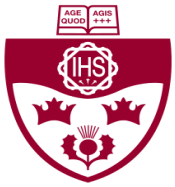


SAINT MARY'S UNIVERSITY SINCE 1802

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

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

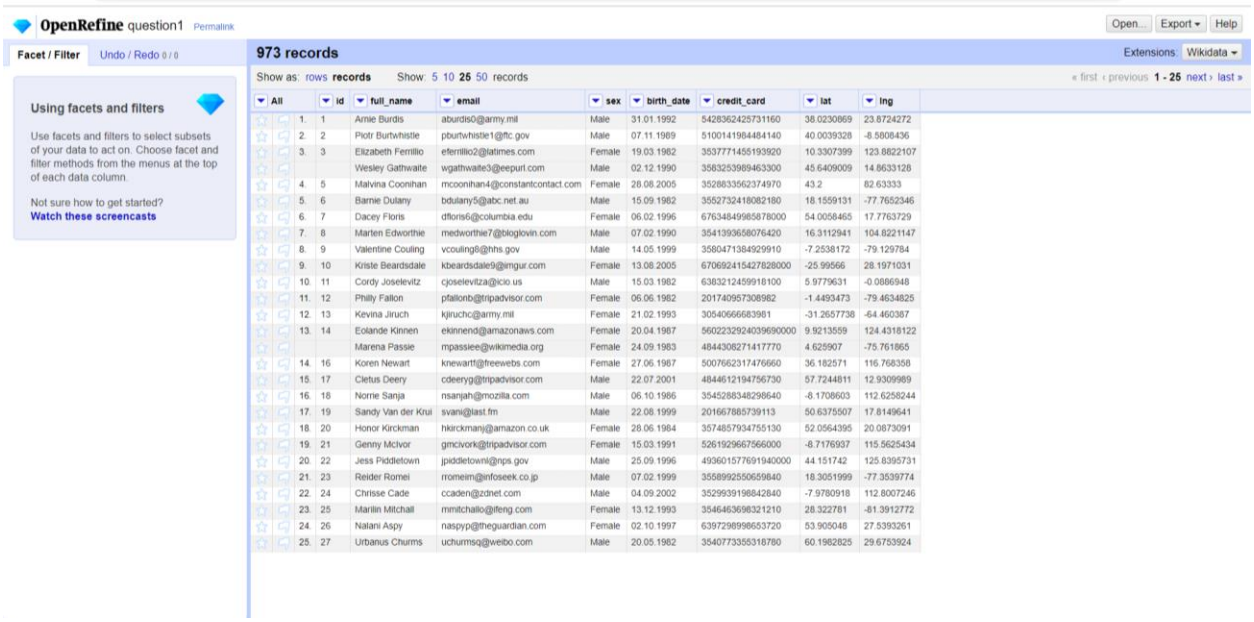
After successful installation of OpenRefine**, "question1.csv"** file is loaded, and hence new project is created and named as **"question1"** as per the provided instructions.

Tasks:

1. Install OpenRefine as shown in the tutorial.

2. Create Project **question1** in OpenRefine.

3. Load **question1.csv**.

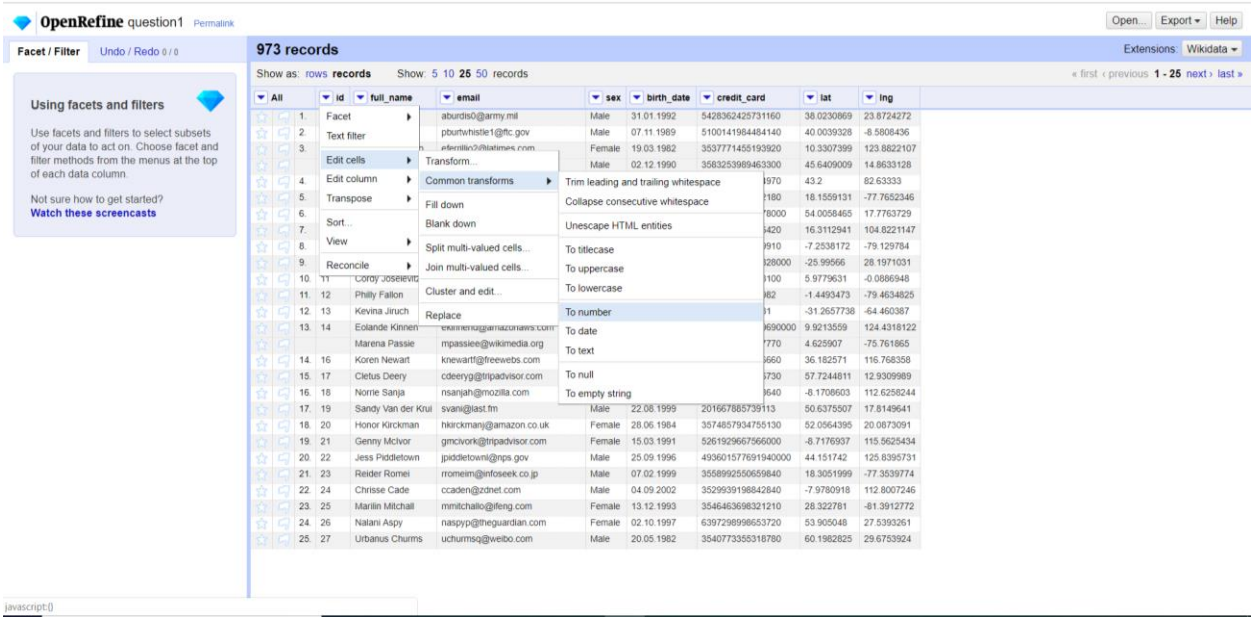


Here is the snapshot after loading 'question1.csv' in 'question1' project.

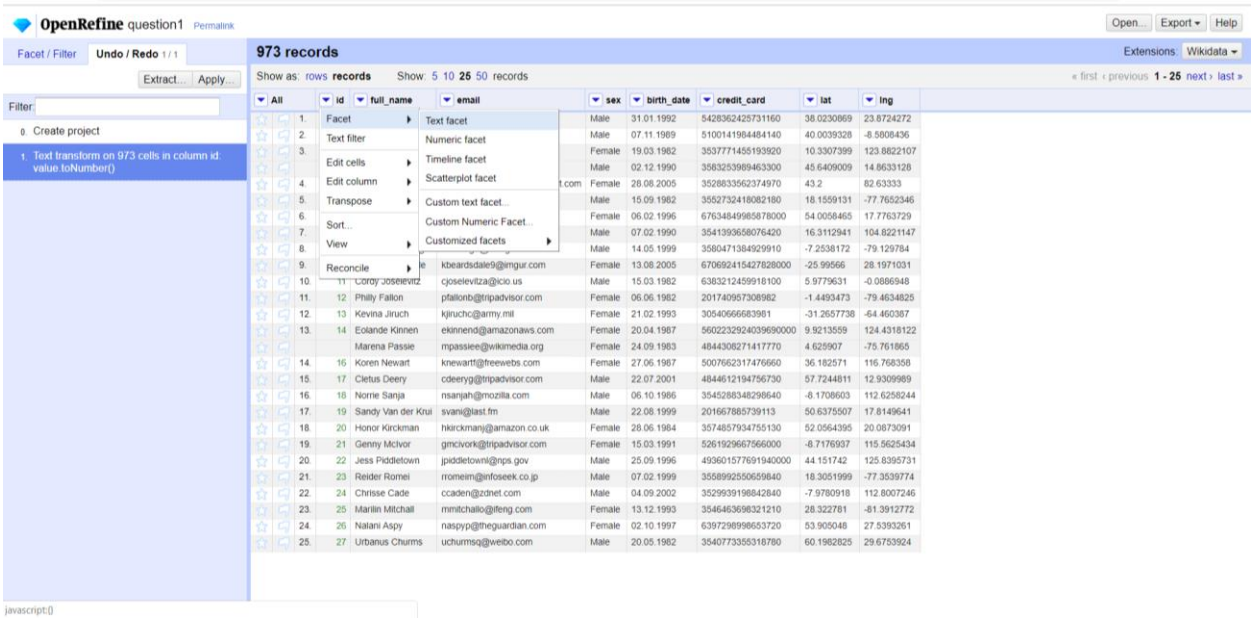
4. Remove all rows in which **id** is empty.

Ans:

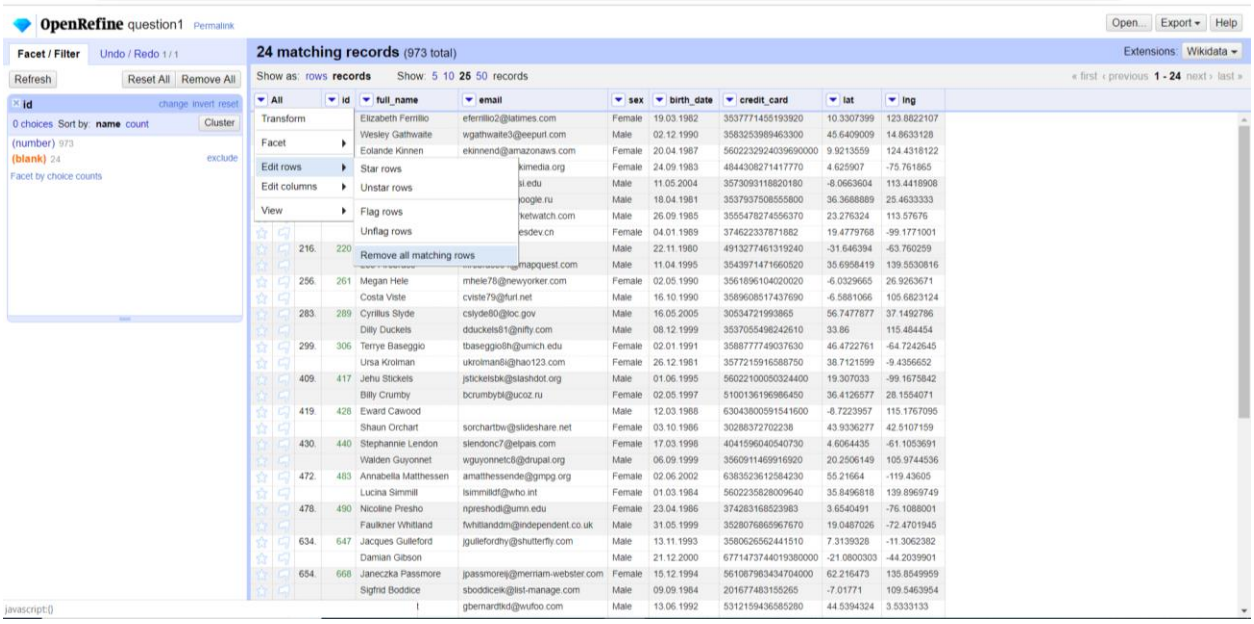
The '**id'** column is first transformed to **'number'** and then '**Text Facet'** is done. Then, '**(blank)'** option is selected and then under the '**All'** dropdown menu, edit rows is selected and finally the option of **"Remove all matching rows"** is selected to ensure that all the empty id rows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



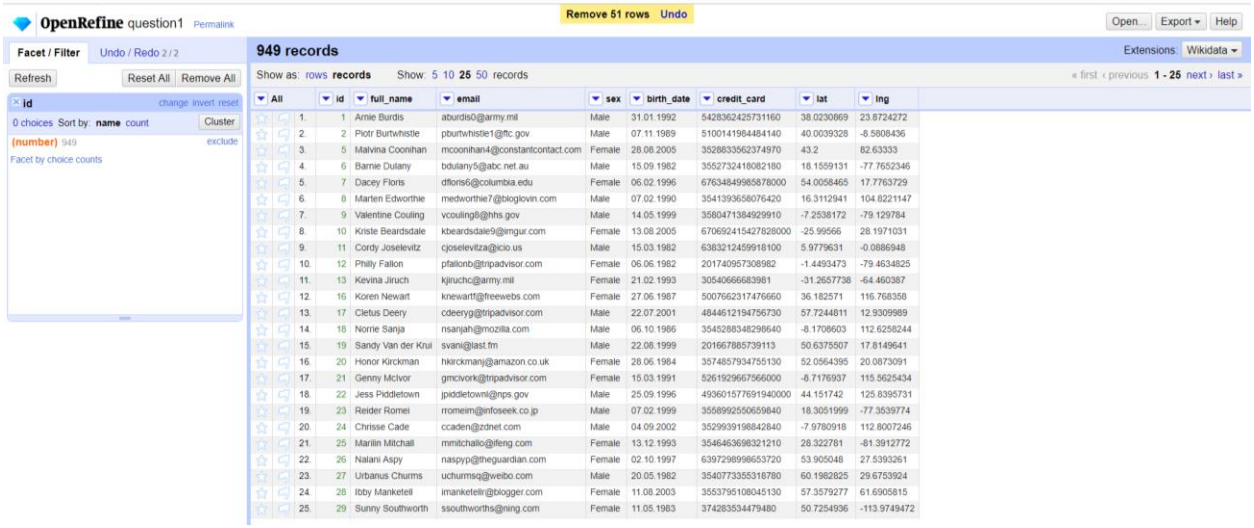
Here is the snapshot that shows the steps to transform the 'id' column into number.



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



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.

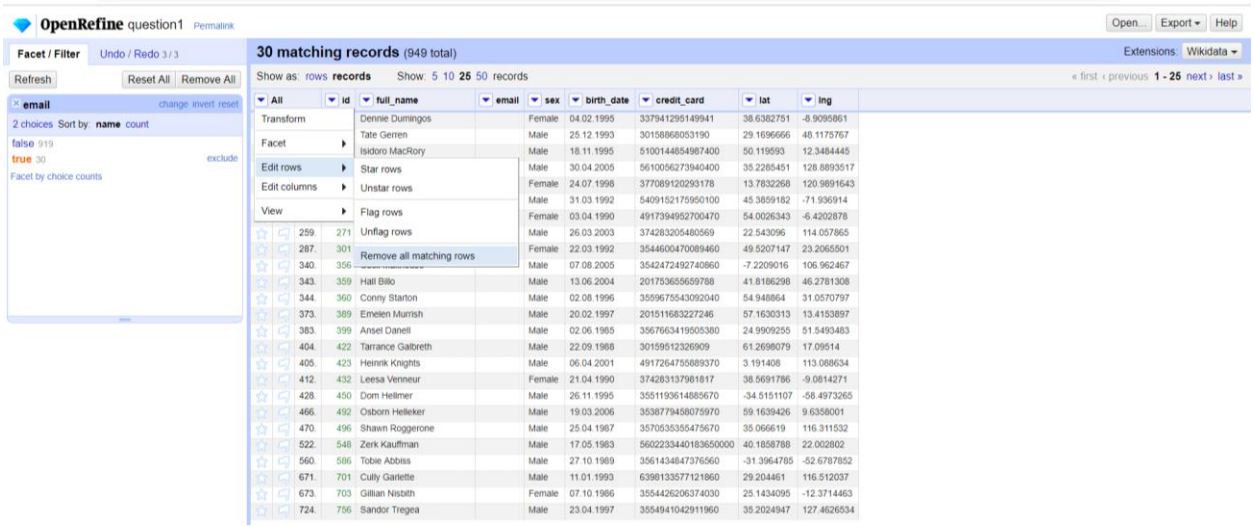


Here is the snapshot after deletion of empty id records where we are then left with 949 records.

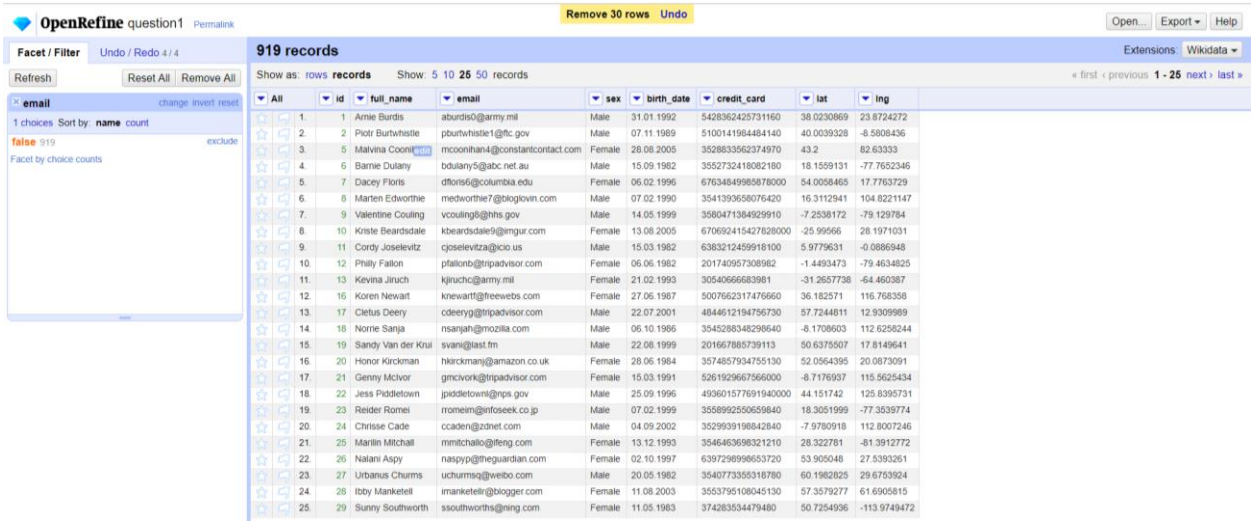
5. Remove all rows in which **email** is empty.

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.



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.



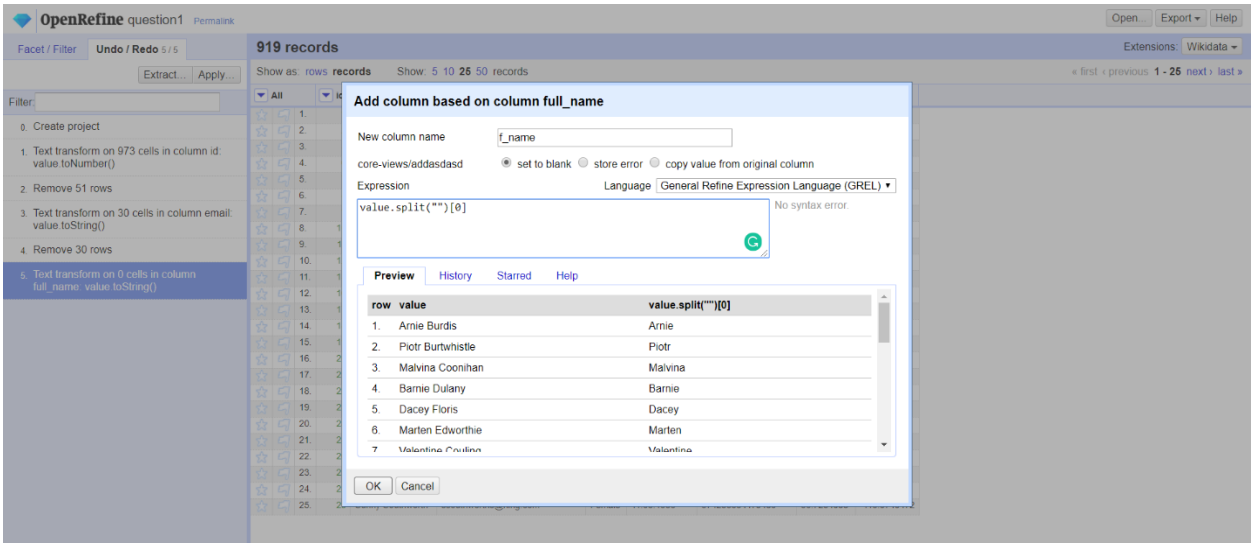
All	id	full_name	email	sex	birth_date	credit_card	lat	lng
1.	1	Arnie Burdis	aburdis0@army.mil	Male	31.01.1992	5428362425731160	38.0230869	23.8724272
2.	2	Piotr Burtwhistle	pburtwhistle1@ftc.gov	Male	07.11.1989	5100141984484140	40.0039328	-8.5808436
3.	5	Malvina Coonihan	mcoonihan4@constantcontact.com	Female	28.08.2005	3528833562374970	43.2	82.63333
4.	6	Barnie Dulany	bdulany5@abc.net.au	Male	15.09.1982	3552732418082180	18.1559131	-77.7652346
5.	7	Dacey Floris	dfloris6@columbia.edu	Female	06.02.1996	6763484998587800	54.0058465	17.7763729
6.	8	Marten Edworthie	medworthie7@bloglovin.com	Male	07.02.1990	3541393658076420	16.3112941	104.8221147
7.	9	Valentine Couling	vcouling8@hhs.gov	Male	14.05.1999	3580471384929910	-7.2538172	-79.129784
8.	10	Kriste Beardsdale	kbeardsdale9@imgur.com	Female	13.08.2005	6706924154278280000	-25.99566	28.1971031
9.	11	Cordy Joselevitz	cjoselevitza@icio.us	Male	15.03.1982	6383212459918100	5.9779631	-0.0886948
10.	12	Philly Fallon	pfallonb@tripadvisor.com	Female	06.06.1982	201740957308982	-1.4493473	-79.4634825
11.	13	Kevina Jiruch	kjiruchc@army.mil	Female	21.02.1993	30540666683981	-31.2657738	-64.460387
12.	16	Koren Newart	knewartf@freewebs.com	Female	27.06.1987	5007662317476660	36.182571	116.768358
13.	17	Cletus Deery	cdeeryg@tripadvisor.com	Male	22.07.2001	4844612194756730	57.7244811	12.9309989
14.	18	Norrie Sanja	nsanjah@mozilla.com	Male	06.10.1986	3545288348298640	-8.1708603	112.6258244
15.	19	Sandy Van der Krui	svani@last.fm	Male	22.08.1999	201667885739113	50.6375507	17.8149641
16.	20	Honor Kirckman	hkirckmanj@amazon.co.uk	Female	28.06.1984	3574857934755130	52.0564395	20.0873091
17.	21	Genny McIvor	gmcivork@tripadvisor.com	Female	15.03.1991	5261929667566000	-8.7176937	115.5625434
18.	22	Jess Piddletown	jpiddletowni@nps.gov	Male	25.09.1996	4936015776919400000	44.151742	125.8395731
19.	23	Reider Romei	rromeim@infoseek.co.jp	Male	07.02.1999	3558992550659840	18.3051999	-77.3539774
20.	24	Chrisse Cade	ccaden@zdnet.com	Male	04.09.2002	3529939198842840	-7.9780918	112.8007246
21.	25	Marilin Mitchall	mmitchallo@ifeng.com	Female	13.12.1993	3546463698321210	28.322781	-81.3912772
22.	26	Nalani Aspy	naspyp@theguardian.com	Female	02.10.1997	6397298998653720	53.905048	27.5393261
23.	27	Urbanus Churms	uchurmsq@weibo.com	Male	20.05.1982	3540773355316780	60.1982825	29.6753924
24.	28	Ibby Manketell	imanketellr@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.

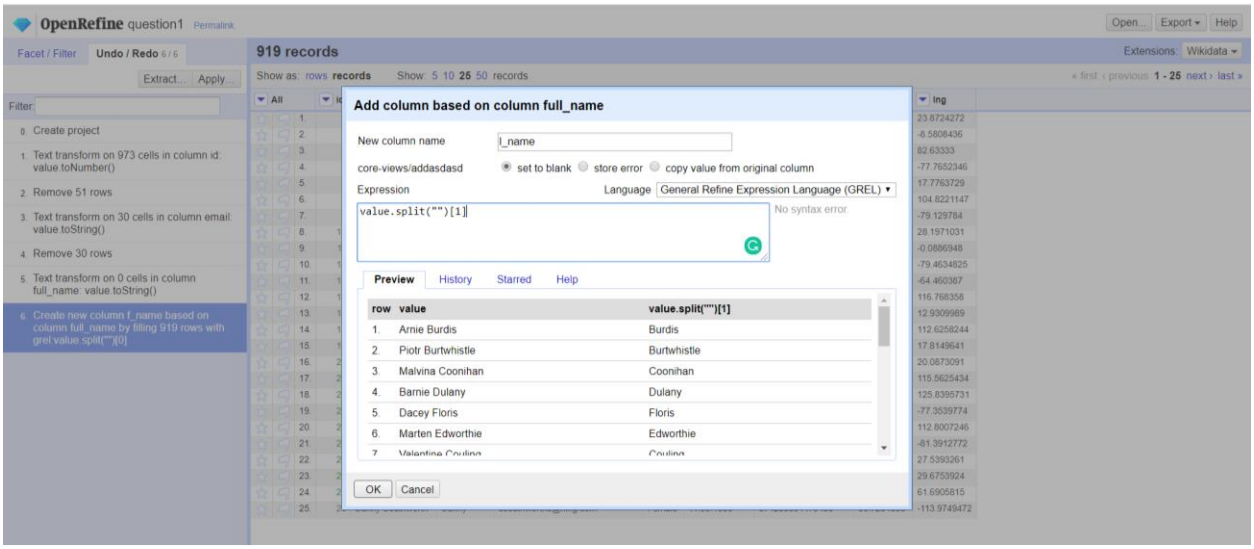
6. Split Full Name in to First name and Last Name. Store first name into **f_name** and last name into **l_name** (you will create these two columns).

Ans:

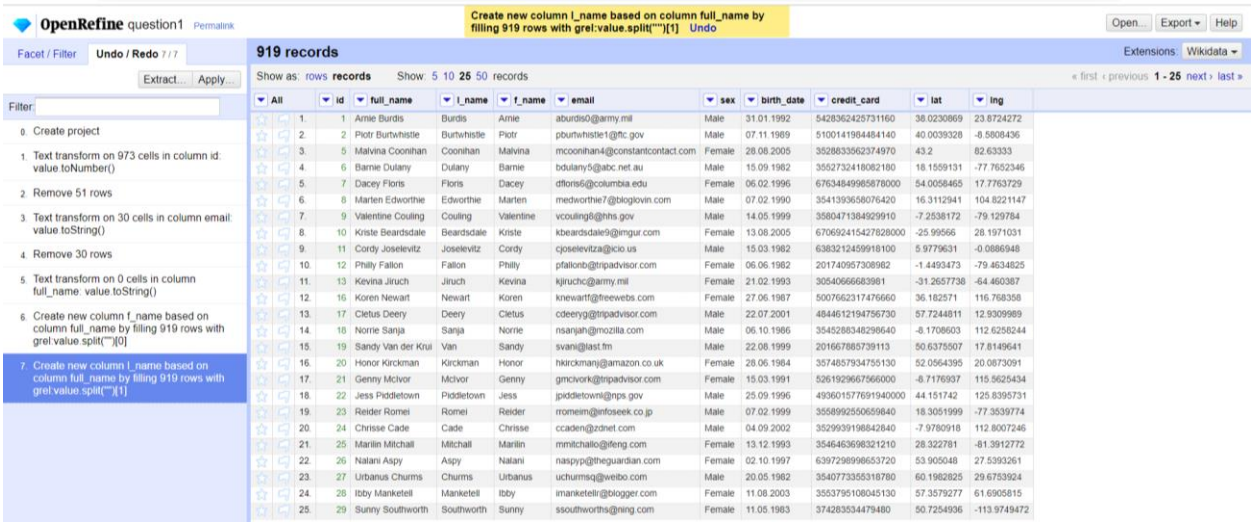
The '**full_name'** column is first transformed to 'text' and then under the **'Edit Column'**, **'Add column based on this column…'** option is selected. Then, new column names: **'f_name'** and **'l_name'** are defined with the GREL expression of **'value.split("")[0]'** and '**value.split("")[1]'** 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.



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



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

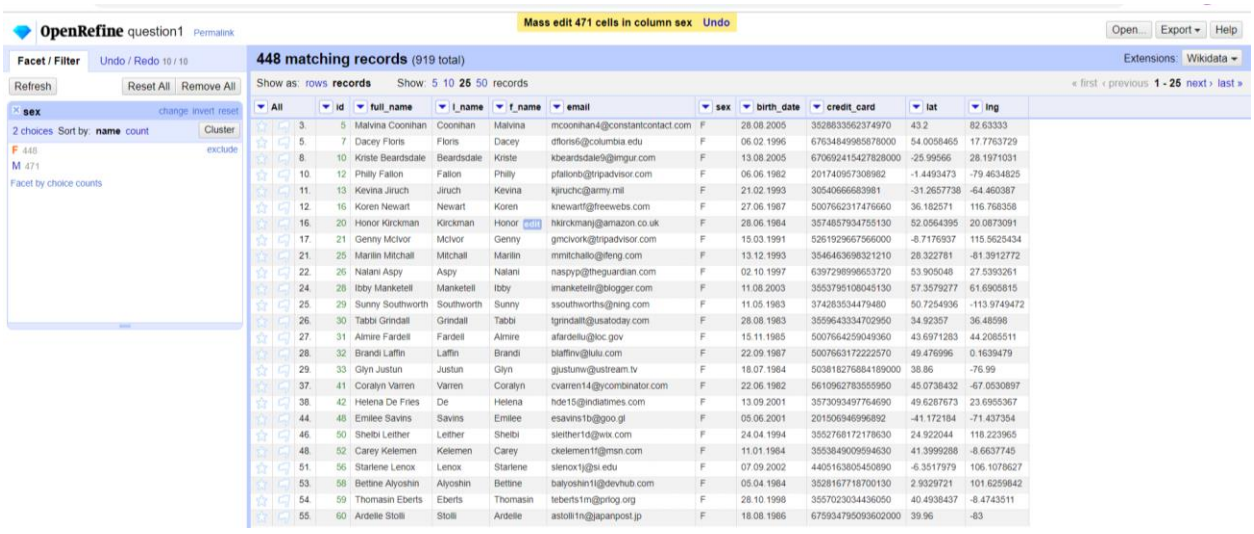


Here is the snapshot after splitting of 'full_name' column into 'f_name' and 'l_name' columns.

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.

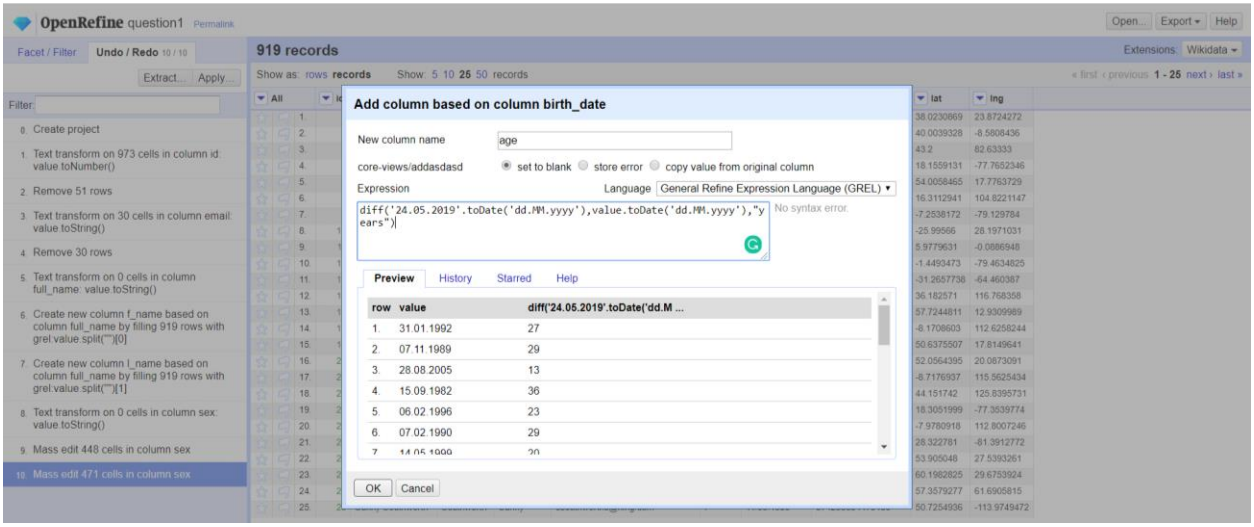


Here is the snapshot that illustrates the after the coding of 'sex' column into 'M' and 'F' for male and female respectively.

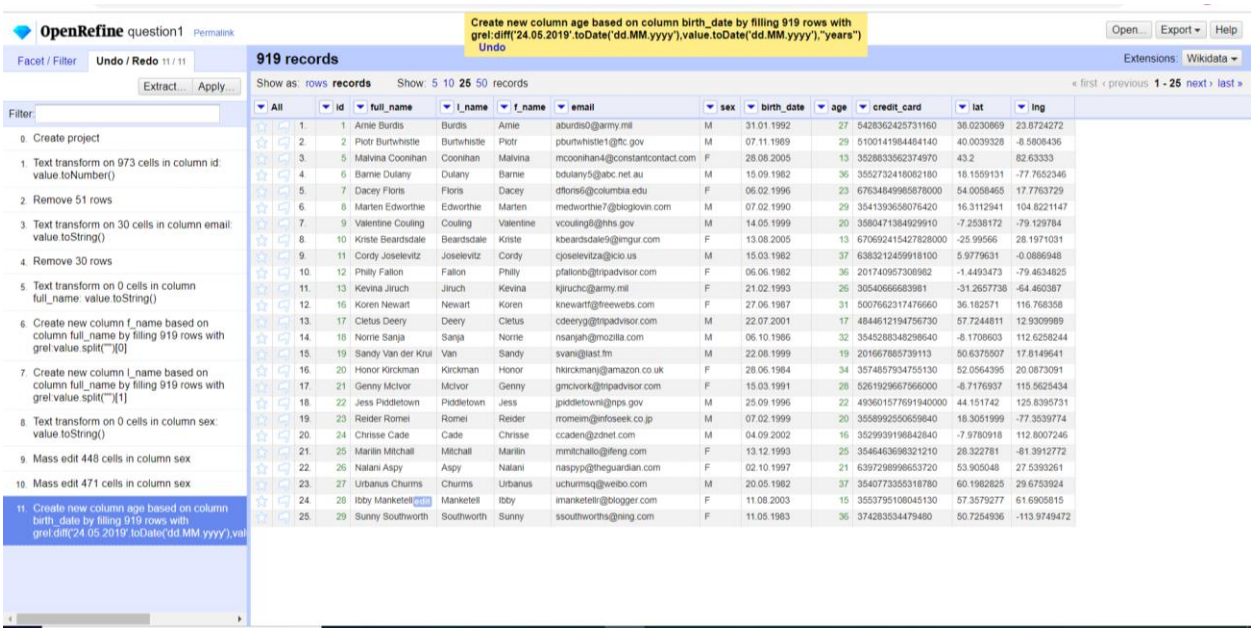
8. Calculate age of the person on 24th May 2019 from the **birth_date** column. Store calculated age in **age** column (You will create this column).

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.

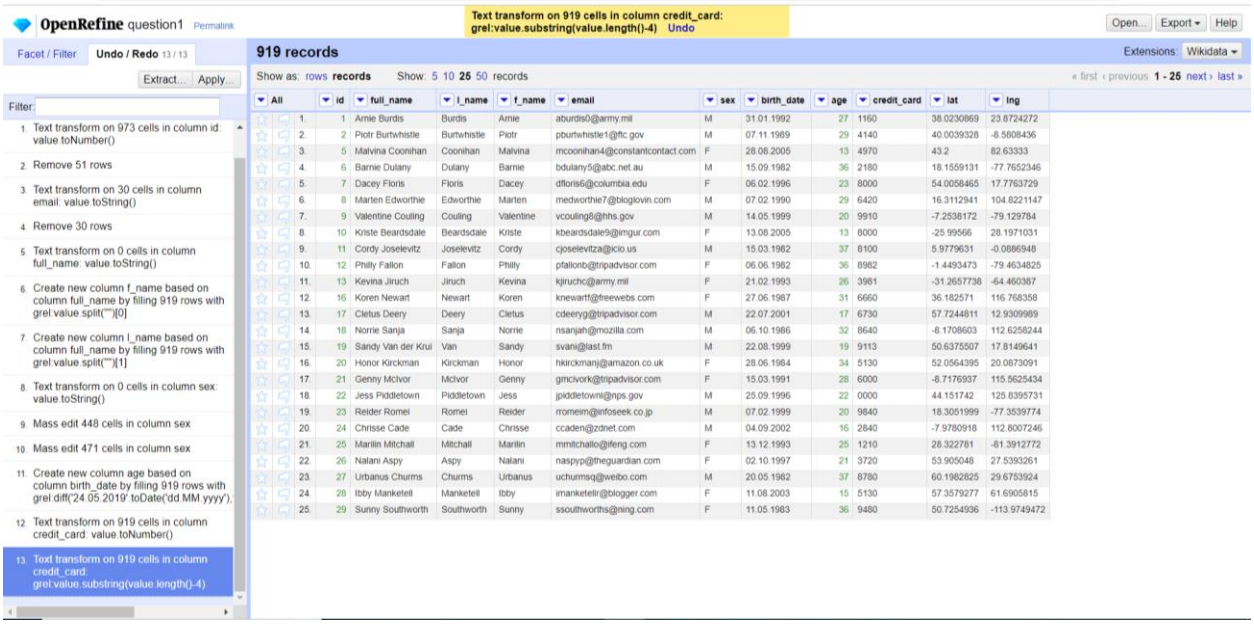


Here is the snapshot that illustrates the after derivation of age column after calculating age from 'birth_date' with respect to 24th May 2019 as a base date.

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.

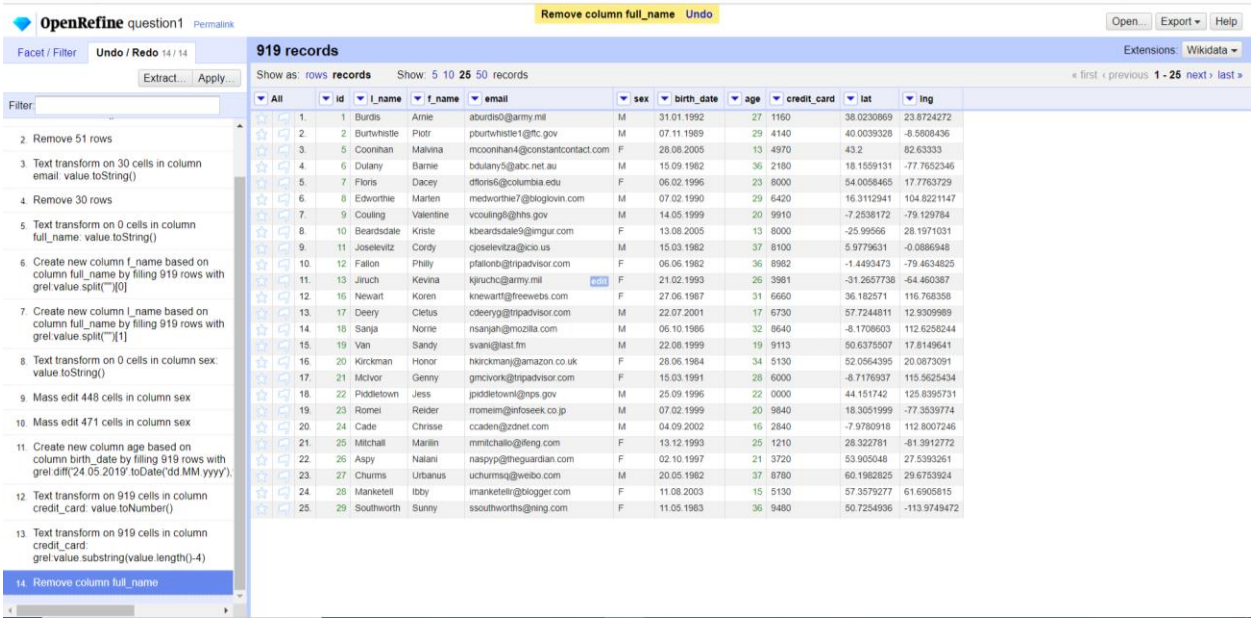


Here is the snapshot that illustrates after extraction of only last 4 digits of 'credit_card' column.

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.

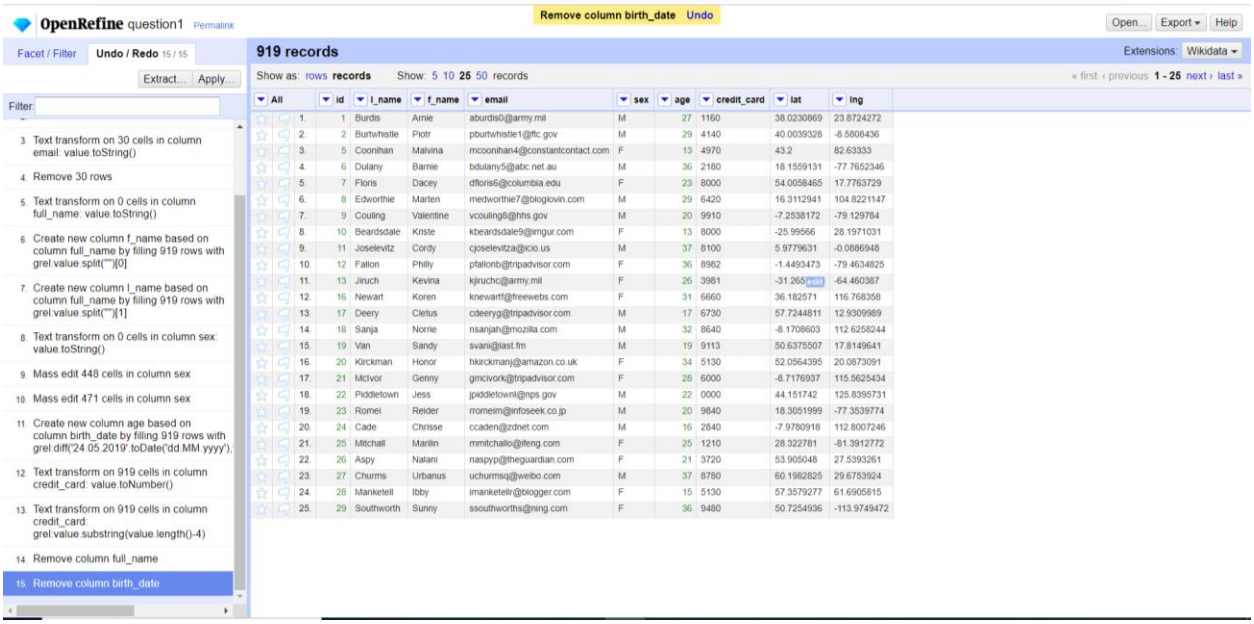


Here is the snapshot that illustrates after the deletion of 'full_name' column.

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.

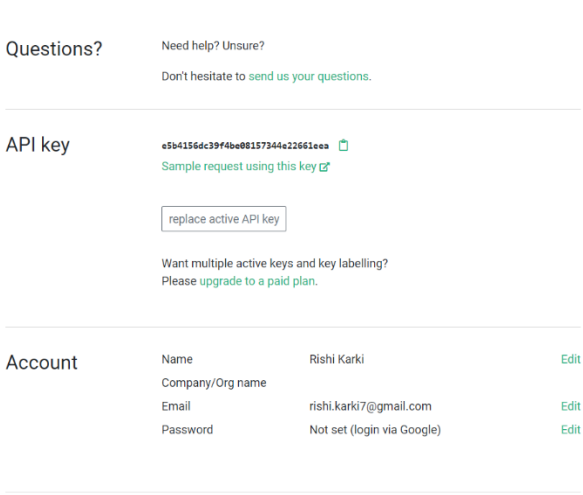


Here is the snapshot that illustrates after the deletion of 'birth_date' column.

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**.



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

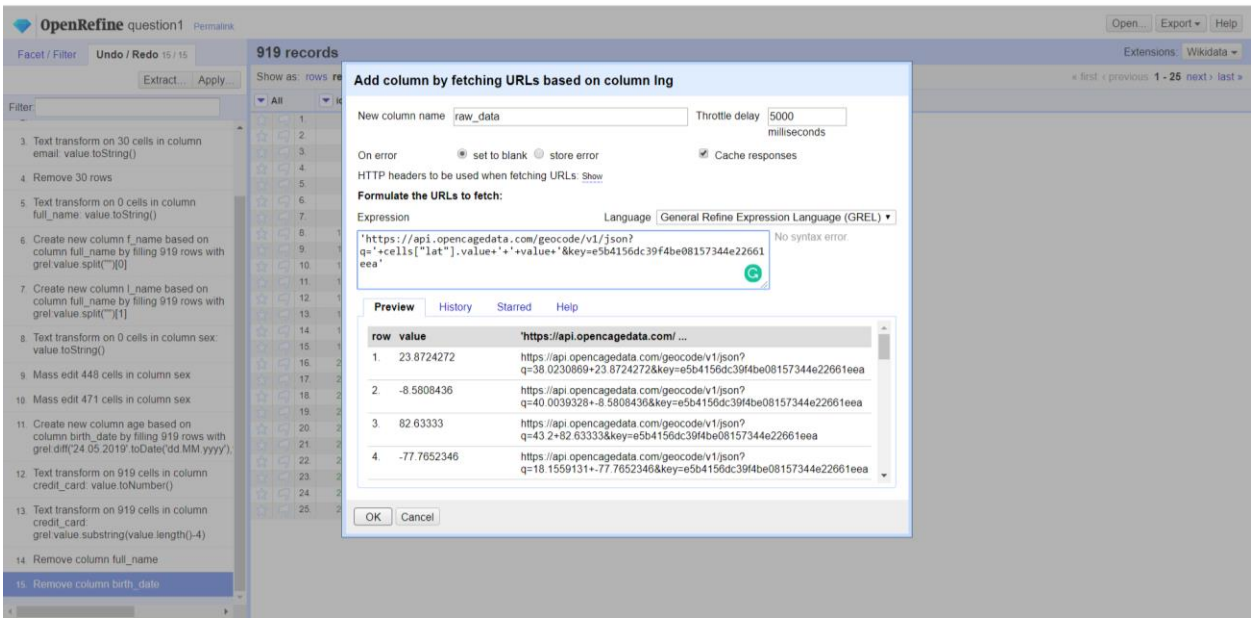
14. You can replace the text highlighted in red using expression and retrieve location detail (JSON data). You can store this JSON data into **raw_data** column (You will create this column).

Ans:

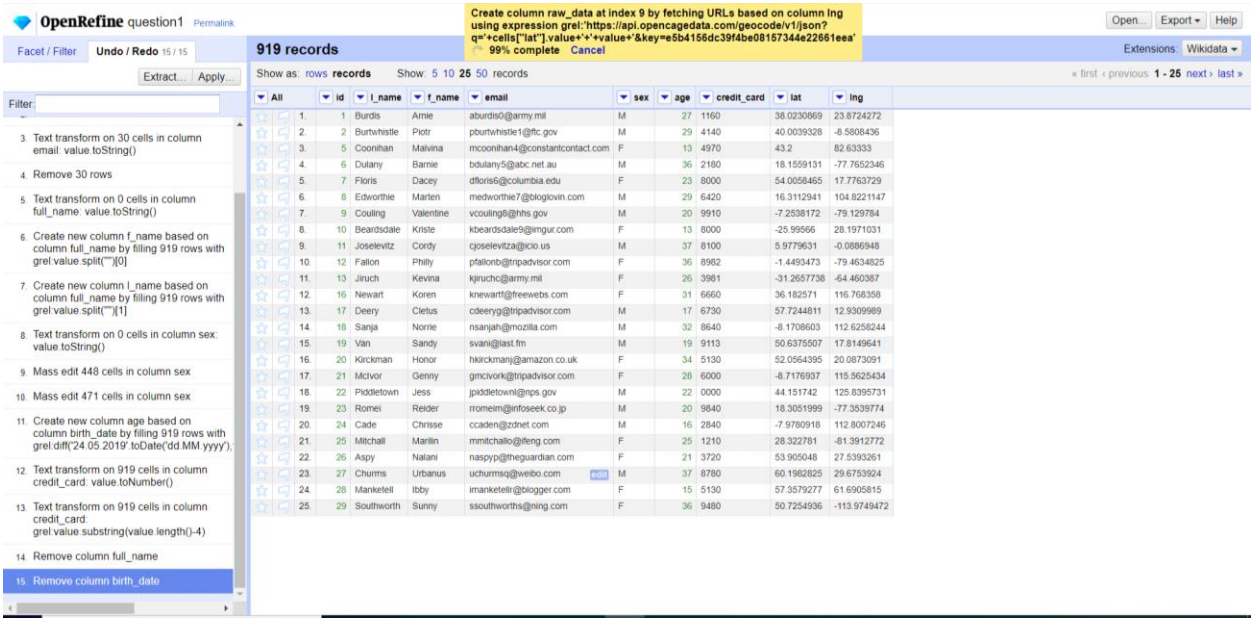
The **'lng'** column is first selected and under **'Edit Column', 'Add Column by fetching URLs…'** option is selected. Then, new column name of **'raw_data'** is defined with the GREL expression of

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

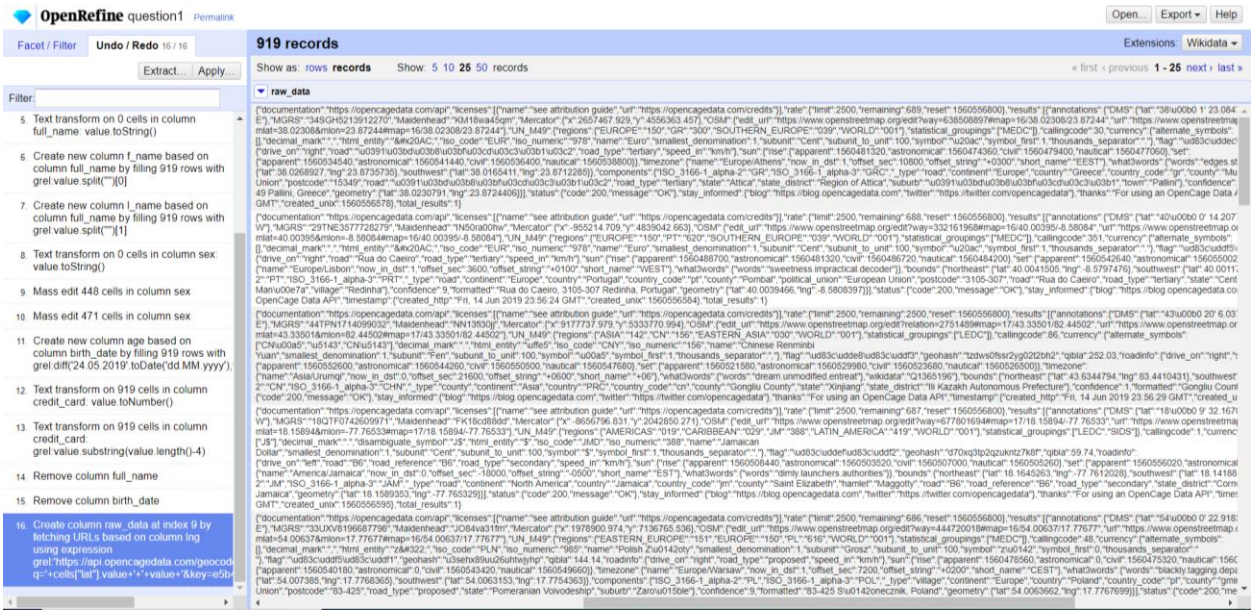
in order to retrieve location JSON data by appending **'lat'** column value, **'lng'** column value and '**API Key'** as mentioned in Q.No. 13's URL link. Hence, web scrapping is successfully done and location JSON data is obtained in our dataset.



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.



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

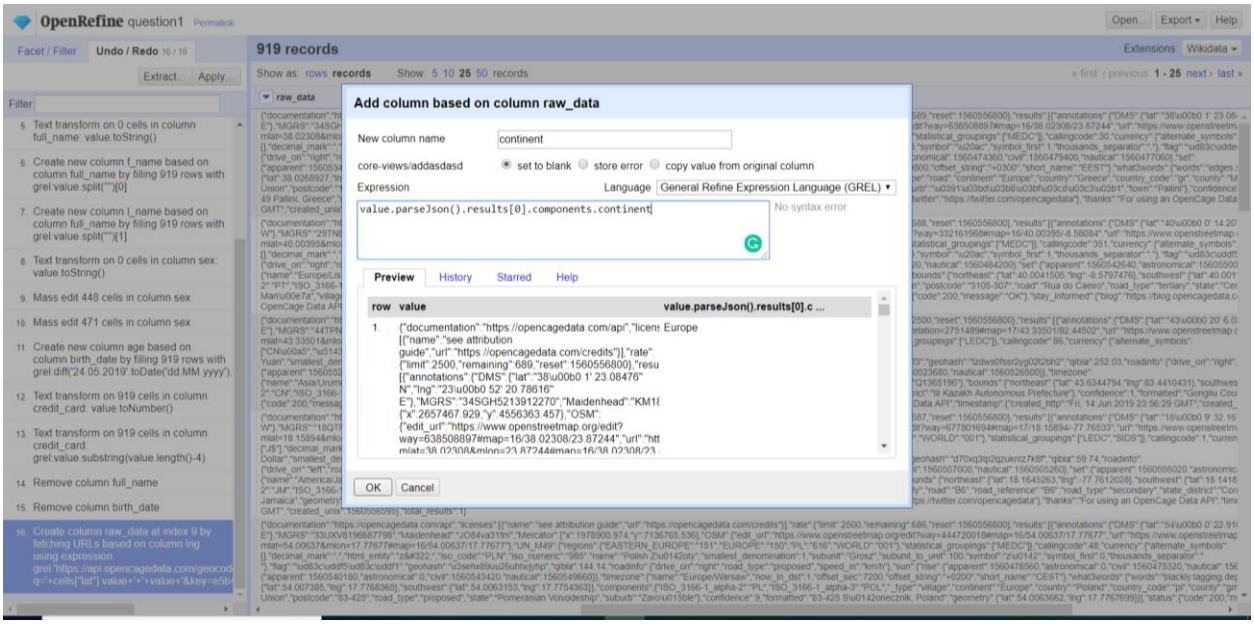


Here is the snapshot that illustrates after the derivation of 'raw_data' colum from location JSON data.

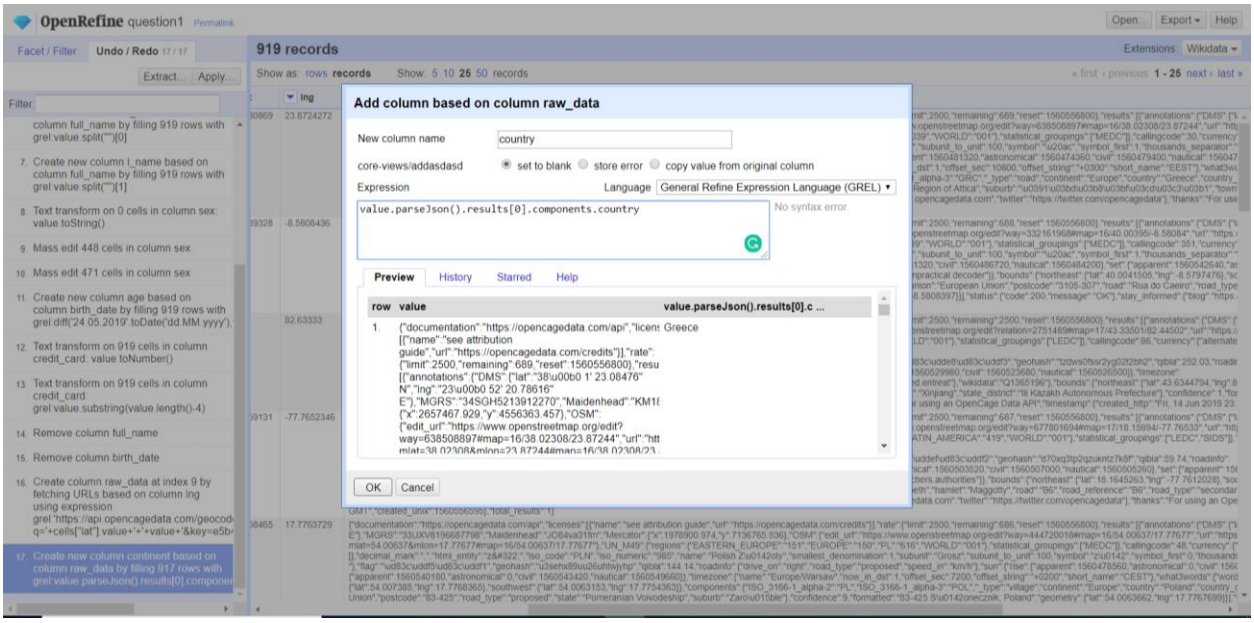
15. Filter continent and country from the JSON data and store into **continent and country** columns (You will create these two columns).

Ans:

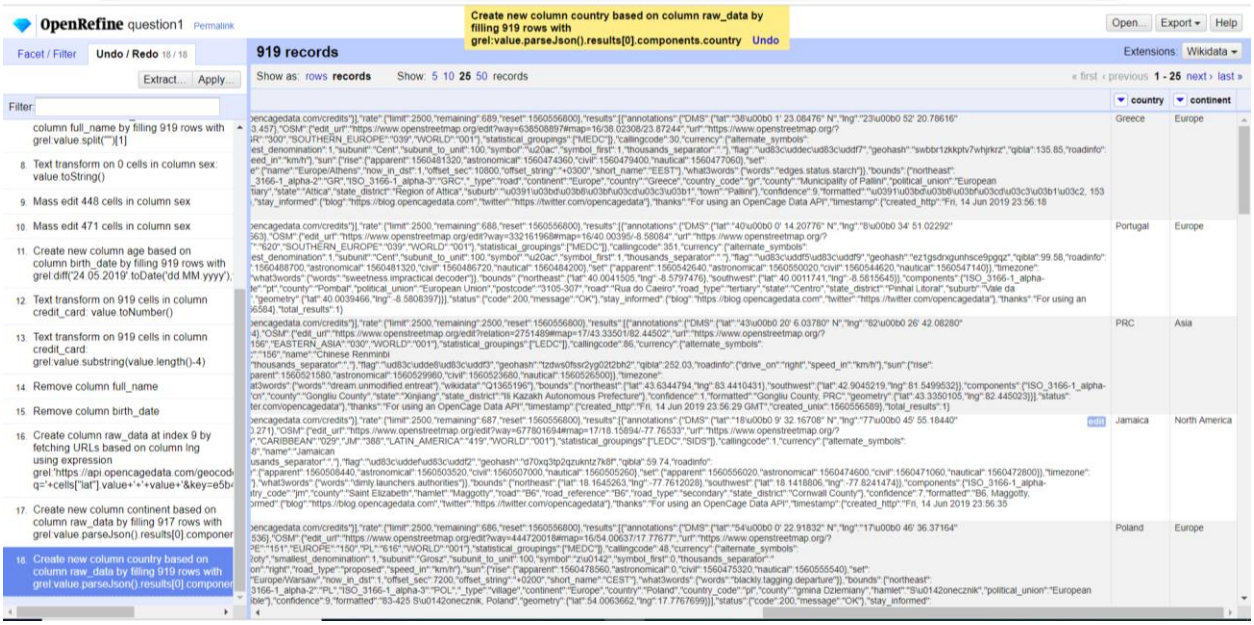
The **'raw_data'** column is first selected and then under the **'Edit Column'**, **'Add column based on this column…'** option is selected. Then, new column names: **'continent'** and **'country'** are defined with the GREL expression of **'value.parseJson().results[0].components.continent'** and **'value.parseJson().results[0].components.country'** 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.

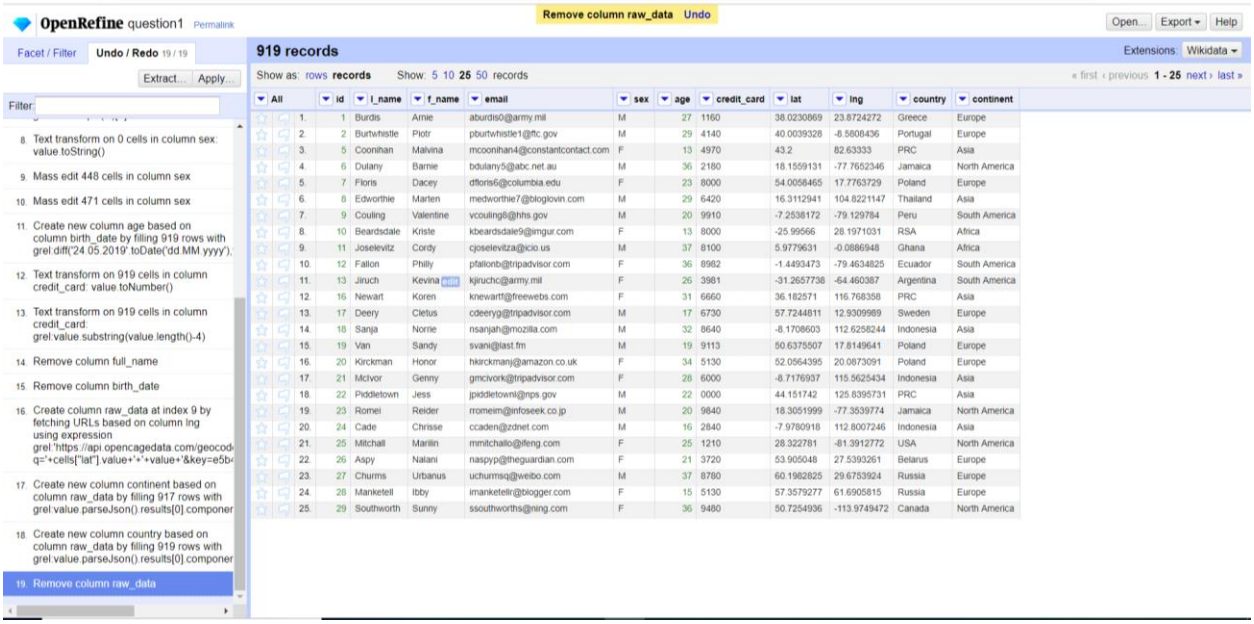


Here is the snapshot that illustrates the result of two new derived columns 'country' and 'continent'.

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.



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**

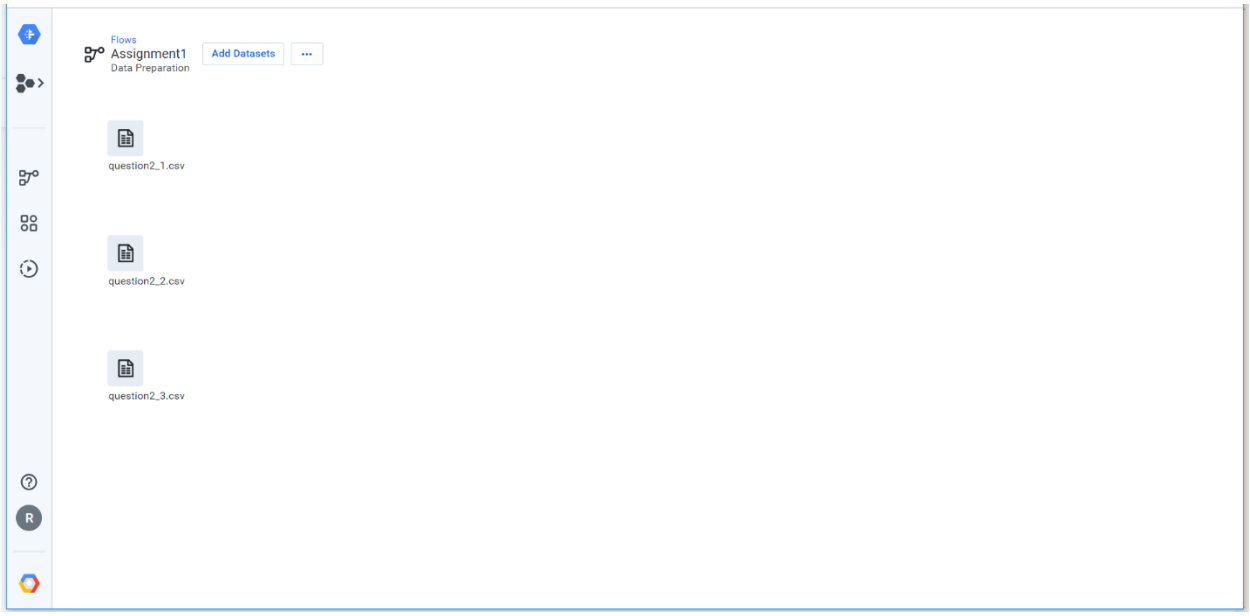
Finally, the project as well as CSV file are exported and renamed project file as '**solution1.tar.gz'** and CSV file as '**solution1.csv'**.
The '**solution1.csv'** has total of 920 records including header with 11 column names as: id, l_name, f_name, email, sex, age, credit_card, lat, lng, country and continent.

3.2 Data Preparation using Google Cloud Dataprep

Question 2: Data Preparation using Google Cloud Dataprep.

Preliminary Steps:

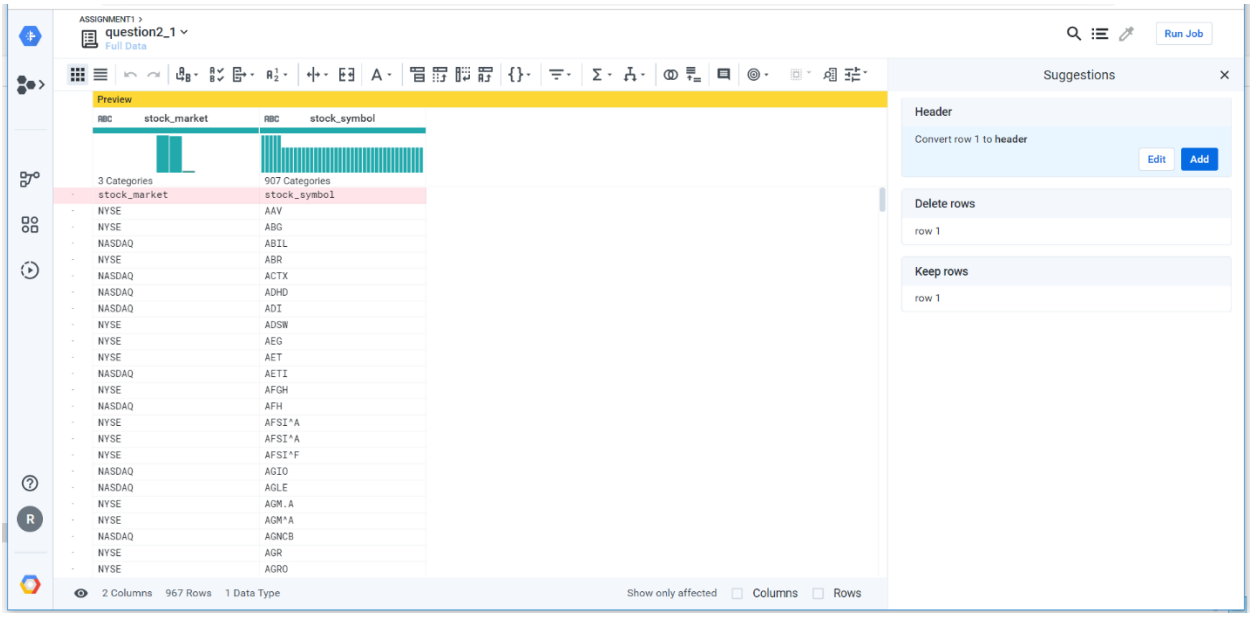
1. First of all, we have done successful signup for Google Cloud Dataprep.
2. Then, we go to following URL:
   https://cloud.google.com/dataprep/
3. Select "View Console"
4. Under sidebar, Select Flows/ Create…/Create Flow with flow description as:
   Flow Name: Assignment1
   Flow Description: Data 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.

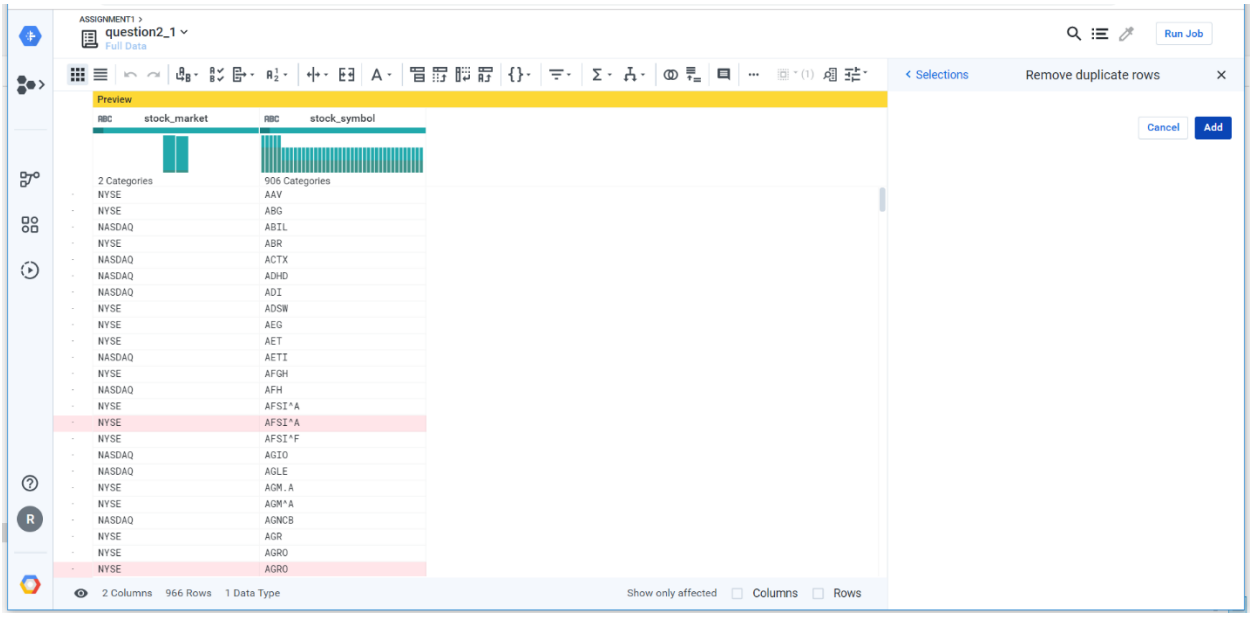
Procedural Steps:

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.



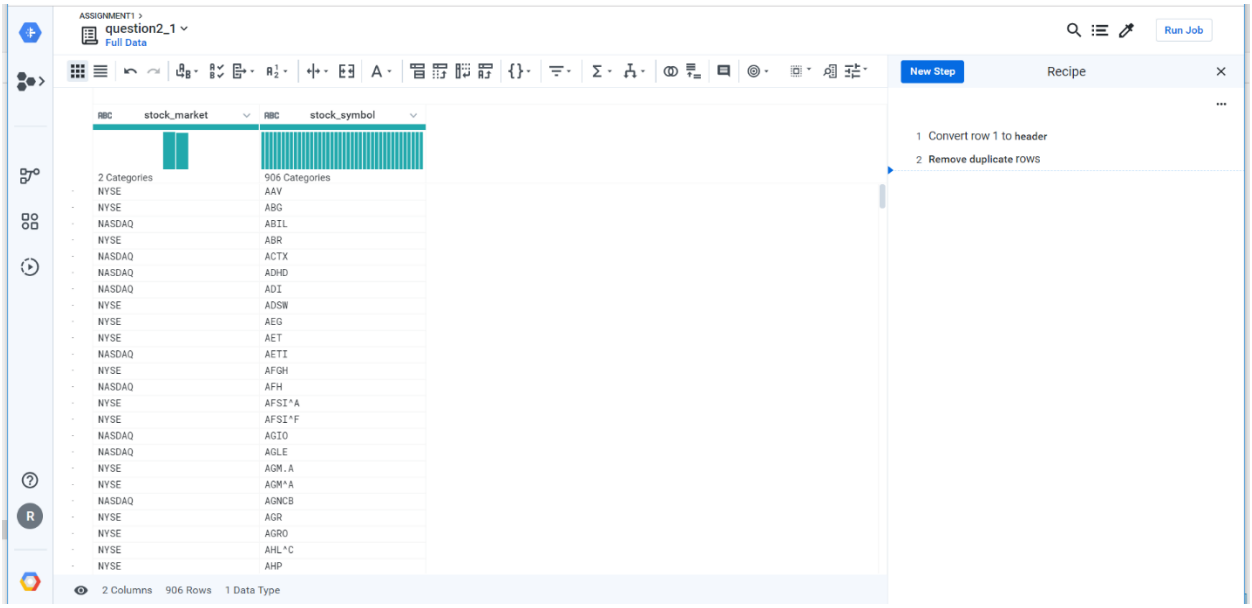
Here is the snapshot for step to add header after clicking on first rows on dataset. We have initially of 967 rows including header rows.

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



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.



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

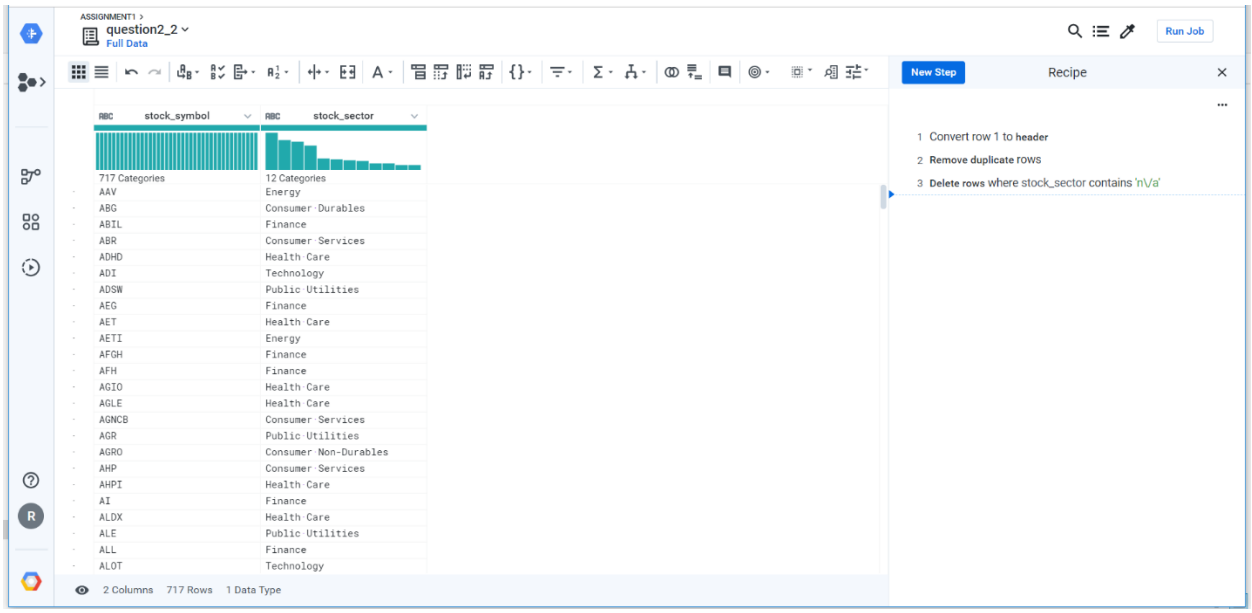
4. Similarly, select **question2_2.csv**, choose “Add new Recipe” and then “Edit Recipe”.

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.

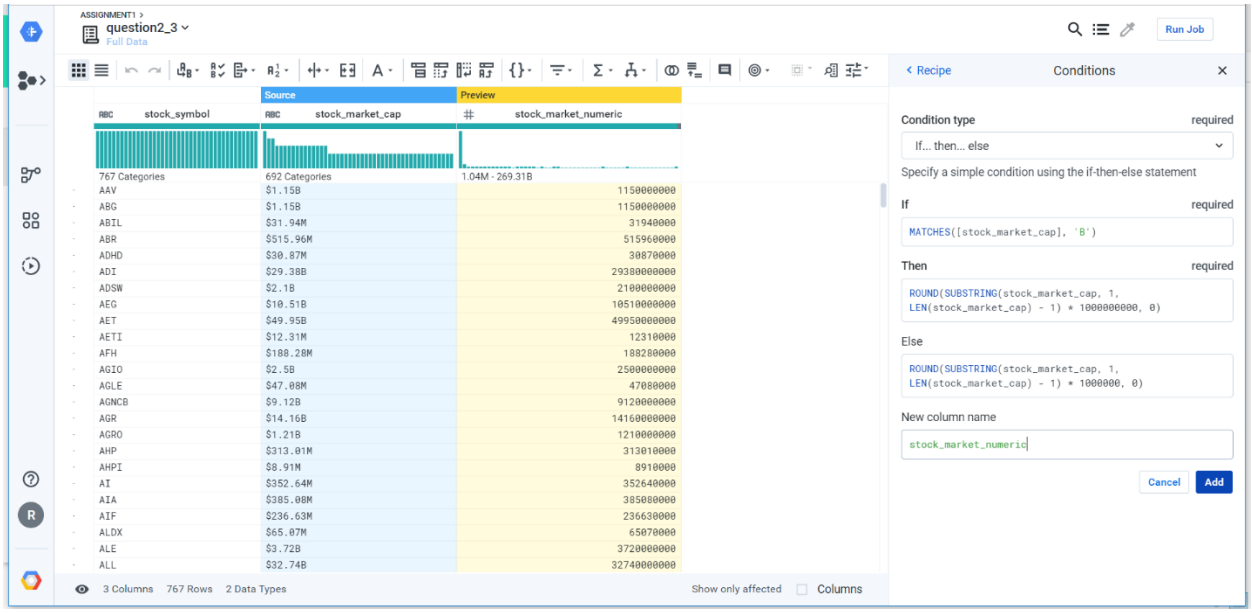
10. Under ‘stock_market_cap’ drop down, Select Filter rows/Remove duplicate rows/Add to remove all the 60 duplicate pairs of stock_sector with stock_symbol.
By this, we remove all the duplicate pairs of stock_market_capt with stock_symbol.

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.

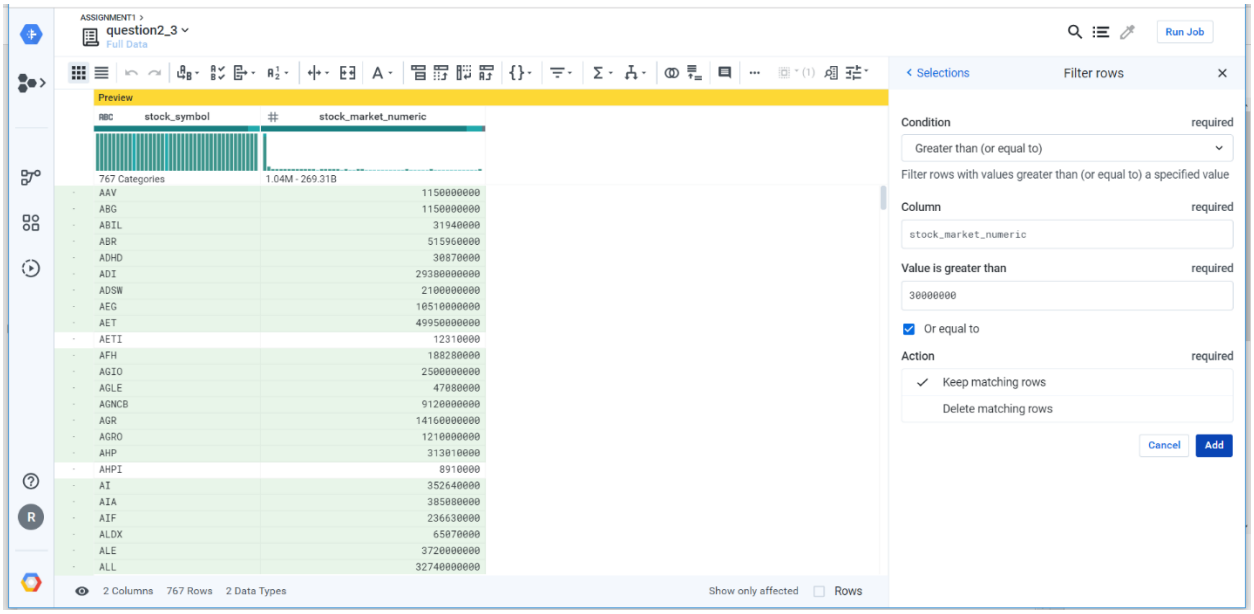
12. Under ‘Search’ menu, Select ‘Conditional Column’ as a transformation option followed by “If… then…else’ condition type with following values for each clause:
if: MATCHES([stock_market_cap],'B')
Then:ROUND(SUBSTRING(stock_market_cap,1,LEN(stock_market_cap)-1)*1000000000,0)
Else: ROUND(SUBSTRING(stock_market_cap, 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.



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.

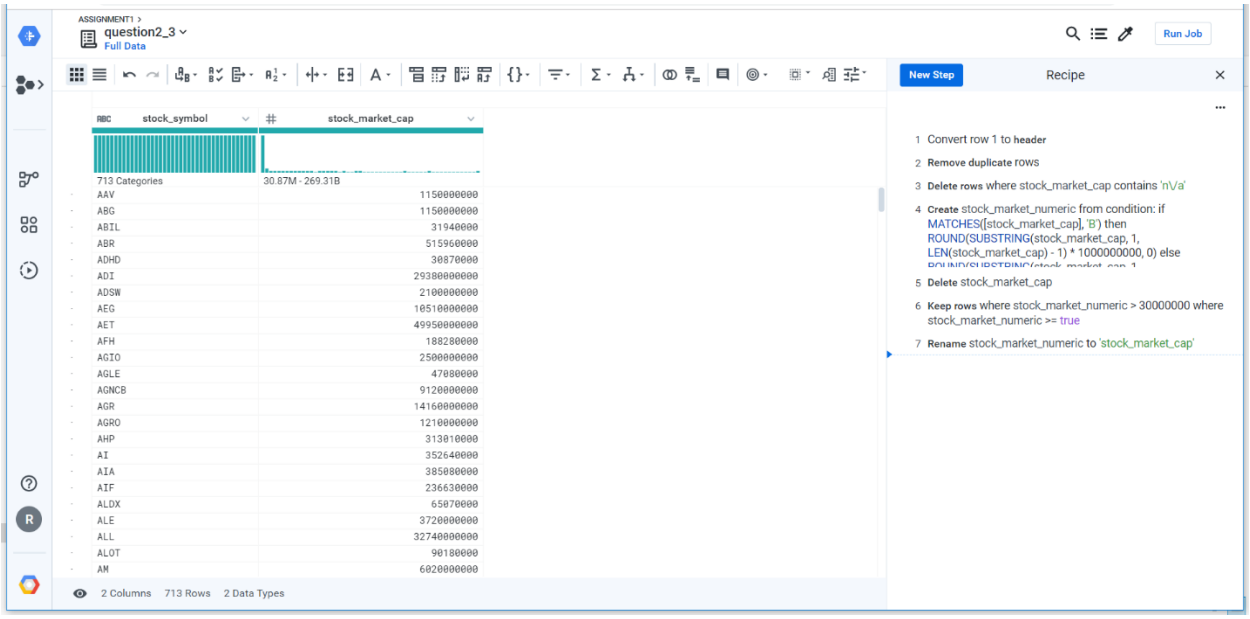
Under, 'stock_market_numeric' dropdown menu, Select Filter rows/On column values/Greater than (or equal to). In 'Value is greater than' option, type 30000000 and check the label 'Or equal to' with action as 'Keep matching rows' and click 'Add'.



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.

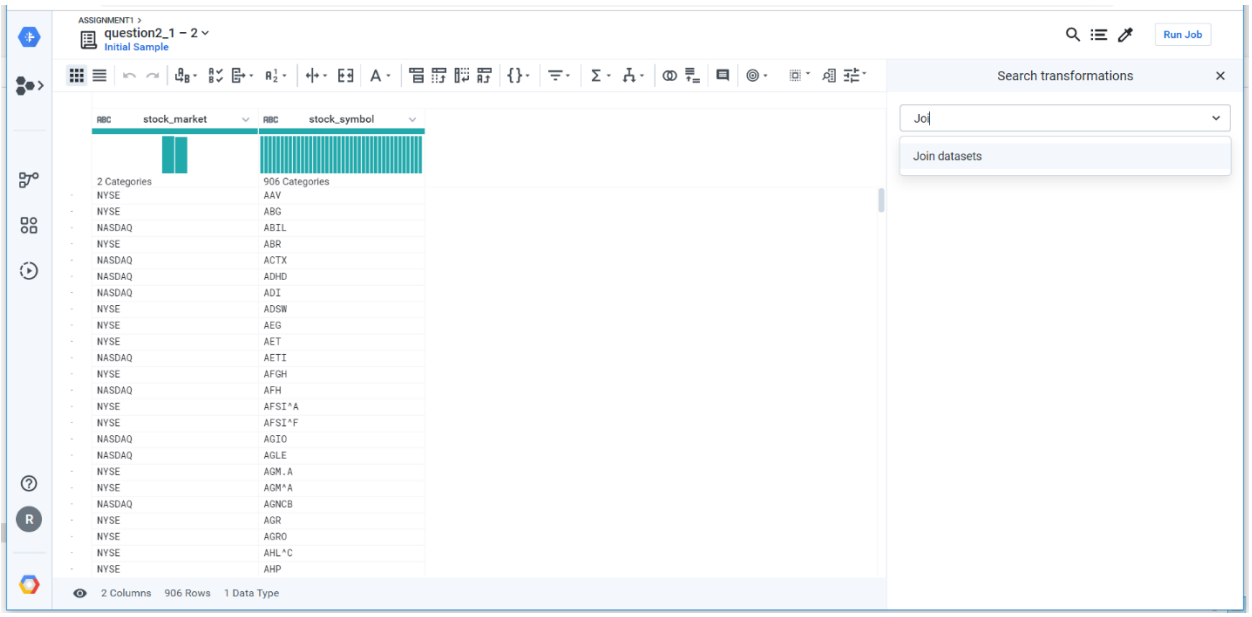


Here is the snapshot of question 2_3.csv after cleaning and transformation process where we are left with only 713 rows.

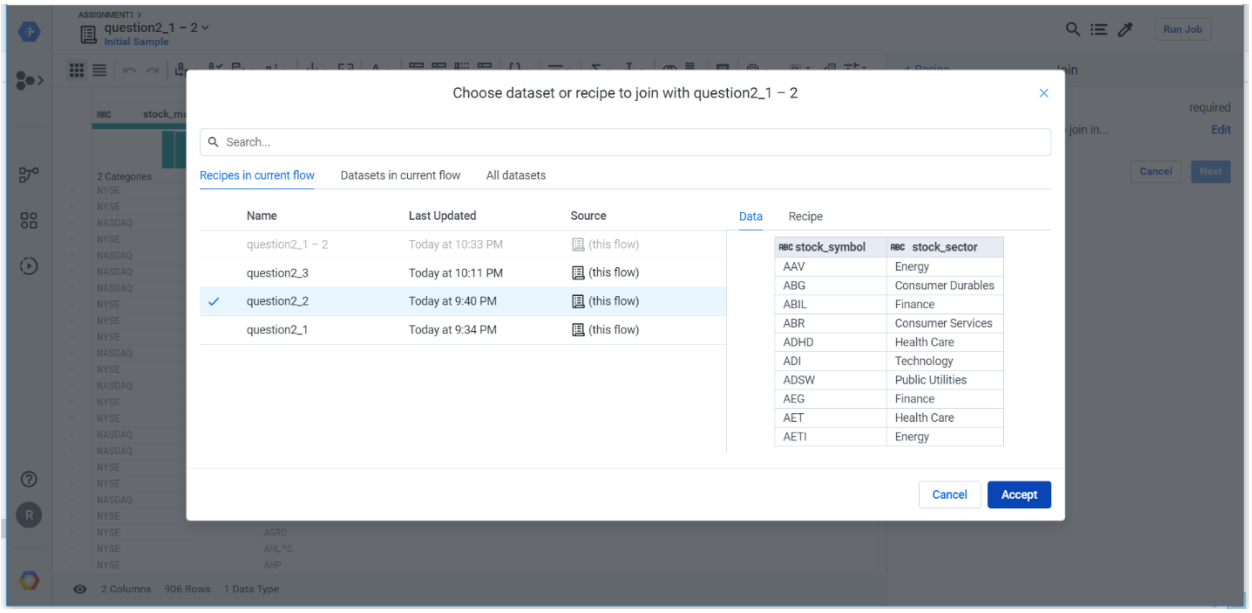
Now, the next challenge is to join all the three datasets into one, with joining two adjacent datasets and finally deriving one complete dataset. Firstly, lets join two datasets: 'question2_1' and 'question2_2' which are discussed below:

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'.



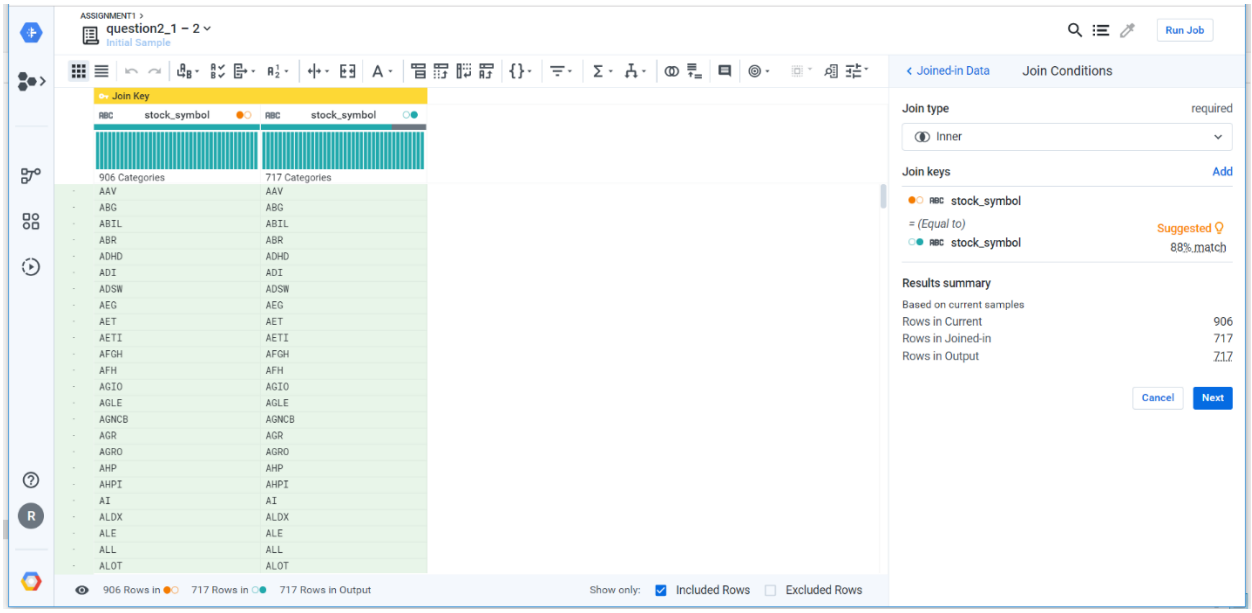
Here is the snapshot for the initial process to join two datasets. The newly created recipe is ready to join to another dataset.



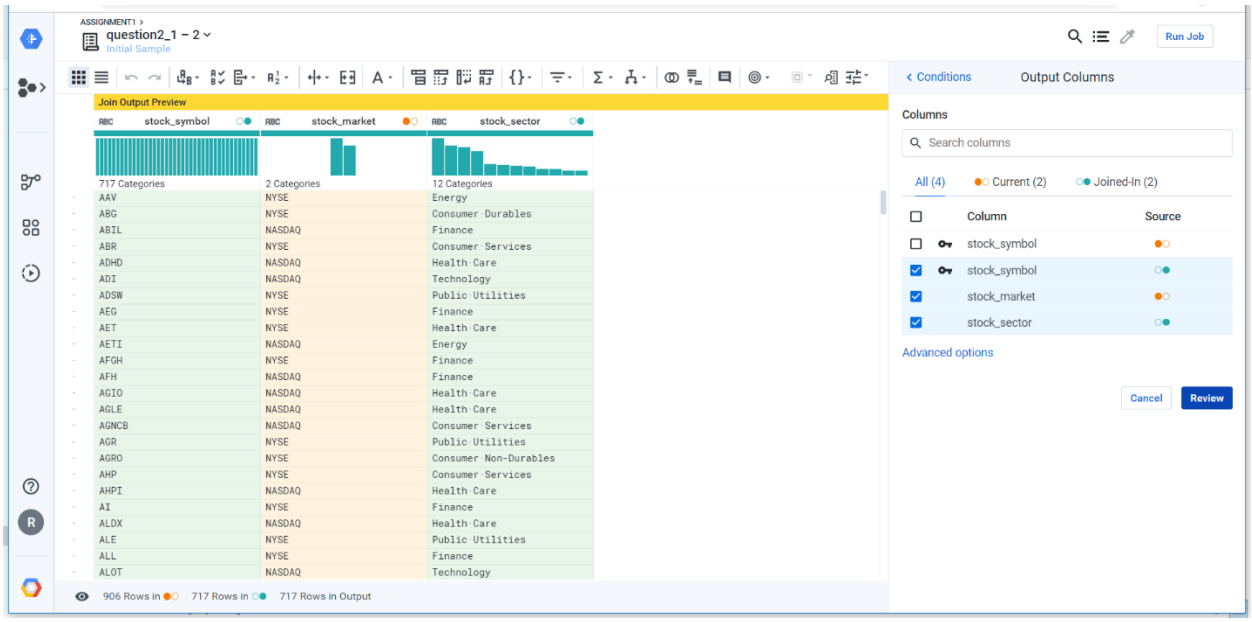
Here is the snapshot to join newly created recipe to another dataset and question2_2 dataset is selected for join operation.

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

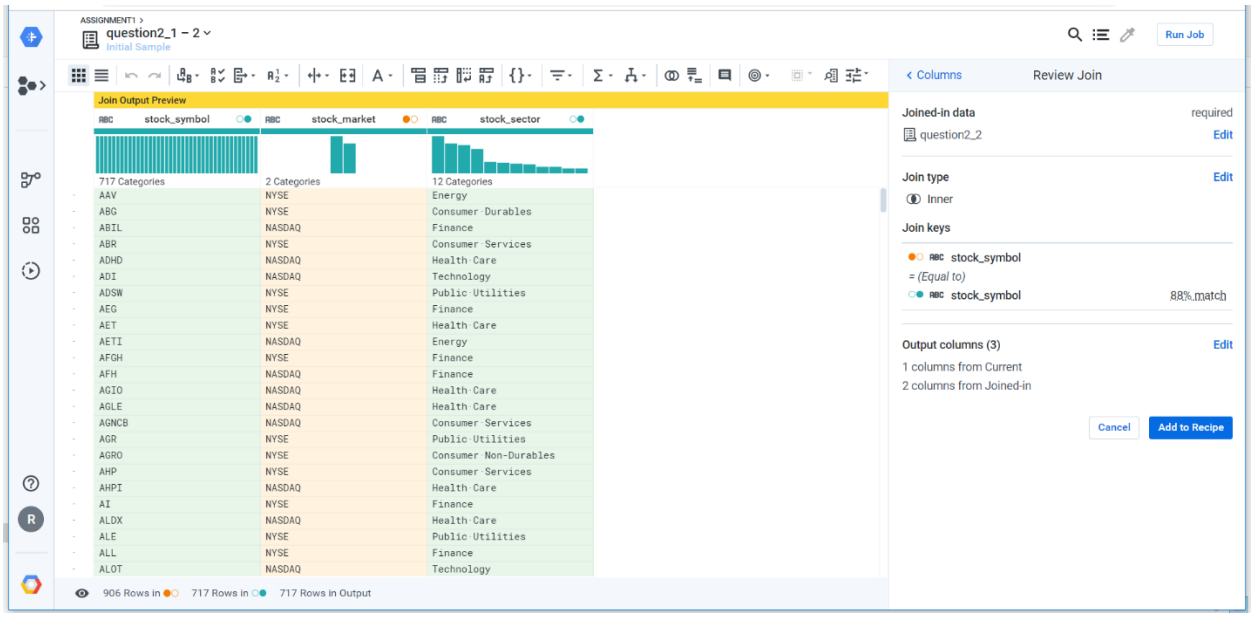
of the primary key column i.e. stock_symbol. Finally, click on ‘Add to Recipe’ to confirm execution of the join operation.



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

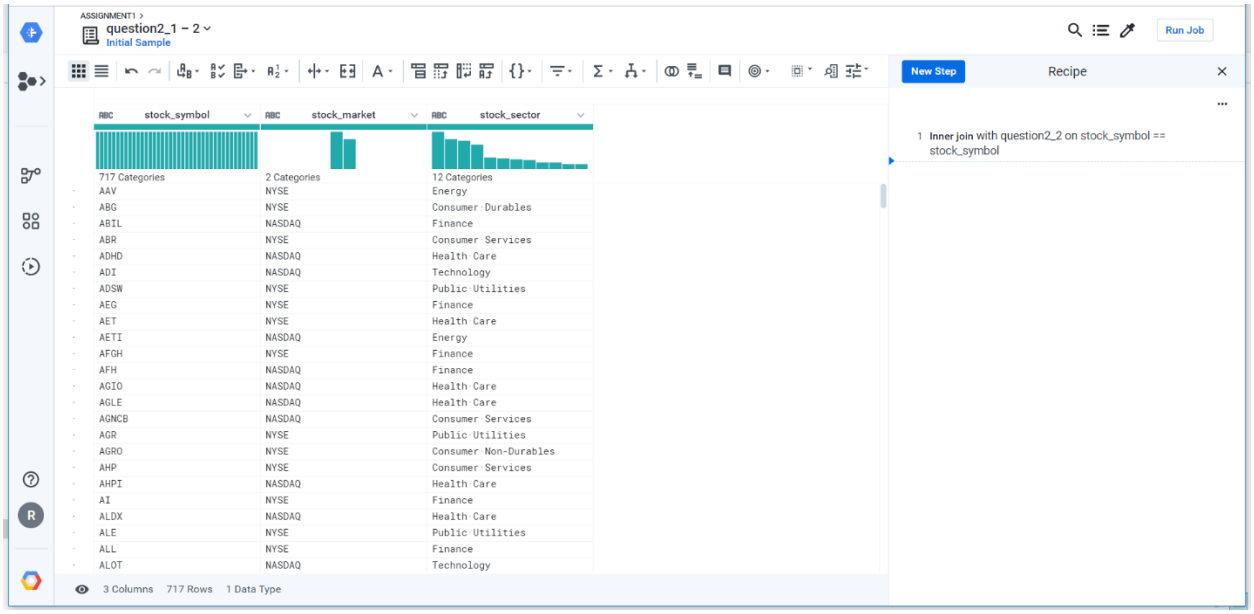


Here is the snapshot for output column where we specify only one of the common columns i.e. stock_symbol as a basis for join operation.



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 question 2_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:

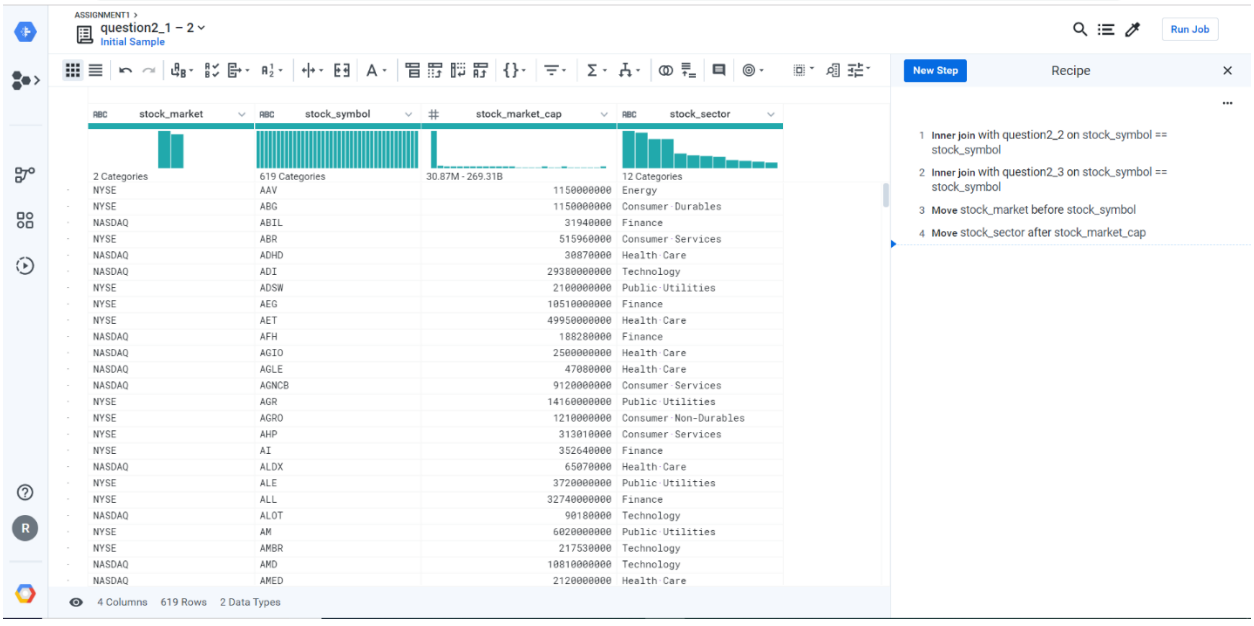
18. Again, select ‘question2_1-2’ dataset and then select ‘Edit Recipe’.

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.

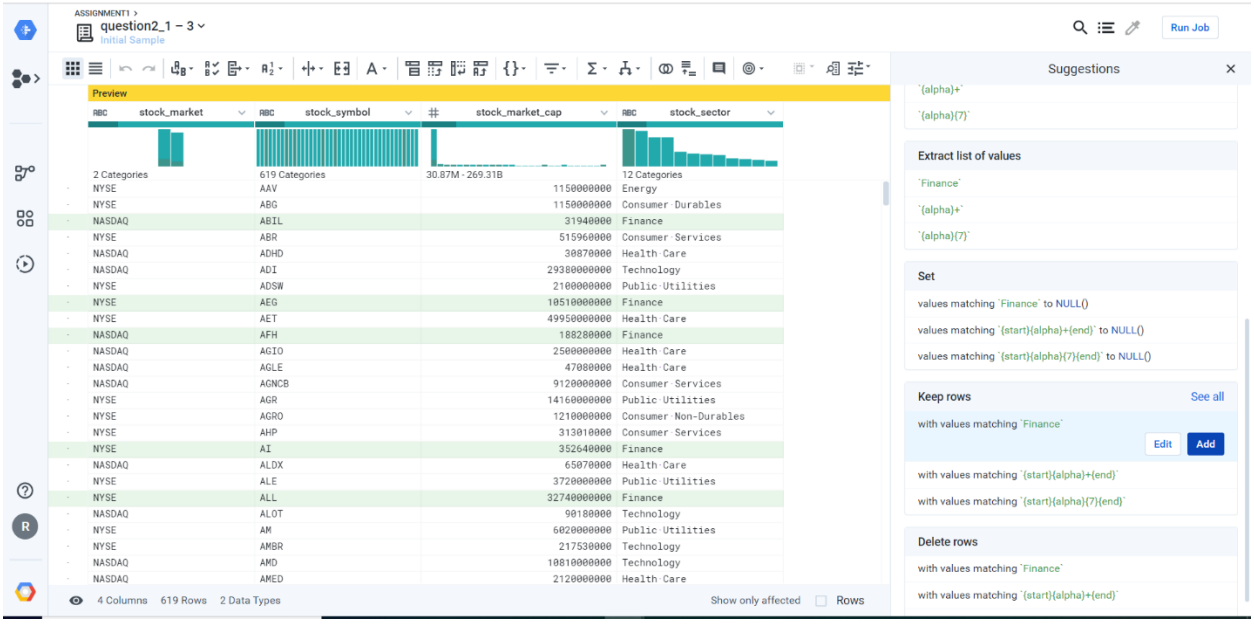
Now, the next challenge is to extract the datasets according to output dataset requirement in reference to stocks related to:

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'.

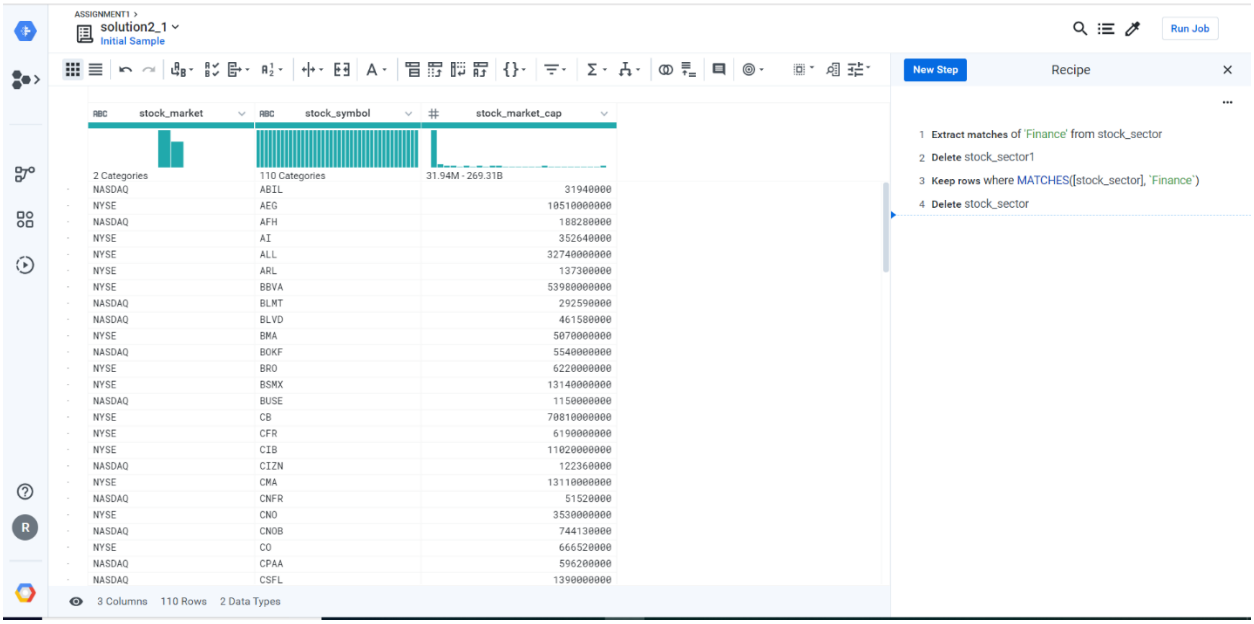
With this, it will keep all rows matching 'Finance' as stock_sector discarding rows other than 'Finance' sector.



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.

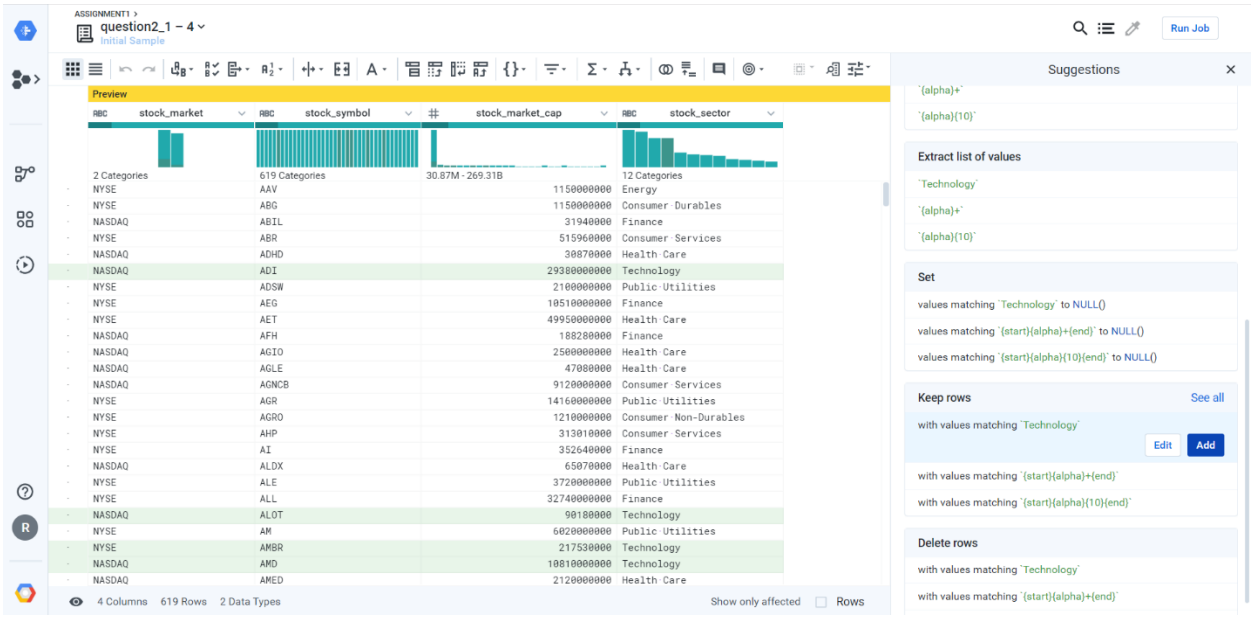


Here is the snapshot after removing filtered rows from 'Finance' stock_sector which is our final output dataset 1(solution2_1) containing 110 records of data.

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'.

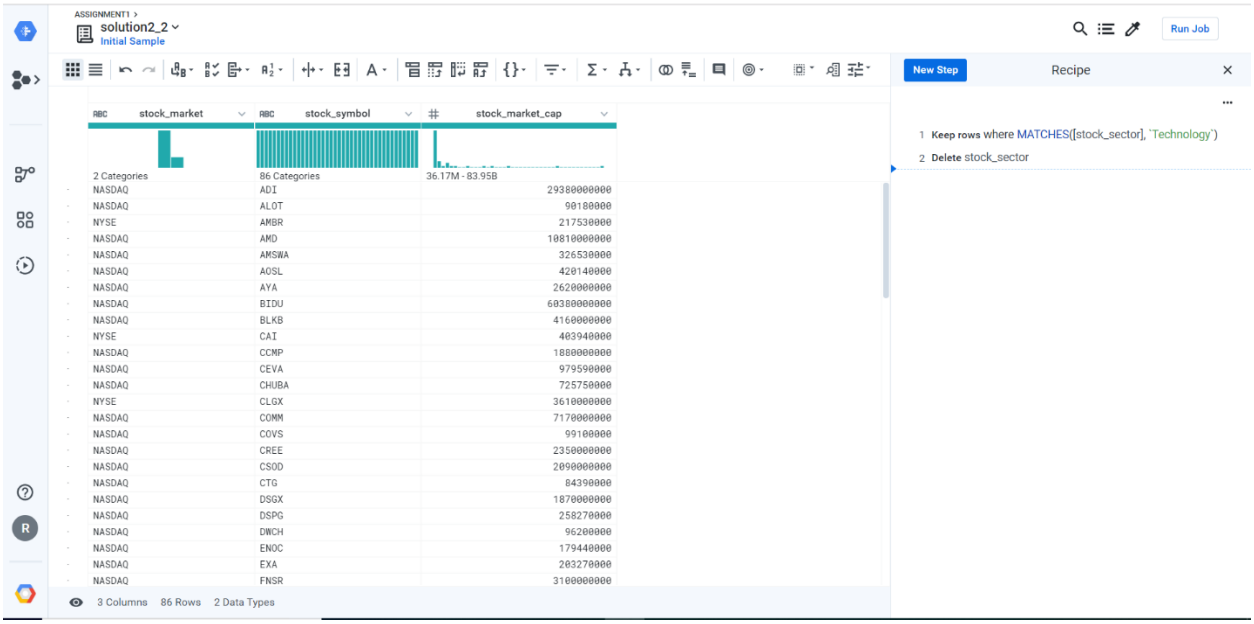
With this, it will keep all rows matching 'Technology' as stock_sector discarding rows other than 'Technology' sector.



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.

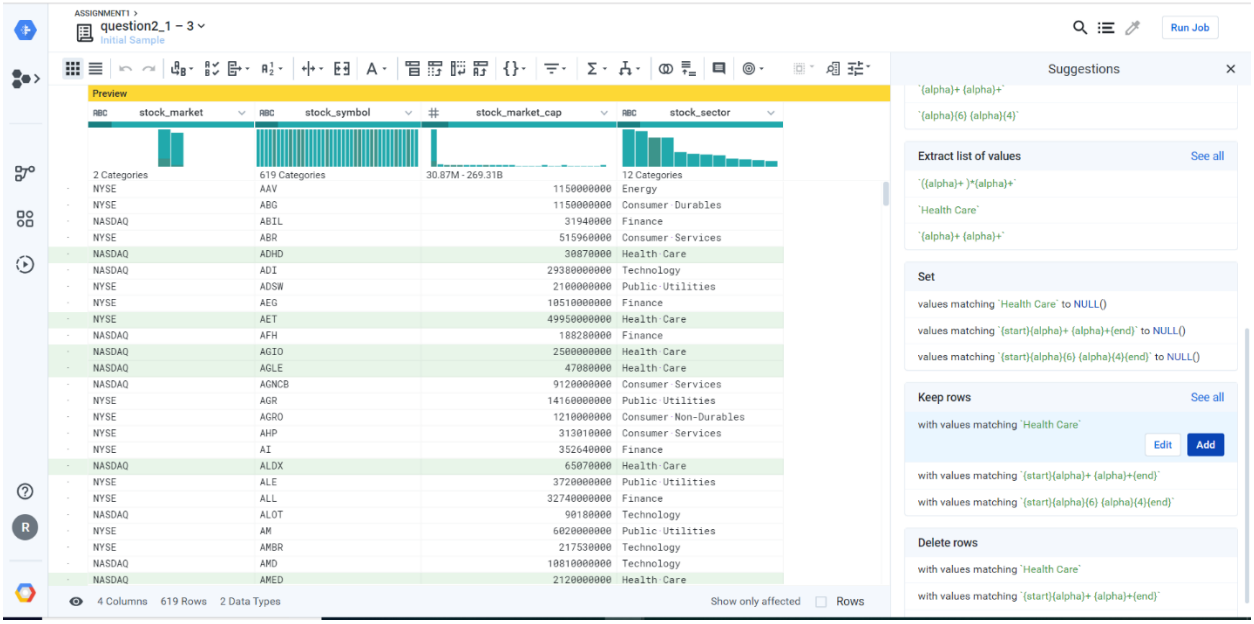


Here is the snapshot after removing filtered rows from 'Technology' stock_sector which is our final output dataset 2(solution2_2) containing 86 records of data.

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'.

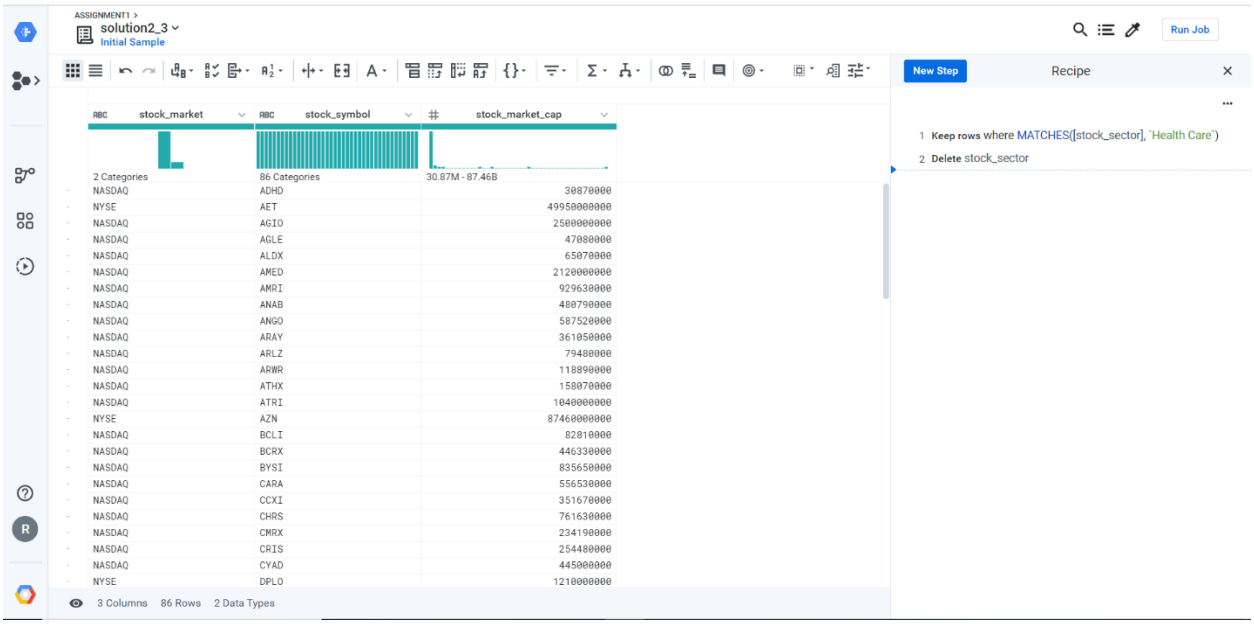
With this, it will keep all rows matching 'Health Care' as stock_sector discarding rows other than 'Health Care' sector.



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.

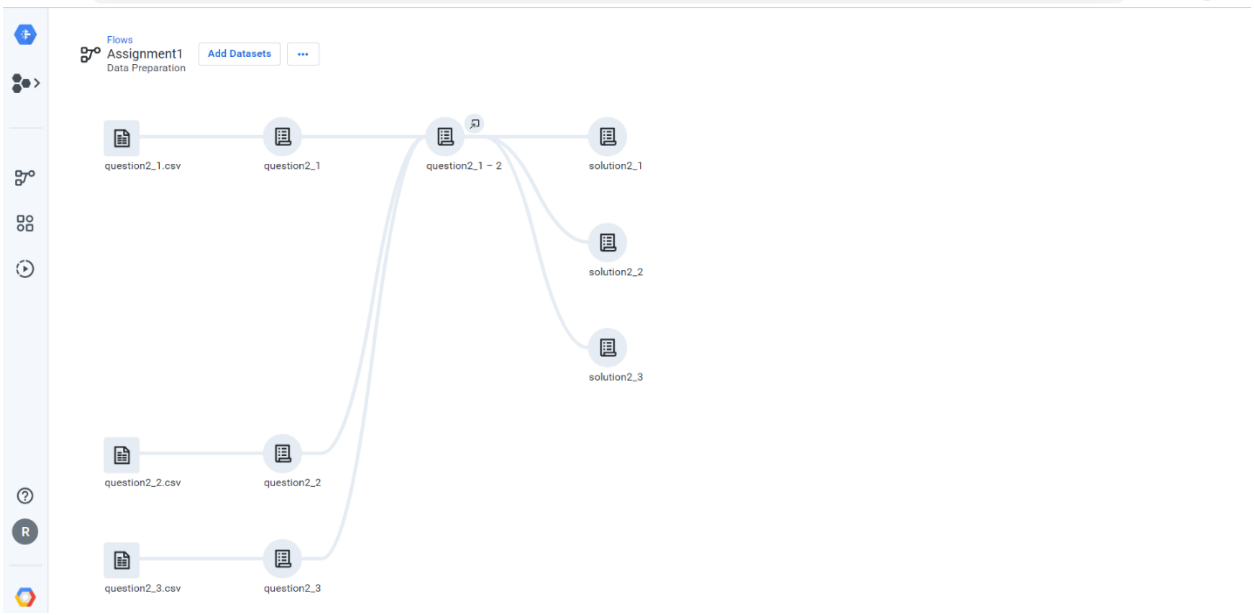
29. Then, delete the 'stock_sector' column.

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.



Here is the snapshot after removing filtered rows from 'Health Sector' stock_sector which is our final output dataset 3(solution2_3) containg 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 our 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:

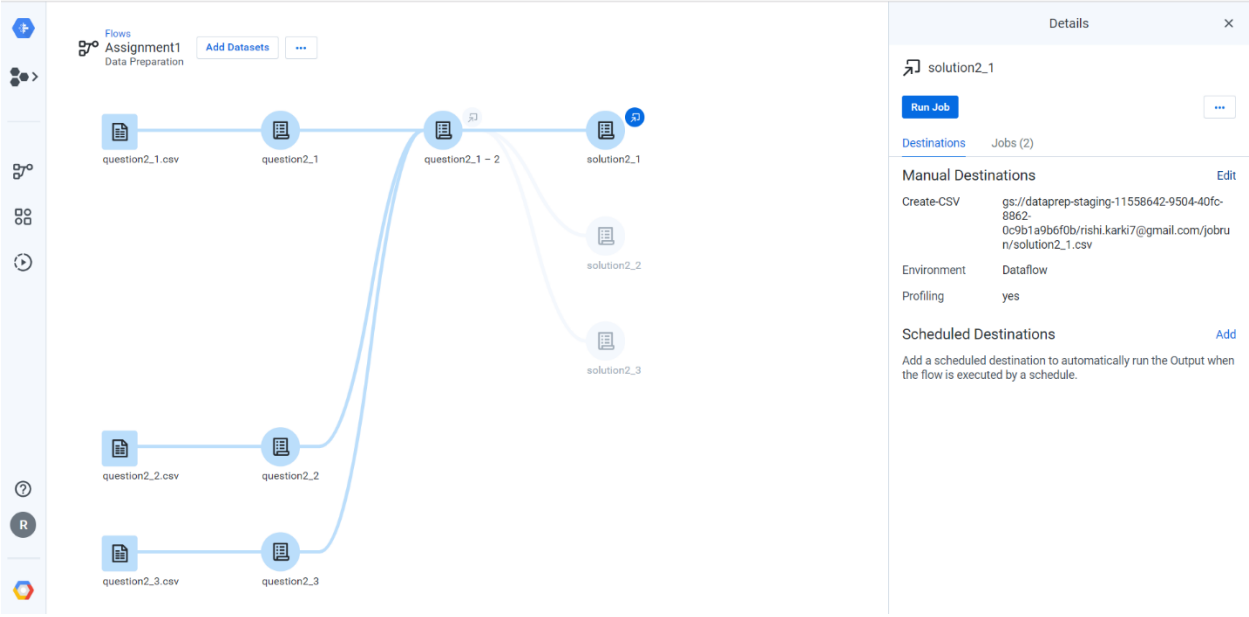


Here is the snapshot of the final flow of data preparation, which clearly previews our final output datasets.

Now, the next challegne 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 preparataion.

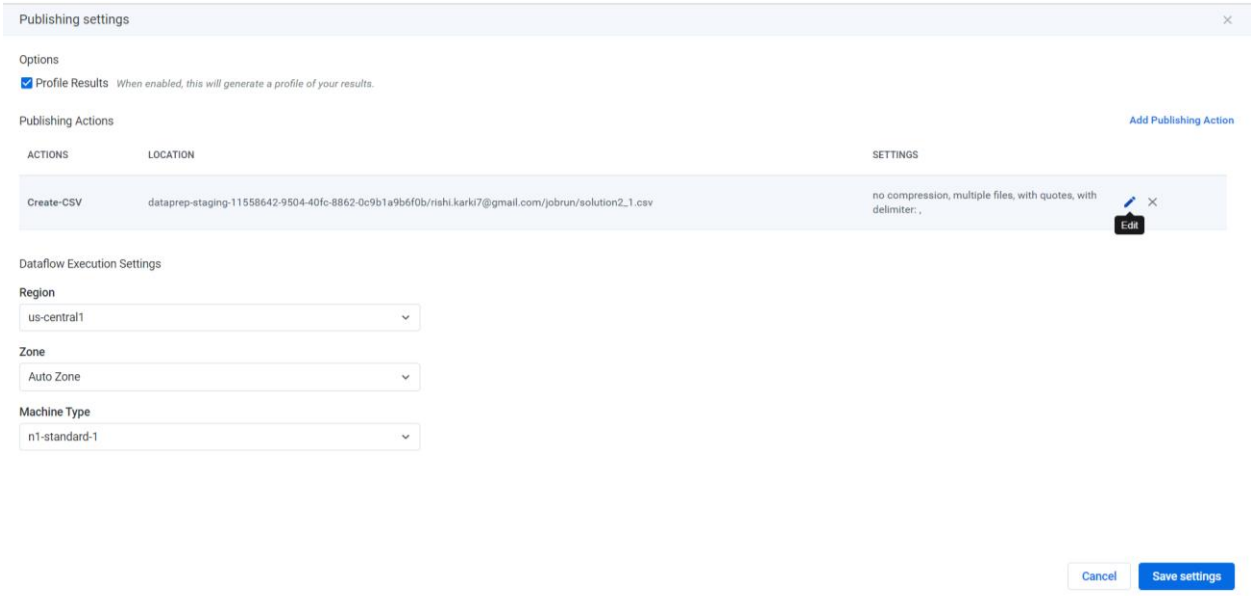
The process of generating final output dataset 'solution2_1.csv' is shortly described below:

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 datset 'solution2_1.csv'

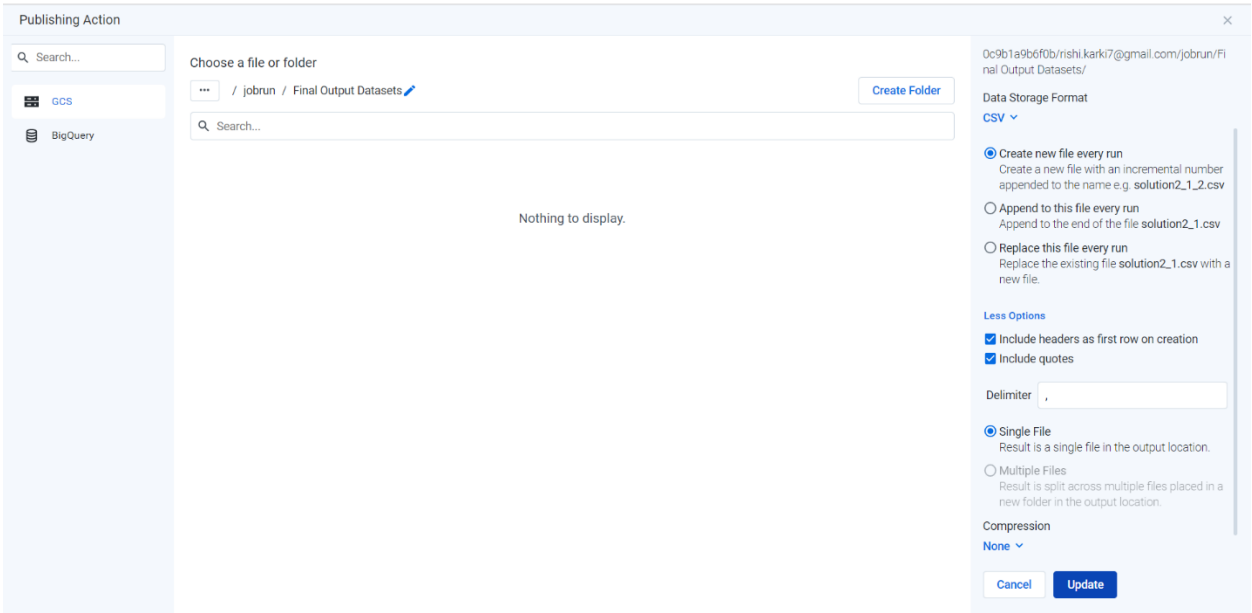
31. Under 'Destination' tab, Click on 'Edit' menu just left to 'Manual Destinations'. Select 'Edit' option in setting tab.



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

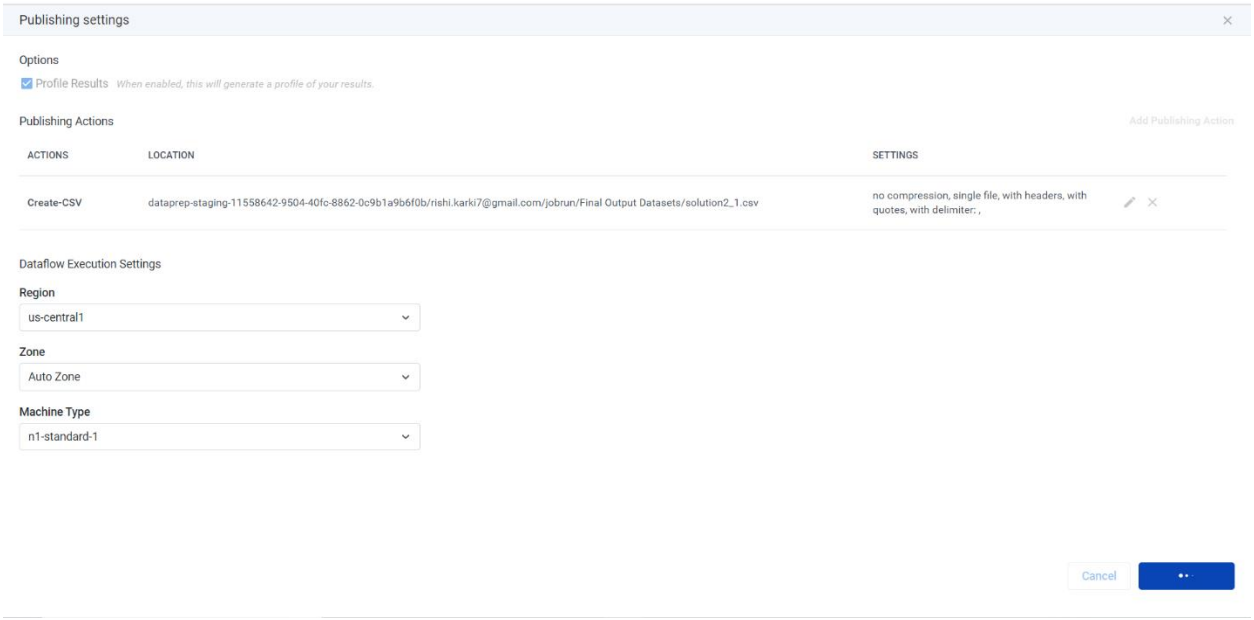
32. In this page, configure the necessary setting and make sure to check 'Include headers as first row on creation' option under 'More Option' menu. Then, click on 'Update' to save settings.

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.

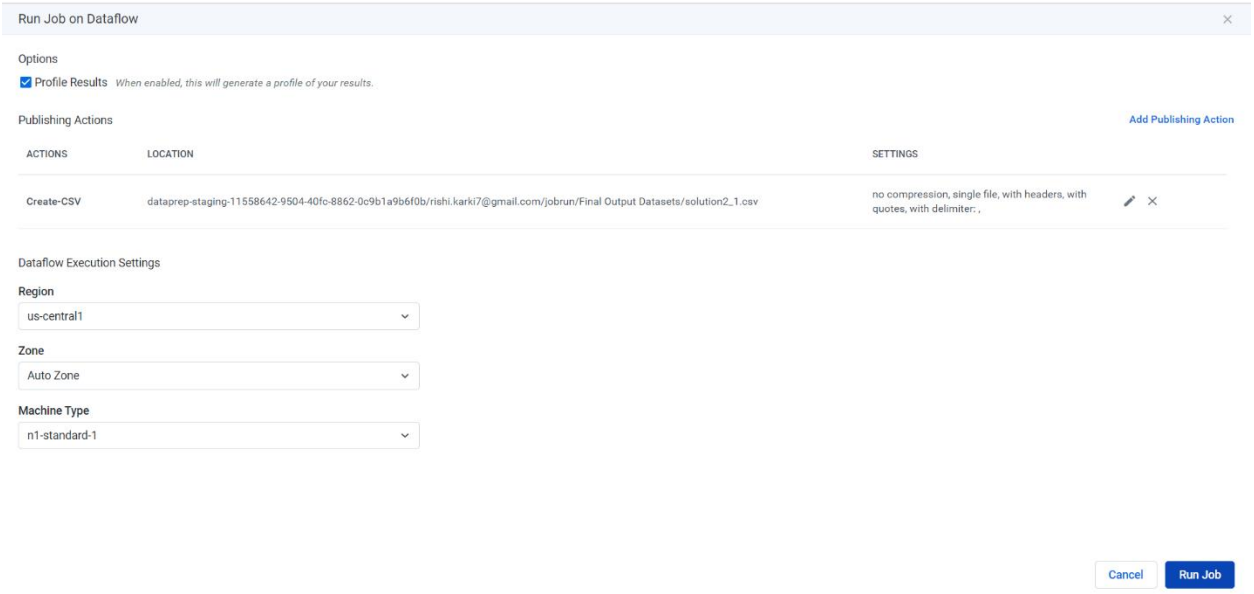
33. Then, Click on Save settings to confirm the previous step settings configuration.



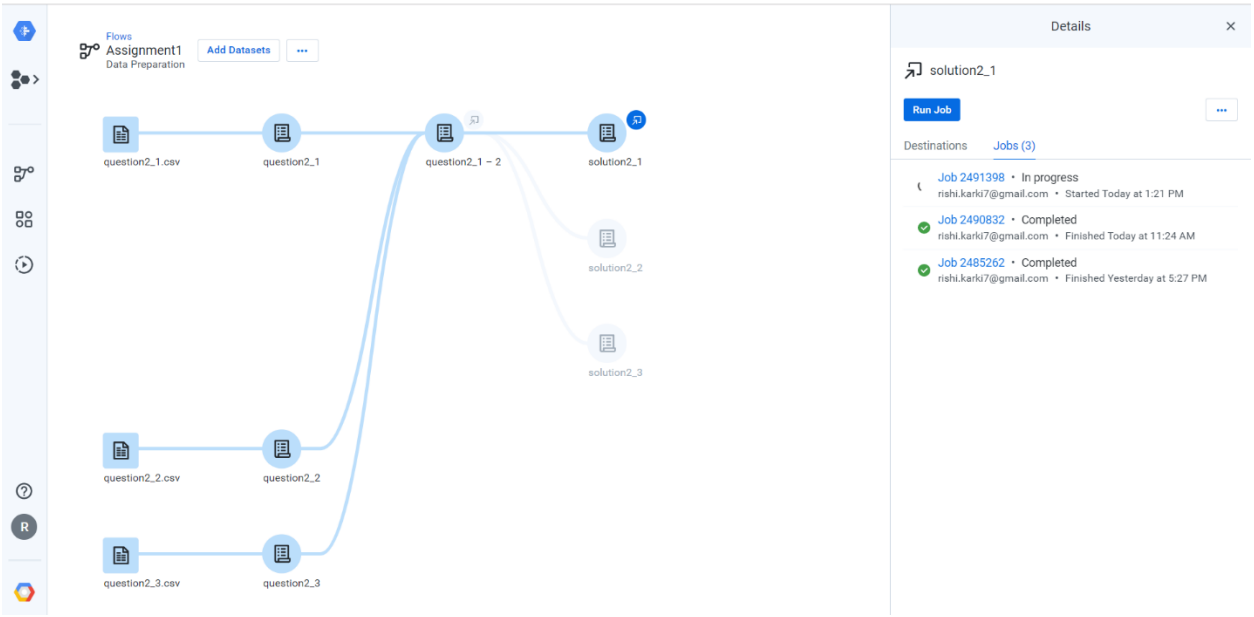
Here is the snapshot to confirm the previous step settings configuration.

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.

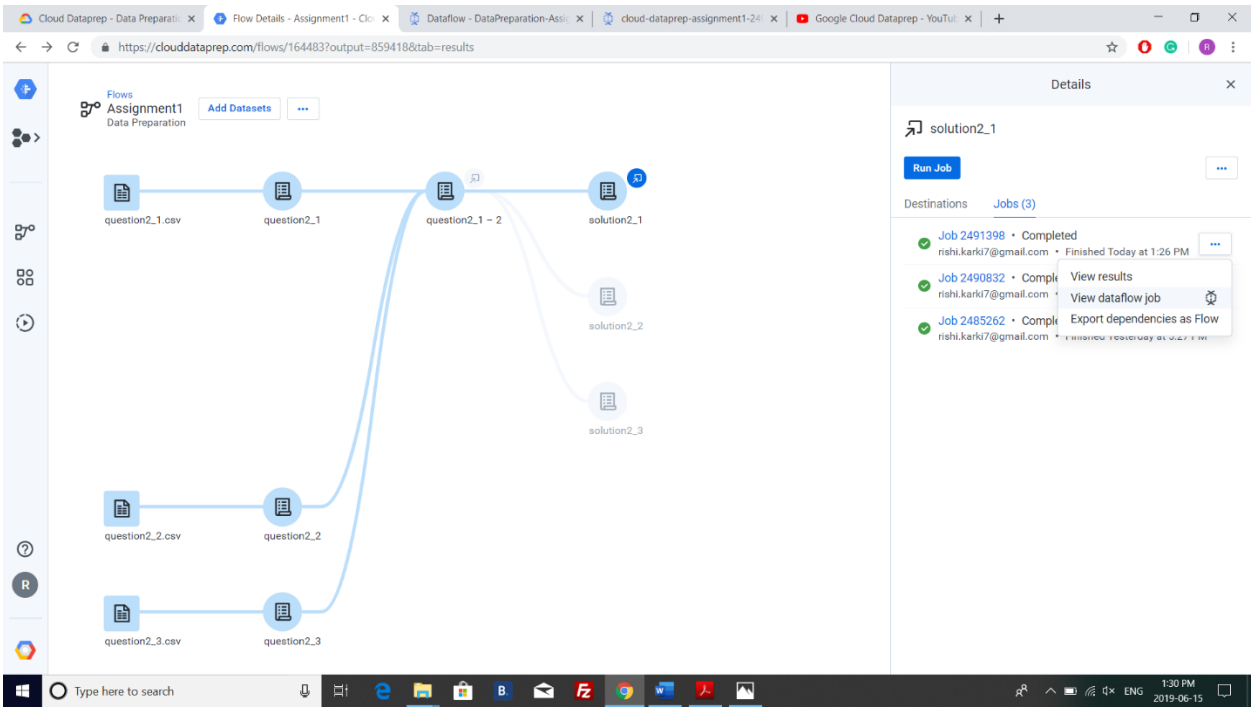


Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

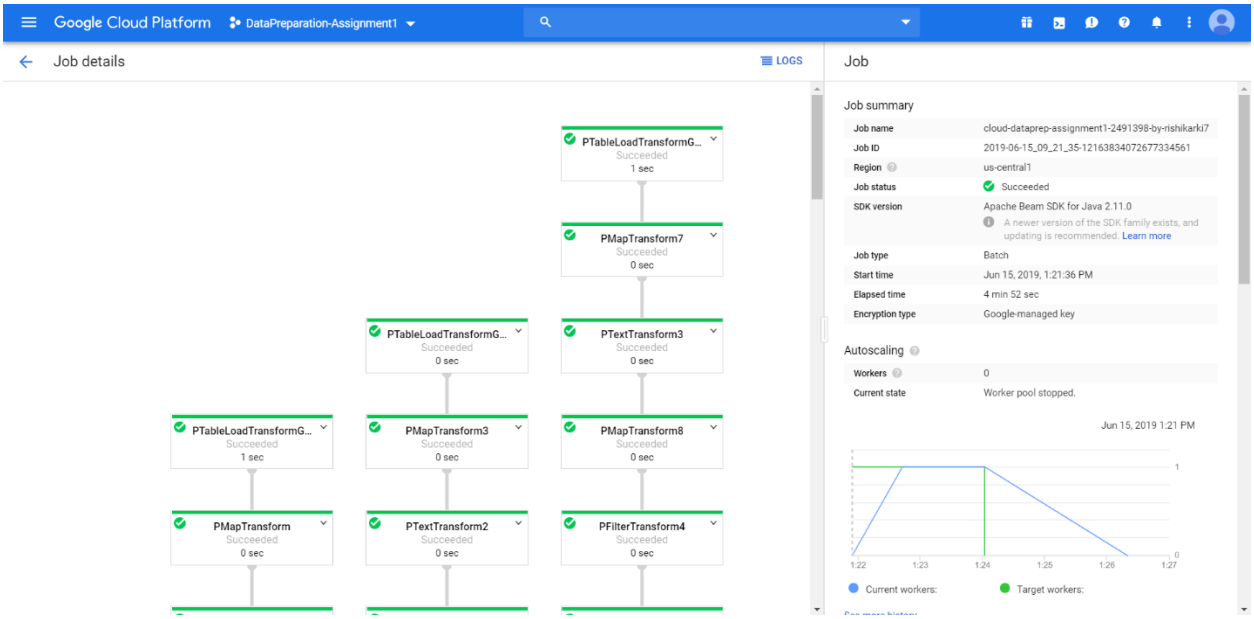


Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

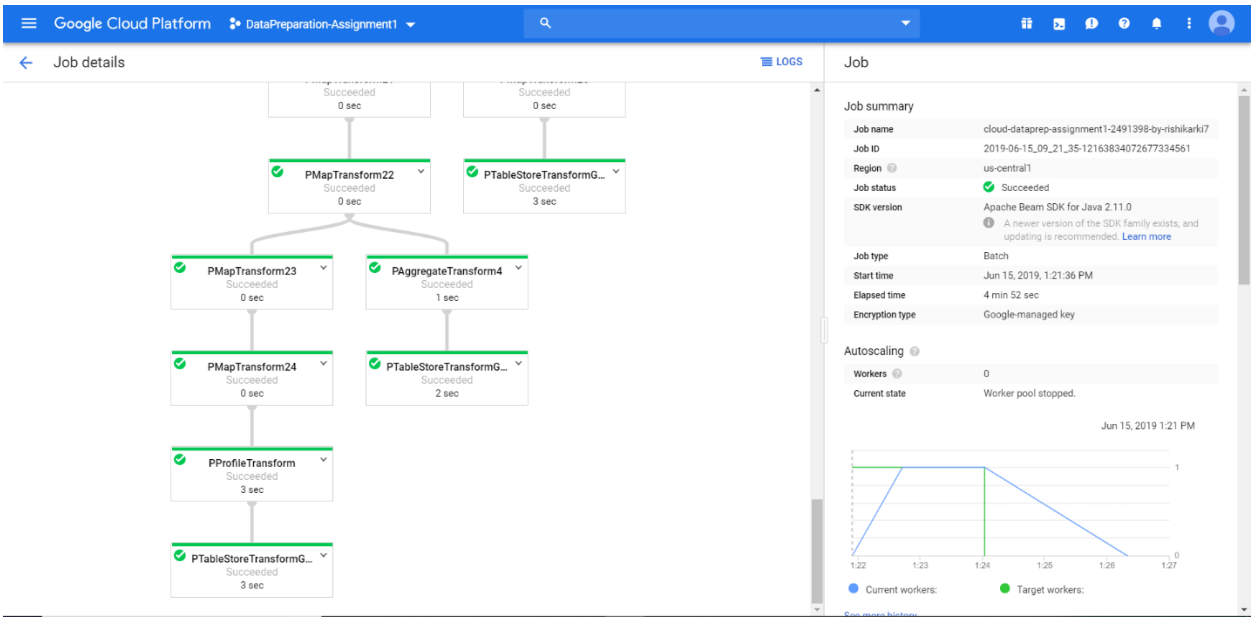
35. Once the 'Run Job' function gets completed, you can view the dataflow job for your output dataset 'solution 2_1' by clicking on 'View dataflow job' from the drop down menu just right to it.



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



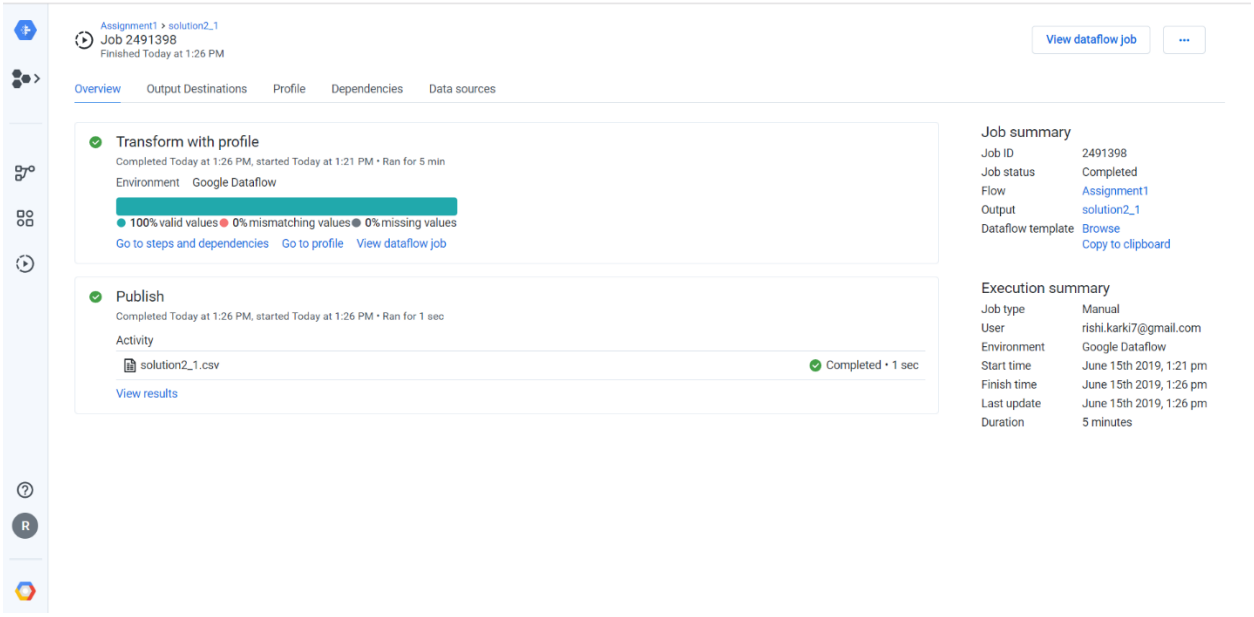
Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_1.



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

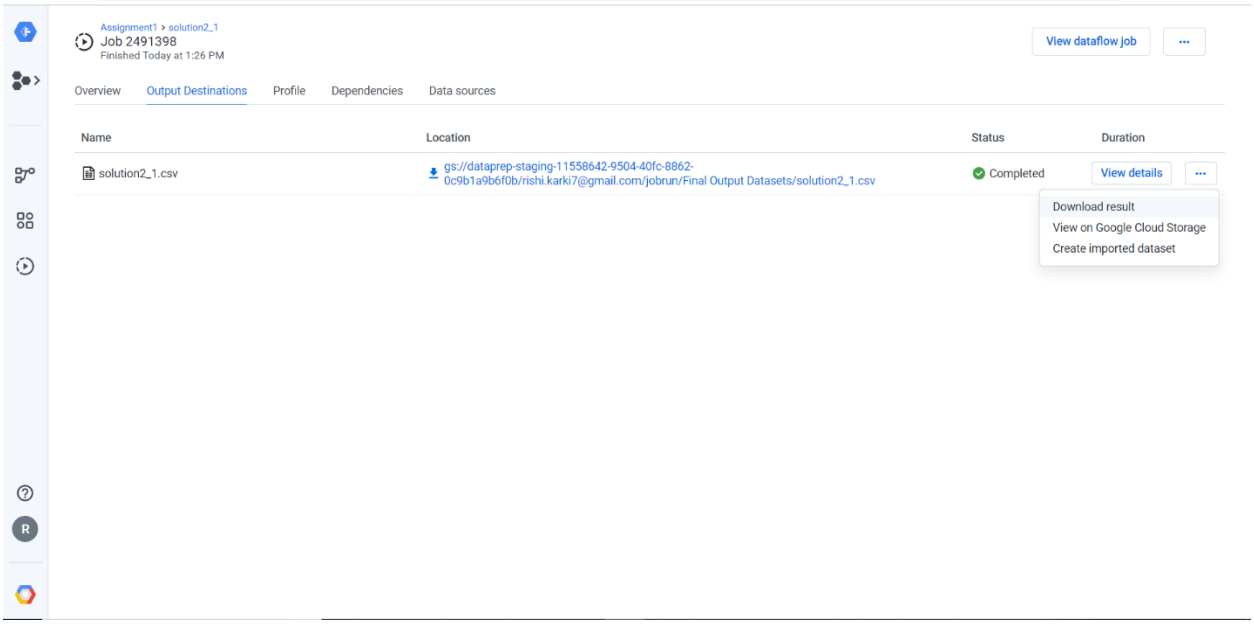
36. Now, to view the results, just click on 'View results' option from drop down menu that is just at top right.

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



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.



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

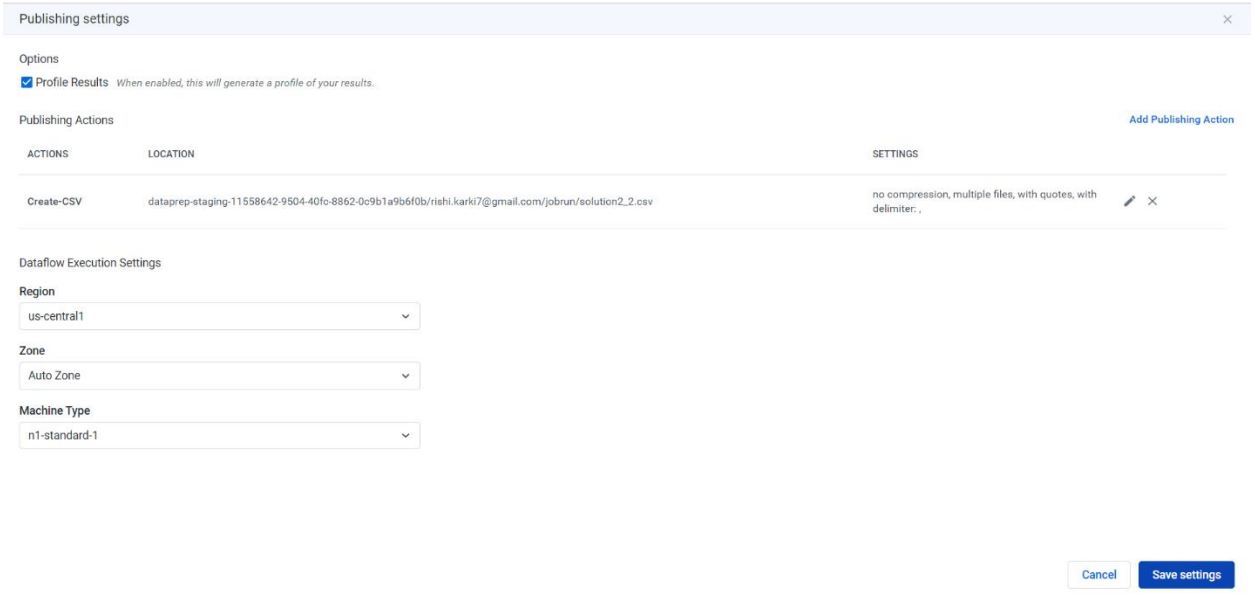
Now, we just download the final output dataset ‘solution2_1.csv’ in our local PC directory by clicking on ‘Download result’ option.

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 datset ‘solution2_2.csv’

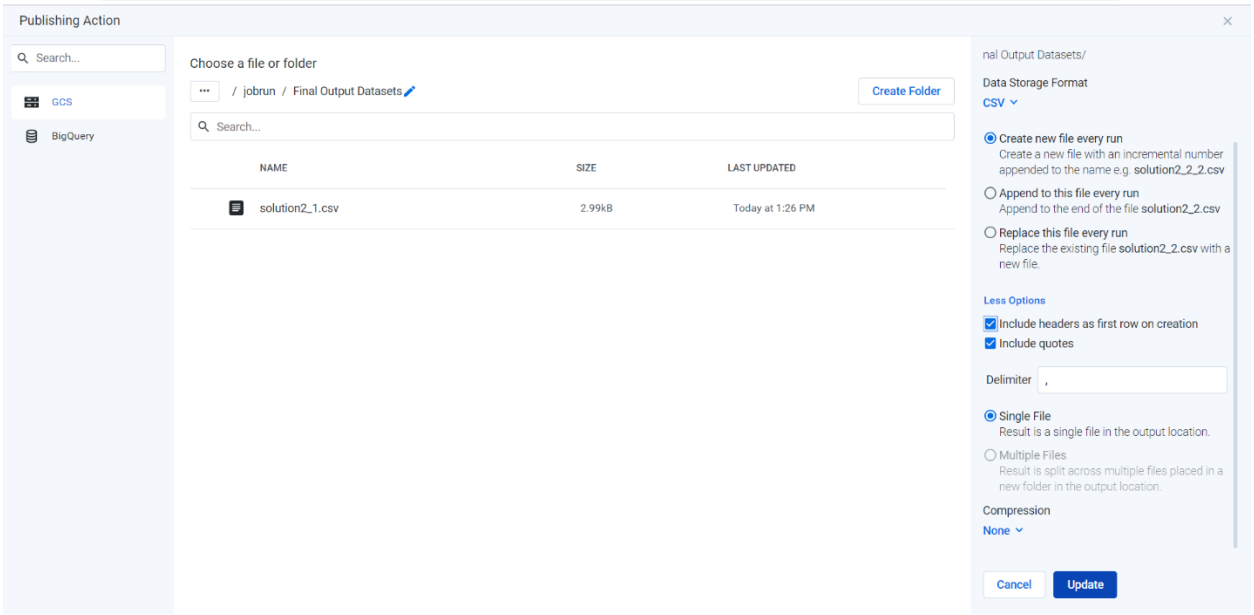
39. Under ‘Destination’ tab, Click on ‘Edit’ menu just left to ‘Manual Destinations’. Select ‘Edit’ option in setting tab.



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

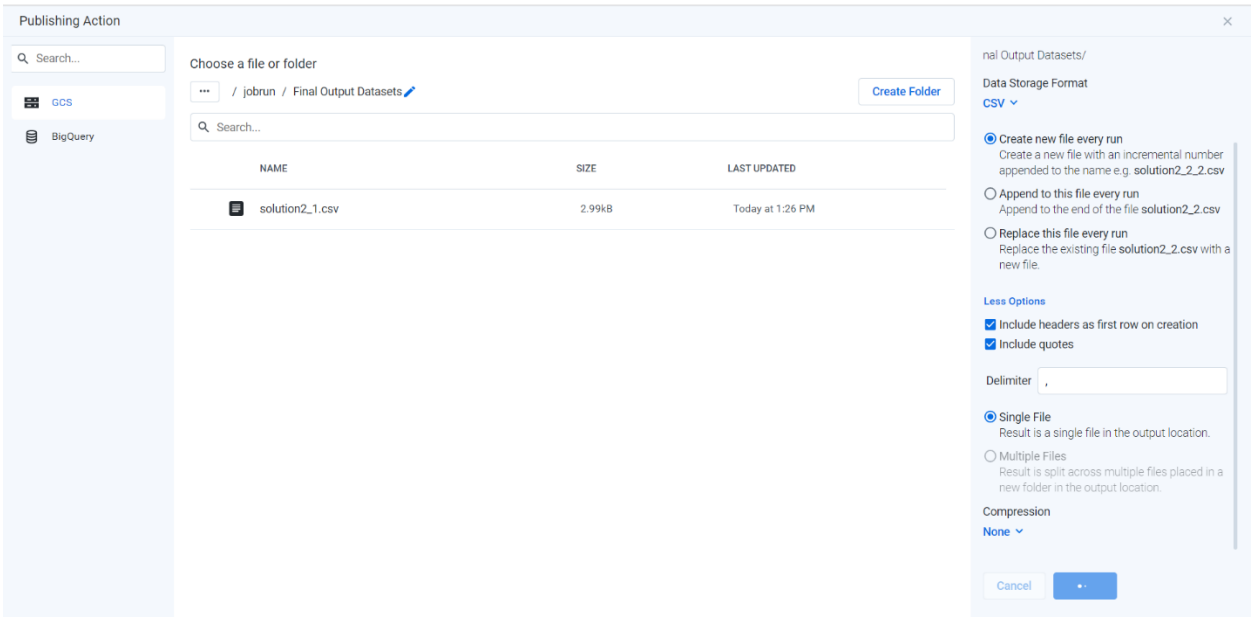
40. In this page, configure the necessary setting and make sure to check 'Include headers as first row on creation' option under 'More Option' menu. Then, click on 'Update' to save settings.

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



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

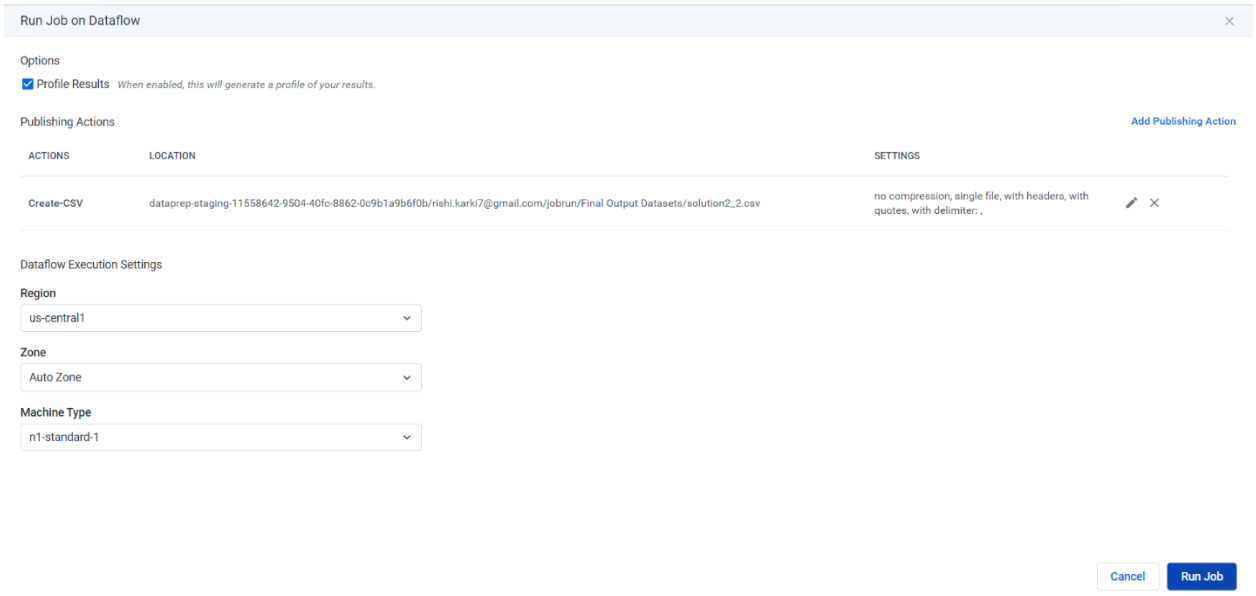
41. Then, Click on Save settings to confirm the previous step settings configuration.



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

42. 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.

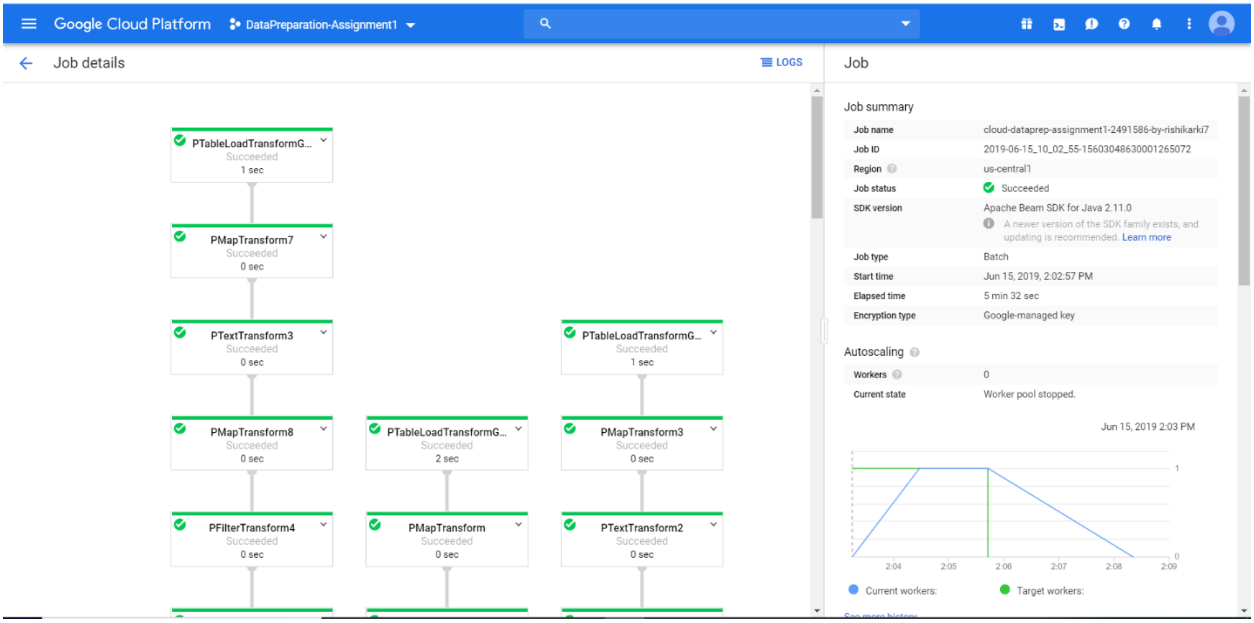


Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

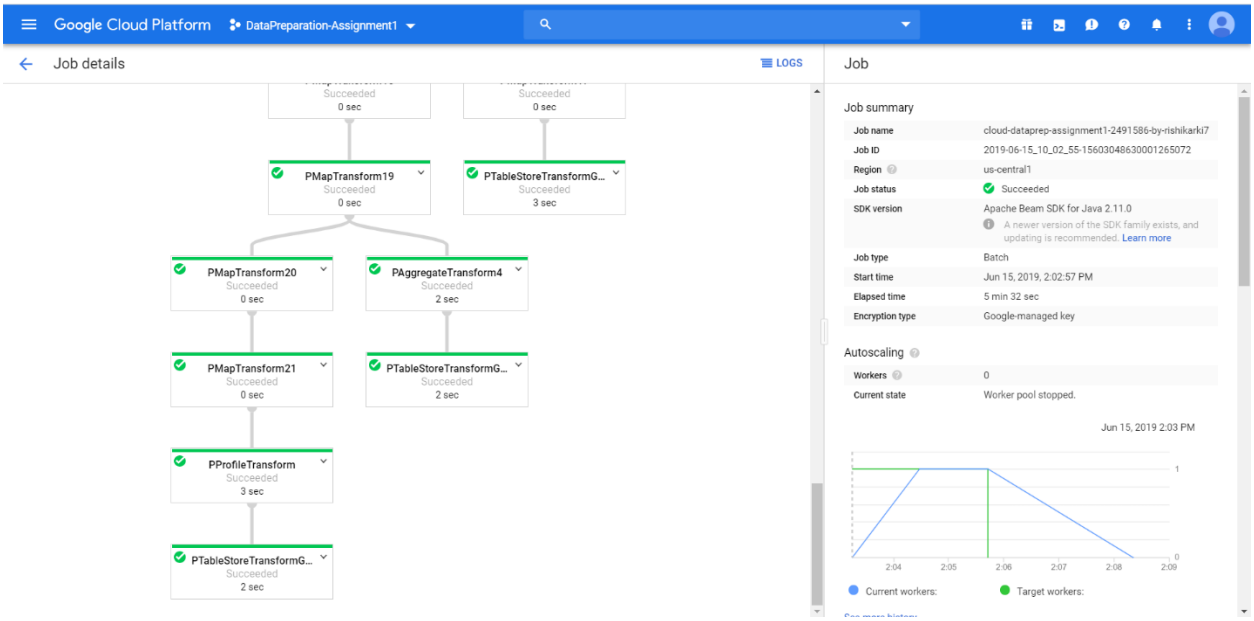
Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

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.



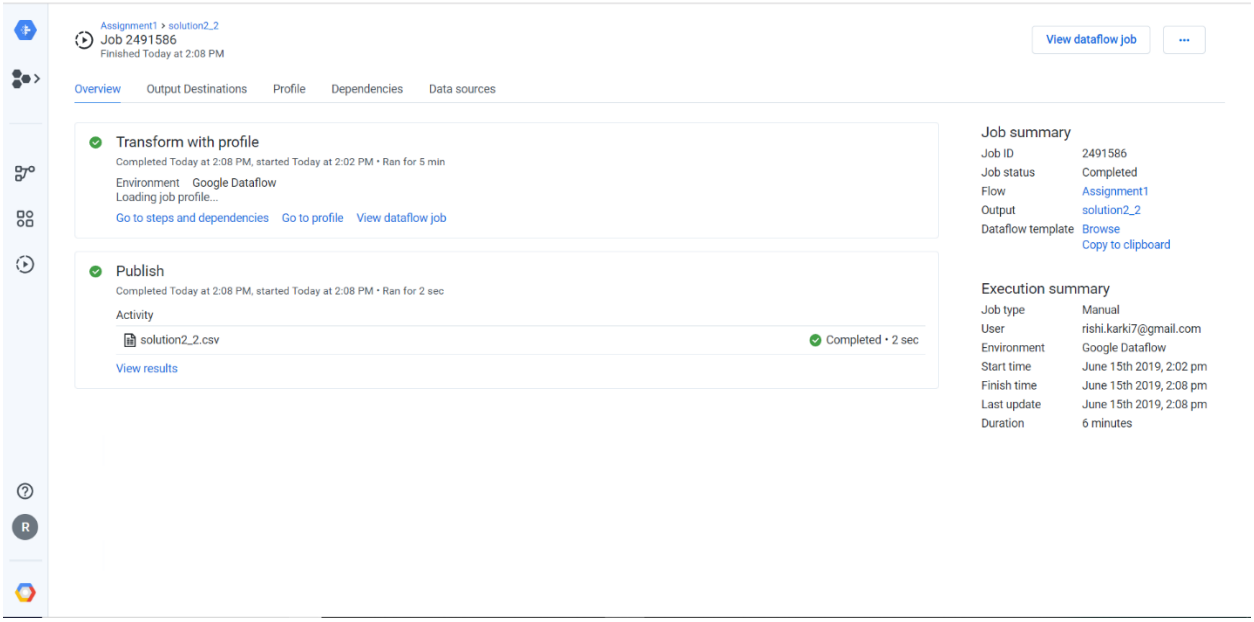
Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_2.



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

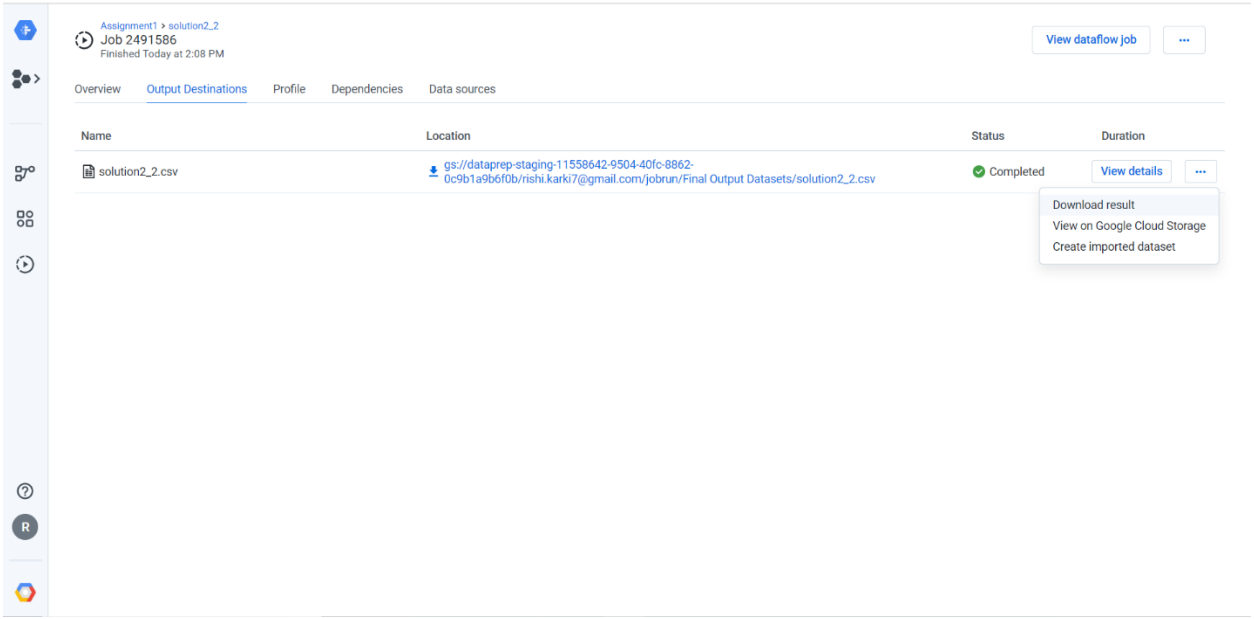
44. Now, to view the results, just click on 'View results' option from drop down menu that is just at top right.

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



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.



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

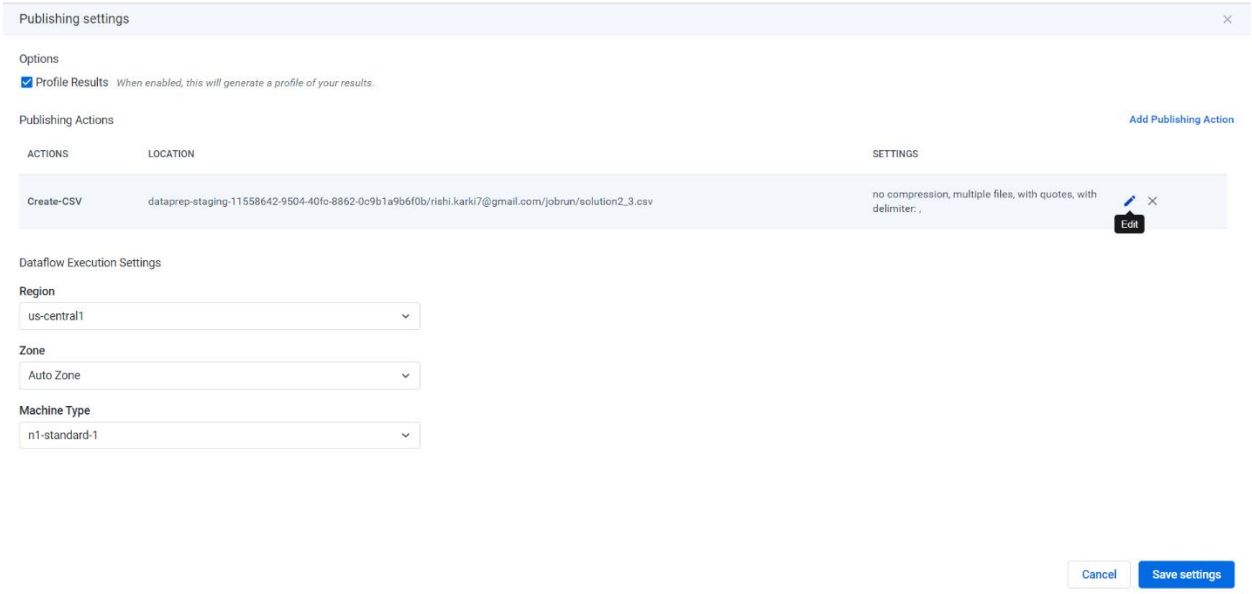
Now, we just download the final output dataset ‘solution2_2.csv’ in our local PC directory by clicking on ‘Download result’ option.

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 datset ‘solution2_3.csv’

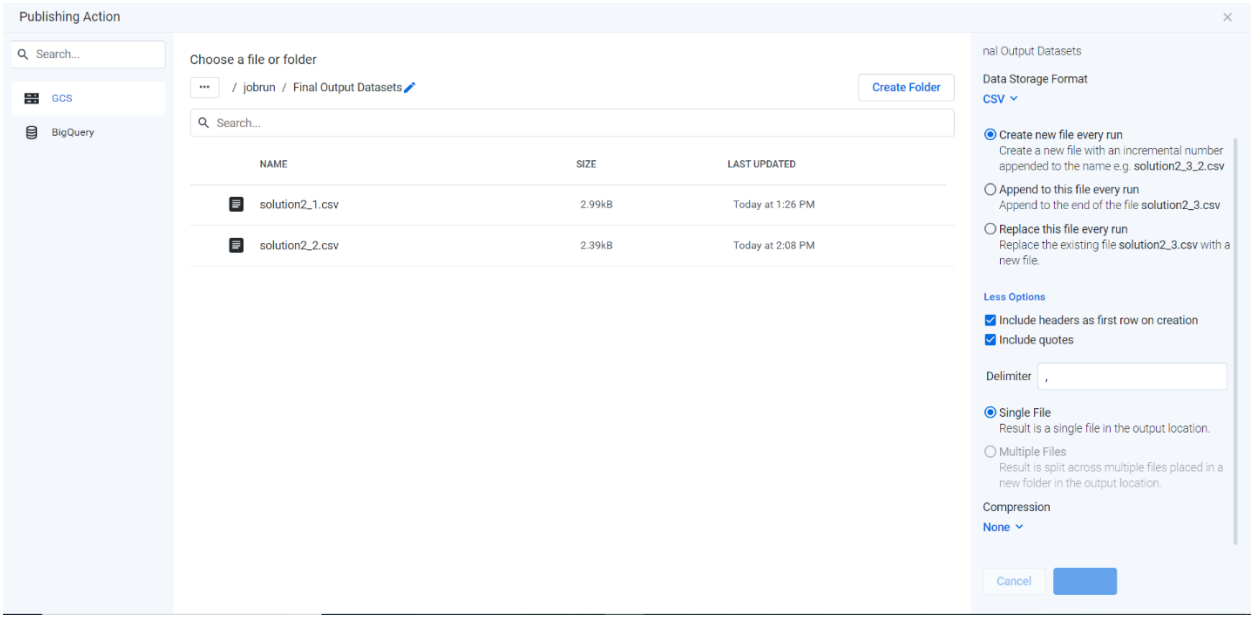
47. Under ‘Destination’ tab, Click on ‘Edit’ menu just left to ‘Manual Destinations’. Select ‘Edit’ option in setting tab.



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

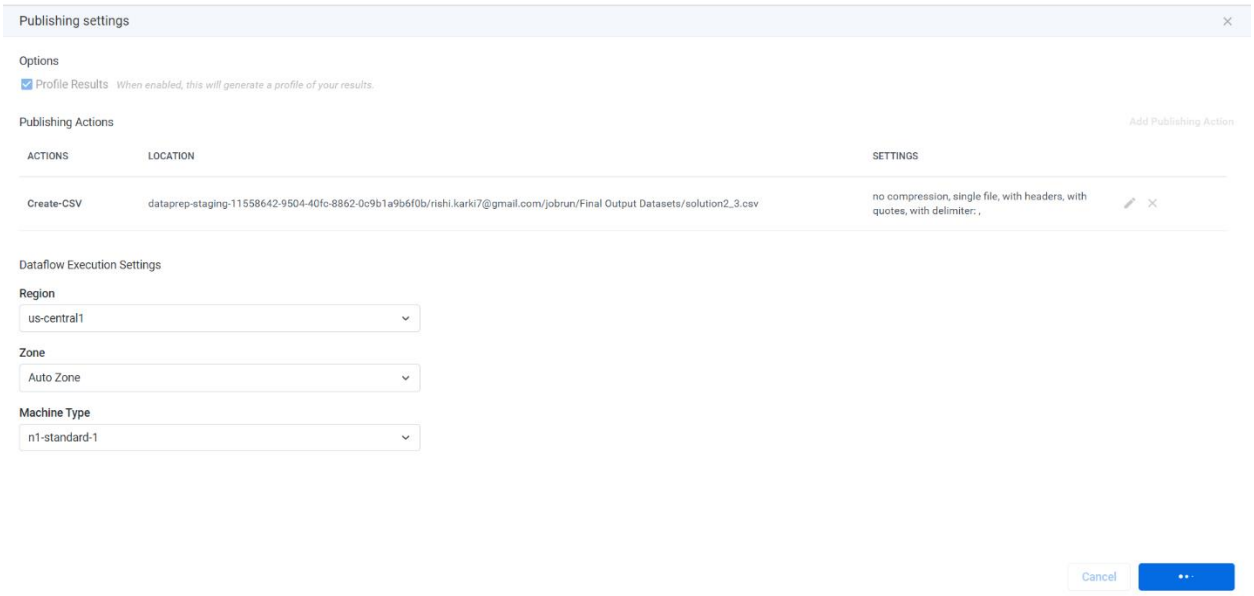
48. In this page, configure the necessary setting and make sure to check 'Include headers as first row on creation' option under 'More Option' menu. Then, click on 'Update' to save settings.

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_3.

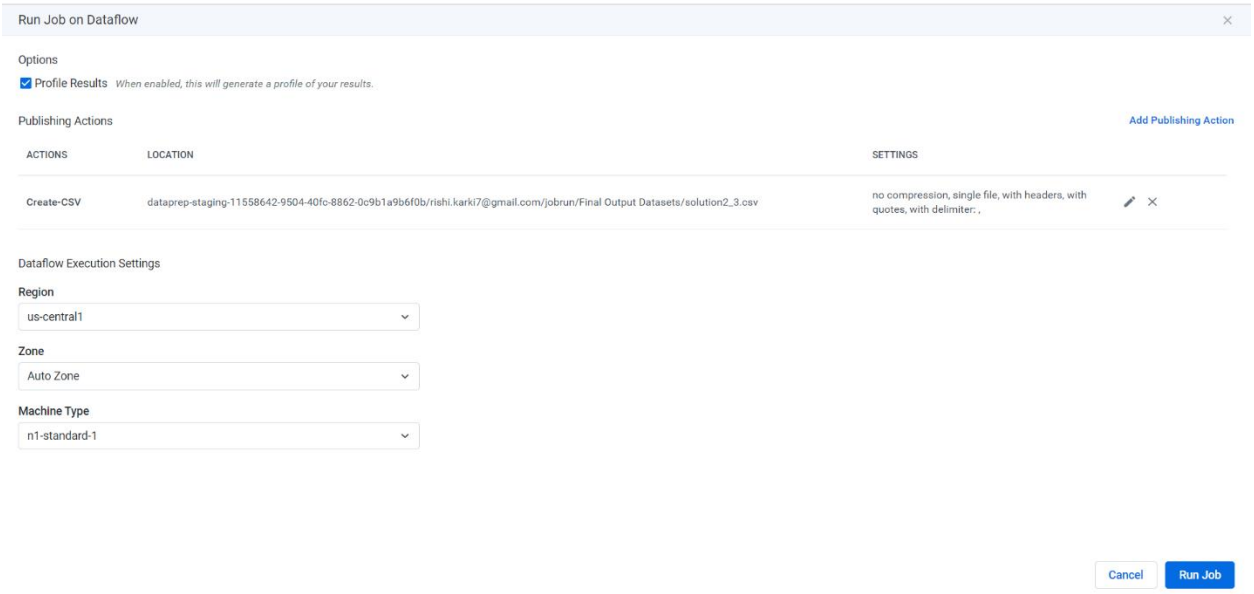
49. Then, Click on Save settings to confirm the previous step settings configuration.



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.

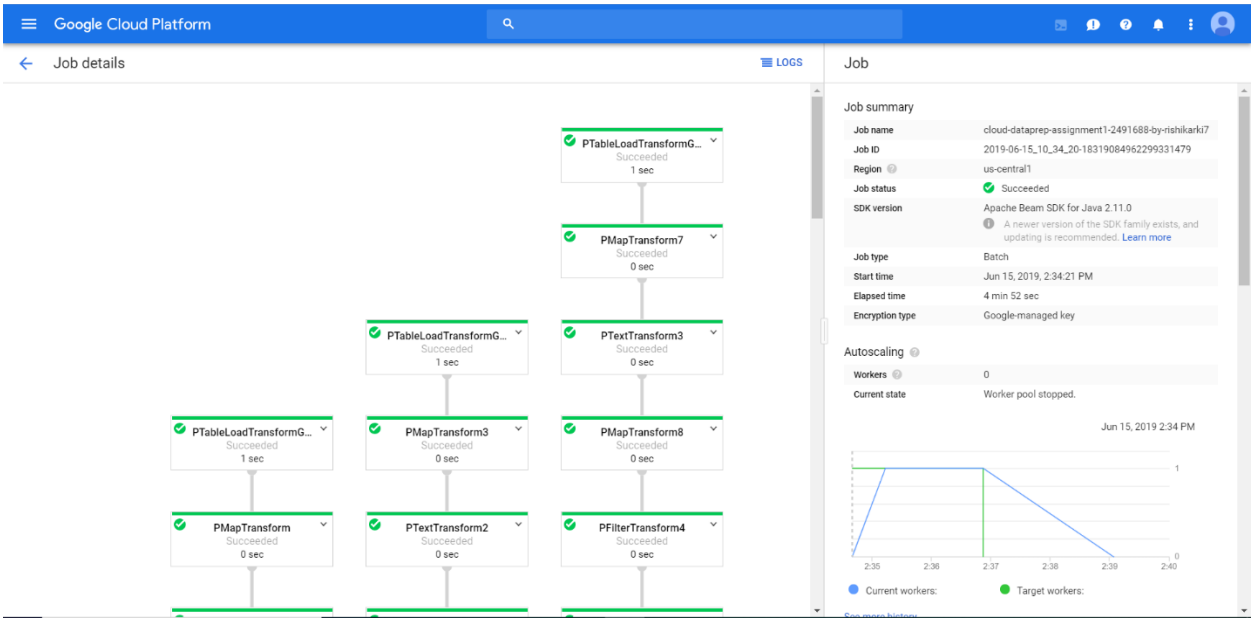


Here is the snapshot to illustrate the continuation step for the actual start of 'Run Job' function

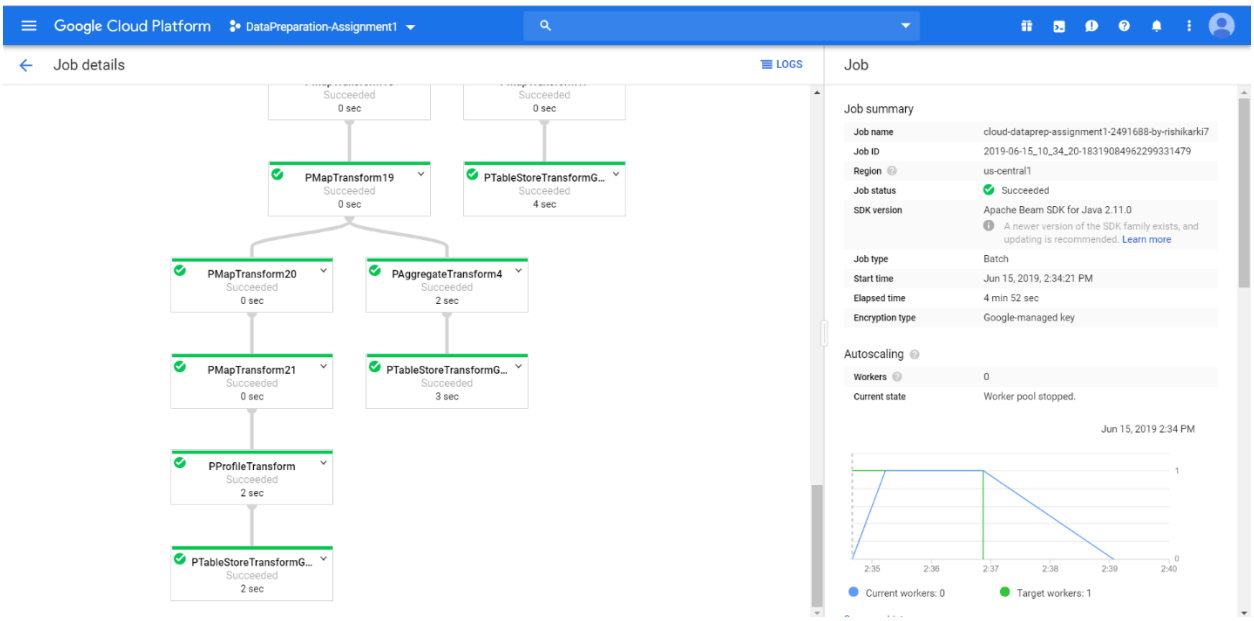
Here is the snapshot to illustrate the in progress of the start of 'Run Job' function

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.



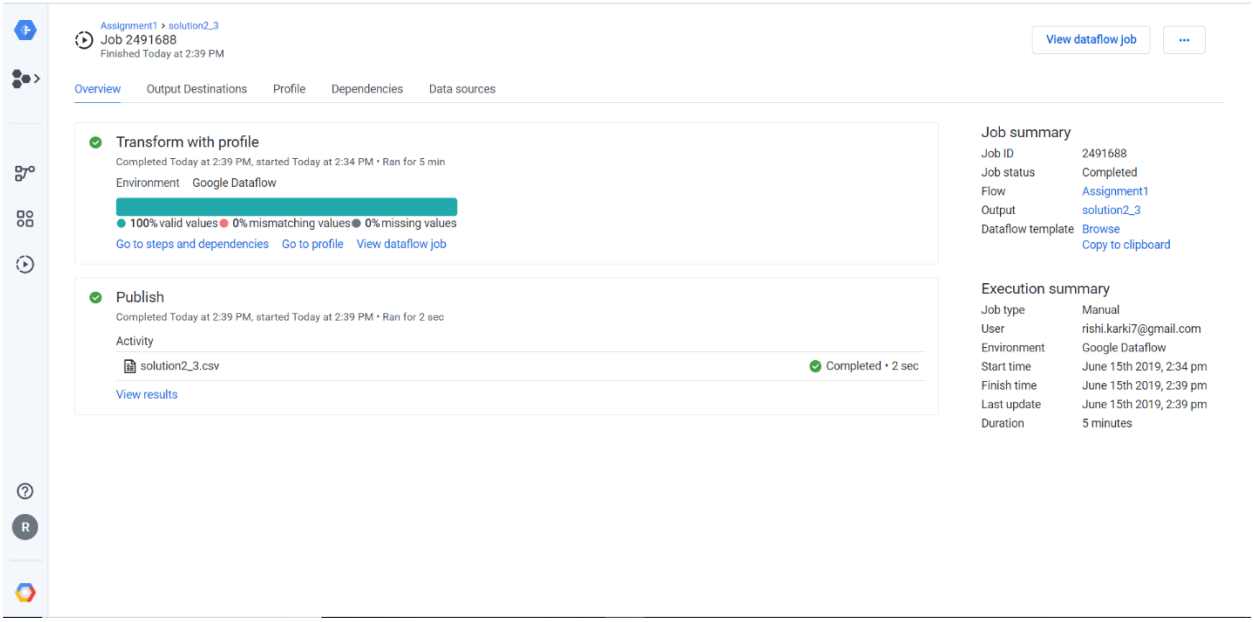
Here is the snapshot that illustrates the initial sample of dataflow job for output dataset solution2_3.



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

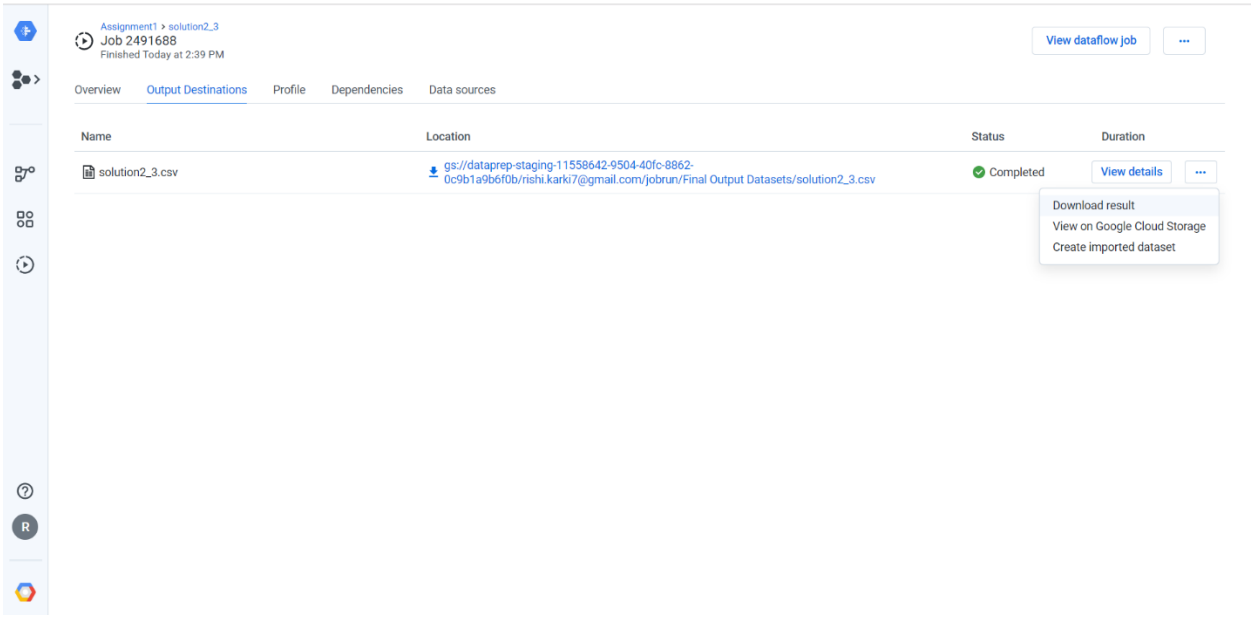
52. Now, to view the results, just click on 'View results' option from drop down menu that is just at top right.

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



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.



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 NewYork 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, 1015 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.

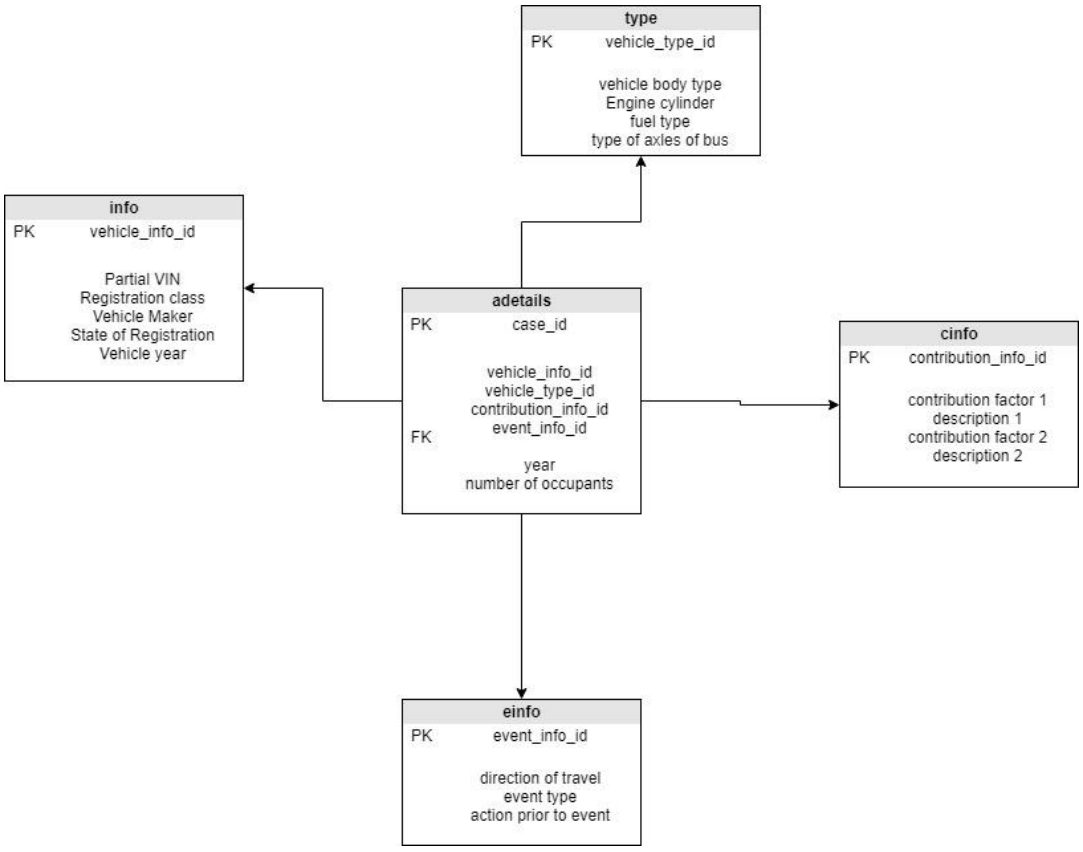


Fig: The Star schema design for data preprocessing

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.

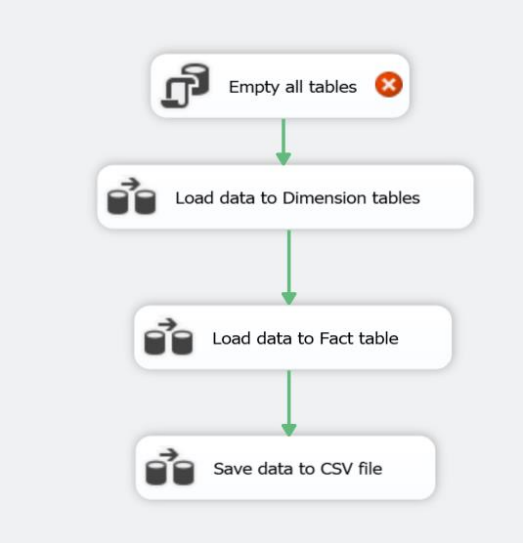


Fig: The ETL pipeline flow

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.

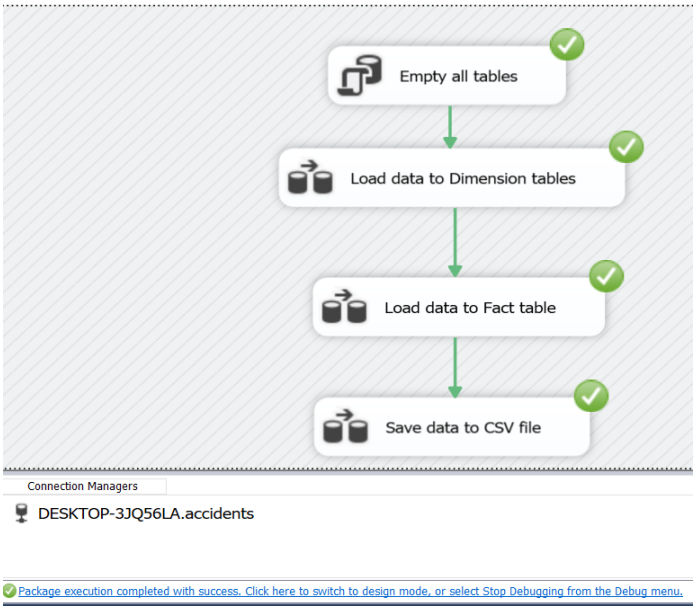


Fig: Successful execution of truncation

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.

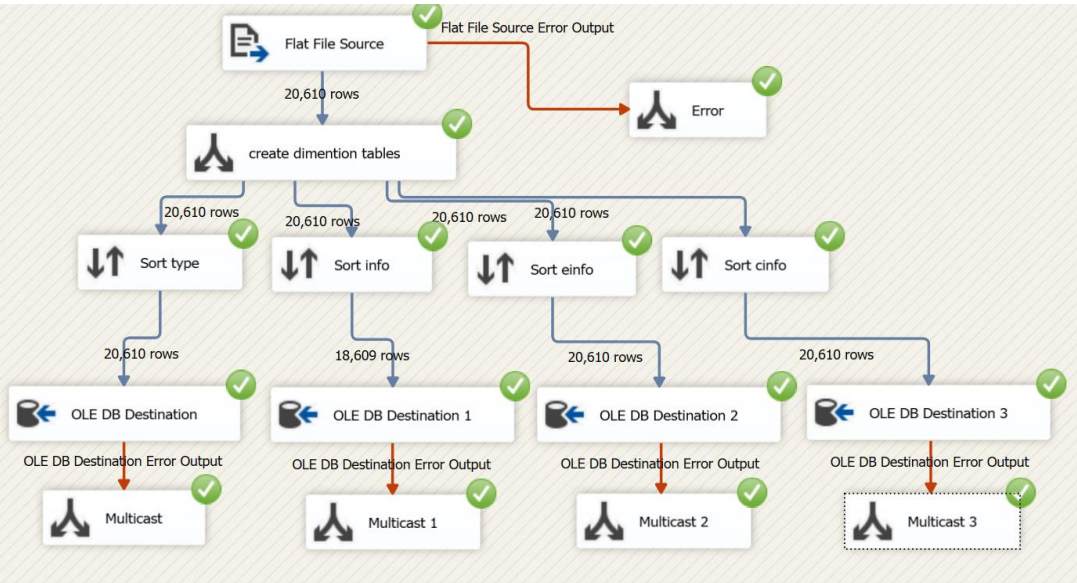


Fig: Saving data into the dimension tables.

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.

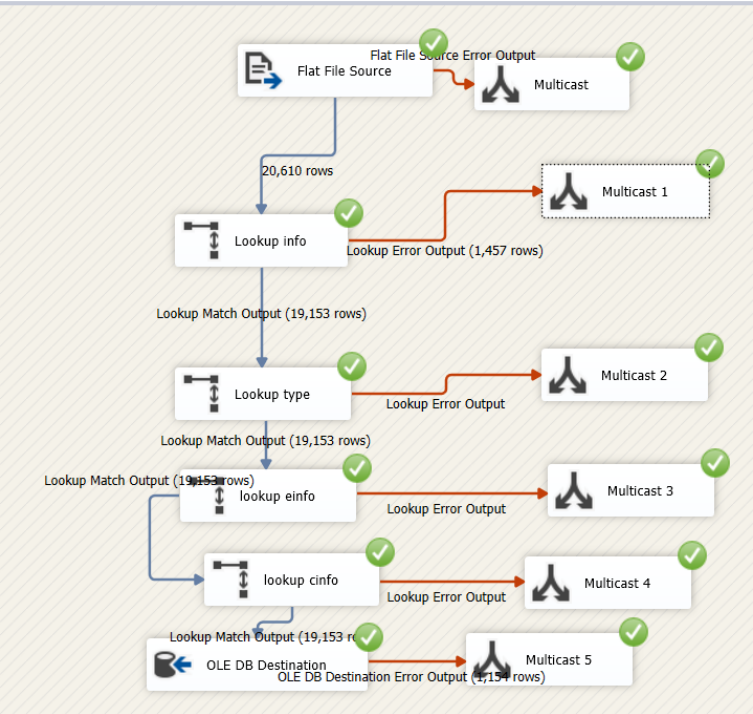


Fig: Saving data to the fact table

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.

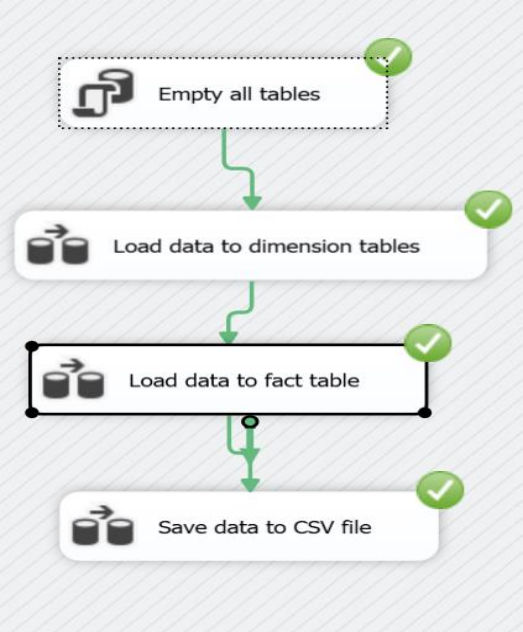


Fig: The main program workflow

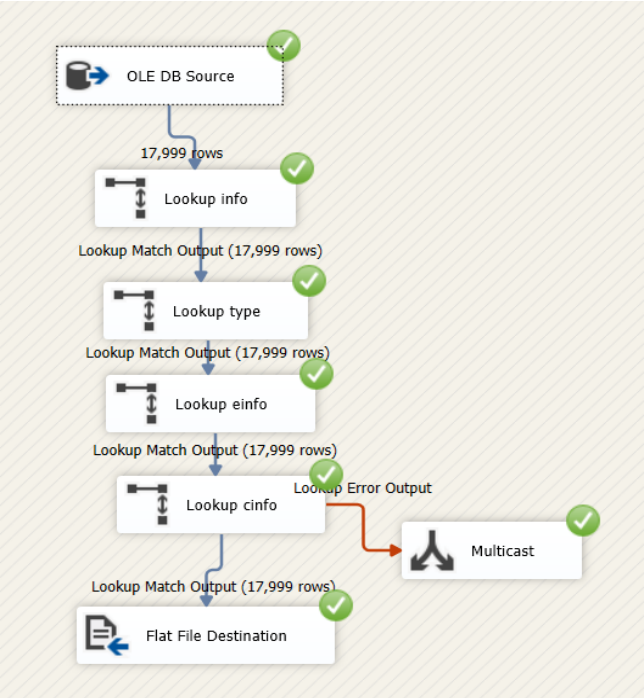


Fig: Steps in writing data into a CSV file

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.

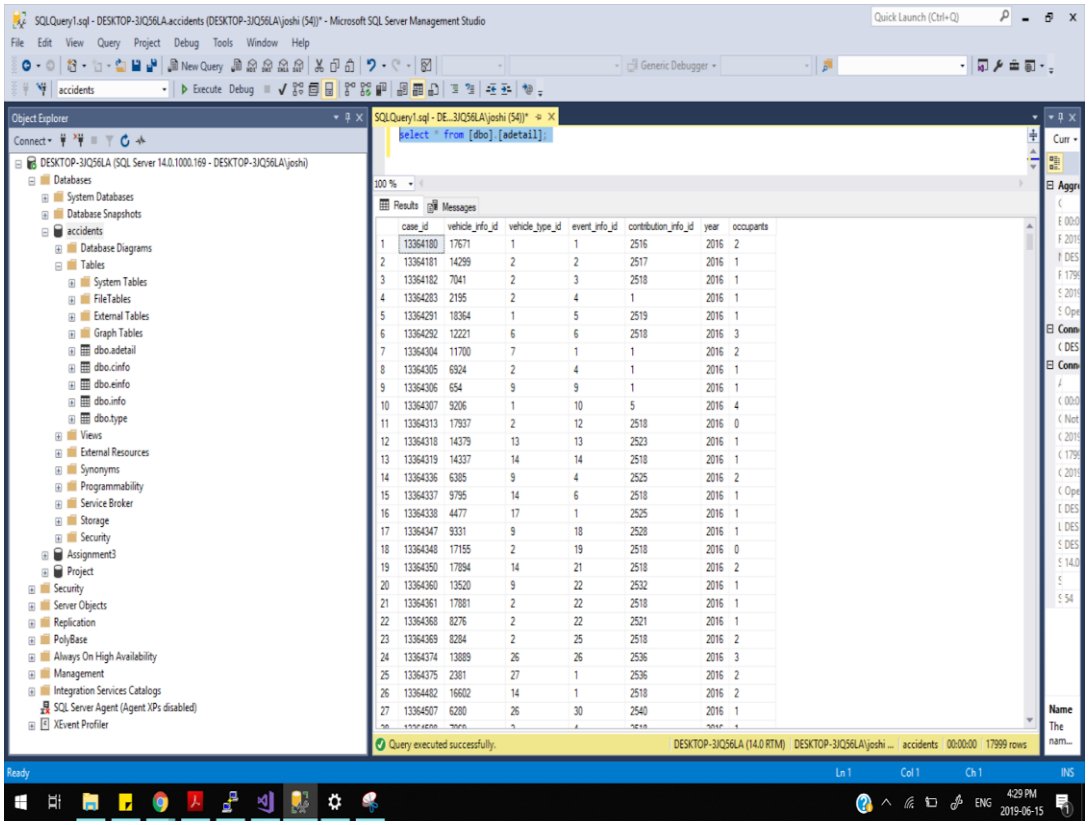


Fig: Loading the fact table into the database

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-one 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
```