeery9@ipadvisor.com	Male	22.07.2001	4844612194756730	57.7244811	12.9309889
14.	18	Normie Sanja	Sanja	Normie	nsanjan@mozilla.com	Male	06.10.1966	3545288548296940	-8.1708603	112.6256244
15.	19	Sandy Van der Kruif	Van	Sandy	svan1@ast.fm	Male	22.08.1999	20166788579911	50.6379507	17.8149641
16.	20	Honor Kirkman	Kirkman	Honor	hkirkman1@amazon.co.uk	Female	28.06.1984	3574857934755130	52.0564395	20.0873091
17.	21	Genny McIvor	McIvor	Genny	gmcivor9@ipadvisor.com	Female	15.03.1991	526192967566000	-8.7176937	115.5625434
18.	22	Jess Piddleton	Piddleton	Jess	jpiddleton1@ips.gov	Male	25.09.1996	4936015776919400	44.151742	125.8395731
19.	23	Reider Romeo	Romeo	Reider	rromeim@infoseek.co.jp	Male	07.02.1999	3558992500595940	18.3051999	-77.3539774
20.	24	Chrissie Cade	Cade	Chrissie	ccaden1@zohet.com	Male	04.09.2002	3529939198642804	-9.780918	112.8007246
21.	25	Martin Mitchell	Mitchell	Martin	mmitchalo1@feng.com	Female	13.12.1993	3546463698321210	28.322781	-81.3912772
22.	26	Natani Aspy	Aspy	Natani	nasypy1@theguardian.com	Female	02.10.1997	639729996653720	53.905048	27.5939261
23.	27	Urbanus Churms	Churms	Urbanus	uchurms1@weebio.com	Male	20.05.1982	354077335518780	60.1982825	29.6753924
24.	28	Ibby Mankell	Mankell	Ibby	imankell1@blogger.com	Female	11.08.2003	3553795108045130	57.3579277	61.6905815
25.	29	Sunny Southworth	Southworth	Sunny	ssouthworths@ning.com	Female	11.05.1983	37428354479480	50.7254996	-113.7479472

Here is the snapshot after splitting of 'full_name' column into 'f_name' and 'l_name' columns.

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

Ans:

The 'sex' column is first transformed to 'text' and then 'Text Facet' is done. Then, options of 'Female' and 'Male' are selected and then edited with 'F' and 'M' respectively in order to code gender of Female and Male. Hence, in sex column, 448 females are coded with F and 471 males are coded with M in our dataset.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 10 / 10 Refresh Reset All Remove All

448 matching records (919 total)

Show as: rows records Show 5 10 25 50 records

2 choices Sort by: name count Cluster exclude

sex F 448 M 471 Facet by choice counts

Mass edit 471 cells in column sex Undo

Extensions Wikidata ▾

448 matching records (919 total)

Show as: rows records Show 5 10 25 50 records

2 choices Sort by: name count Cluster exclude

sex F 448 M 471 Facet by choice counts

All	id	full_name	l_name	f_name	email	sex	birth_date	credit_card	lat	lng
3.	5	Malina Coonihan	Coonihan	Malina	mcooniha1@constantcontact.com	F	28.08.2005	3528833962374970	43.2	82.63333
4.	7	Dacey Florts	Florts	Dacey	dflorts6@columbia.edu	F	06.02.1995	6763489985878000	54.0058465	17.7763729
5.	8.	Kriste Beardsdale	Beardsdale	Kriste	kbeardsdale9@imgur.com	F	13.08.2005	670692415427828000	-25.99566	28.1971031
6.	10.	Philly Fallon	Fallon	Philly	pfallon8@ipadvisor.com	F	06.06.1982	20174095730896	-1.4493473	-79.4634825
7.	11.	Keivina Jiruch	Jiruch	Keivina	kjiruch12@army.mil	F	21.02.1993	30540666685986	-31.2657738	-64.460387
8.	12.	Koren Newart	Newart	Koren	knewart1@freewebs.com	F	27.06.1987	5007662317476660	36.162571	116.768358
9.	16.	Honor Kirkman	Kirkman	Honor	hkirkman1@amazon.co.uk	F	28.06.1984	3574857934755130	52.0564395	20.0873091
10.	21.	Genny McIvor	McIvor	Genny	gmcivor9@ipadvisor.com	F	15.03.1991	526192967566000	-8.7176937	115.5625434
11.	25.	Martin Mitchell	Mitchell	Martin	mmitchalo1@feng.com	F	13.12.1993	3546463698321210	28.322781	-81.3912772
12.	26.	Natani Aspy	Aspy	Natani	nasypy1@theguardian.com	F	02.10.1997	639729996653720	53.905048	27.5939261
13.	28.	Ibby Mankell	Mankell	Ibby	imankell1@blogger.com	F	11.08.2003	3553795108045130	57.3579277	61.6905815
14.	29.	Sunny Southworth	Southworth	Sunny	ssouthworths@ning.com	F	11.05.1983	37428354479480	50.7254996	-113.7479472
15.	30.	Tabbi Grindall	Grindall	Tabbi	tgrindall1@zatacademy.com	F	28.08.1983	3559643334729930	34.92357	36.48598
16.	27.	Alinne Fardell	Fardell	Alinne	afardell1@ic.gov	F	15.11.1985	500766425904930	43.6971283	44.2085511
17.	32.	Branda Laffin	Laffin	Branda	blaffin1@ulu.com	F	22.09.1987	500766317222570	49.476996	0.1630479
18.	33.	Glyn Justun	Justun	Glyn	gjustun1@ustream.tv	F	18.07.1984	5038182688418900	38.86	-76.99
19.	37.	Corayn Varen	Varen	Corayn	cvaren11@combinator.com	F	22.06.1982	5610962783595960	45.0736432	-67.0530897
20.	38.	Helena De	De	Helena	hde1@indiatimes.com	F	13.09.2001	357305349776469	49.6287673	23.6958367
21.	44.	Emilee Savins	Savins	Emilee	esavins1@google.gl	F	05.06.2001	20150694999892	-41.172184	-71.437354
22.	45.	Shebti Leither	Leither	Shebti	shebti10@wix.com	F	24.04.1994	3552768172178630	24.922044	118.223965
23.	48.	Carey Kelemen	Kelemen	Carey	cklement1@msn.com	F	11.01.1984	355384909954630	41.39992988	-8.6637745
24.	51.	Stariene Lenox	Lenox	Stariene	slenox11@si.edu	F	07.09.2002	4403163805450890	-6.3517979	106.1078627
25.	53.	Bettine Alyoshin	Alyoshin	Bettine	balyoshin11@devhub.com	F	05.04.1984	3528167718700150	2.90329721	101.62599642
26.	54.	Thomasin Eberts	Eberts	Thomasin	tebents11@ping.org	F	28.10.1998	3557023034436050	40.4938437	-8.4743511
27.	55.	Ardelle Stoll	Stoll	Ardelle	astoll1n@panpost.jp	F	18.08.1986	6793479509560200	39.96	-83

Here is the snapshot that illustrates the after the coding of 'sex' column into 'M' and 'F' for male and female respectively.

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).

Ans:

The ‘**birth_date**’ column is first selected to ‘**number**’ and then then under the ‘**Edit Column**’, ‘**Add column based on this column...**’ option is selected. Then, new column name of ‘**age**’ is defined with the GREL expression of ‘`diff('24.05.2019'.toDate('dd.MM.yyyy'),value.toDate('dd.MM.yyyy'),"years")`’ in order to calculate the age of person on 24th May 2019 from the birth date. Hence, age of person with respect to years is successfully calculated against birth date for all **birth_date** column in our dataset.

Here is the snapshot to derive a new column ‘age’ based on calculation age from base date of 24th May 2019 where GREL expression is used to derive the age in years.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 11 / 11

Extract... Apply...

Create new column age based on column birth_date by filling 919 rows with
grel:diff('24.05.2019'.toDate('dd.MM.yyyy'),value.toDate('dd.MM.yyyy')),'years')

Undo

Extensions: Wikidata

919 records

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

Filter:

1. Amie Burdis Burdis Amie aburdts0@army.mil M 31.01.1992 27 5428362425731160 38.023069 23.8724272

2. Piotr Burwhistle Burwhistle Piotr pburwhistle@ftc.gov M 07.11.1969 29 5100141964484140 40.009328 -8.5808496

3. Malvina Coonihan Coonihan Malvina mcoonthan@constantcontact.com F 28.08.2005 13 35288335623274970 43.2 82.63333

4. Dame Dulany Dulany Dame bdulany5@abc.net.au M 15.09.1982 36 3552732418082180 16.1559131 -77.755346

5. Dacey Floris Floris Dacey dfloris6@columbia.edu F 06.02.1996 23 67534649695878000 54.0059865 17.7753729

6. Marten Edworthie Edworthie Marten medworthie7@bloglovin.com M 07.02.1990 29 3541393658076420 16.3112941 104.8221147

7. Valentine Couling Couling Valentine vcouling8@fhs.gov M 14.05.1999 20 3580471984929910 -7.2581727 -79.129784

8. Kriste Beardsdale Beardsdale Kriste kbeardsdale9@imgur.com F 13.08.2009 17 67059241542782800 -25.99566 28.1971031

9. Cordy Joselevitz Joselevitz Cordy cjoselevitz10@cio.us M 15.03.1982 37 6383212459918100 5.9776931 -0.086948

10. Philly Fallon Fallon Philly pfallon11@ipadvisor.com F 06.06.1982 36 2017409573089862 -1.4493473 -79.4634625

11. Kevina Jiruch Jiruch Kevina kjiruch12@army.mil F 21.02.1993 26 305406665853981 -31.2657738 -64.460387

12. Koren Newart Newart Koren knewart13@freewebs.com F 27.06.1987 31 5007662317476660 36.182571 116.768338

13. Cletus Deery Deery Cletus cderry14@ipadvisor.com M 22.07.2001 17 4844612194756730 57.7244811 12.9309869

14. Norrie Sanja Sanja Norrie nsanjah15@mozilla.com M 06.10.1986 32 3545285348299640 -8.1706601 112.6258244

15. Sandy Van der Kruif Van Sandy svan16@last.fm M 22.08.1999 19 2016678857599191 50.6375907 17.8149641

16. Honor Kirkman Kirkman Honor hkirkman17@amazon.co.uk F 28.06.1984 34 3574857934755130 52.0564395 20.0873091

17. Genny McIvor McIvor Genny gmcivor18@ipadvisor.com F 15.03.1991 28 52619296576566000 -8.7176937 115.5625434

18. Jess Piddetown Piddetown Jess jpiddetown19@ips.gov M 15.09.1996 22 49360157769194000 44.151742 125.639731

19. Reider Romeo Romeo Reider rromein1@oseek.co.jp M 07.02.1999 20 35589925506596840 16.3051999 -77.3539774

20. Chrissie Cade Cade Chrissie ccadeng1@onet.com M 04.09.2002 16 3529939198642640 -7.970918 112.8007246

21. Marlin Mitchell Mitchell Marlin mmitchall10@feng.com F 13.12.1999 25 35464636598321210 28.322781 -81.3917722

22. Nalani Aspy Aspy Nalani nasypy11@guardian.com F 02.10.1997 21 639729899653720 53.905048 27.5393261

23. Urbanus Churms Churms Urbanus uchurms12@weibo.com M 10.05.1982 37 3540773355318780 60.1982625 29.6753924

24. Itby Manketell Manketell Itby imanketell13@blogger.com F 11.08.2003 15 3553795106045130 57.3579277 61.6905815

25. Sunny Southworth Southworth Sunny ssouthworts14@ng.com F 11.05.1983 36 374283534479480 50.7254936 -113.9749472

Here is the snapshot that illustrates the after derivation of age column after calculating age from 'birth_date' with respect to 24th May 2019 as a base date.

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

Ans:

The '**credit_card**' column is first transformed to '**number**' and then under the '**Edit Cells**', '**Transform...**' option is selected. Then, GREL expression of '**value.substring(value.length()-4)**' is defined in order to keep only last 4 digits of credit card. Hence, only last 4 digits of credit card is kept for all the credit_card column in our dataset.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 13 / 13 Extract... Apply...

Text transform on 919 cells in column credit_card:
grel:value.substring(value.length()-4) Undo

Extensions Wikidata

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

Filter:

1. Text transform on 973 cells in column id:
value.toIntNumber()

2. Remove 51 rows

3. Text transform on 30 cells in column email:
value.toString()

4. Remove 30 rows

5. Text transform on 0 cells in column full_name:
value.toString()

6. Create new column f_name based on
column full_name by filling 919 rows with
grel:value.split("")[0]

7. Create new column l_name based on
column full_name by filling 919 rows with
grel:value.split("")[1]

8. Text transform on 0 cells in column sex:
value.toString()

9. Mass edit 448 cells in column sex

10. Mass edit 471 cells in column sex

11. Create new column age based on
column birth_date by filling 919 rows with
grel:diff('24.05.2019' toDate('dd/MM/yyyy'))

12. Text transform on 919 cells in column
credit_card:
value.toIntNumber()

13. Text transform on 919 cells in column
credit_card:
grel:value.substring(value.length()-4)

1 All id full_name l_name f_name email sex birth_date age credit_card lat lng

1. 1 Amie Burdis Burdis Amie aburdiso@army.mil M 31.01.1992 27 1160 38.0230669 23.8724272

2. 2 Piotr Burwhistle Burwhistle Piotr pburwhistle@ftc.gov M 07.11.1969 29 4140 40.0039528 -8.5608436

3. 5 Malvina Coonihan Coonihan Malvina mcoonthan@constantcontact.com F 28.08.2005 13 4970 43.2 82.63333

4. 6 Barrie Dulany Dulany Barrie bdulany@abc.net.au M 15.09.1982 36 2160 18.1559131 -77.7652346

5. 7 Dacey Floris Floris Dacey dfloris6@columbia.edu F 06.02.1996 23 8000 54.0058465 17.7763729

6. 8 Marten Edwirthie Edwirthie Marten medwirthie@bloglovin.com M 07.02.1990 29 6420 16.3112941 104.0221147

7. 9 Valentine Couling Couling Valentine vcouling@jhs.gov M 14.05.1999 20 9910 -7.2538172 -79.129784

8. 10 Kriste Beardsdale Beardsdale Kriste kbeardsdale9@imgur.com F 13.08.2005 13 8000 -25.99566 28.1971031

9. 11 Cordy Joselevitz Joselevitz Cordy cjoselevitz@cio.us M 15.03.1982 37 8100 5.9779631 -0.0866948

10. 12 Philly Fallon Fallon Philly pfallon9@tripadvisor.com F 06.06.1982 36 8982 -1.4493473 -79.4634825

11. 13 Keivina Jiruch Jiruch Keivina kjruch@army.mil F 21.02.1993 26 3961 -31.2657738 -64.460387

12. 16 Karen Newart Newart Karen knewart@freewebs.com F 27.06.1987 31 6660 36.182571 116.768358

13. 17 Cletus Deery Deery Cletus cdeerryg@tripadvisor.com M 22.07.2001 17 6730 57.7244811 12.9309989

14. 18 Norie Sanja Sanja Norie nsanjan@mozilla.com M 06.10.1966 32 8640 -8.1708603 112.6256244

15. 19 Sandy Van der Kruif Van Sandy svan@last.fm M 22.08.1999 19 9113 50.6375507 17.8149641

16. 20 Honor Kirkman Kirkman Honor hkirkman9@amazon.co.uk F 28.06.1984 34 5130 52.0564395 20.0873091

17. 21 Genny McIvor McIvor Genny gmcivor@tripadvisor.com F 15.03.1991 28 6000 -8.7176937 115.5625434

18. 22 Jess Piddleton Piddleton Jess jpiddleton9@ips.gov M 25.09.1996 22 0000 44.151742 125.8395731

19. 23 Reider Romei Romei Reider rromeim@fsoseek.co.jp M 07.02.1999 20 9840 18.3051999 -77.3539774

20. 24 Chrissie Cade Cade Chrissie ccaden9@zmet.com M 04.09.2002 16 2840 -7.9780918 112.8007246

21. 25 Martin Mitchell Mitchell Martin mmitchall9@feng.com F 13.12.1993 25 1210 28.322781 -81.3912772

22. 26 Nalani Aspy Aspy Nalani nasypy@reguardian.com F 02.10.1997 21 3720 53.905048 27.5393261

23. 27 Urbanus Churms Churms Urbanus uchurms9@weibo.com M 20.05.1982 37 8780 60.1982825 29.6753924

24. 28 Itby Manketell Manketell Itby imanketell9@blogger.com F 11.08.2003 15 5130 57.3579277 61.6905815

25. 29 Sunny Southworth Southworth Sunny ssouthworths@ning.com F 11.05.1983 36 9480 50.7254936 -113.9749472

Here is the snapshot that illustrates after extraction of only last 4 digits of 'credit_card' column.

10. Delete full_name column.

Ans:

The 'full_name' column is first selected and under the 'Edit Column', 'Remove this column' option is selected in order to delete full_name column from our dataset. Hence, full_name column is completed deleted from our dataset.

OpenRefine question1 Permalink

Facet / Filter Undo / Redo 14 / 14 Extract... Apply...

Remove column full_name Undo

Extensions Wikidata

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

Filter:

1. Remove 51 rows

3. Text transform on 30 cells in column email:
value.toString()

4. Remove 30 rows

5. Text transform on 0 cells in column full_name:
value.toString()

6. Create new column f_name based on
column full_name by filling 919 rows with
grel:value.split("")[0]

7. Create new column l_name based on
column full_name by filling 919 rows with
grel:value.split("")[1]

8. Text transform on 0 cells in column sex:
value.toString()

9. Mass edit 448 cells in column sex

10. Mass edit 471 cells in column sex

11. Create new column age based on
column birth_date by filling 919 rows with
grel:diff('24.05.2019' toDate('dd/MM/yyyy'))

12. Text transform on 919 cells in column
credit_card:
value.toIntNumber()

13. Text transform on 919 cells in column
credit_card:
grel:value.substring(value.length()-4)

14. Remove column full_name

All id l_name f_name email sex birth_date age credit_card lat lng

1. 1 Burdis Amie aburdiso@army.mil M 31.01.1992 27 1160 38.0230669 23.8724272

2. 2 Burwhistle Piotr pburwhistle@ftc.gov M 07.11.1969 29 4140 40.0039528 -8.5608436

3. 5 Coonihan Malvina mcoonthan@constantcontact.com F 28.08.2005 13 4970 43.2 82.63333

4. 6 Dulany Barrie bdulany@abc.net.au M 15.09.1982 36 2160 18.1559131 -77.7652346

5. 7 Floris Floris Dacey dfloris6@columbia.edu F 06.02.1996 23 8000 54.0058465 17.7763729

6. 8 Edwirthie Marten Marten medwirthie@bloglovin.com M 07.02.1990 29 6420 16.3112941 104.0221147

7. 9 Couling Valentine Valentine vcouling@jhs.gov M 14.05.1999 20 9910 -7.2538172 -79.129784

8. 10 Beardsdale Kriste Kriste kbeardsdale9@imgur.com F 13.08.2005 13 8000 -25.99566 28.1971031

9. 11 Joselevitz Cordy Cordy cjoselevitz@cio.us M 15.03.1982 37 8100 5.9779631 -0.0866948

10. 12 Fallon Philly Philly pfallon9@tripadvisor.com F 06.06.1982 36 8982 -1.4493473 -79.4634825

11. 13 Jiruch Keivina Keivina kjruch@army.mil F 21.02.1993 26 3961 -31.2657738 -64.460387

12. 16 Newart Karen Karen knewart@freewebs.com F 27.06.1987 31 6660 36.182571 116.768358

13. 17 Deery Cletus Cletus cdeerryg@tripadvisor.com M 22.07.2001 17 6730 57.7244811 12.9309989

14. 18 Sanja Norie Norie nsanjan@mozilla.com M 06.10.1966 32 8640 -8.1708603 112.6256244

15. 19 Van Sandy Sandy svan@last.fm M 22.08.1999 19 9113 50.6375507 17.8149641

16. 20 Kirkman Honor Honor hkirkman9@amazon.co.uk F 28.06.1984 34 5130 52.0564395 20.0873091

17. 21 McIvor Genny Genny gmcivor@tripadvisor.com F 15.03.1991 28 6000 -8.7176937 115.5625434

18. 22 Piddleton Jess Jess jpiddleton9@ips.gov M 25.09.1996 22 0000 44.151742 125.8395731

19. 23 Romei Reider Reider rromeim@fsoseek.co.jp M 07.02.1999 20 9840 18.3051999 -77.3539774

20. 24 Cade Chrissie Chrissie ccaden9@zmet.com M 04.09.2002 16 2840 -7.9780918 112.8007246

21. 25 Mitchell Marlin Marlin mmitchall9@feng.com F 13.12.1993 25 1210 28.322781 -81.3912772

22. 26 Aspy Nalani Nalani nasypy@reguardian.com F 02.10.1997 21 3720 53.905048 27.5393261

23. 27 Churms Urbanus Urbanus uchurms9@weibo.com M 20.05.1982 37 8780 60.1982825 29.6753924

24. 28 Manketell Itby Itby imanketell9@blogger.com F 11.08.2003 15 5130 57.3579277 61.6905815

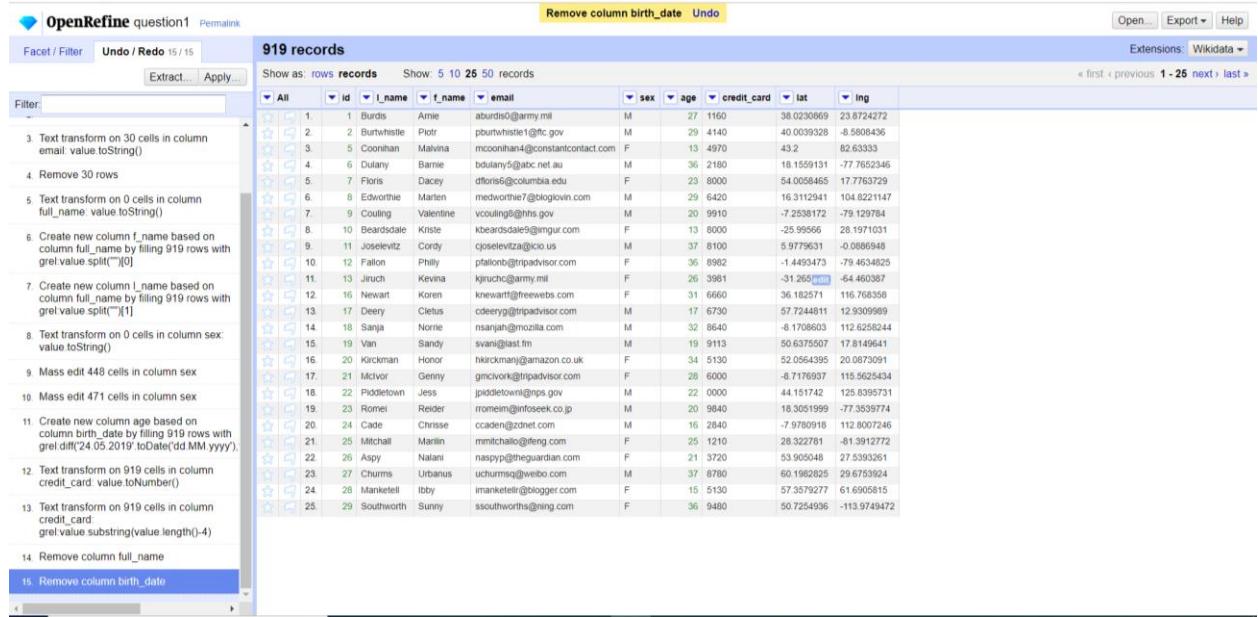
25. 29 Southworth Sunny Sunny ssouthworths@ning.com F 11.05.1983 36 9480 50.7254936 -113.9749472

Here is the snapshot that illustrates after the deletion of 'full_name' column.

11. Delete birth_date column.

Ans:

The 'birth_date' column is first selected and under the 'Edit Column', 'Remove this column' option is selected in order to delete birth_date column from our dataset. Hence, birth_date column is completed deleted from our dataset.



The screenshot shows the OpenRefine interface with the following details:

- Header:** OpenRefine question1 Permalink
- Toolbar:** Extract... Apply, Undo / Redo (15 / 15)
- Table Headers:** All, id, l_name, f_name, email, sex, age, credit_card, lat, lng
- Table Data:** A list of 919 records with columns for id, l_name, f_name, email, sex, age, credit_card, lat, and lng. The birth_date column has been removed.
- Left Panel:** A sidebar with 16 numbered steps for transforming the dataset, starting with "Text transform on 30 cells in column email value toString()". Step 15 is "Remove column birth_date".
- Bottom Bar:** Open..., Export..., Help, Extensions: Wikidata, first < previous 1 - 25 next > last

Here is the snapshot that illustrates after the deletion of 'birth_date' column.

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

Ans:

The API Key after successful signup for for OpenCage Reverse Geocoding API is **e5b4156dc39f4be08157344e22661eea**.

The screenshot shows the OpenCage API key retrieval interface. At the top, there's a 'Questions?' section with a link to help and a note about sending questions. Below it is the 'API key' section, which displays the generated key: **e5b4156dc39f4be08157344e22661eea**. There's also a link to a sample request using this key. A button labeled 'replace active API key' is present. Below this, there's a note about wanting multiple active keys and key labelling, with a link to upgrade to a paid plan. At the bottom, there's an 'Account' section showing details: Name (Rishi Karki), Company/Org name (empty), Email (rishi.karki7@gmail.com), Password (Not set (login via Google)), and two 'Edit' links. The entire interface has a clean, modern design with light colors and clear typography.

Here is the snapshot of retrieval of API key upon the successful signup for OpenCage Reverse Geocoding API.

13. URL format to retrieve location (JSON Data):

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

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).

Ans:

The '**lNg**' column is first selected and under '**Edit Column**', '**Add Column by fetching URLs...**' option is selected. Then, new column name of '**raw_data**' is defined with the GREL expression of

'[https://api.opencagedata.com/geocode/v1/json?q='+cells\["lat"\].value+'+'+value+'&key=e5b4156dc39f4be08157344e22661eea](https://api.opencagedata.com/geocode/v1/json?q='+cells[)'

in order to retrieve location JSON data by appending ‘lat’ column value, ‘lng’ column value and ‘API Key’ as mentioned in Q.No. 13’s URL link. Hence, web scrapping is successfully done and location JSON data is obtained in our dataset.

Here is the snapshot to derive new column as ‘raw_data’ from location JSON data using GREL expression appending ‘lat’, ‘lng’ and previously derived ‘API Key’ to URL, commonly known as web scrapping.

	All	id	l_name	f_name	email	sex	age	credit_card	lat	lng
1	1	Burdis	Arnie		aburdis@army.mil	M	27	1160	38.023069	23.8724272
2	2	Burtwhistle	Piotr		pbourwhistle@itc.gov	M	29	4140	40.0039528	-5.5808436
3	3	Coonihan	Malvina		mcoonihan4@constantcontact.com	F	13	4970	43.2	82.63333
4	4	Dutany	Bartee		bdutany@abc.net.au	M	36	2180	18.1559131	-77.7652346
5	5	Floris	Dacey		dfloris@columbia.edu	F	23	8000	54.0058465	17.7763729
6	6	Edworthie	Marlen		medworthie@plogovin.com	M	29	6420	16.3112941	104.8221147
7	7	Couling	Valentine		vcouling@hhs.gov	M	20	9910	-7.2581172	-79.129784
8	8	Beardsdale	Kristle		kbeardsdale@gmぐur.com	F	13	8000	-26.99566	28.1971031
9	9	Joselevitz	Cordy		cjoselevitz@cio.us	M	37	8100	5.9779651	-0.0866548
10	10	Fallon	Philly		pfallon0@tripadvisor.com	F	36	8982	-1.4493473	-79.4634825
11	11	Jiruchi	Keinya		kjiruchi@army.mil	F	26	3981	-31.2657738	-64.4605987
12	12	Newhart	Koren		knewhart@freewebs.com	F	31	6660	36.162571	116.768358
13	13	Deeny	Cletus		cdeeny@tripadvisor.com	M	17	6730	57.7244811	12.9309869
14	14	Sanja	Norrie		nsanjah@mozilla.com	M	32	8640	-8.1708603	112.6256244
15	15	Van	Sandy		svan@last.fm	M	19	9113	50.6375907	17.8149641
16	16	Kirkelman	Honor		hkirkelman@amazon.co.uk	F	34	5130	52.0564395	20.0873091
17	17	McTigue	Genny		gmctigue@tripadvisor.com	F	28	6000	-8.7176937	115.5625434
18	18	Piddleton	Jess		jpiddleton@nps.gov	M	22	0000	44.151742	125.6395731
19	19	Romei	Reider		rromeini@infoseek.co.jp	M	20	9640	18.3051999	-77.3599774
20	20	Cade	Chrissie		ccaden@znet.com	M	16	2840	-7.9780918	112.8007246
21	21	Mitchall	Marlin		mmitchall@feng.com	F	25	1210	28.322781	-61.3912772
22	22	Aspy	Natalia		naspy@theeguardian.com	F	21	3720	53.905044	27.7593261
23	23	Churms	Urbanus		uchurms@webo.com	M	37	8780	60.1962825	29.6753924
24	24	Manktelow	Ibby		imanktelow@blogger.com	F	15	5130	57.3579277	61.6905815
25	25	Southworth	Sunny		ssouthworths@ning.com	F	36	9480	50.7254936	-113.9749472

Here is the snapshot illustrating the progress of web scrapping process to derive ‘raw_data’ column from location JSON data.

The screenshot shows the OpenRefine interface with the following details:

- Facet / Filter**: Shows a list of filters applied to the 'raw_data' column, including:
 - 5. Text transform on 0 cells in column full_name: valueToString()
 - 6. Create new column f_name based on column full_name by filling 919 rows with gref.value.split()[0]
 - 7. Create new column l_name based on column full_name by filling 919 rows with gref.value.split()[1]
 - 8. Text transform on 0 cells in column sex: valueToString()
 - 9. Mass edit 448 cells in column sex
 - 10. Mass edit 471 cells in column sex
 - 11. Create new column age based on column birth_date by filling 919 rows with gref.diff('24/05 2019') toDate('dd MM yy')
 - 12. Text transform on 919 cells in column credit_card: valueToNumber()
 - 13. Text transform on 919 cells in column credit_card: gref.value.substring(value.length() - 4)
 - 14. Remove column full_name
 - 15. Remove column birth_date
 - 16. Create column raw_data at index 9 by fetching URLs based on column img using expression gref['https://api.opencagedata.com/geocod q=+cells["lat"] value+"+value &key=e5b
- Annotations**: A large JSON object representing geographical features like roads, rivers, and administrative boundaries.
- Extensions**: Shows 'Wikidata' and other available extensions.
- Help**: Standard OpenRefine help links.

Here is the snapshot that illustrates after the derivation of ‘raw_data’ column from location JSON data.

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

Ans:

The ‘**raw_data**’ column is first selected and then under the ‘**Edit Column**’, ‘**Add column based on this column...**’ option is selected. Then, new column names: ‘**continent**’ and ‘**country**’ are defined with the GREL expression of ‘**value.parseJson().results[0].components.continent**’ and ‘**value.parseJson().results[0].components.country**’ in order to filter continent and country from JSON data respectively. Hence, filtering of JSON data is done and new columns with continent and country is successfully added in our dataset.

Here is the snapshot that shows GREL expression to derive a new column ‘continent’ from ‘raw_data’ column.

Here is the snapshot that shows GREL expression to derive a new column ‘country’ from ‘raw data’ column.

OpenRefine question1		Permalink	Create new column country based on column raw_data by filling 919 rows with gref:value.parseJson().results[0].components.country	Undo	Extensions	Wikidata
Facet / Filter	Undo / Redo	18 / 18			first < previous 1 - 25 next > last	
919 records						
Show as:	rows	records	Show:	5 10 25 50 records		
Filter:						
1. column full_name by filling 919 rows with gref:value.split("[1])					Greece	Europe
8. Text transform on 0 cells in column sex: value.toLowerCase()					Portugal	Europe
9. Mass edit 448 cells in column sex					PRC	Asia
10. Mass edit 471 cells in column sex					Jamaica	North America
11. Create new column age based on column birth_date by filling 919 rows with gref:diff('24/05/2019' toDate('dd MM yyyy'))					Poland	Europe
12. Text transform on 919 cells in column credit_card: value.toUpperCase()						
13. Text transform on 919 cells in column credit_card: gref:value.substring(value.length()-4)						
14. Remove column full_name						
15. Remove column birth_date						
16. Create column raw_data at index 9 by fetching URLs based on column Ing using expression: gref:https://api.opendatakit.org/geocoding?q=+\${cells['lat']}+\${value}+\${value}+&key=e5b-						
17. Create new column continent based on column raw_data by filling 917 rows with gref:value.parseJson().results[0].components						
18. Create new column country based on column raw_data by filling 919 rows with gref:value.parseJson().results[0].components						
19. Remove column raw_data						

Here is the snapshot that illustrates the result of two new derived columns ‘country’ and ‘continent’.

16. Delete raw_data column.

Ans:

The ‘raw_data’ column is first selected and under the ‘Edit Column’, ‘Remove this column’ option is selected in order to delete raw_data column from our dataset. Hence, raw_data column is completed deleted from our dataset.

OpenRefine question1		Permalink	Remove column raw_data	Undo	Extensions	Wikidata
Facet / Filter	Undo / Redo	19 / 19			first < previous 1 - 25 next > last	
919 records						
Show as:	rows	records	Show:	5 10 25 50 records		
Filter:						
1. Burdiso, Arme aburdiso@army.mil	M	27	1160	38.023069	23.8724272	Greece Europe
2. Burwinski, Piotr pburwinski1@tfc.net	M	29	4140	40.0039328	-8.59008436	Portugal Europe
3. Coonihan, Malvina mcoonihan@constantcontact.com	F	13	4970	43.2	82.63333	PRC Asia
4. Dutany, Dacei dfloris6@columbia.edu	M	36	2180	18.1559131	-77.7652346	Jamaica North America
5. Floris, Dacei dfloris6@columbia.edu	F	23	800	54.0058465	17.7763729	Poland Europe
6. Edworthy, Marlen medworth@otogivm.com	M	29	6420	16.3112941	104.6221147	Thailand Asia
7. Couling, Valentine vcouling@hhs.gov	M	20	9910	-7.2558172	-7.9929784	Pere South America
8. Beardstone, Kristie kbeardstone9@imgur.com	F	13	8000	-25.99566	26.1971031	RSA Africa
9. Joselevitz, Cory cjoselevitz@cio.us	M	37	8100	5.9779631	0.0866948	Ghana Africa
10. Fallon, Philly ptalton@tripadvisor.com	F	36	8962	-1.4493473	-7.99438025	Ecuador South America
11. Jiruchi, Kevena kjruchi@army.mil	F	26	3981	-31.2657738	-64.460387	Argentina South America
12. Newark, Karen knewatt@freewebs.com	F	31	6660	36.162571	116.768358	PRC Asia
13. Deepy, Cletus cdeerry@tripadvisor.com	M	17	6730	57.7244811	12.9309989	Sweden Europe
14. Sanja, Norrie nsanjah@mozilla.com	M	32	8640	-8.1708603	112.6258244	Indonesia Asia
15. Van, Sandy svani@last.fm	M	19	9113	50.6375007	17.8149641	Poland Europe
16. Kirkman, Honor hrkrmn@amazon.co.uk	F	34	5130	52.0564395	20.0673091	Poland Europe
17. McIvor, Gemma gmcwork@tripadvisor.com	F	28	6000	-8.7176937	115.5625434	Indonesia Asia
18. Piddleton, Jess jpiddleton1@ips.gov	M	22	0000	44.151742	125.6395733	PRC Asia
19. Rome, Reider rromeim@forsseek.co.jp	M	20	9840	18.3051999	-77.3559774	Jamaica North America
20. Cade, Chrissie ccaden@zmet.com	M	16	2840	-7.9780918	112.0007245	Indonesia Asia
21. Mitchell, Marlin mmitchalo@fifeng.com	F	25	1210	28.322781	-81.3912772	USA North America
22. Aspy, Natani naspp@theguardian.com	F	21	3720	53.905048	27.53930261	Belarus Europe
23. Churns, Urbanus uchurns@weibo.com	M	37	8780	60.1962625	29.6753924	Russia Europe
24. Manekell, Itby imanekelle@blogger.com	F	15	5130	57.3579277	61.6905815	Russia Europe
25. Southworth, Sunny ssouthworths@ning.com	F	36	9480	50.7254936	-113.9749472	Canada North America

Here is the snapshot that illustrates the final dataset after the deletion of ‘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**

Finally, the project as well as CSV file are exported and renamed project file as ‘**solution1.tar.gz**’ and CSV file as ‘**solution1.csv**’.

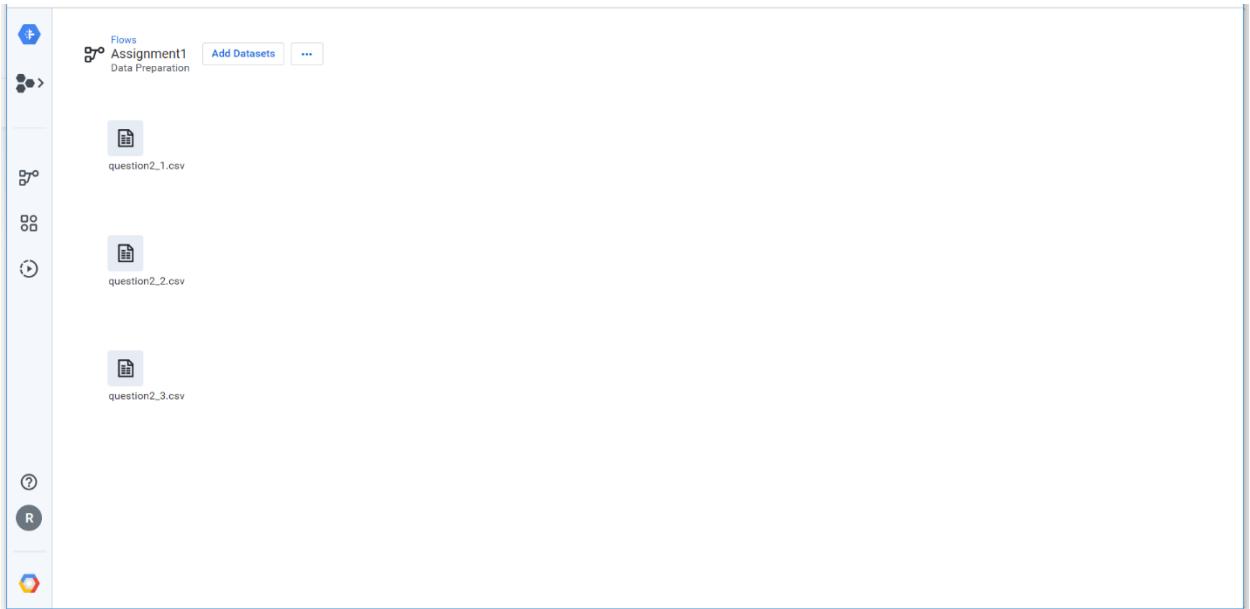
The ‘**solution1.csv**’ has total of 920 records including header with 11 column names as: id, l_name, f_name, email, sex, age, credit_card, lat, lng, country and continent.

3.2 Data Preparation using Google Cloud Dataprep

Question 2: Data Preparation using Google Cloud Dataprep.

Preliminary Steps:

1. First of all, we have done successful signup for Google Cloud Dataprep.
2. Then, we go to following URL:
<https://cloud.google.com/dataprep/>
3. Select “View Console”
4. Under sidebar, Select Flows/ Create.../Create Flow with flow description as:
Flow Name: Assignment1
Flow Description: Data Preparation
5. Choose Add Datasets/ Import Dataset/Choose a file. Browse the file and import all 3 datasets (question2_1.csv, question2_2.csv, question2_3.csv) and add it to a flow.
Note: We imported files by converting given .xlsx file into .csv file.



Here is the snapshot of initial flow after 3 datasets are loaded.

Procedural Steps:

1. Select **question2_1.csv**, choose “Add new Recipe” and then “Edit Recipe”.
2. Click on first rows, then Select ‘Add’ to convert row (stock_market, stock_symbol) to header.

The screenshot shows the Data Preparation interface with the following details:

- Left Sidebar:** Includes icons for Flows, ASSIGNMENT1 > question2_1 > Full Data, and three dots.
- Content Area:** Shows the 'Preview' of the 'question2_1' dataset with two columns: 'stock_market' and 'stock_symbol'.
- Right Sidebar:** Shows the 'Header' recipe step with the following configuration:
 - Header:** Convert row 1 to header
 - Delete rows:** row 1
 - Keep rows:** row 1

Here is the snapshot for step to add header after clicking on first rows on dataset. We have initially of 967 rows including header rows.

3. Under ‘stock_symbol’ dropdown, Select **Filter rows/Remove duplicate rows/Add** to remove all the duplicate pairs of stock_symbol and stock_market.

The screenshot shows a data processing interface with a preview window displaying two columns: 'stock_market' and 'stock_symbol'. The 'stock_symbol' column is highlighted with a yellow header. The sidebar on the left shows category counts: '2 Categories' for 'stock_market' (NYSE, NASDAQ) and '906 Categories' for 'stock_symbol'. The toolbar at the top includes a 'Remove duplicate rows' button. The bottom status bar indicates '2 Columns', '966 Rows', and '1 Data Type'.

Here is the snapshot to remove the duplicate rows where we had total duplicate of 60 rows.

By this, we are done cleaning for question2_1.csv as there are no any duplicate records, empty records or any other inconsistent records as we have also ensured it by thoroughly examining each record.

The screenshot shows a data cleaning interface with a table containing two columns: 'stock_market' and 'stock_symbol'. The 'stock_symbol' column lists 906 unique entries. A sidebar on the right displays two steps: 'Convert row 1 to header' and 'Remove duplicate rows'.

Here is the snapshot of question2_1.csv after doing necessary cleaning where we are able to get total of 906 records.

4. Similarly, select **question2_2.csv**, choose “Add new Recipe” and then “Edit Recipe”.
5. Click on first rows, then Select ‘Add’ to convert row (stock_symbol, stock_sector) to header.
6. Under ‘stock_sector’ drop down, Select Filter rows/Remove duplicate rows/Add to remove all the duplicate pairs of stock_sector with stock_symbol.
By this, we remove all the 60 duplicate pairs of stock_sector with stock_symbol.
7. Under ‘stock_sector’ drop down, Select Filter rows/On Column values/Contains.
Input in ‘Pattern to match’ with “n/a” and make sure to select ‘Delete matching rows’ under ‘Action’ option and click on ‘Add’ to remove all the pairs of stock_symbol and stock_sector with stock_sector having ‘n/a’ values.
By this, we delete all 189 records of stock_sector with ‘n/a’ values.

By this, we are done cleaning for question2_2.csv as there are no any duplicate records, empty records, records with ‘n/a’ values or any other inconsistent records as we have also ensured it by thoroughly examining each record.



Here is the snapshot of question2_2.csv after doing necessary cleaning where we are able to get total of 717 records.

8. Similarly, select **question2_3.csv**, choose “Add new Recipe” and then “Edit Recipe”.
9. Click on first rows, then Select ‘Add’ to convert row (stock_symbol, stock_market_cap) to header.
10. Under ‘stock_market_cap’ drop down, Select Filter rows/Remove duplicate rows/Add to remove all the 60 duplicate pairs of stock_sector with stock_symbol.
By this, we remove all the duplicate pairs of stock_market_cap with stock_symbol.
11. Under ‘stock_market_cap’ drop down, Select Filter rows/On Column values/Contains.

Input in ‘Pattern to match’ with “n/a” and make sure to select ‘Delete matching rows’ under ‘Action’ option and click on ‘Add’ to remove all the pairs of stock_symbol and stock_market_cap with stock_market_cap having ‘n/a’ values.

By this, we delete all 139 records from stock_market_cap with ‘n/a’ values.

By this, we are done cleaning for question2_2.csv as there are no any duplicate records, empty records, records with ‘n/a’ values or any other inconsistent records as we have also ensured it by thoroughly examining each record.

Since, all our desired output dataset should have ‘stock_market_cap’ equal or more than \$30M, the transformation of capital figure with the given constraint will be done just after finishing the above cleaning process.

12. Under ‘Search’ menu, Select ‘Conditional Column’ as a transformation option followed by “If... then...else” condition type with following values for each clause:

if: MATCHES([stock_market_cap],'B')

Then: ROUND(SUBSTRING(stock_market_cap,1,LEN(stock_market_cap)-1)*1000000000,0)

Else: ROUND(SUBSTRING(stock_market_cap, 1,LEN(stock_market_cap)-1)*1000000,0)

New column name: stock_market_numeric

By this, we are transforming ‘stock_market_cap’ into respective numerical figure of millions and billions for the remaining 767 records. And, now we delete the ‘stock_market_cap’ column.

The screenshot shows a data processing interface with a preview of a dataset and a transformation recipe.

Preview:

RBC	stock_symbol	RBC	stock_market_cap	#	stock_market_numeric
767 Categories	AAV	\$1.15B	1.04M - 269.31B		1150000000
	ABG	\$1.15B			1150000000
	ABIL	\$31.94M			31940000
	ABR	\$515.96M			515960000
	ADHD	\$30.87M			30870000
	ADI	\$29.38B			2938000000
	ADSW	\$2.18			2100000000
	AEG	\$10.51B			10510000000
	AET	\$49.95B			4995000000
	AETI	\$12.31M			12310000
	AFH	\$188.28M			188280000
	AGIO	\$2.5B			2500000000
	AGLE	\$47.88M			47088000
	AGNCB	\$9.12B			9120000000
	AGR	\$114.16B			11416000000
	AGRO	\$1.21B			1210000000
	AHP	\$313.01M			313810000
	AHPI	\$8.91M			8910000
	AI	\$352.64M			352640000
	AIA	\$385.88M			385880000
	AIF	\$236.63M			236630000
	ALDX	\$65.87M			650780000
	ALE	\$3.72B			3720000000
	ALL	\$32.74B			32740000000

Condition type: required
If: `MATCHES([stock_market_cap], 'B')`
Then: `ROUND(SUBSTRING(stock_market_cap, 1, LEN(stock_market_cap) - 1) * 100000000, 0)`
Else: `ROUND(SUBSTRING(stock_market_cap, 1, LEN(stock_market_cap) - 1) * 1000000, 0)`
New column name: `stock_market_numeric`

Here is the snapshot of conditional transformation of 'stock_market_cap' to 'stock_market_numeric' through if-then-else condition type for dataset question2_3 where we have extracted the million and billion value to numerical figures by the help of substring function along with round function and created new column 'stock_market_numeric' to store the computed value.

13. Since, 'stock_market_cap' value for all the output file should be equal or greater than \$30 M (30000000), now we filter the rows with this constraint.

Under, 'stock_market_numeric' dropdown menu, Select Filter rows/On column values/Greater than (or equal to). In 'Value is greater than' option, type 30000000 and check the label 'Or equal to' with action as 'Keep matching rows' and click 'Add'.

The screenshot shows a data processing interface with a preview of a dataset named 'question2_3'. The preview table has two columns: 'stock_symbol' and 'stock_market_numeric'. The data shows various stock symbols and their market values. A filter dialog box is open on the right, set to filter rows where 'stock_market_numeric' is greater than or equal to 30,000,000. The dialog also includes options for 'Or equal to', 'Keep matching rows', and 'Delete matching rows'.

Here is the snapshot to filter 'stock_market_numeric' value with the constraint of equal or greater than \$30M for dataset question2_3. By this, we ignore all the 54 rows where the constraint is not matched and we are left with only 713 rows.

14. Now, let's rename the 'stock_market_numeric' to 'stock_market_cap' by selecting 'Rename' option under drop down menu with new column name as 'stock_market_cap' and Click on 'Add' to confirm.

By this, we are done with either of the cleaning and transformation process.

The screenshot shows a data processing interface with a table containing 713 rows. The columns are labeled 'stock_symbol' and 'stock_market_cap'. The data includes various stock symbols like AAV, ABG, ABIL, ABR, ADHD, ADI, ADSW, AEG, AET, AFH, AGIO, AGLE, AGNCB, AGR, AGRO, AHP, AI, AIA, ATF, ALDX, ALE, ALL, ALOT, and AM, along with their corresponding market caps. A sidebar on the right displays a sequence of 7 steps:

- 1 Convert row 1 to header
- 2 Remove duplicate rows
- 3 Delete rows where stock_market_cap contains 'nVa'
- 4 Create stock_market_numeric from condition: if MATCHES([stock_market_cap], B) then ROUND(SUBSTRING(stock_market_cap, 1, LEN(stock_market_cap) - 1) * 1000000000) else stock_market_cap
- 5 Delete stock_market_cap
- 6 Keep rows where stock_market_numeric > 300000000 where stock_market_numeric >= true
- 7 Rename stock_market_numeric to 'stock_market_cap'

Here is the snapshot of question 2_3.csv after cleaning and transformation process where we are left with only 713 rows.

Now, the next challenge is to join all the three datasets into one, with joining two adjacent datasets and finally deriving one complete dataset. Firstly, lets join two datasets: ‘question2_1’ and ‘question2_2’ which are discussed below:

- 15.** Go back to ‘Flows’ window. Select ‘question2_1.csv’ and Click on ‘Add new Recipe’ to create as ‘question2_1-2’ and then select ‘Edit Recipe’.
- 16.** Under ‘Search’ transformation menu, type ‘Join datasets’ and then select ‘question2_2’ and click on ‘Accept’.

ASSIGNMENT1 > question2_1 – 2 > Initial Sample

Search transformations Run Job

Join datasets

Join

2 Categories

stock_market	stock_symbol
NYSE	AAV
NYSE	ABG
NASDAQ	ABIL
NYSE	ABR
NASDAQ	ACTX
NASDAQ	ADHD
NASDAQ	ADI
NYSE	ADSW
NYSE	AEG
NYSE	AET
NASDAQ	AETI
NYSE	AFGH
NASDAQ	AFH
NYSE	AFSI^A
NYSE	AFSI^F
NASDAQ	AGIO
NASDAQ	AGLE
NYSE	AGM_A
NASDAQ	AGNCB
NYSE	AGR
NYSE	AGRO
NYSE	AHL^C
NYSE	AHP

2 Columns 906 Rows 1 Data Type

Here is the snapshot for the initial process to join two datasets. The newly created recipe is ready to join to another dataset.

ASSIGNMENT1 > question2_1 – 2 > Initial Sample

Choose dataset or recipe to join with question2_1 – 2

Recipes in current flow Datasets in current flow All datasets

Name Last Updated Source Data Recipe

rec_stock_symbol	rec_stock_sector
AAV	Energy
ABG	Consumer Durables
ABIL	Finance
ABR	Consumer Services
ADHD	Health Care
ADI	Technology
ADSW	Public Utilities
AEG	Finance
AET	Health Care
AETI	Energy

join in... Edit

Cancel Next

Here is the snapshot to join newly created recipe to another dataset and question2_2 dataset is selected for join operation.

17. Make sure Join type is selected as ‘Inner’, Click ‘Next’. Then, make sure to check the label for ‘stock_sector’ which is one of the column of ‘question2_2’ dataset and uncheck one

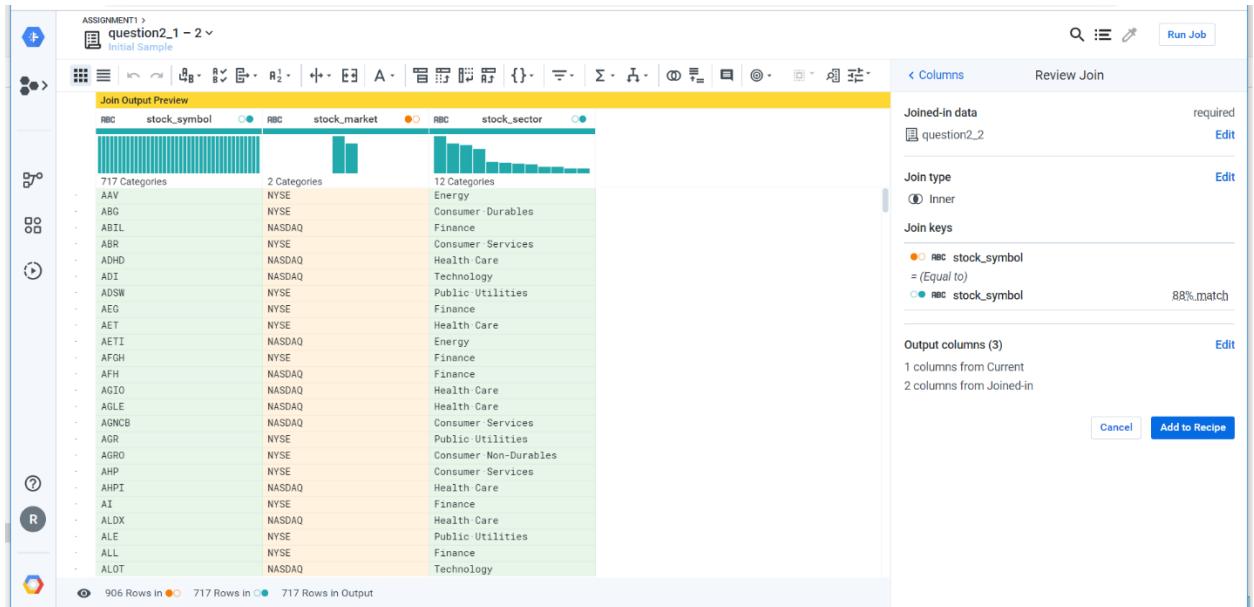
of the primary key column i.e. stock_symbol. Finally, click on ‘Add to Recipe’ to confirm execution of the join operation.

The screenshot shows a data processing interface with a toolbar at the top and a main workspace below. In the workspace, there is a 'Join Key' section where two datasets are joined on the 'stock_symbol' column. The left dataset has 906 categories and the right dataset has 717 categories. The results summary indicates 906 rows in the current dataset, 717 rows in the joined dataset, and 717 rows in the output. On the right side, there are tabs for 'Joined-in Data' and 'Join Conditions'. Under 'Join Conditions', the join type is set to 'Inner' and the join key is 'RBC stock_symbol = (Equal to) RBC stock_symbol'. A note says 'Suggested 98% match'. Below this is a 'Results summary' table showing the count of rows based on current samples.

Here is the snapshot which shows that two datasets are joined together by common column in each dataset i.e. ‘stock_symbol’.

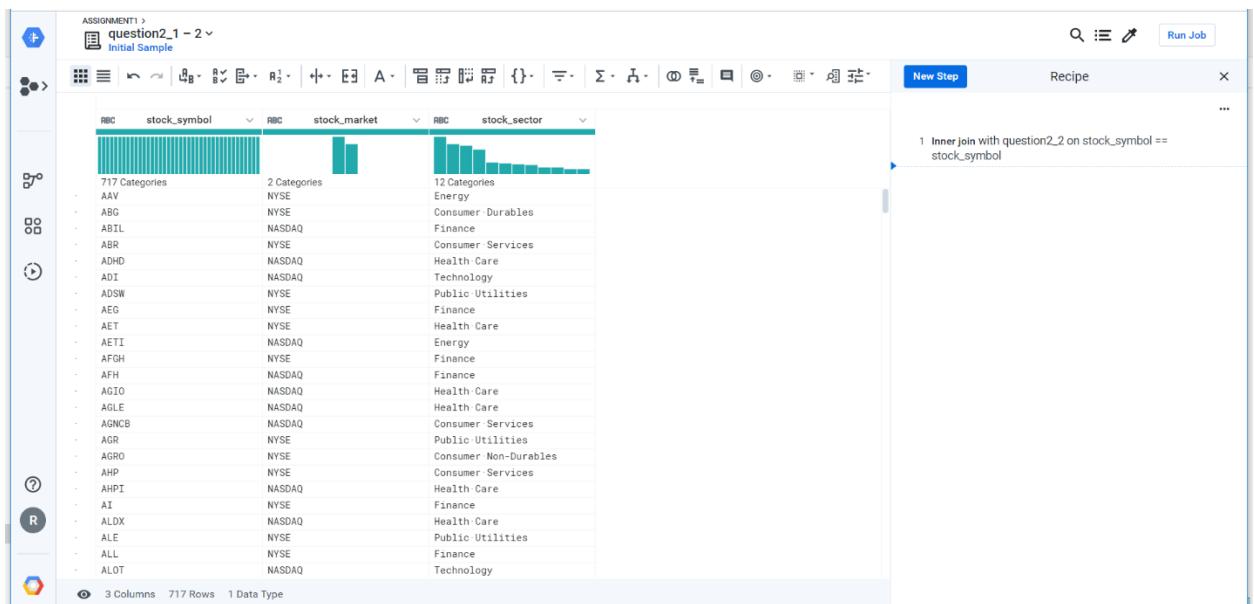
This screenshot shows the same interface but with a different join configuration. The 'Join Output Preview' section now includes the 'stock_market' and 'stock_sector' columns from the second dataset. The results summary remains the same (906 rows in current, 717 rows in joined, 717 rows in output). The 'Output Columns' tab on the right is selected, showing the columns 'Column', 'Source', and 'Current (2)' status for 'stock_symbol', 'stock_market', and 'stock_sector'. There are also tabs for 'Conditions' and 'Advanced options'.

Here is the snapshot for output column where we specify only one of the common columns i.e. stock_symbol as a basis for join operation.



Here is the snapshot review indicating that two datasets are successfully able to join together, and it is ready to be added in the newly created recipe.

Here, stock_symbol column will be the base for joining two datasets: question2_1 and question2_2.



Here is the snapshot as a result of join operation of datasets question2_1 and question2_2 where we have total of 717 rows as a result of join operation.

Now, lets join the resultant dataset with ‘question2_3’ dataset. For this process, steps are mentioned below:

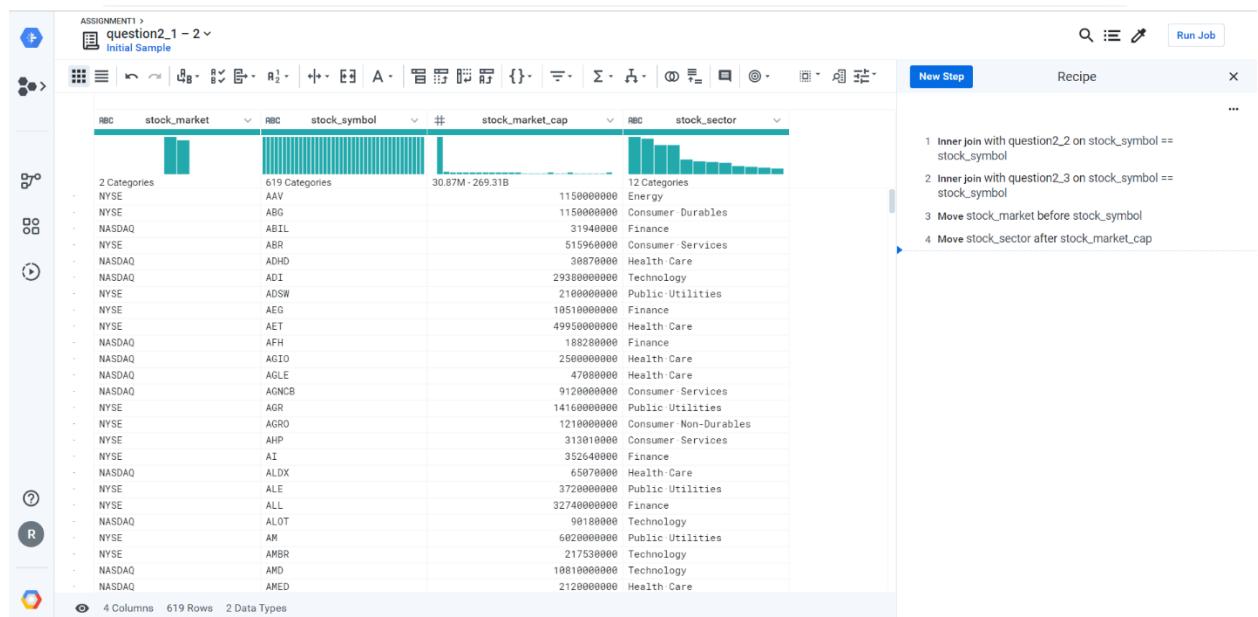
18. Again, select ‘question2_1-2’ dataset and then select ‘Edit Recipe’.

19. Under ‘Search’ transformation menu, type ‘Join datasets’ and then select ‘question2_3’ and click on ‘Accept’.

20. Make sure Join type is selected as ‘Inner’, Click ‘Next’. Then, make sure to check the label for ‘stock_market_cap’ which is one of the column of ‘question2_3’ dataset and uncheck one of the primary key column i.e. stock_symbol. Finally, click on ‘Add to Recipe’ to confirm execution of the joint operation.

Here, stock_symbol column will be the base for joining two datasets: question2_1-2 and question 2_3.

Now, joining of three datasets into one is done and we are successfully able to join and merge it together. For the sake of representation of final resultant dataset, we move ‘stock_market’ column to the beginning and ‘stock_sector’ column to the end.



Here is the snapshot of combined datasets of question2_1, question2_2 and question2_3 as a result of series of join operation with it's adjacent dataset, first join with question2_1 and question2_2 datasets and then resultant dataset with question2_3 and as a result we are able to get total of 619 rows altogether as a result of combined join operation.

Now, the next challenge is to extract the datasets according to output dataset requirement in reference to stocks related to:

- a. finance sector
- b. technology sector.
- c. health care sector.

21. For the output dataset 1 as of stocks related to **finance sector** (solution2_1.csv), we create a new recipe as ‘solution2_1’ and edit recipe.

22. Double click on one of the records with ‘Finance’ as stock_sector, then at the right side of the window with ‘Keep rows’ tab having option of ‘with value matching ‘Finance’, select ‘Add’.

With this, it will keep all rows matching ‘Finance’ as stock_sector discarding rows other than ‘Finance’ sector.

The screenshot shows a data processing interface with a preview of a dataset named 'question2_1 - 3'. The preview shows 619 rows across four columns: stock_market, stock_symbol, stock_market_cap, and stock_sector. The stock_sector column contains values like Energy, Finance, Consumer Durables, Health Care, Technology, and Public Utilities. A sidebar on the right provides filtering options for the 'stock_sector' column, specifically for the 'Finance' sector. It includes suggestions like '(alpha)+', '(alpha)(7)', and 'Extract list of values' for 'Finance'. The 'Keep rows' section has an 'Edit' button and an 'Add' button, both of which are currently disabled. The 'Delete rows' section also has an 'Edit' button and an 'Add' button, both of which are currently disabled.

Here is the snapshot of filtering 'Finance' stock_sector for solution2_1 dataset where we are able to get total of 110 records meeting the filtered criteria.

23. Then, delete the 'stock_sector' column.

By this, our desired output dataset 1 containing the stocks related to **finance sector** is prepared having three columns of **stock_market**, **stock_symbol** and **stock_market_cap** with stock market capital equal or more than \$30M.

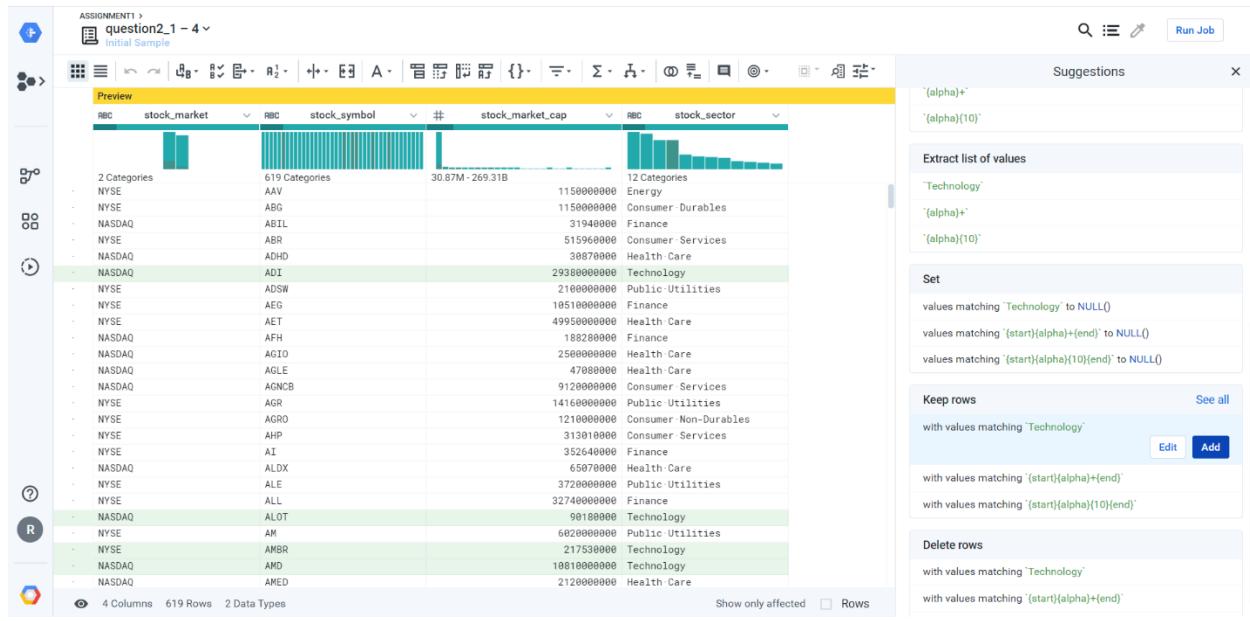
stock_market	stock_symbol	stock_market_cap
NASDAQ	ABIL	319400000
NYSE	AEG	1051000000
NASDAQ	AFH	188280000
NYSE	AI	352640000
NYSE	ALL	3274000000
NYSE	ARL	137360000
NASDAQ	BBVA	5398000000
NASDAQ	BLMT	292590000
NASDAQ	BLVD	461580000
NYSE	BMA	507000000
NASDAQ	BOKF	554000000
NYSE	BRO	622000000
NYSE	BSMX	1314000000
NASDAQ	BUSE	115000000
NYSE	CB	7081000000
NYSE	CFR	619000000
NYSE	CIB	1102000000
NASDAQ	CIZN	122360000
NYSE	CMA	1311000000
NASDAQ	CNFR	51520000
NYSE	CNO	353000000
NASDAQ	CNOB	744130000
NYSE	CO	666520000
NASDAQ	CPAA	596200000
NASDAQ	CSFL	1390000000

Here is the snapshot after removing filtered rows from 'Finance' stock_sector which is our final output dataset 1(solution2_1) containing 110 records of data.

24. Similarly, for the output dataset 2 as of stocks related to **technology sector** (solution2_2.csv), we create a new recipe as 'solution2_2' and edit recipe.

25. Double click on one of the records with 'Technology' as stock_sector, then at the right side of the window with 'Keep rows' tab having option of 'with value matching 'Finance'', select 'Add'.

With this, it will keep all rows matching 'Technology' as stock_sector discarding rows other than 'Technology' sector.



Here is the snapshot of filtering 'Technology' stock_sector for solution2_2 dataset where we are able to get total of 86 records meeting the filtered criteria.

26. Then, delete the 'stock_sector' column.

By this, our desired output dataset 2 containing the stocks related to **technology sector** is prepared having three columns of **stock_market**, **stock_symbol** and **stock_market_cap** with stock market capital equal or more than \$30M.

ASSIGNMENT1 > solution2_2 > Initial Sample

New Step Recipe X ...

1 Keep rows where MATCHES([stock_sector], 'Technology')
2 Delete stock_sector

stock_market	stock_symbol	stock_market_cap
2 Categories		
NASDAQ	AD1	29380000000
NASDAQ	ALOT	90180000
NYSE	AMBR	217530000
NASDAQ	AMD	10810000000
NASDAQ	AMSWA	326530000
NASDAQ	AOSL	429148000
NASDAQ	AYA	2620000000
NASDAQ	BIDU	6038000000
NASDAQ	BLKB	4160000000
NYSE	CAT	403940000
NASDAQ	CCMP	1880000000
NASDAQ	CEVA	975950000
NASDAQ	CHUBA	725750000
NYSE	CLGX	3610000000
NASDAQ	COMM	7170000000
NASDAQ	COVS	99100000
NASDAQ	CREE	2350000000
NASDAQ	CSOD	2090000000
NASDAQ	CTG	84390000
NASDAQ	DSGX	1870000000
NASDAQ	DSPG	258270000
NASDAQ	DWCH	95260000
NASDAQ	ENOC	179440000
NASDAQ	EXA	203270000
NASDAQ	FNSR	3100000000

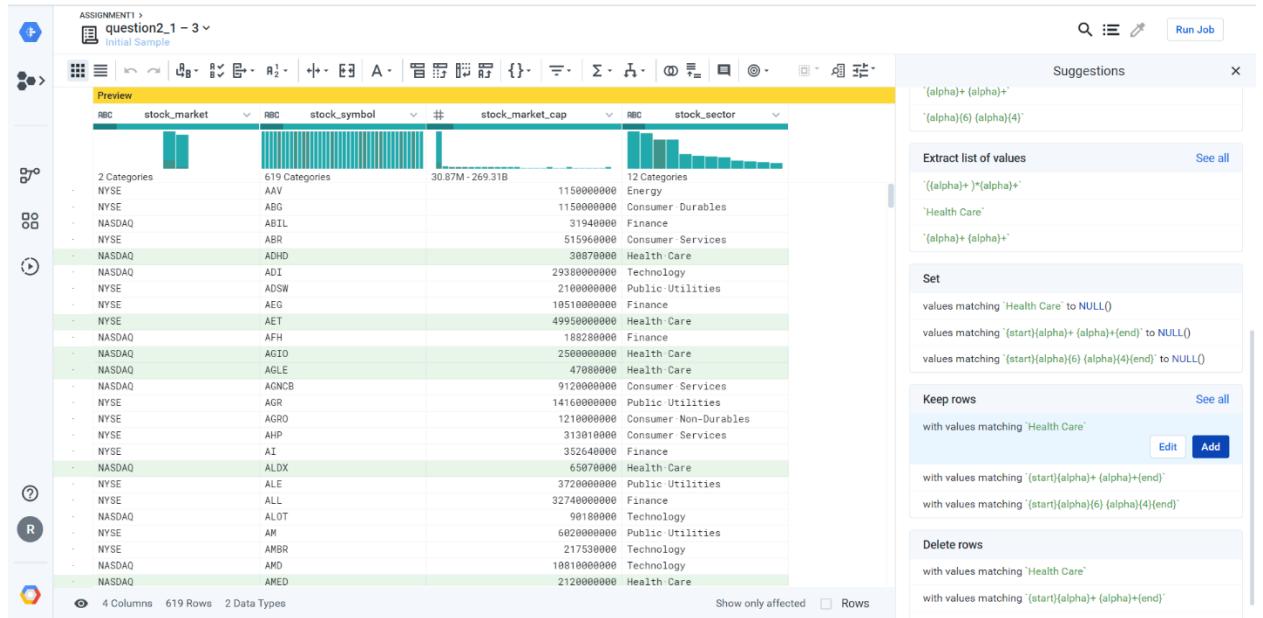
3 Columns 86 Rows 2 Data Types

Here is the snapshot after removing filtered rows from 'Technology' stock_sector which is our final output dataset 2(solution2_2) containing 86 records of data.

27. Similarly, for the output dataset 2 as of stocks related to **health care sector** (solution2_3.csv), we create a new recipe as 'solution2_3' and edit recipe.

28. Double click on one of the records with 'Health Care' as stock_sector, then at the right side of the window with 'Keep rows' tab having option of 'with value matching 'Health Care'', select 'Add'.

With this, it will keep all rows matching 'Health Care' as stock_sector discarding rows other than 'Health Care' sector.



Here is the snapshot of filtering 'Health Care' stock_sector for solution2_3 dataset where we are able to get total of 86 records meeting the filtered criteria.

29. Then, delete the 'stock_sector' column.

By this, our desired output dataset 3 containing the stocks related to **health care sector** is prepared having three columns of **stock_market**, **stock_symbol** and **stock_market_cap** with stock market capital equal or more than \$30M.

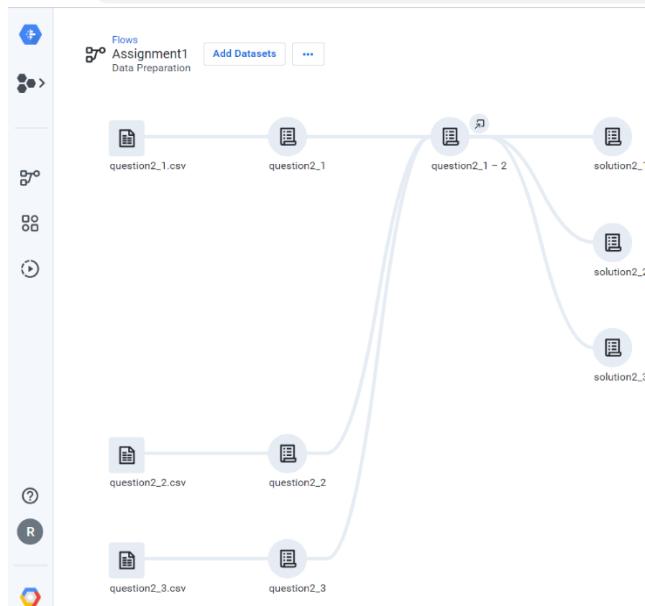
ASSIGNMENT1 > solution2_3 > Initial Sample

RBC	stock_market	RBC	stock_symbol	#	stock_market_cap
	NASDAQ		ADHD	30.87M - 87.46B	30870000
	NYSE		AET		499500000000
-	NASDAQ		AGIO		2500000000
-	NASDAQ		AGLE		47080000
-	NASDAQ		ALDX		65070000
-	NASDAQ		AMED		2120000000
-	NASDAQ		AMRI		929630000
-	NASDAQ		ANAB		48079000
-	NASDAQ		ANGO		587520000
-	NASDAQ		ARAY		361050000
-	NASDAQ		ARLZ		79480000
-	NASDAQ		ARWR		118890000
-	NASDAQ		ATHX		150070000
-	NASDAQ		ATRI		1040060000
-	NYSE		AZN		87460000000
-	NASDAQ		BCLI		82810000
-	NASDAQ		BCRX		446330000
-	NASDAQ		BYSI		835650000
-	NASDAQ		CARA		556530000
-	NASDAQ		CCXI		351670000
-	NASDAQ		CHRS		761630000
-	NASDAQ		CMRX		234190000
-	NASDAQ		CRIS		254480000
-	NASDAQ		CYAD		445000000
-	NYSE		DPLQ		12100000000

3 Columns 86 Rows 2 Data Types

Here is the snapshot after removing filtered rows from 'Health Sector' stock_sector which is our final output dataset 3(solution2_3) containing 86 records of data.

By this, we are successfully able to prepare the three different desired output datasets i.e. solution2_1.csv, solution2_2.csv and solution2_3.csv which satisfies all the given condition and constraints. The final flow of our solution can be visualized as follows which is sum of all the above discussed steps:



Here is the snapshot of the final flow of data preparation, which clearly previews our final output datasets.

Now, the next challenge is to generate the output dataset as we have already got the three output datasets (solution2_1, solution2_2 and solution2_3) reflected in the final flow of data preparation.

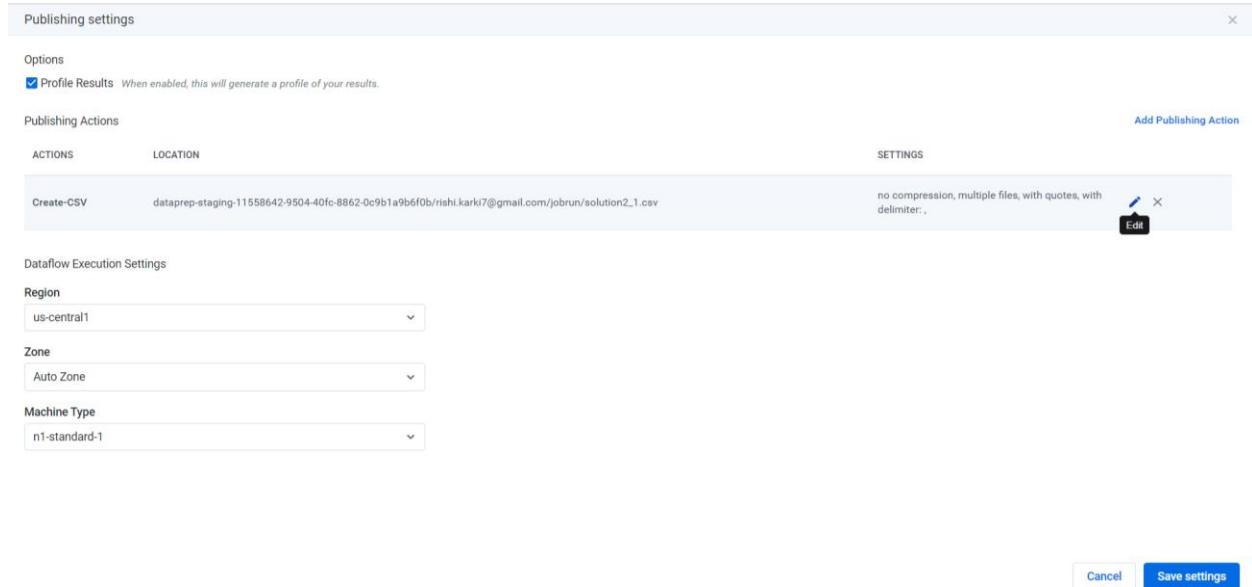
The process of generating final output dataset ‘solution2_1.csv’ is shortly described below:

- 30.** Under the ‘Flows’ menu, ‘Run Job’ function is activated , then select the dataset ‘solution2_1’ and click on ‘tray’ icon which is just above the dataset icon.



Here is the snapshot of the initial step to ‘edit’ menu of ‘Run Job’ function in generating output dataset ‘solution2_1.csv’

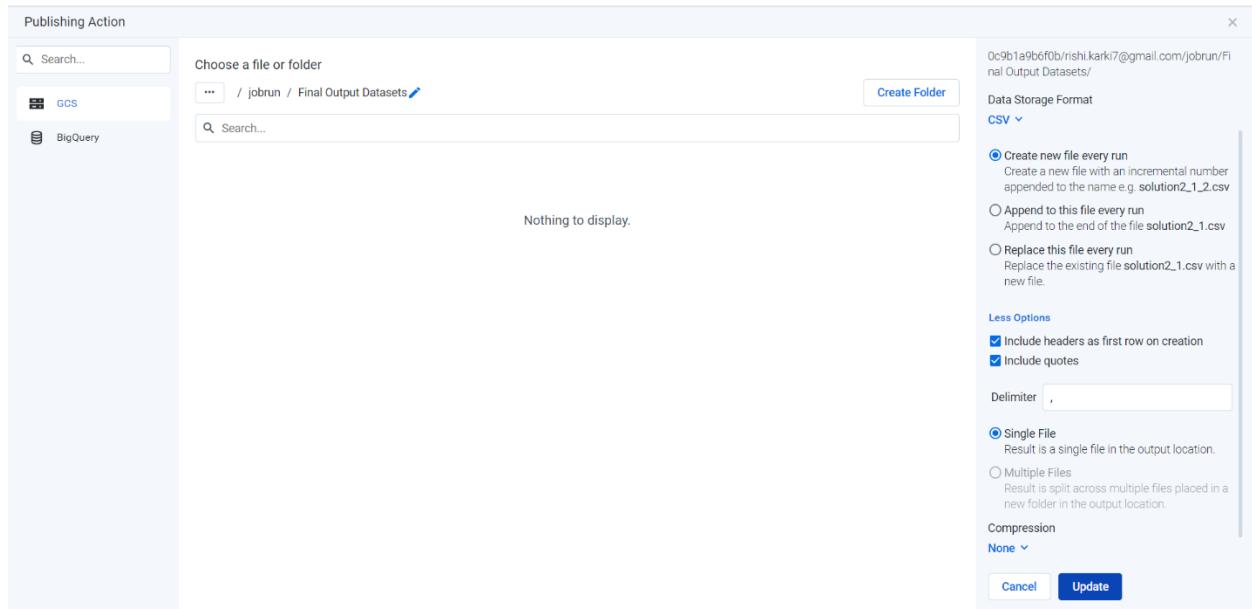
- 31.** Under ‘Destination’ tab, Click on ‘Edit’ menu just left to ‘Manual Destinations’. Select ‘Edit’ option in setting tab.



Here is the snapshot after clicking on edit menu from previous step which prompts to go to another ‘edit’ option.

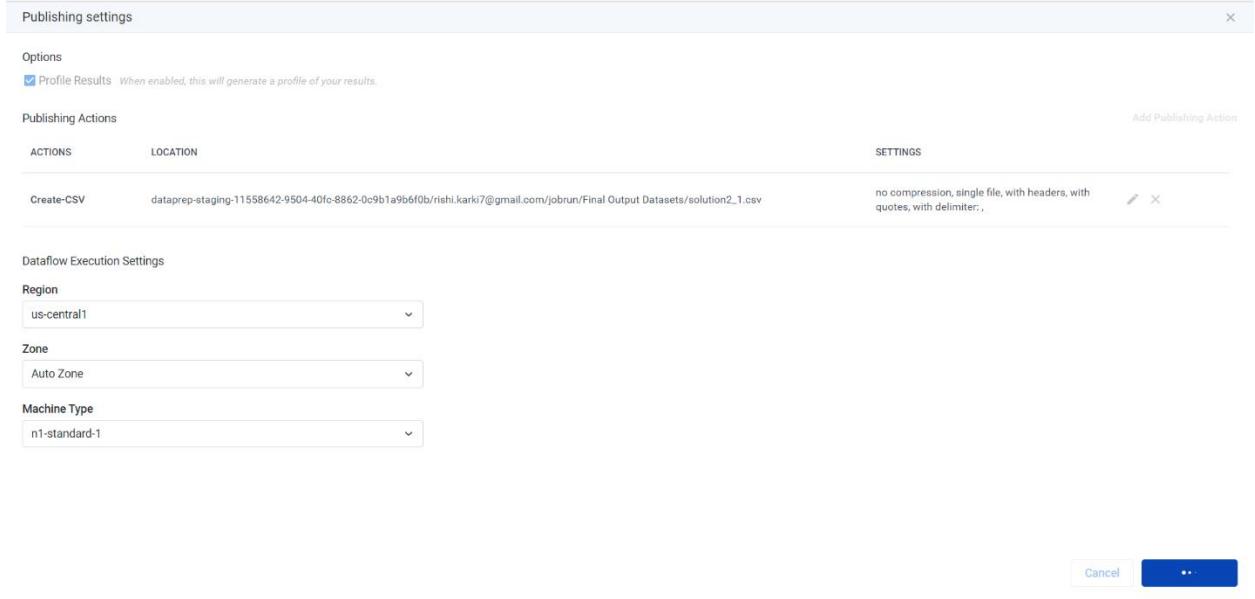
- 32.** In this page, configure the necessary setting and make sure to check ‘Include headers as first row on creation’ option under ‘More Option’ menu. Then, click on ‘Update’ to save settings.

For our purpose, we create new folder as ‘Final Output Datasets’ to store our final output datasets.



Here is the snapshot to configure the output type and location for output dataset solution2_1.

33. Then, Click on Save settings to confirm the previous step settings configuration.

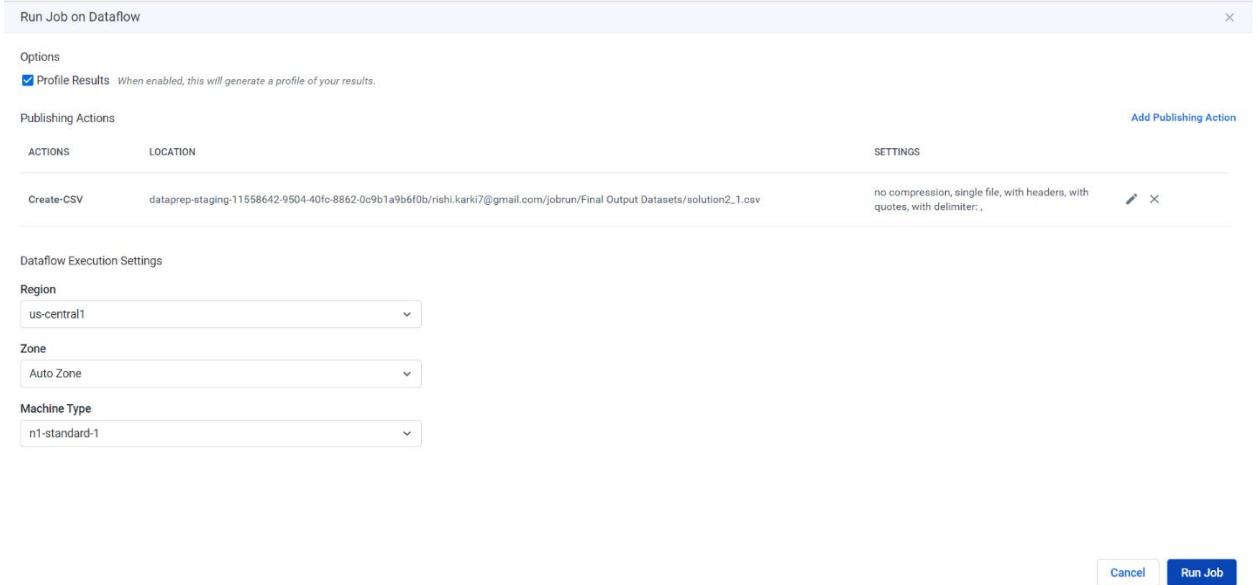


Here is the snapshot to confirm the previous step settings configuration.

34. Then, Click on ‘Run Job’ option to start processing your final output dataset creation.



Here is the initial snapshot to illustrate the actual start of ‘Run Job’ function.

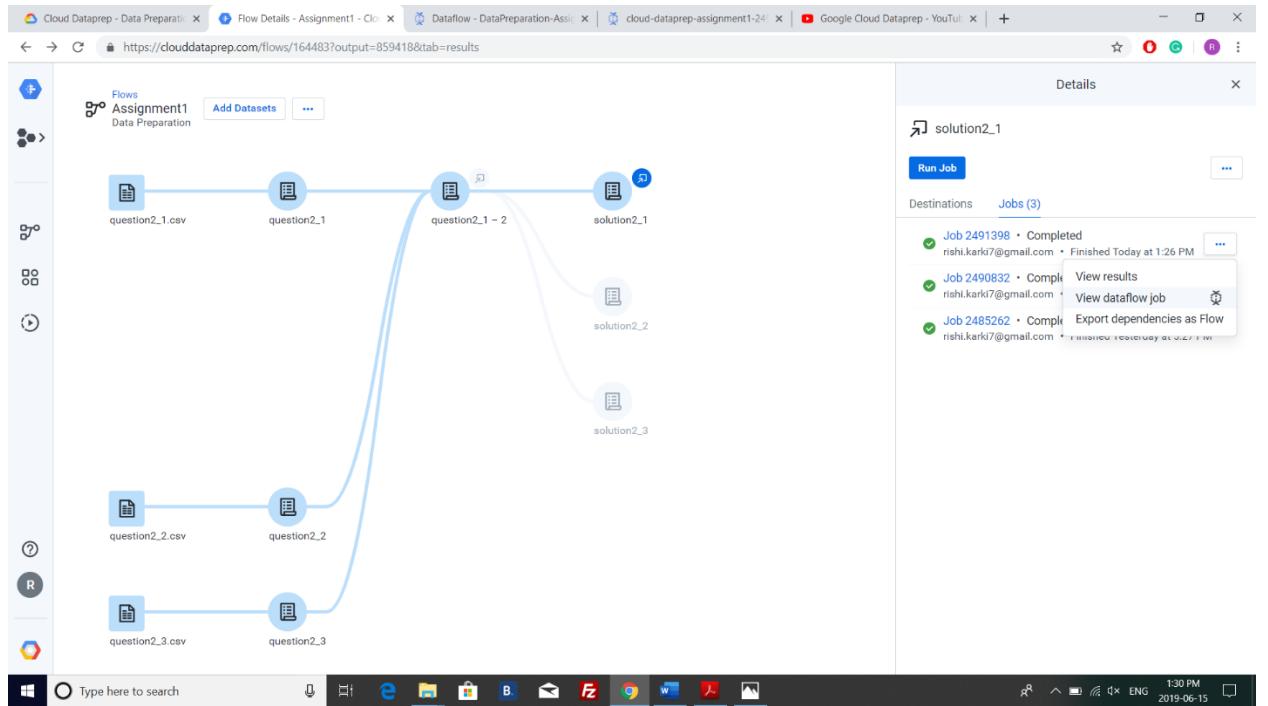


Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

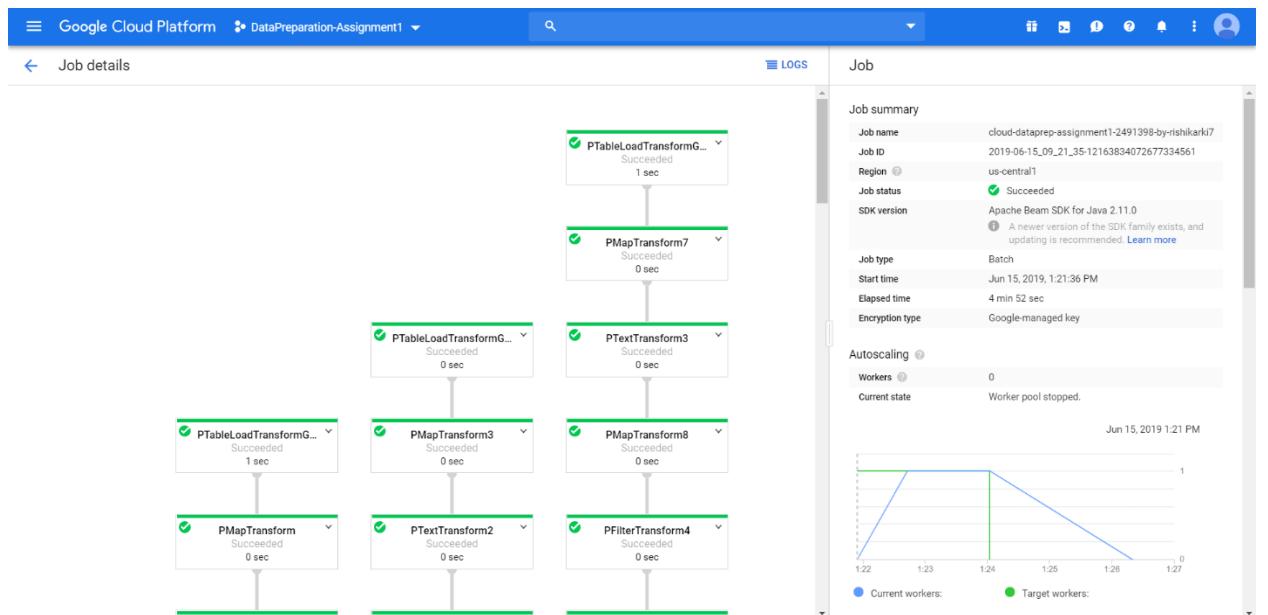


Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

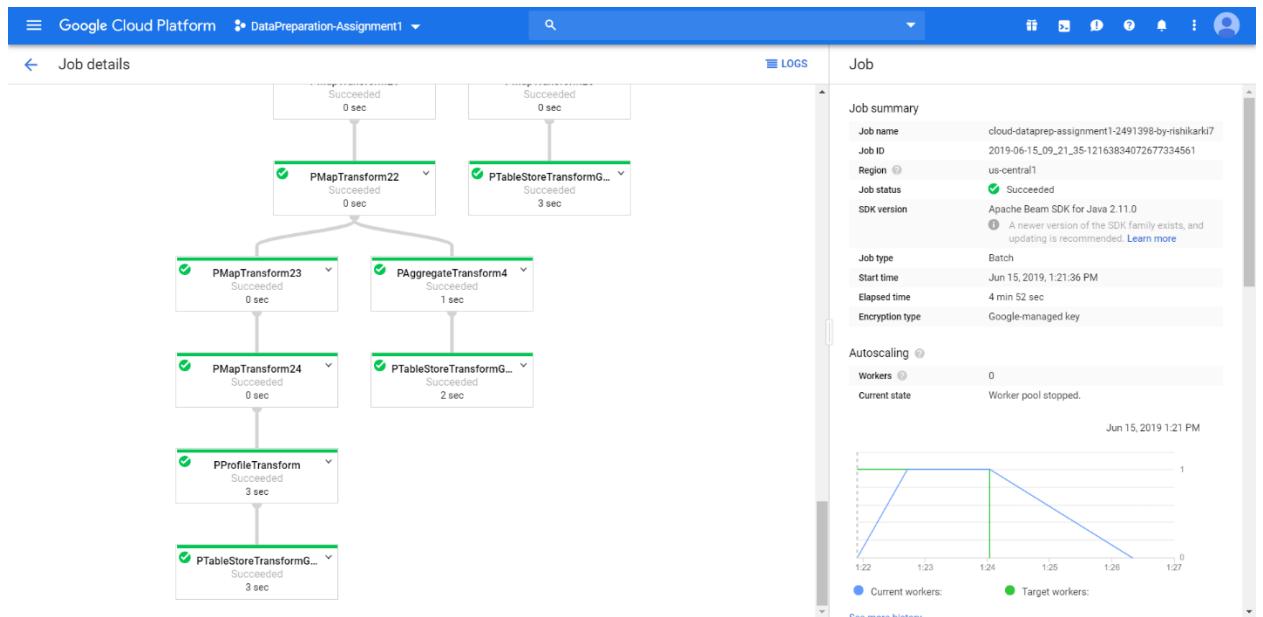
35. Once the 'Run Job' function gets completed, you can view the dataflow job for your output dataset 'solution 2_1' by clicking on 'View dataflow job' from the drop down menu just right to it.



Here is the snapshot that illustrates the step to view the dataflow job after 'Run Job' progress gets completed.

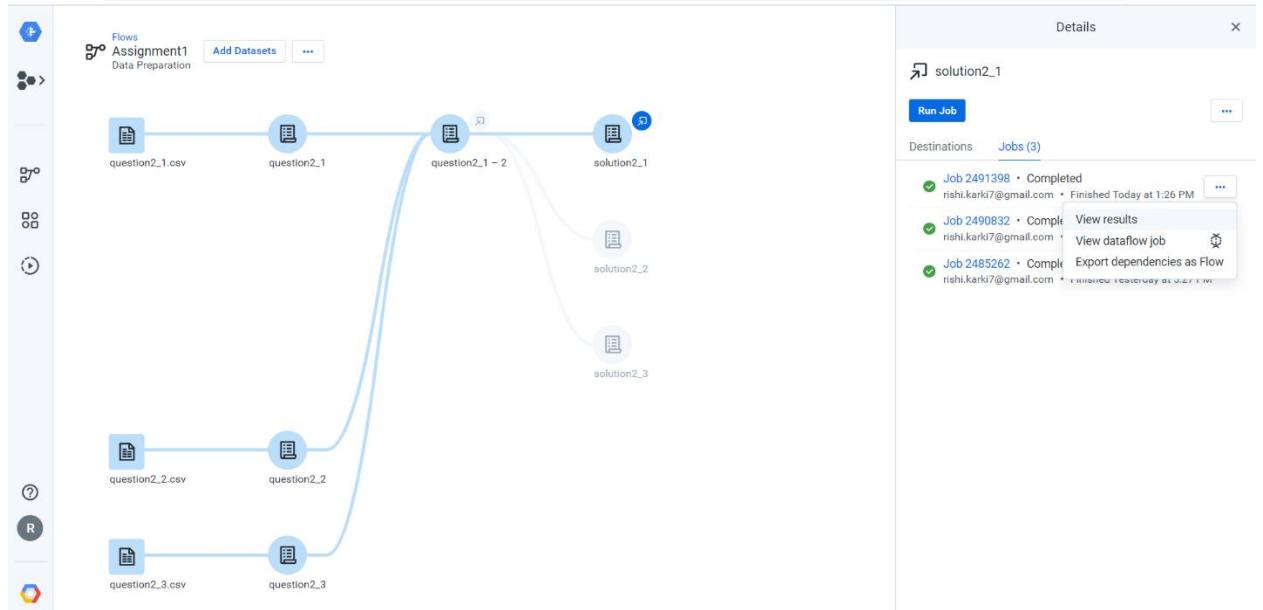


Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_1.



Here is the snapshot that illustrates the final sample of dataflow job for output dataset solution2_1.

- 36.** Now, to view the results, just click on ‘View results’ option from drop down menu that is just at top right.



Here is the snapshot that illustrates the step to view results for output dataset solution2_1.

The screenshot shows the Google Dataflow job overview page for Assignment1 > solution2_1, Job 2491398, finished today at 1:26 PM. The 'Output Destinations' tab is selected. A card for the 'solution2_1.csv' output shows it is completed with 1 sec duration. The 'View results' link is highlighted. To the right, the 'Job summary' and 'Execution summary' sections provide detailed information about the job's status and execution.

Job summary	Value
Job ID	2491398
Job status	Completed
Flow	Assignment1
Output	solution2_1
Dataflow template	Browse Copy to clipboard

Execution summary	Value
Job type	Manual
User	rishi.karki7@gmail.com
Environment	Google Dataflow
Start time	June 15th 2019, 1:21 pm
Finish time	June 15th 2019, 1:26 pm
Last update	June 15th 2019, 1:26 pm
Duration	5 minutes

Here is the snapshot that illustrates the screen after clicking on ‘View results’ from the previous step.

37. Then, you can see your output dataset ‘solution2_1.csv’ that is ready to be downloaded under the ‘Output Destinations’ tab.

The screenshot shows the Google Dataflow job overview page for Assignment1 > solution2_1, Job 2491398, finished today at 1:26 PM. The 'Output Destinations' tab is selected. A table lists the output datasets, including 'solution2_1.csv' which is completed and located at gs://dataprep-staging-11558642-9504-40fc-8862-0c9b1a9b6f0b/rishi.karki7@gmail.com/jobrun/Final Output Datasets/solution2_1.csv. A context menu for this dataset includes options to 'Download result', 'View on Google Cloud Storage', and 'Create imported dataset'.

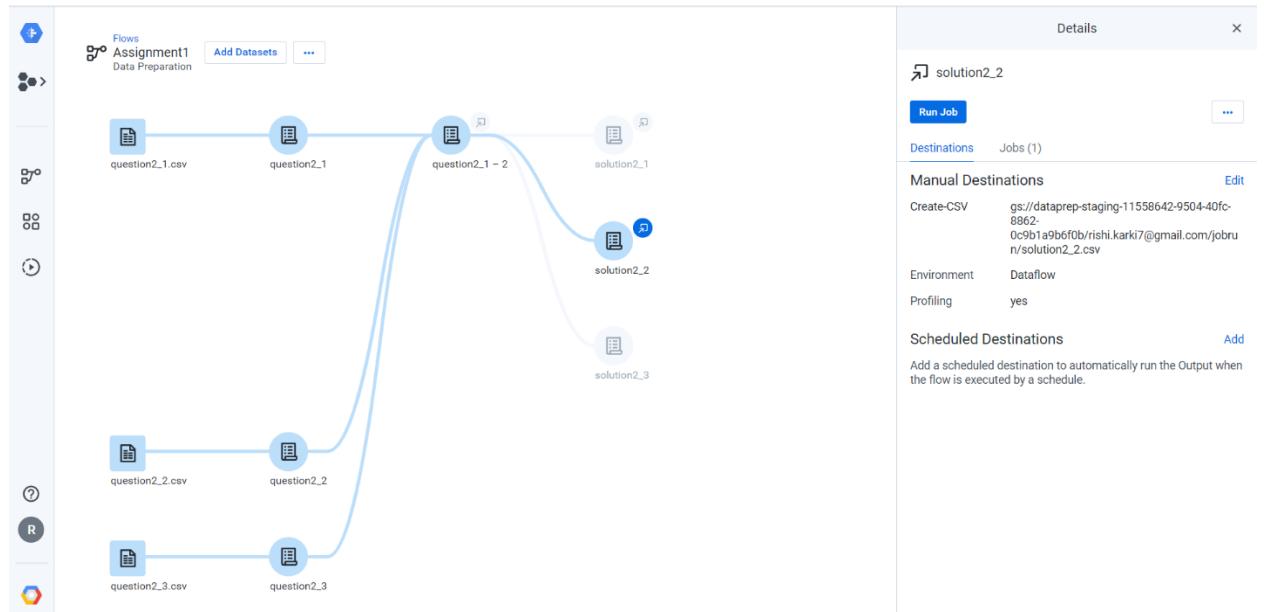
Name	Location	Status	Duration
solution2_1.csv	gs://dataprep-staging-11558642-9504-40fc-8862-0c9b1a9b6f0b/rishi.karki7@gmail.com/jobrun/Final Output Datasets/solution2_1.csv	Completed	

Here is the snapshot that illustrates the final output dataset ‘solution2_1.csv’ that is ready to be downloaded.

Now, we just download the final output dataset ‘solution2_1.csv’ in our local PC directory by clicking on ‘Download result’ option.

The process of generating final output dataset ‘solution2_2.csv’ is shortly described below:

38. Under the ‘Flows’ menu, ‘Run Job’ function is activated , then select the dataset ‘solution2_2’ and click on ‘tray’ icon which is just above the dataset icon.



Here is the snapshot of the initial step to ‘edit’ menu of ‘Run Job’ function in generating output dataset ‘solution2_2.csv’

39. Under ‘Destination’ tab, Click on ‘Edit’ menu just left to ‘Manual Destinations’. Select ‘Edit’ option in setting tab.

Publishing settings

Options

Profile Results When enabled, this will generate a profile of your results.

Publishing Actions

ACTIONS	LOCATION	SETTINGS
Create-CSV	dataprep-staging-11558642-9504-40fc-8862-0c9b1a9b6f0b@iehi.karki7@gmail.com/jobrun/solution2_2.csv	no compression, multiple files, with quotes, with delimiter: ,

Dataflow Execution Settings

Region: us-central1

Zone: Auto Zone

Machine Type: n1-standard-1

Cancel Save settings

Here is the snapshot after clicking on edit menu from previous step which prompts to go to another ‘edit’ option.

40. In this page, configure the necessary setting and make sure to check ‘Include headers as first row on creation’ option under ‘More Option’ menu. Then, click on ‘Update’ to save settings.

For our purpose, we create new folder as ‘Final Output Datasets’ to store our final output datasets.

Publishing Action

Choose a file or folder

... / jobrun / Final Output Datasets

Create Folder

Search...

NAME	SIZE	LAST UPDATED
solution2_1.csv	2.99kB	Today at 1:26 PM

Final Output Datasets/

Data Storage Format: CSV

Create new file every run
Create a new file with an incremental number appended to the name e.g. solution2_2_2.csv

Append to this file every run
Append to the end of the file solution2_2.csv

Replace this file every run
Replace the existing file solution2_2.csv with a new file.

Less Options

Include headers as first row on creation

Include quotes

Delimiter: ,

Single File
Result is a single file in the output location.

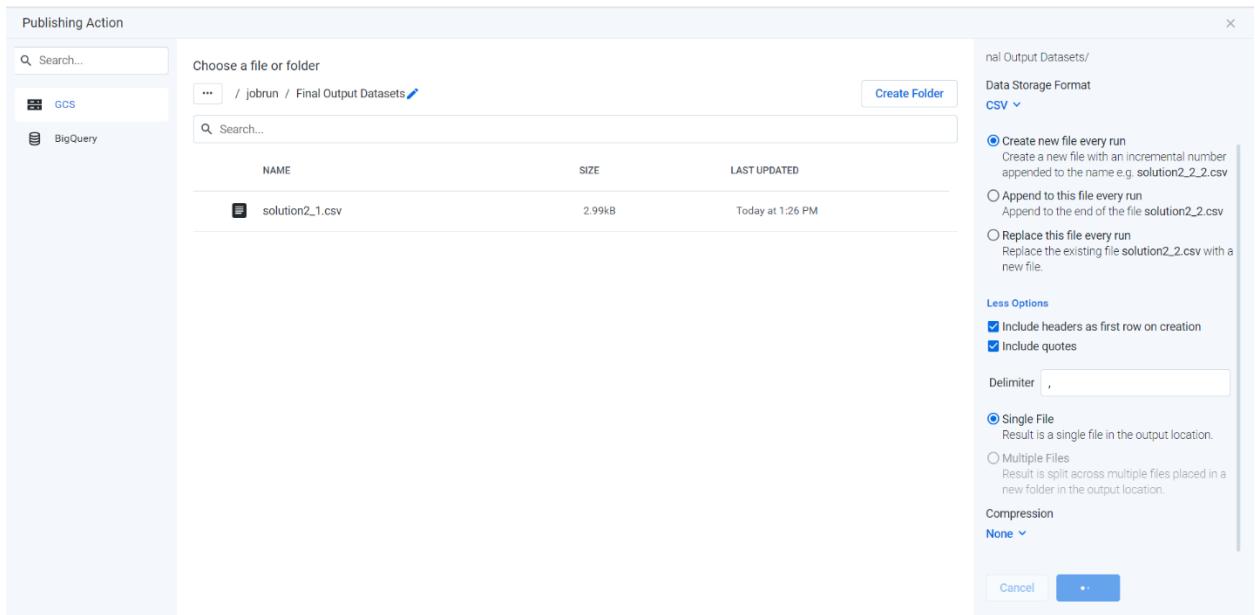
Multiple Files
Result is split across multiple files placed in a new folder in the output location.

Compression: None

Cancel Update

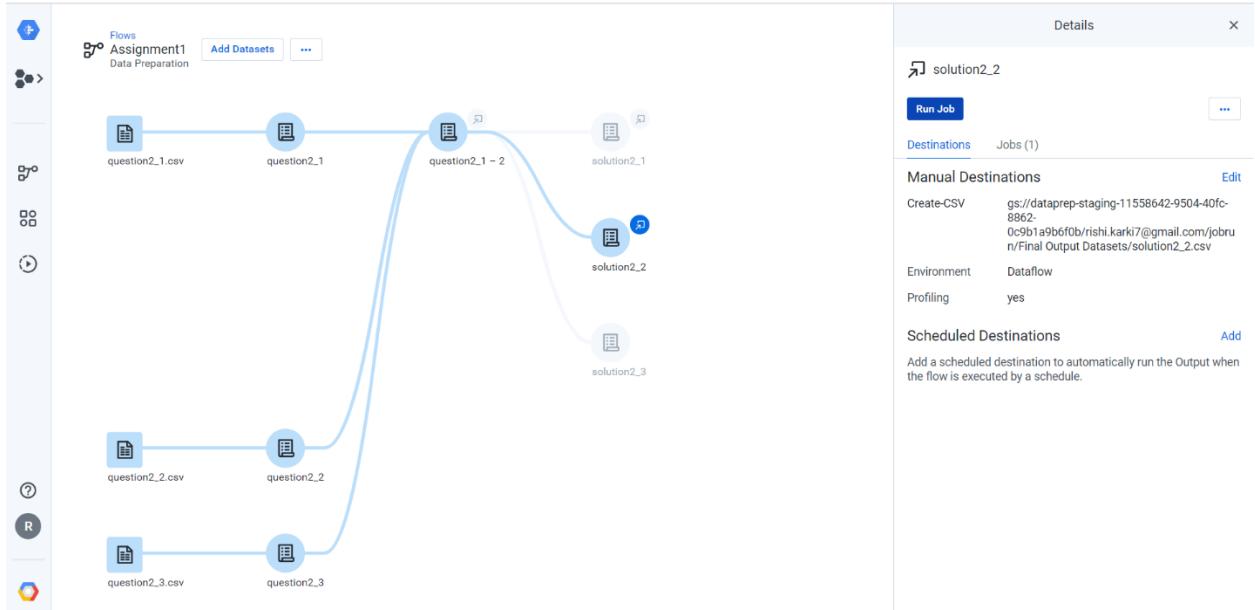
Here is the snapshot to configure the output type and location for output dataset solution2_2.

- Then, Click on Save settings to confirm the previous step settings configuration.

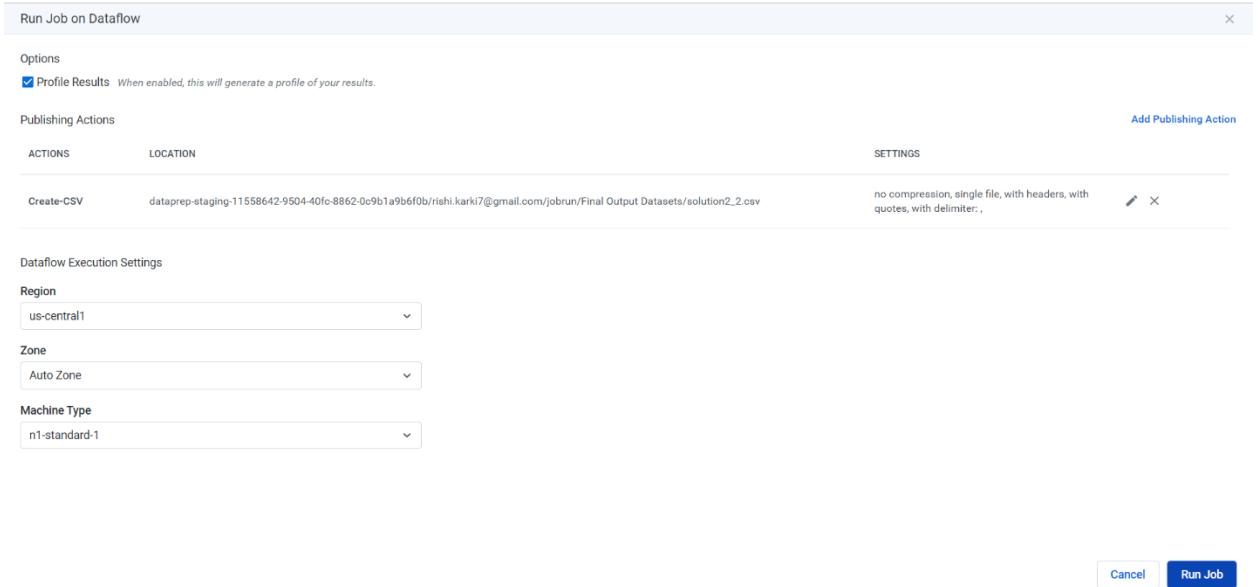


Here is the snapshot to confirm the previous step settings configuration.

- Then, Click on 'Run Job' option to start processing your final output dataset creation.



Here is the initial snapshot to illustrate the actual start of 'Run Job' function.



Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

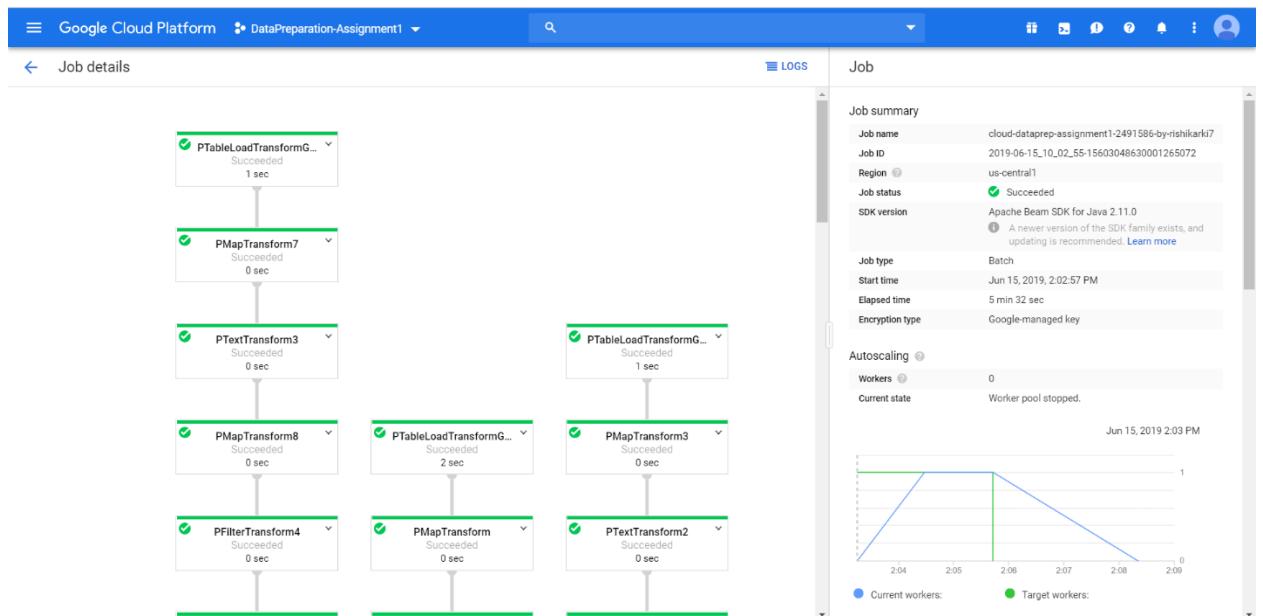


Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

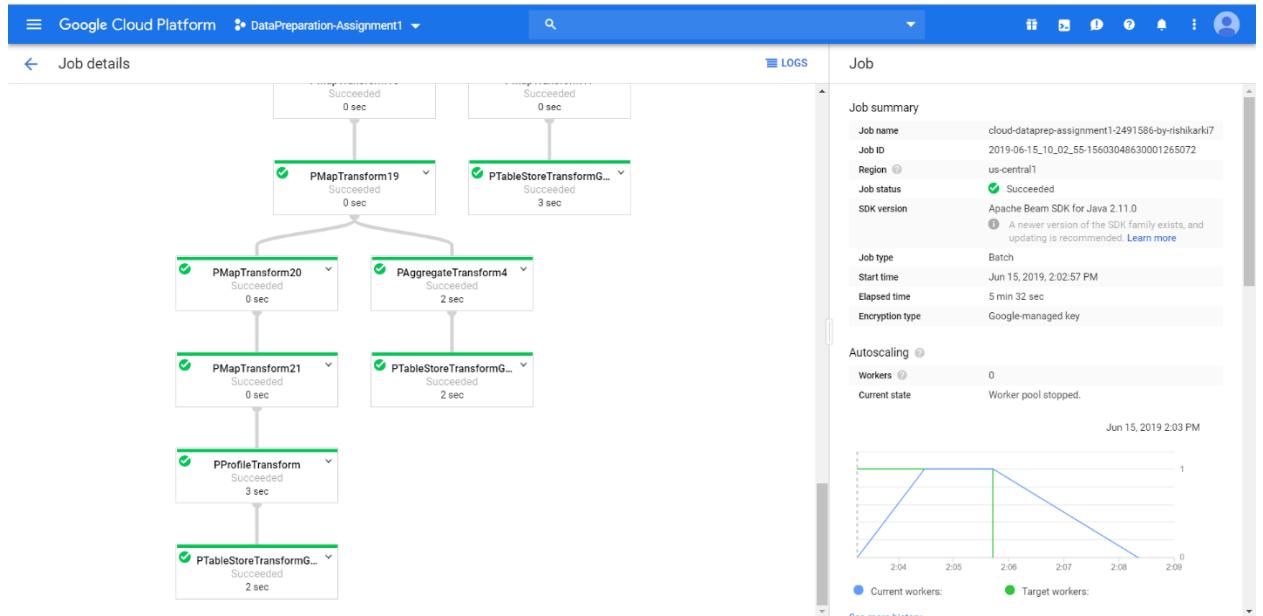
43. Once the 'Run Job' function gets completed, you can view the dataflow job for your output dataset 'solution 2_2' by clicking on 'View dataflow job' from the drop down menu just right to it.



Here is the snapshot that illustrates the step to view the dataflow job after the 'Run Job' progress gets completed.



Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_2.



Here is the snapshot that illustrates the final sample of dataflow job for output dataset solution2_2.

44. Now, to view the results, just click on ‘View results’ option from drop down menu that is just at top right.



Here is the snapshot that illustrates the step to view results for output dataset solution2_2.

Assignment1 > solution2_2
Job 2491586
Finished Today at 2:08 PM

[View dataflow job](#) [...](#)

[Overview](#) [Output Destinations](#) [Profile](#) [Dependencies](#) [Data sources](#)

Transform with profile
Completed Today at 2:08 PM, started Today at 2:02 PM • Ran for 5 min
Environment Google Dataflow
Loading job profile...
[Go to steps and dependencies](#) [Go to profile](#) [View dataflow job](#)

Publish
Completed Today at 2:08 PM, started Today at 2:08 PM • Ran for 2 sec
Activity
 [solution2_2.csv](#) Completed • 2 sec
[View results](#)

Job summary
Job ID 2491586
Job status Completed
Flow Assignment1
Output solution2_2
Dataflow template [Browse](#) [Copy to clipboard](#)

Execution summary
Job type Manual
User rishi.karki7@gmail.com
Environment Google Dataflow
Start time June 15th 2019, 2:02 pm
Finish time June 15th 2019, 2:08 pm
Last update June 15th 2019, 2:08 pm
Duration 6 minutes

Here is the snapshot that illustrates the screen after clicking on 'View results' from the previous step.

45. Then, you can see your output dataset 'solution2_2.csv' that is ready to be downloaded under the 'Output Destinations' tab.

Assignment1 > solution2_2
Job 2491586
Finished Today at 2:08 PM

[View dataflow job](#) [...](#)

[Overview](#) [Output Destinations](#) [Profile](#) [Dependencies](#) [Data sources](#)

Name	Location	Status	Duration
solution2_2.csv	gs://dataprep-staging-11558642-9504-40fc-8862-0c9b1a9b6f0b/rishi.karki7@gmail.com/jobrun/Final Output Datasets/solution2_2.csv	Completed	View details ...

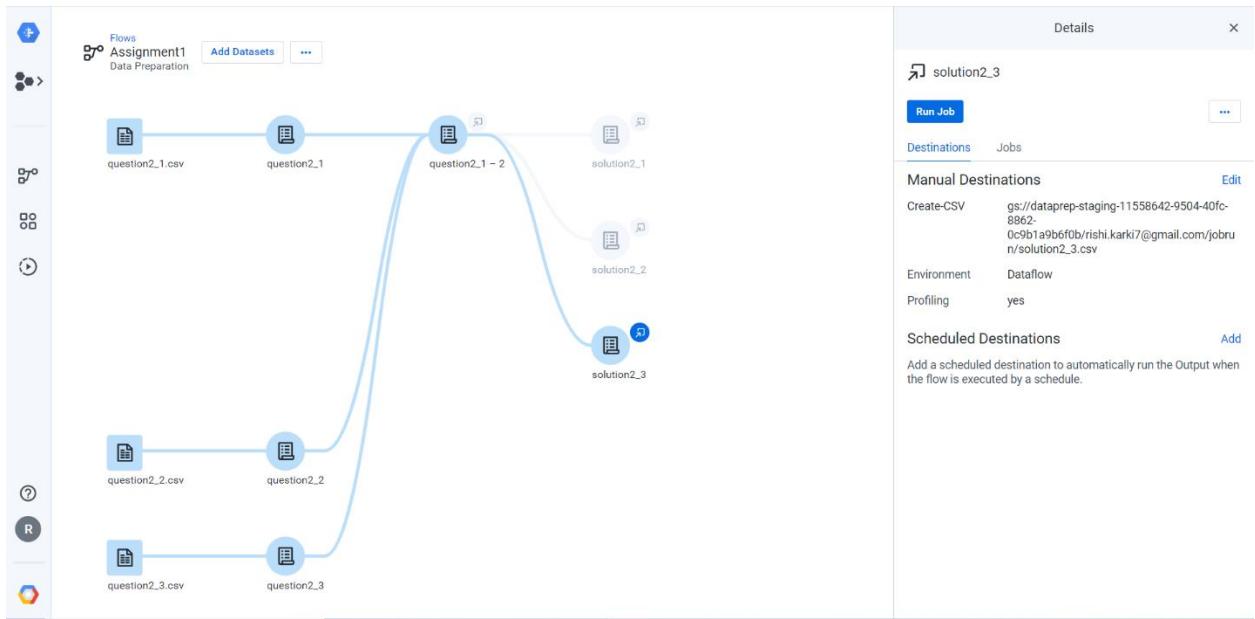
[Download result](#) [View on Google Cloud Storage](#) [Create imported dataset](#)

Here is the snapshot that illustrates the final output dataset 'solution2_2.csv' that is ready to be downloaded.

Now, we just download the final output dataset ‘solution2_2.csv’ in our local PC directory by clicking on ‘Download result’ option.

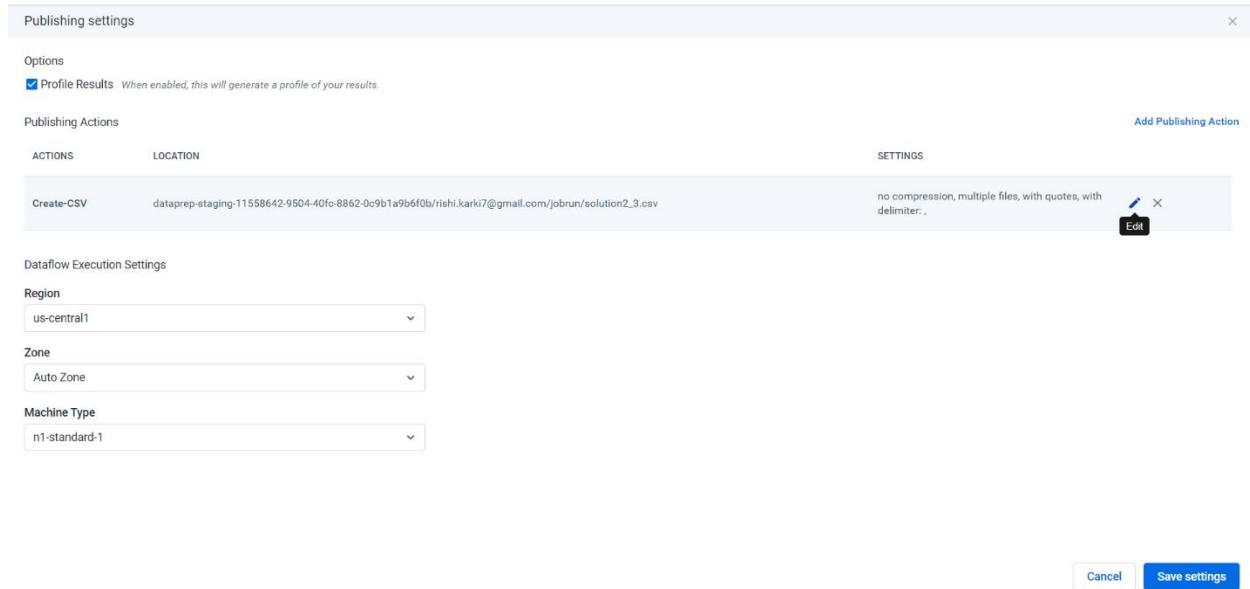
The process of generating final output dataset ‘solution2_3.csv’ is shortly described below:

- 46.** Under the ‘Flows’ menu, ‘Run Job’ function is activated , then select the dataset ‘solution2_3’ and click on ‘tray’ icon which is just above the dataset icon.



Here is the snapshot of the initial step to ‘edit’ menu of ‘Run Job’ function in generating output dataset ‘solution2_3.csv’

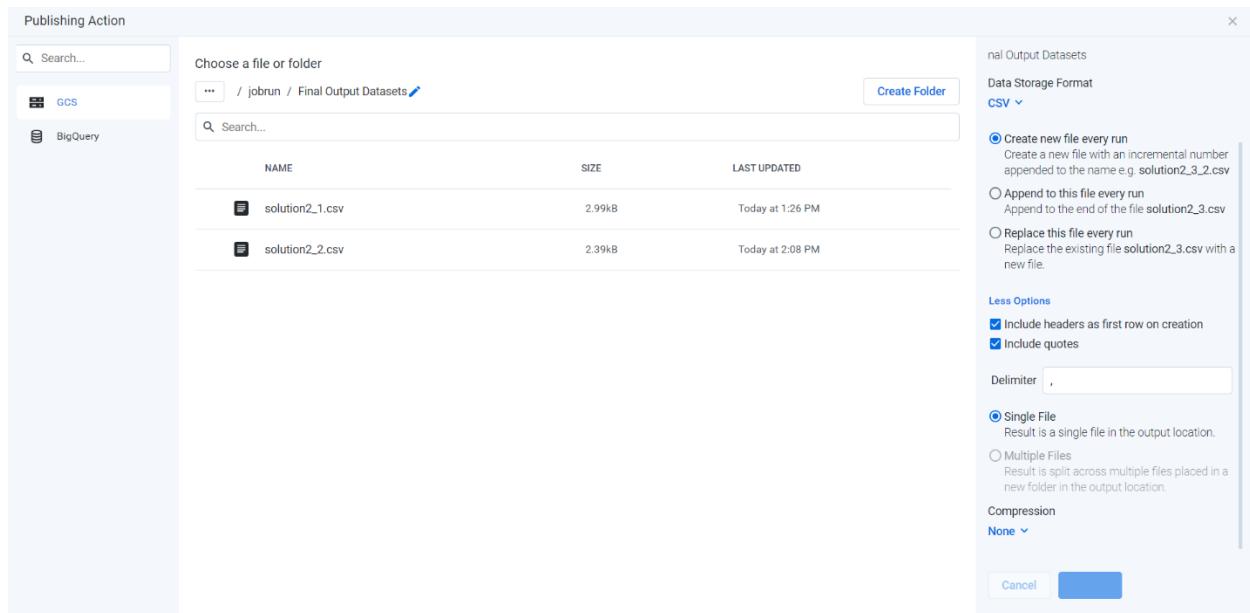
- 47.** Under ‘Destination’ tab, Click on ‘Edit’ menu just left to ‘Manual Destinations’. Select ‘Edit’ option in setting tab.



Here is the snapshot after clicking on edit menu from previous step which prompts to go to another ‘edit’ option.

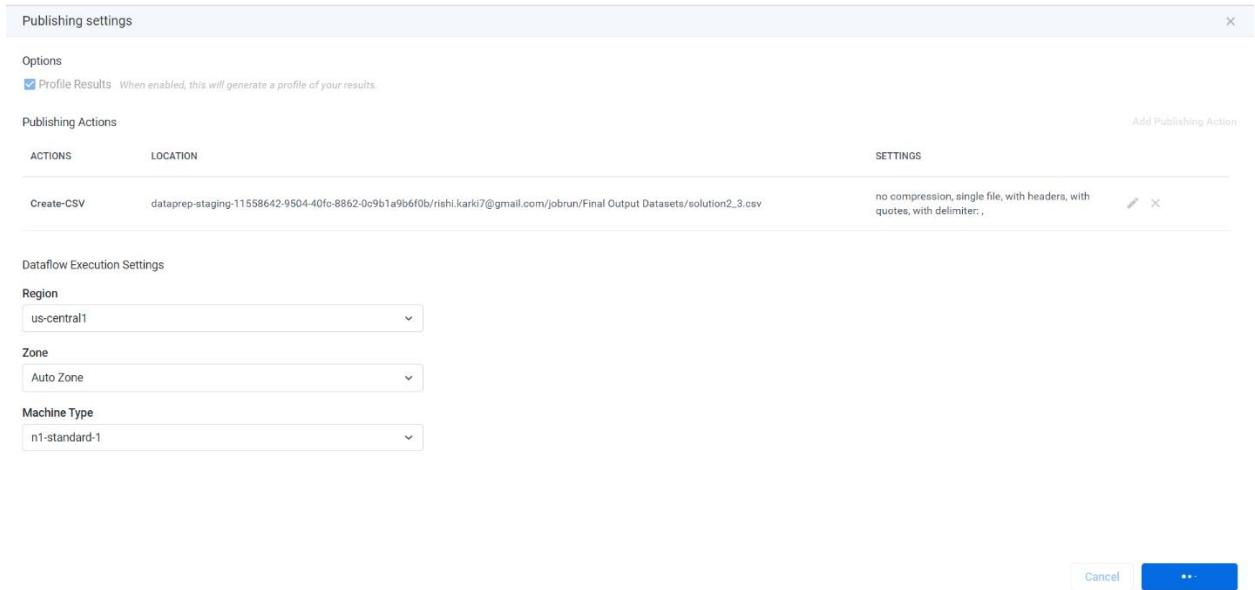
48. In this page, configure the necessary setting and make sure to check ‘Include headers as first row on creation’ option under ‘More Options’ menu. Then, click on ‘Update’ to save settings.

For our purpose, we create new folder as ‘Final Output Datasets’ to store our final output datasets.



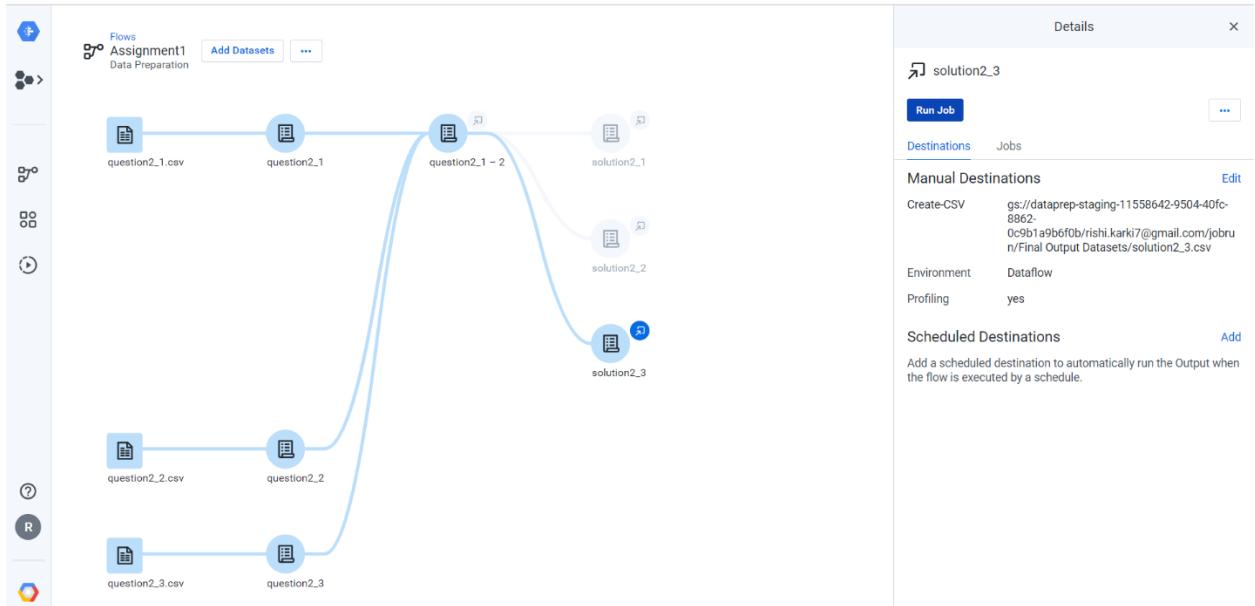
Here is the snapshot to configure the output type and location for output dataset solution2_3.

49. Then, Click on Save settings to confirm the previous step settings configuration.

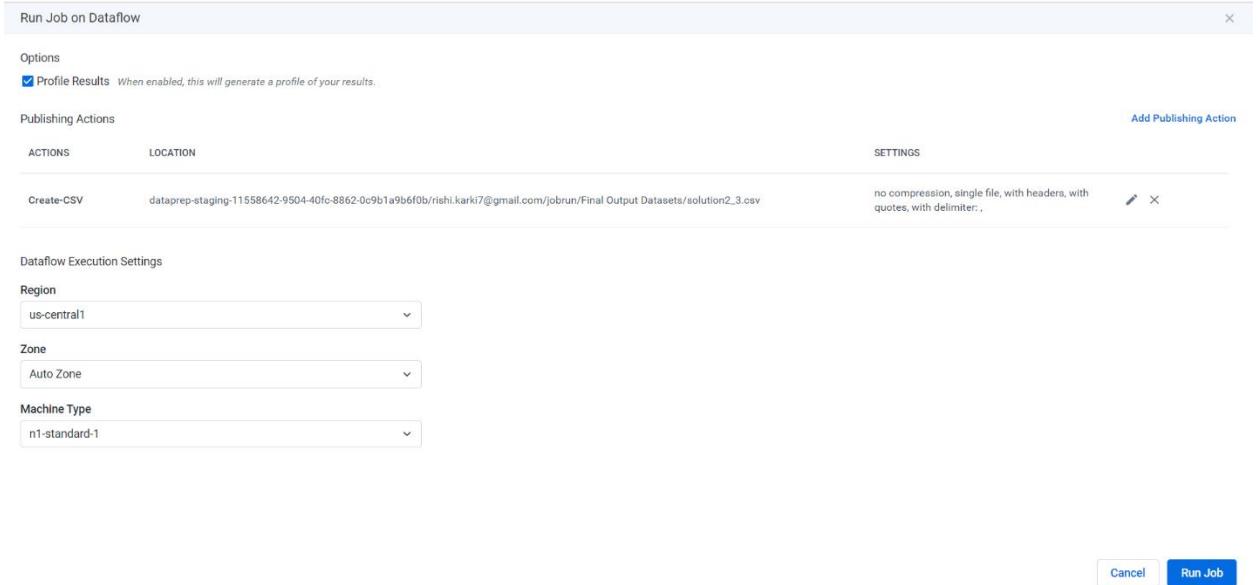


Here is the snapshot to confirm the previous step settings configuration.

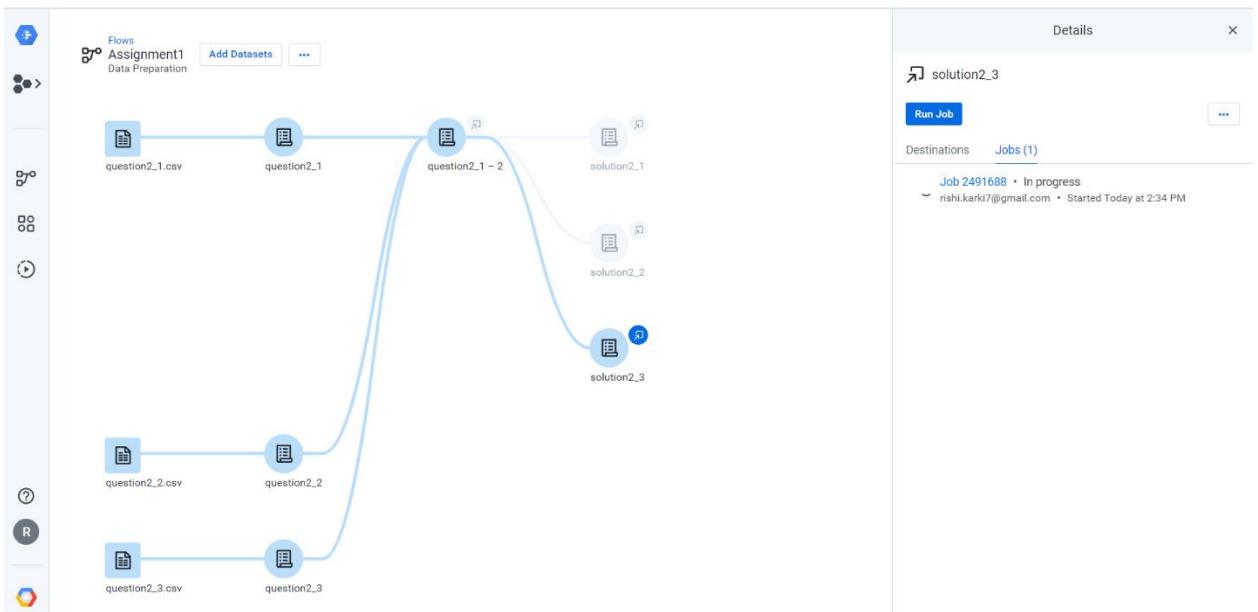
50. Then, Click on ‘Run Job’ option to start processing your final output dataset creation.



Here is the initial snapshot to illustrate the actual start of ‘Run Job’ function.



Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

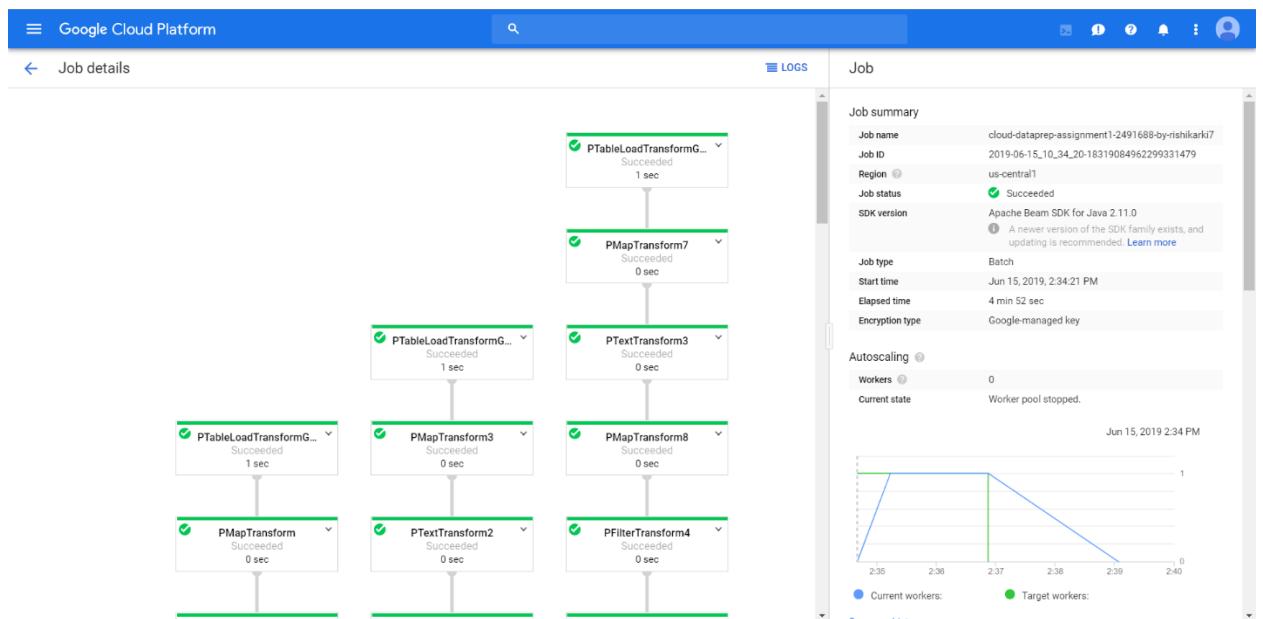


Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

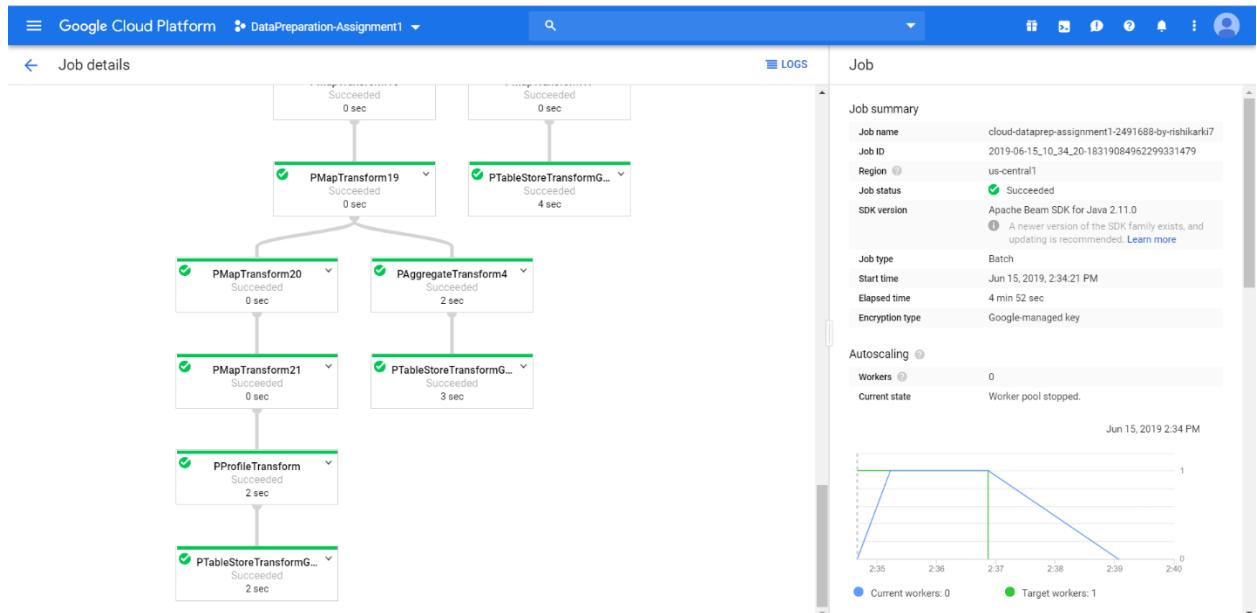
51. Once the 'Run Job' function gets completed, you can view the dataflow job for your output dataset 'solution 2_3' by clicking on 'View dataflow job' from the drop down menu just right to it.



Here is the snapshot that illustrates the step to view the dataflow job after progress gets completed.



Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_3.



Here is the snapshot that illustrates the final sample of dataflow job for output dataset solution2_3.

- 52.** Now, to view the results, just click on ‘View results’ option from drop down menu that is just at top right.



Here is the snapshot that illustrates the step to view results for output dataset solution2_3.

The screenshot shows the Google Dataflow job overview page for Assignment1 > solution2_3, Job 2491688, finished today at 2:39 PM. The 'Overview' tab is selected. A summary table on the right provides details like Job ID, Status, Flow, Output, and Dataflow template. The main area displays two steps: 'Transform with profile' and 'Publish'. The 'Publish' step has a green checkmark and a duration of 2 seconds. Below it, a 'View results' link is visible.

Here is the snapshot that illustrates the screen after clicking on 'View results' from the previous step.

53. Then, you can see your output dataset ‘solution2_3.csv’ that is ready to be downloaded under the ‘Output Destinations’ tab.

The screenshot shows the Google Dataflow job overview page for Assignment1 > solution2_3, Job 2491688, finished today at 2:39 PM. The 'Output Destinations' tab is selected. It lists a single output named 'solution2_3.csv' located at 'gs://dataprep-staging-11558642-9504-40fc-8862-0c9b1a9b6f0b/rishi.karki7@gmail.com/jobrun/Final Output Datasets/solution2_3.csv'. The status is 'Completed' and the duration is 2 seconds. A 'View details' button and a 'Download result' button are present.

Here is the snapshot that illustrates the final output dataset ‘solution2_3.csv’ that is ready to be downloaded.

Now, we just download the final output dataset ‘solution2_3.csv’ in our local PC directory by clicking on ‘Download result’ option.

3.3 Data Preparation (ETL Pipeline) using Microsoft SSIS

The general ETL pipeline refers to the process of data Extraction from one or more sources and Transforming that data in order to load it into a destination with different visualization and format in comparison to the source data, loading all the data from the data warehouse [1]. The ETL process is an important part of the Business Intelligence studies which includes the learning of the basic SQL and it’s working as it’s result is an automated Loading process into a SQL database [2]. It basically starts with designing a schema for the table.

Initially, we chose a dataset from a database which provided us with adequate information to take it as a source of ETL process. We look through various datasets and after a thorough research we agreed on a vehicle accident dataset of New York city was finalized to be worked on. We took the dataset from a governmental open data source <https://catalog.data.gov/dataset/motor-vehicle-crashes-vehicle-information-beginning-2009> [3] where there was data regarding the accident cases of the vehicles for the three consecutive years but since the file was around 200MB and it took quite a time to operate as a CSV we divided the dataset on three parts according to year 2014, 2015 and 2016 and **took the dataset of a total of 20,610 records of 2016 for building our ETL pipeline.**

For the design part we mostly use the star schema which is a schema design consisting of fact and dimension tables. The fact table is the centralized table which is connected to every small dimension table taking their primary key as foreign key and acts as a parent to all the junior dimension tables [4].

3.3.1. Pre-processing task: Schema Design

The first task was to group various instances together to create a fact and multiple dimension table and connect them as a star schema. Here, in accordance to our dataset we

segregated the design into 1 fact table **adetail** and four-dimension tables as **type**, **info**, **cinfo** and **einfo** which consecutively described the whole dataset and its contents.

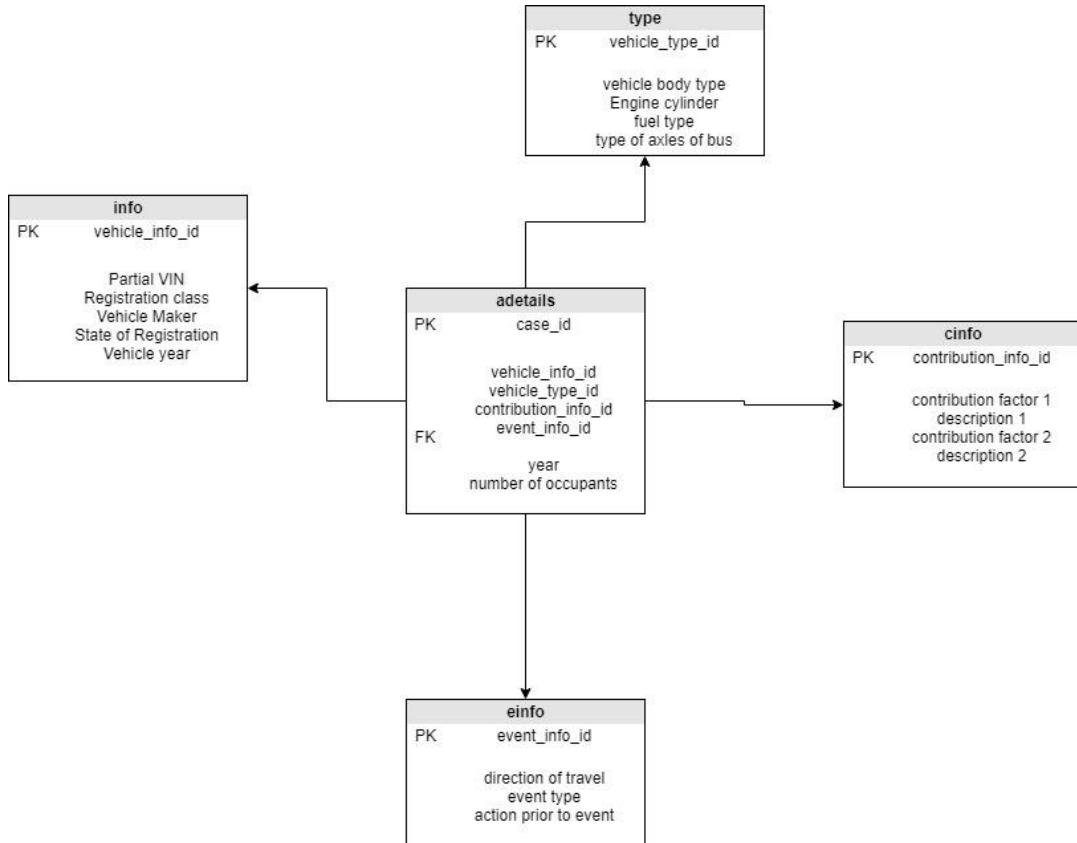


Fig: The Star schema design for data preprocessing

Here, **Fact table:** adetails

Dimension tables: info, cinfo, type and einfo

3.3.2. ETL pipeline: Generating target dataset

We use Visual studio 2017 in order to generate the ETL pipeline. For this first of all, we created the flow of the entire pipeline as shown in the figure below.

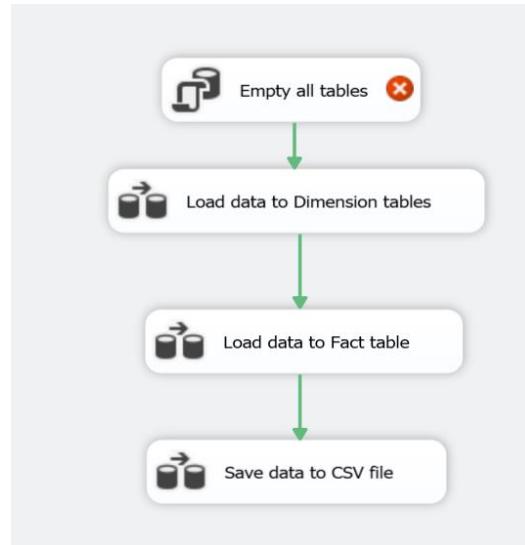


Fig: The ETL pipeline flow

The first and foremost task was to empty all the table. It was necessary to truncate through all the tables so as to empty the relationship between the tables.

We wrote a sql query to truncate all the tables.

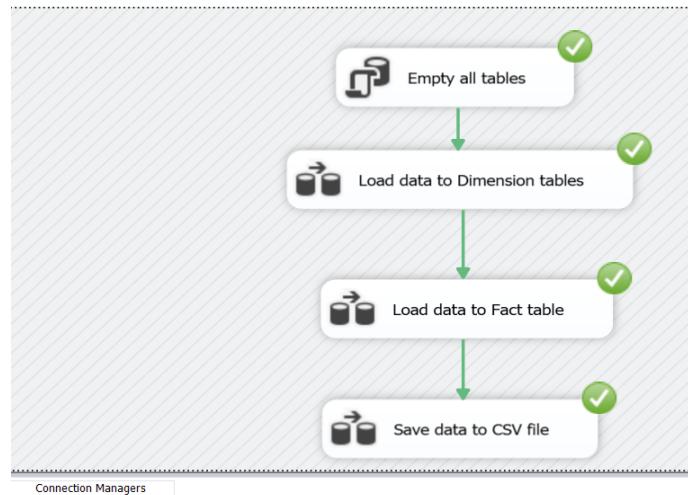


Fig: Successful execution of truncation

Now, for the second step i.e. to read the CSV file into the dimension table. We took the whole CSV file as a dataset and then mapped it according to it's DBO dimension tables. First, we did it for the dbo.info table. Similarly, we repeat this step for all the dimension tables required for the schema and we made sure to revert the rows in case of errors.

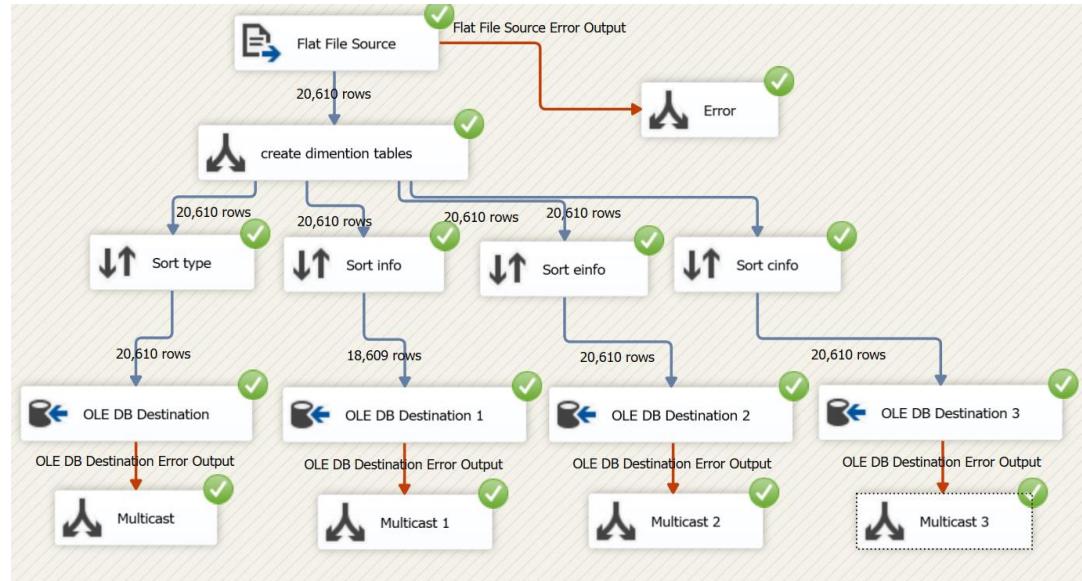


Fig: Saving data into the dimension tables.

Now here we can see that out of 20,611 rows **only 18,609 rows** get saved into the info table because we have removed the duplicate values of the case ID because the Case number cannot be the same for two different accidents. we save the data to the fact table following the procedures as taught in the class. We map the features of the dimension tables into the fact table and redirect the error.

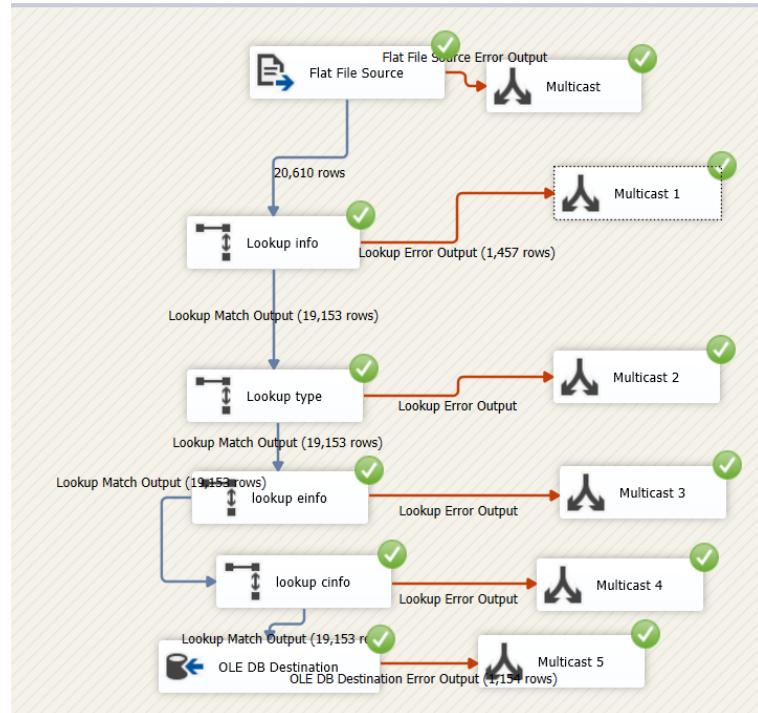


Fig: Saving data to the fact table

During saving the data into the fact table, we came across the fact that while taking the input only 19,153 rows were looked up and passed on to the first table. As the lookup for each dimension table in accordance with the fact table continues, we can see that only out of 19,153, 1,154 rows were thrown as error and the remaining 17,999 rows are input for loading the csv file or loading the tables in the sql database.

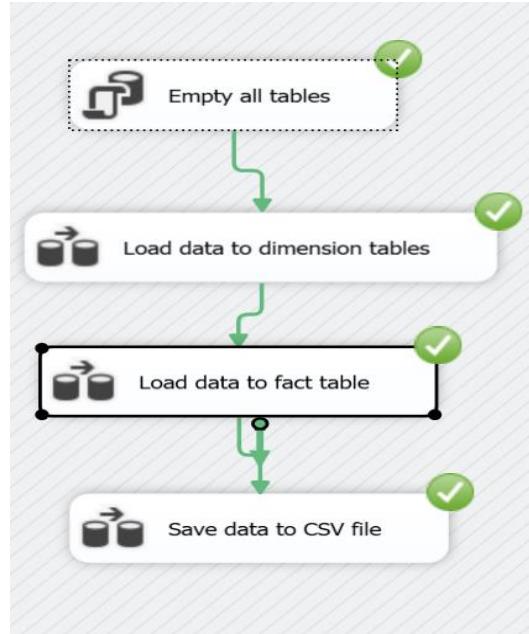


Fig: The main program workflow

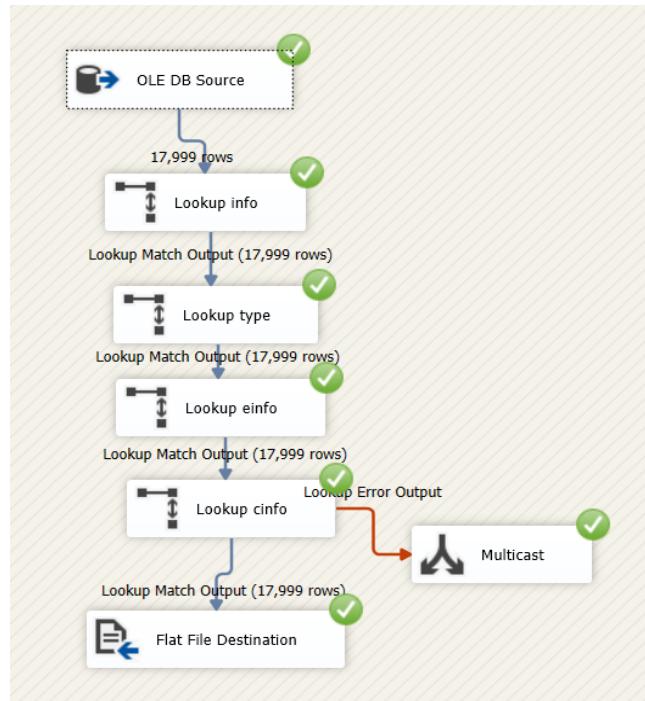


Fig: Steps in writing data into a CSV file

The total of 17,999 rows are written into a CSV file as an output. We can either do this or load the resultant CSV in the sql server. To check the Load process we wrote a sql query in order to check if the data is loaded in the table or not. After the workflow of saving data to the tables is saved successfully we execute the whole program which runs successfully as well.

```
select * from [dbo].[adetail];
```

The result is presented below which shows that the data was loaded successfully.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - DESKTOP-3IQ56LA.accidents (DESKTOP-3IQ56LA\joshi (S4)) - Microsoft SQL Server Management Studio". The main window displays the results of a query:

```
select * from [dbo].[adetail];
```

The results grid shows the following columns: case_id, vehicle_info_id, vehicle_type_id, event_info_id, contributor_info_id, year, and occupants. The data consists of 17,999 rows, each containing unique values for the specified columns. The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-3IQ56LA (14.0 RTM) DESKTOP-3IQ56LA\joshi ... accidents 00:00:00 17999 rows".

Fig: Loading the fact table into the database

3.3.3. General Workflow

- **Extract**

Input the data from the dataset and create a connection with the database.

Extract all possible rows and read them in accordance to their header. **Read the data from the dataset.**

- **Transform**

Sort the values and remove the redundancy. Implement the functions such as merge if you need to generalize the column or implement split if you need to perform specification. In our case, the dataset we took had no such derived attribute so we could not perform the merge or split functionalities. We could just filter the accident case by case ID but we do understand the concept of merge, join and split and can implement it on any other dataset. **Change the functionality of the data in the dataset and recreate the set.**

- **Load**

For this step, we established a connection with the SQL server and pushed the connected tables as we created the fact and dimension tables. Finally, data was loaded as an output to the fact and dimension table and can be seen in the screenshot above or a screen recording provided. **Data loaded and presented in the table.**

4. Comparison

Overall, we have used the three tools for data preparation and gained a basic understanding of them. So, here is the comparison after our experience with these tools which is presented below in tabular form:

	OpenRefine	Google Cloud Dataprep	Microsoft SSIS
Fees	Open source and completely no charge included	Not free (get free credits when a new user register)	Not free (available for university students)
Backer	Founded by Google and it is supported by the community now.	Google	Microsoft
Functions	Process data format(excel-like , but more suitable for large data)	Join and compare data(database-like but more visual)	Define the control flow and data flow , drag components to finish the whole ETL process in automation.
Installation and Configuration	Easy to install, requires less effort and memory.	Signup process is very tedious although no any installation required.	Very difficult. The whole installation is verbose and also requires more time, effort and memory.
Ease of use	The feature is straight forward, you can get the direct information from tables.	Easy to use, the control flow can make am amateur join many tables easily.	Difficult. A newbie must at least go through many tutorials and documentation to learn the working skills.
GUI	Functionality provided in simple and clear.	Functionality provided in more concise and user friendly.	Functionality provided in convenient, easy and flexible way.

5. Conclusion

Hence, while working on this assignment we learned the installation and usage of three different tools which are essential for data preparation that is the first step in analyzing the data. We got hands-on experience on using them to clean the data, and organize them in a proper way to implement the ETL(Extract->Transform->Load). Although our dataset was somewhat different from others, every step was neatly performed and despite taking so much time to learn in the end it gave an efficient result.

In practice, we should avoid using only one tool to solve all the difficulties. But on the contrary, we should explore and analyze the various tools available in order to solve various problems depending on the context and scenario.

6. References

- [1] Alooma.com. (2019). *What is the difference between a data pipeline and an ETL pipeline? / Alooma.* [online] Available at: <https://www.alooma.com/answers/what-is-the-difference-between-a-data-pipeline-and-an-etl-pipeline> [Accessed 12 Jun. 2019].
- [2] Panoply. (2019). *3 Ways to Build An ETL Process.* [online] Available at: <https://panoply.io/data-warehouse-guide/3-ways-to-build-an-etl-process/> [Accessed 13 Jun. 2019].
- [3] Data, N. (2019). *Motor Vehicle Crashes - Vehicle Information: Three Year Window - Data.gov.* [online] Catalog.data.gov. Available at: <https://catalog.data.gov/dataset/motor-vehicle-crashes-vehicle-information-beginning-2009> [Accessed 14 Jun. 2019].
- [4] Datawarehouse4u.info. (2019). *Data Warehouse Schema Architecture - star schema / Datawarehouse4u.info.* [online] Available at: <https://www.datawarehouse4u.info/Data-warehouse-schema-architecture-star-schema.html> [Accessed 12 Jun. 2019].

7. Appendix [SQL Script]

SQL for Fact table: ‘adetail’

USE [accidents]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[adetail](

[case_id] [int] NOT NULL,
[vehicle_info_id] [int] NOT NULL,
[vehicle_type_id] [int] NOT NULL,
[event_info_id] [int] NOT NULL,
[contribution_info_id] [int] NOT NULL,
[year] [int] NULL,
[occupants] [int] NULL,

CONSTRAINT [PK_adetail] PRIMARY KEY CLUSTERED

(

[case_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT
[FK_adetail_info] FOREIGN KEY([vehicle_info_id])

```
REFERENCES [dbo].[info] ([vehicle_info_id])
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_info]
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT
```

```
[FK_adetail_type] FOREIGN KEY([vehicle_type_id])
```

```
REFERENCES [dbo].[type] ([vehicle_type_id])
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_type]
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT
```

```
[FK_adetail_einfo] FOREIGN KEY([event_info_id])
```

```
REFERENCES [dbo].[einfo] ([event_info_id])
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_einfo]
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT
```

```
[FK_adetail_cinfo] FOREIGN KEY([contribution_info_id])
```

```
REFERENCES [dbo].[cinfo] ([contribution_info_id])
```

```
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_cinfo]
```

```
GO
```

SQL for Dimension table: ‘cinfo’

USE [accidents]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[cinfo](

 [contribution_info_id] [int] IDENTITY(1,1) NOT NULL,

 [contribution_one] [varchar](255) NULL,

 [describe_one] [varchar](255) NULL,

 [contribution_two] [varchar](255) NULL,

 [describe_two] [varchar](255) NULL,

 CONSTRAINT [PK_cinfo] PRIMARY KEY CLUSTERED

(

 [contribution_info_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,

IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS

= ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SQL for Dimension table: ‘einfo’

USE [accidents]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[einfo](

 [event_info_id] [int] IDENTITY(1,1) NOT NULL,

 [event_type] [varchar](255) NULL,

 [action_prior] [varchar](255) NULL,

 [direction_travel] [varchar](255) NULL,

 CONSTRAINT [PK_einfo] PRIMARY KEY CLUSTERED

(

 [event_info_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,

IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SQL for Dimension table: ‘info’

USE [accidents]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[info](

 [vehicle_info_id] [int] IDENTITY(1,1) NOT NULL,

 [partial_VIN] [varchar] (255) NULL,

 [registration_class] [varchar](255) NULL,

 [vehicle_maker] [varchar](255) NULL,

 [state_of_registration] [varchar](255) NULL,

 [vehicle_year] [varchar](255) NULL,

 CONSTRAINT [PK_info] PRIMARY KEY CLUSTERED

(

 [vehicle_info_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,

IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS
= ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SQL for Dimension table: ‘type’

USE [accidents]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

CREATE TABLE [dbo].[type](

 [vehicle_type_id] [int] IDENTITY(1,1) NOT NULL,

 [body_type] [varchar](255) NULL,

 [engine_cylinder] [int] NULL,

 [fuel_type] [varchar](255) NULL,

 [axles_of bus] [varchar](255) NULL,

 CONSTRAINT [PK_type] PRIMARY KEY CLUSTERED

(

 [vehicle_type_id] ASC

)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,

IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS

= ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SQL for truncating all tables

GO

```
ALTER TABLE [dbo].[adetail] DROP CONSTRAINT [FK_adetail_info]
```

GO

```
ALTER TABLE [dbo].[adetail] DROP CONSTRAINT [FK_adetail_type]
```

GO

```
ALTER TABLE [dbo].[adetail] DROP CONSTRAINT [FK_adetail_cinfo]
```

GO

```
ALTER TABLE [dbo].[adetail] DROP CONSTRAINT [FK_adetail_einfo]
```

GO

```
truncate table info;
```

```
truncate table adetail;
```

```
truncate table type;
```

```
truncate table einfo;
```

```
truncate table cinfo;
```

GO

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT [FK_adetail_info]
```

```
FOREIGN KEY([vehicle_info_id])
```

```
REFERENCES [dbo].[info] ([vehicle_info_id])
```

GO

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_info]
```

GO

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT  
[FK_adetail_type] FOREIGN KEY([vehicle_type_id])  
REFERENCES [dbo].[type] ([vehicle_type_id])  
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_type]  
GO
```

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT  
[FK_adetail_einfo] FOREIGN KEY([event_info_id])  
REFERENCES [dbo].[einfo] ([event_info_id])  
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_einfo]  
GO
```

```
ALTER TABLE [dbo].[adetail] WITH CHECK ADD CONSTRAINT  
[FK_adetail_cinfo] FOREIGN KEY([contribution_info_id])  
REFERENCES [dbo].[cinfo] ([contribution_info_id])  
GO
```

```
ALTER TABLE [dbo].[adetail] CHECK CONSTRAINT [FK_adetail_cinfo]  
GO
```