RESEARCH

Introduction of Profile Areas of Data Science :Project 5

Frenny Macwan, Aman Jain and Suresh Kumar Choudhary

Full list of author information is available at the end of the article

Abstract

Goal of the Project: To execute SQL queries using Python programming language with BigQuery and estimate scan size using helper package, along with plotting the most meaningful data graphically from a huge chunk of data.

Main Result of the Project: Successfully running the query(s) on Google Colab using BigQuery API, in a very quick succession of time.

Personal Key Leanings: We understood the Big Query API, how fast it works, and to use those APIs using Python Notebooks (Google Colab).

Estimate working hours: 8

Project Evaluation: 3

Keywords: BigQuery API; SQL; Large Dataset; Quick execution

Scientific Background

In the real world, we don't get simple data sets which are small in size or very clean. Generally we get huge chunk of data, which might be meaningful, not meaningful,balanced, unbalanced, and could have multiple missing datas in the data set. Inorder to plot the information and show it in a meaningful graphical representation we will have to process those huge chunk of data, which is generally not possible with a single system, thus comes Google's BigQuery API into picture, where it processes huge chunk of data in just few seconds.

Google BigQuery is a Serverless, highly scalable, and cost-effective cloud storage service (Platform as a Service (PaaS)) that allows you to collect all your data in one system and easily analyze it using ANSI SQL queries. It has capabilties to manage data, query data, access control and get sense from data(machine learning). It can be integrated with any language that can work with its REST API or client libraries. The key features of BigQuery are:

- BigQuery ML (Machine Learning) : It allows you to create and execute machine learning models on planet-scale structured or semi-structured data, directly inside BigQuery using simple SQL in a fraction of the time.
- 2. BigQuery GIS (Geographic Information Systems) : It enables you to analyze and visualize geospatial data in BigQuery by using geography data types and standard SQL geography functions.
- BigQuery BI Engine: With the help of it, you can build rich, interactive dashboards and reports in Data Studio without compromising performance, scale, security, or data freshness. Having a fast and in-memory analysis service, it can analyze data stored in BigQuery with sub-second query response time

and with high concurrency. BI Engine integrates with familiar Google tools like Google Data Studio to accelerate data exploration and analysis.

- Connected Sheets: It enables users to analyze billions of rows and petabytes of data of live BigQuery data in Google Sheets without requiring specialized knowledge of computer languages like SQL. Users can apply familiar tools-like pivot tables, charts, and formulas to easily derive insights from big data.

To use Google BigQuery, you need to create a project in Google Cloud Platform (GCP). Upon registration, you'll receive access to all Cloud Platform products during a free trial period and $300 to spend on these products within the next 90 days with get 10 GB storage and up to 1 TB queries/month. After creating a project in Google Cloud Platform, you need to add at least one dataset to Google BigQuery.A dataset is a top-level container that's used to organize and control access to your data. In simple terms, it's a kind of folder in which your information is stored in the form of tables and views. We can use it with python by installing the BigQuery Python client library version 1.9.0 or higher and the BigQuery Storage API Python client library.

Data

Task 1 : llumina Platinum Genomes data set contains 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree with approx. 10 milion rows with 22 columns. On which we tried to find different variant counts on different samples/chromosomes using Big Query.
Task 2 : 1,000 Genomes dataset has data for 3,500 individuals with Ethnicity, Continental Origin, gender, and family relationship information containing 62 features. There are 26 different populations from many different locations around the globe like CHB (Han Chinese in Beijing, China), JPT (Japanese in Tokyo, Japan) , etc. These populations have been divided into 5 super populations :

AFR (African) AMR (Ad Mixed American) EAS (East Asian) EUR (European) SAS (South Asian).

Goal

Our goal is to write query(s) on python which we connect to the google's BigQuery API and visualize the most important parts of the data.

Result

Task 1 : We have gone through the tutorial "Advanced guide to analyzing variants using BigQuery" and executed 5 queries which we have presented in the figure 1

QUERY 1 (scan size : 1.53)

IT interpretation: In this query number of variants per sample is counted (COUNT(call.name) AS call_count_for_call_set). Each variant in the table has zero or more values for call.name. We have used GROUP BY clause to group them on basis of call_name and counted the number of variants. The resulting table includes the columns call_name and call_count_for_call_set and result is ordered on the basis of call_name.

Biological representation: From a biological point of view , these columns refer to the sample and the number of variants per sample, respectively.

Figure 1 QUERY 1 (scan size : 1.53)

```
QUERY1 = """
SELECT
  call.name AS call_name,
  COUNT(call.name) AS call_count_for_call_set
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v, v.call
GROUP BY
  call_name
ORDER BY
  call_name
"""
```

QUERY 2 (scan size: 4.24)

Figure 2 QUERY 2 (scan size : 4.24)

```
QUERY2="""
SELECT
  call.name AS call_name,
  COUNT(call.name) AS call_count_for_call_set
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v, v.call
WHERE
  EXISTS (SELECT 1 FROM UNNEST(call.genotype) AS gt WHERE gt > 0)
  AND NOT EXISTS (SELECT 1 FROM UNNEST(call.genotype) AS gt WHERE gt < 0)
GROUP BY
  call_name
ORDER BY
  call_name
  """
```

IT interpretation: In this query number of variants per sample is counted but only those which are true variants. And true variants are denoted with genotype greater than 0 value. So we are filtering out on the basis of genotype using EXIST clause. If any variant exists whose genotype is greater than 0 and not less than 0 then we are making a count.The resulting table includes the columns call_name and call_count_for_call_set whose genotype is greater than 0.
Biological representation: From a biological point of view , these columns refer to the sample and the number of variants per sample, respectively for true variants.

QUERY 3 (scan size = 0.25)

IT interpretation: In this query we are counting number of call_filter. As there can be different values of FILTER column, we are counting number of occurrences of each different value of FILTER using group by clause. The resulting table includes call_filter and count of call_filter as number_of_calls columns, which is in ascending order of number_of_calls column.
Biological representation: The VCF specification describes the FILTER column which can be used to label variant calls of differing qualities. So these columns refer

Figure 3 QUERY 3 (scan size : 0.25)

```
QUERY3 = """
SELECT
  call_filter,
  COUNT(call_filter) AS number_of_calls
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v,
  v.call,
  UNNEST(call.FILTER) AS call_filter
GROUP BY
  call_filter
ORDER BY
  number_of_calls
  """
```

to the FILTER which is nothing but the refClass and PASS, and number of the per-variant-call filter.

QUERY 4 (scan size = 3.34)

Figure 4 QUERY 4 (scan size : 3.34)

```
QUERY4= """
  SELECT
    reference_name,
    COUNT(reference_name) AS number_of_variant_rows
  FROM
    `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v
  WHERE
    EXISTS (SELECT 1
            FROM UNNEST(v.call) AS call, UNNEST(call.genotype) AS gt
          WHERE gt > 0)
  GROUP BY
    reference_name
  ORDER BY
    CASE
      WHEN SAFE_CAST(REGEXP_REPLACE(reference_name, '^chr', '') AS INT64) < 10
        THEN CONCAT('0', REGEXP_REPLACE(reference_name, '^chr', ''))
        ELSE REGEXP_REPLACE(reference_name, '^chr', '')
    END
       """
```

IT interpretation: Using this query, we can count number of variants per chromosome(COUNT(reference_name)). So , all the rows having at least one variant call with genotype greater than 0(gt ¿ 0) are included and grouped by chromosome. To group the variant rows by chromosome, we used ORDER BY CASE WHEN expression. We have used SAFE_CAST function to avoid errors for the string to int conversion.(This function will return NULL instead of an error and allow query to not stop in case a value encountered.) To replace substring of reference_name that match regular expression 'chr' by new " string, REGEX_REPLACE function is used. The resulting table includes the columns reference_name and number_of_variant_rows. The result is in ordered form of chromosome(ORDER BY

CASE).
Biological representation: These columns refer to the name of Chromosome (starting with "chr*"), and the number of variants for each chromosome, respectively.

QUERY 5 (scan size = 3.34)

Figure 5 QUERY 5 (scan size : 3.34)

```
QUERY5="""
SELECT
  REGEXP_REPLACE(reference_name, '^chr', '') AS chromosome,
  COUNT(reference_name) AS number_of_variant_rows
FROM
  `bigquery-public-data.human_genome_variants.platinum_genomes_deepvariant_variants_20180823` v
WHERE
  EXISTS (SELECT 1
            FROM UNNEST(v.call) AS call, UNNEST(call.genotype) AS gt
           WHERE gt > 0)
GROUP BY
  chromosome
ORDER BY
  chromosome
  """
```

IT interpretation: In this query the number of variants per chromosome is counted(COUNT(reference_name) AS number_of_variant_rows). REGEX_REPLACE expression is used to replace substring of reference_name that match 'chr' with new string ". This will display only numbers as reference_name. Here, all the rows having at least one(EXISTS clause) variant call with genotype greater than 0(gt ¿ 0) are included and grouped them using chromosome(GROUP BY). The resulting table includes the columns chromosome and number_of_variant_rows. However, the output still sorts by chromosome which is string(ORDER BY CASE).
Biological representation: These columns refer to the name of Chromosome (replacing 'chr' with empty string "), and the number of variants for each chromosome,respectively.

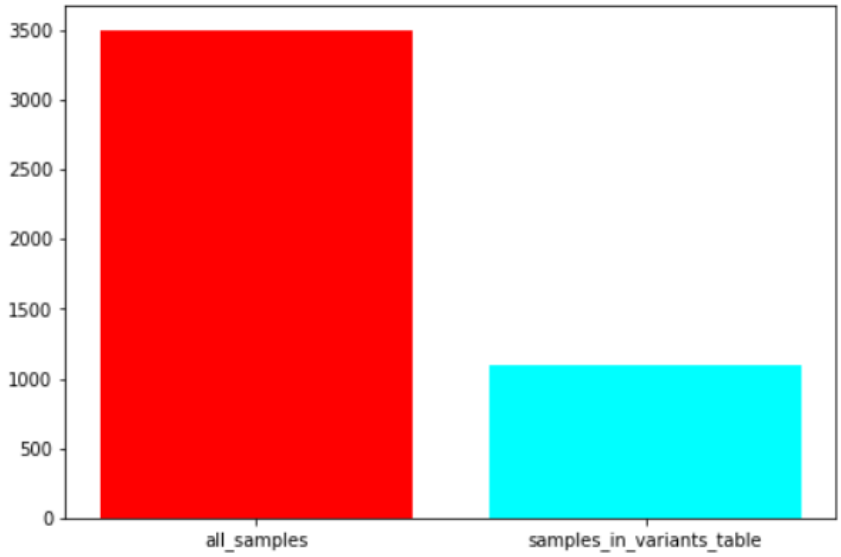
Task 2 : Exploring the Phenotypic Data

Query 1 (Scan Size : 2.385)
We had gone through the exploring-the-phenotypic-data Data Story of 1000 Genomes Project.
IT interpretation : In this query we count the total number of samples present, and along with the count of samples in Phase 1 Integrated Variant Set, through writing a condition SUM(IF(In_Phase1_Integrated_Variant_Set = TRUE, 1, 0)) AS samples_in_variants_table. (**Figure** 6)
Biologic Representation : The query only gives us the count of total samples and a count of a particular variant to enables accurate assessment of common variants. (**Figure** 7)

Figure 6 QUERY 1 (scan size : 2.385)

```
# Count the number of samples in the phenotypic data
QUERY0="""
SELECT
  COUNT(sample) AS all_samples,
  SUM(IF(In_Phase1_Integrated_Variant_Set = TRUE, 1, 0)) AS samples_in_variants_table
FROM
  `genomics-public-data.1000_genomes.sample_info`
  """
```

Figure 7 PLOT 1 : All Samples vs Samples in Variants Table



Query 2 (Scan Size : 2.385)

IT Interpretation : We are selecting the genders, the gender counts and the ratio of the genders respectively for Male and Females. The statement (gender_count/SUM(total_count) OVER (PARTITION BY gender)) as gender_ratio gives us the gender ration by dividing the count of a particular class and dividing by the total samples (1092). (**Figure** 8)

Biologic Representation : The query gives us the gender ratio in the subset of 1092 samples and we found the ratio is almost even, meaning the sample is not biased towards any single class. (**Figure** 9)

Query 3 (Scan Size : 0.0016)

IT Interpretation : From 'genomics-public-data.1000_genomes.sample_info' we are extracting a sub sample of 1092 samples and by using (population_count/SUM(total_count) OVER (PARTITION BY population)) AS population_ratio we are calculating the ratio of the population and also finding out the super population of that particu-

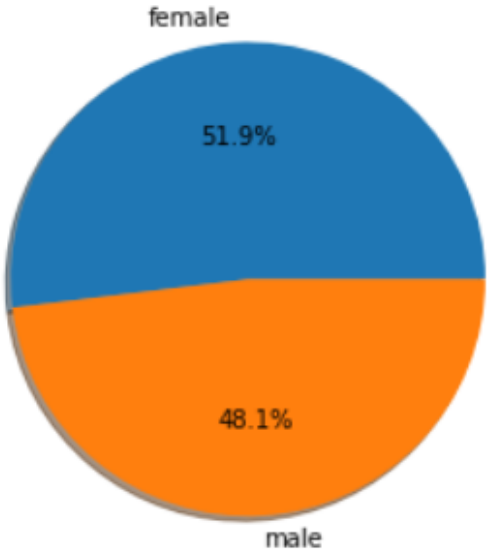
Figure 8 QUERY 2 (scan size : 2.385)

```
QUERY1="""
SELECT
  gender,
  gender_count,
  (gender_count/SUM(total_count) OVER (PARTITION BY gender)) as gender_ratio
FROM (
  SELECT
    gender,
    COUNT(gender) AS gender_count
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY gender
    ),(select count(gender) as total_count from  `genomics-public-data.1000_genomes.sample_info` where In_Phase1_Integrated_Variant_Set = TRUE)
"""
```

```
bq_assistant.estimate_query_size(QUERY1)
```

```
2.3853033781051636e-05
```

Figure 9 PLOT 2 : Female and Male Ratio in the Sub Sample



lar population by extracting the information from the column super_population. (**Figure** 10)

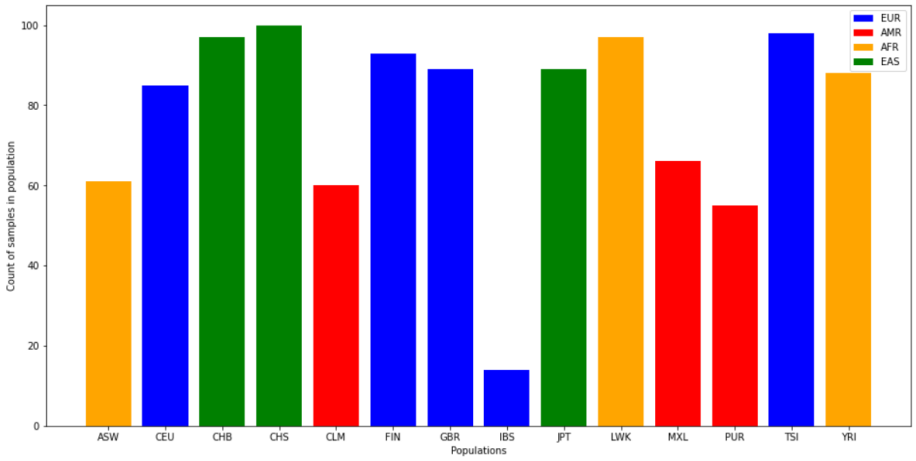
Biologic Representation : We are selecting the population column and the description which tells us about the ancestry and the super population. Ancestry would mean the origin and which country they belong to now : For Example - Finnish in Finland are finnish people who hare living in Finland, where as African Ancestry in Southwest US are people who are African descendants and are living in the Southwest of United States or in simple terms the people who have common sets of tradition, culture, nation etc. Super population will give us the description, from which continent do they actually belong like African, European, East Asians etc. Based on the output we found the population count ranging between 55 to 100,

where IBS (Iberian Population in Spain) being the only exception with the count as 14. (**Figure** 11)

Figure 10 QUERY 3 (scan size : 0.0016)

```
QUERY2= """
SELECT
  population,
  population_description,
  population_count,
  (population_count/SUM(total_count) OVER (PARTITION BY population))  AS population_ratio,
  super_population,
  super_population_description,
from(
  SELECT
    population,
    population_description,
    super_population,
    super_population_description,
    COUNT(population) AS population_count,
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
    population,
    population_description,
    super_population,
    super_population_description),(select count(population) as total_count from  `genomics-public-data.1000_genomes.sample_info` where In_Phase1_Integrated_Variant_Set = TRUE) or
    """
```

Figure 11 PLOT 3 : Count of each ethnicity in the Sub Sample



Query 4 (Scan Size : 6.82)

We are finding the ratio of the super population (Ethnicity/Origin) in the sub sample set. We see the samples ration from 0.16 to 0.34

IT Interpretation : In this query we are calculating the ratio of the super population (Mostly categorized as Continental Population like European, Asian, African etc) through the (super_population_count/SUM(total_count) OVER (PARTITION BY super_population)) AS super_population_ratio and since aggregation is done (Count, Sum) and group by should also be applied, where are are grouping the Population and Population Description in the query GROUP BY super_population, super_population_description . (**Figure** 12)

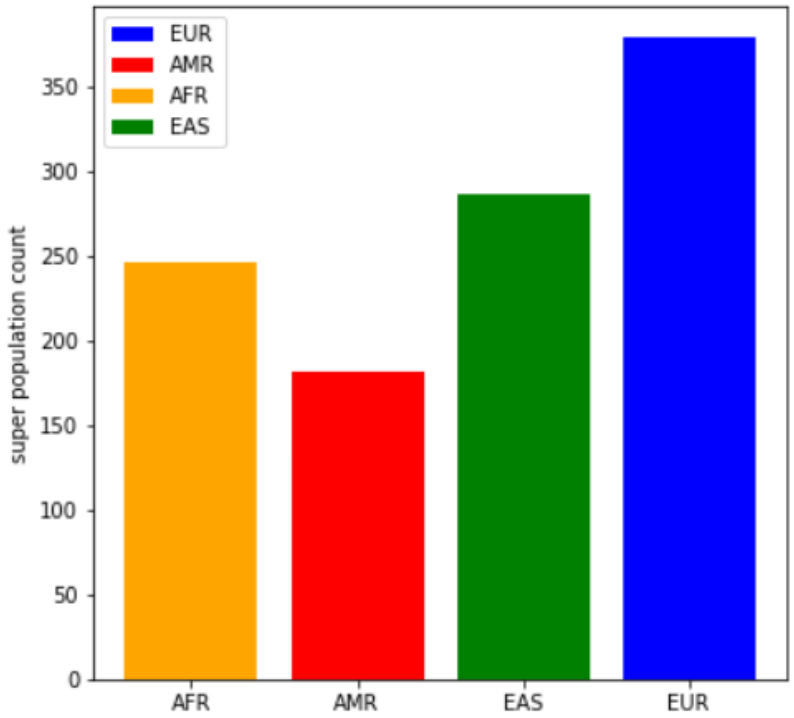
Biologic Representation : The first two columns denote the super population and the description like America, African, European and East Asian which are categorized as Continental Population rather than country specific population. The

column super_population_count gives us the count of the respective continental population and the last column of ratio gives us the ratio of the respective continental population among all the other super population. (**Figure** 13)

Figure 12 QUERY 4 (scan size : 6.82)

```
QUERY3= """
SELECT
  super_population,
  super_population_description,
  super_population_count,
  (super_population_count/SUM(total_count) OVER (PARTITION BY super_population)) AS super_population_ratio
from(
  SELECT
    super_population,
    super_population_description,
    COUNT(population) AS super_population_count,
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
    super_population,
    super_population_description),(select count(super_population) as total_count from  `genomics-public-data.1000_genomes.sample_info` where In_Phase1_Integrated_Variant_Set = TRUE
    """
```

Figure 13 PLOT 4 : Count of Super Population i.e Continental Origin



Query 5 (Scan Size : 4.015)

We subdivided the ethnic population ratio into Males and Females and calculated the ratio of males per ethnic population and female per ethnic population among over all sub sample of population. In query 3, we found IBS ethnicity having the least population of all, and could see the sample of Finnish people living in Finland

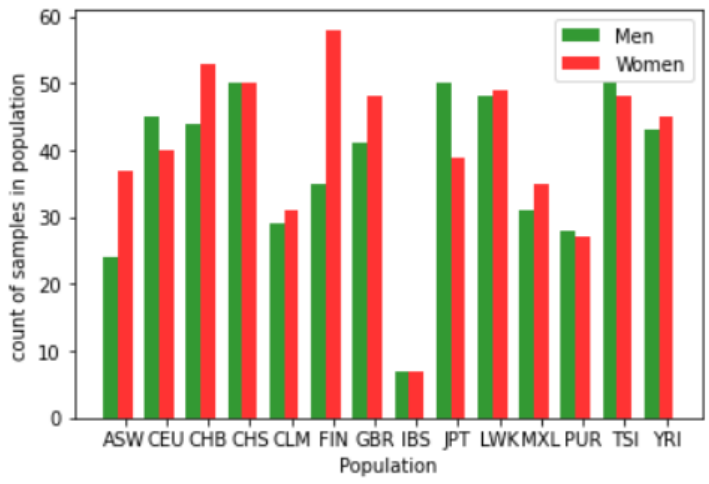
have a lot more number of Females compared to Males.
IT Interpretation : (population_count/SUM(total_count) OVER (PARTITION BY gender)) AS population_ratio - Using this statement we were able to divided the samples into male and female and were able to find the ratio of per gender class, per ethnicity for the whole subsample. Using ORDER BY population, gender we ordered the complete output alphabetically by population and gender to give more sorted and clear output. (**Figure** 14)

Biologic Representation :In this query We found the ratios of Males and Females of each ethnic population out of a total of 14 ethnic population group. (**Figure** 15)

Figure 14 QUERY 5 (scan size : 4.015)

```
QUERY4= """
SELECT
  population,
  gender,
  population_count,
  (population_count/SUM(total_count) OVER (PARTITION BY gender))
  AS population_ratio
from(
  SELECT
    gender,
    population,
    COUNT(population) AS population_count,
  FROM
    `genomics-public-data.1000_genomes.sample_info`
  WHERE
    In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
    gender,
    population)
,(select count(population) as total_count from  `genomics-public-data.1000_genomes.sample_info` where In_Phase1_Integrated_Variant_Set = TRUE)  ORDER BY population, gender
    """
```

Figure 15 PLOT 5 : Female and Male with each Ethnicity Count.



Query 6 (Scan Size : 2.42)
Since there could be a possibility of these people related to each other, or could be a part of family, we will distribute the family sizes as well.
IT Interpretation : By using the GROUP BY family_size, we are grouping and

performing an aggregate function of count on the family size to get the desired output using COUNT(num_family_members) AS num_families_of_size. (**Figure** 16)
Biologic Representation : Through this query we were able to find out how many families are living in the size of 1, 2, 3 and 4 respectively. (**Figure** 17)

Figure 16 QUERY 6 (scan size : 2.42)

```
QUERY5= """
SELECT
num_family_members AS family_size,
COUNT(num_family_members) AS num_families_of_size
FROM (
  SELECT
  family_id,
  COUNT(family_id) AS num_family_members,
  FROM
  `genomics-public-data.1000_genomes.sample_info`
  WHERE
  In_Phase1_Integrated_Variant_Set = TRUE
  GROUP BY
  family_id)
GROUP BY
family_size
    """
```

Discussion

In this project there is no requirement to manage the infrastructure, we can focus on analyzing data to find meaningful insights using familiar SQL without the need for a database administrator. Using Big Query a data warehouse can be set up in seconds and we can start to query data immediately. It can be executed with the help of python and results can be viewed with the help of pandas as well. It is fast, server less and highly salable, so this is a typical project for a data-scientist.

Appendix

Code -

- Aman - Plotted the data of Task2 .
- Suresh - Successfully implemented the Query(s) of Task 2.
- Frenny - Successfully implemented the Query(s) of Task 1

Report -

- Aman - Data and Result(Task 2)
- Suresh - Scientific Background, Discussion
- Frenny - Abstract, Result (Task 1 and Task 2).

Author details
References

Figure 17 PLOT 6 : Count of Families having size of 1,2,3 or 4 Family Members