Experiment 6:- Google Big table

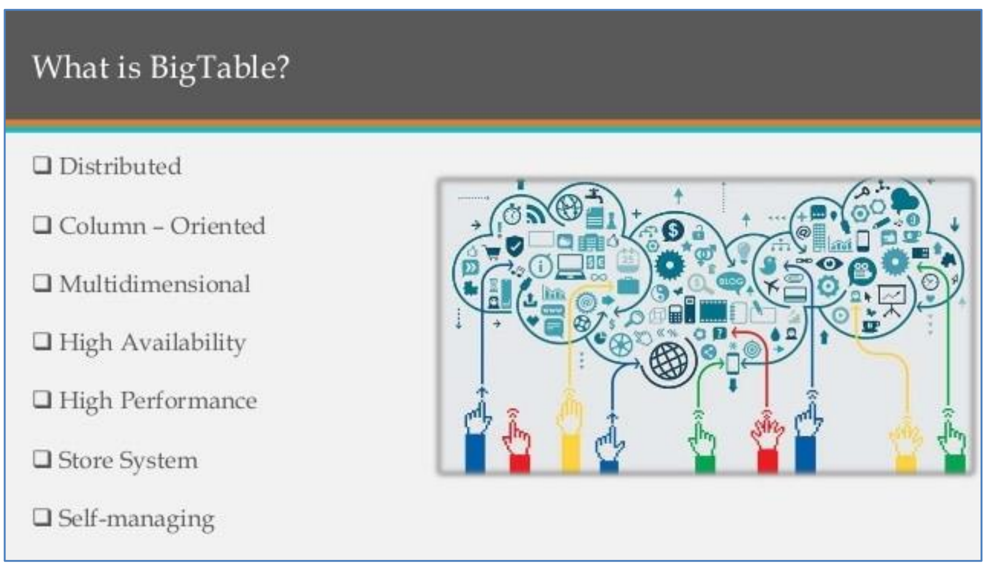
CLASS: BE CMPN A ROLL NO. : 19
NAME: REBECCA DIAS PID: 182027

Aim:

To get handson in Google Big table and work with some of the publically available datasets

Introduction:

Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products. In this paper we describe the simple data model provided by Bigtable, which gives clients dynamic control over data layout and format, and we describe the design and implementation of Bigtable.



Features:

- Fast and performant

Use Cloud Bigtable as the storage engine that grows with you from your first gigabyte to petabyte-scale for low-latency applications as well as high-throughput data processing and analytics.

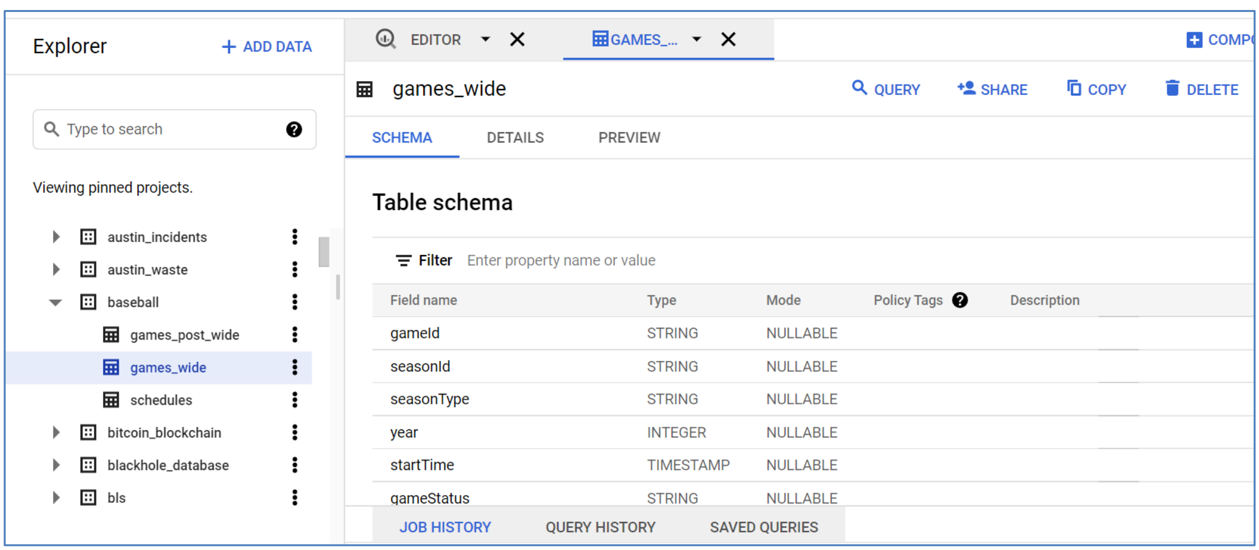
- Seamless scaling and replication
  Start with a single node per cluster, and seamlessly scale to hundreds of nodes dynamically supporting peak demand. Replication also adds high availability and workload isolation for live serving apps.
- Simple and integrated
  Fully managed service that integrates easily with big data tools like Hadoop, Dataflow, and Dataproc. Plus, support for the open source HBase API standard makes it easy for development teams to get started.

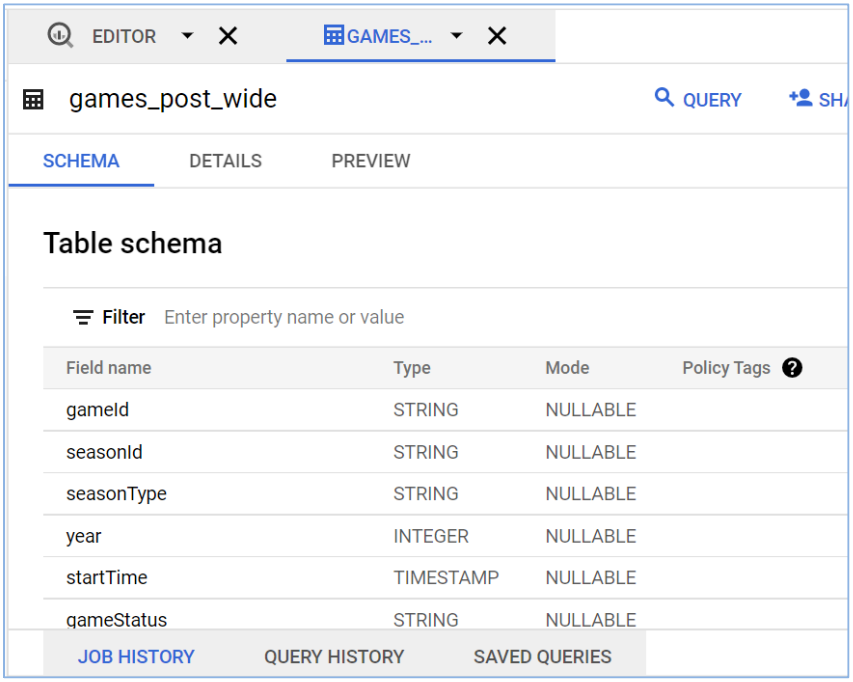
Implementation:

Table Schema

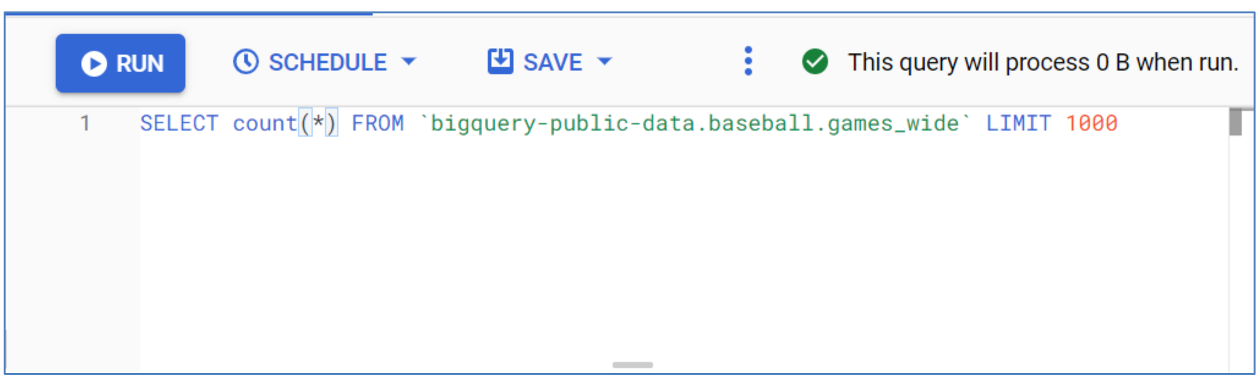
This public data includes pitch-by-pitch data for Major League Baseball (MLB) games in 2016.

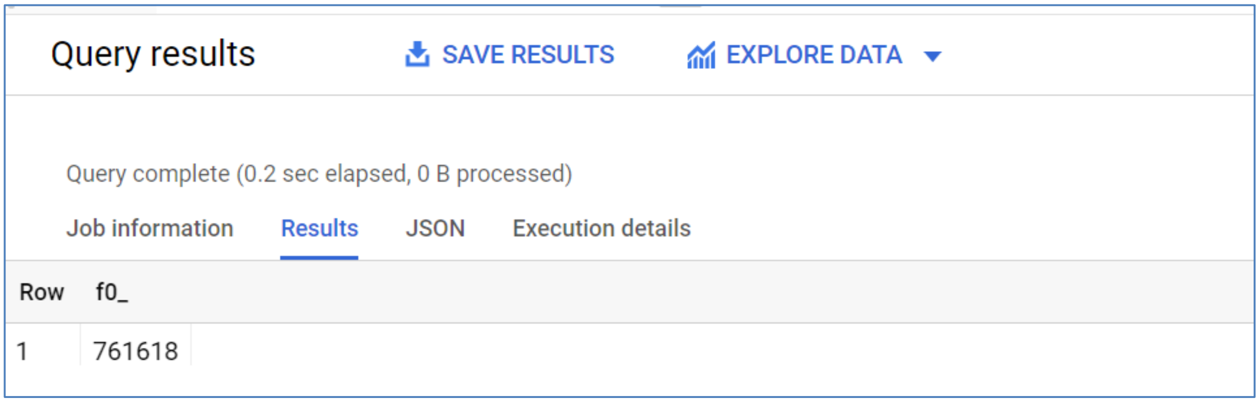
This dataset contains the following tables: games_wide (every pitch, steal, or lineup event for each at bat in the 2016 regular season), games_post_wide(every pitch, steal, or lineup event for each at-bat in the 2016 post season)





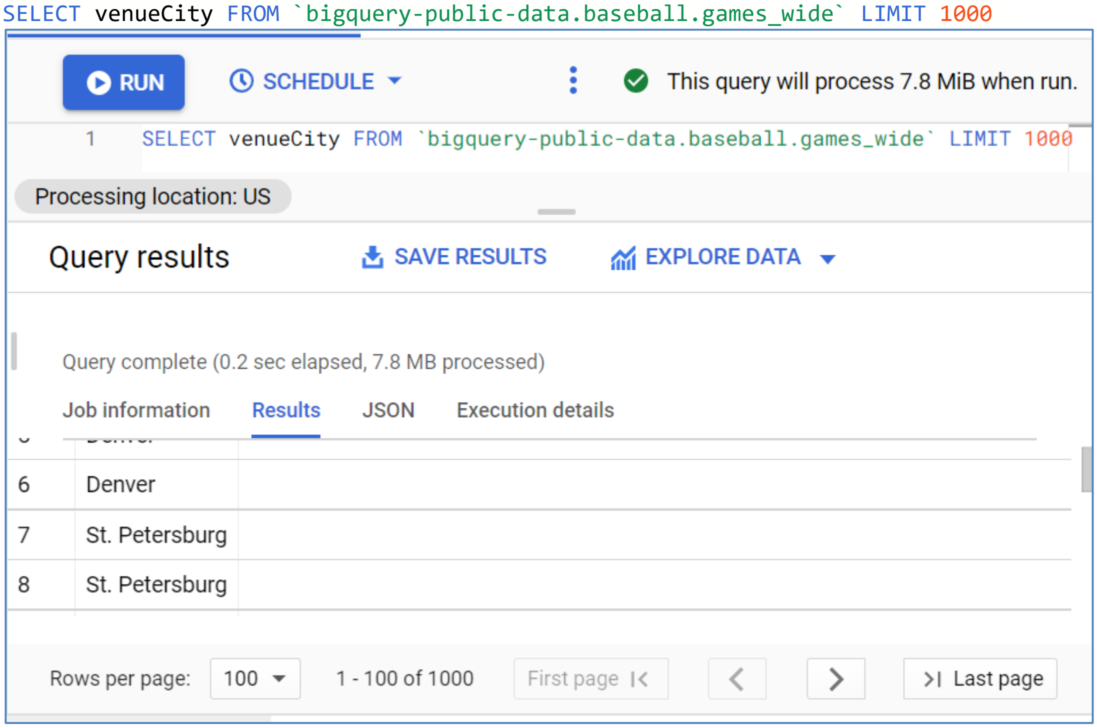
1. Examine the number of records in that dataset





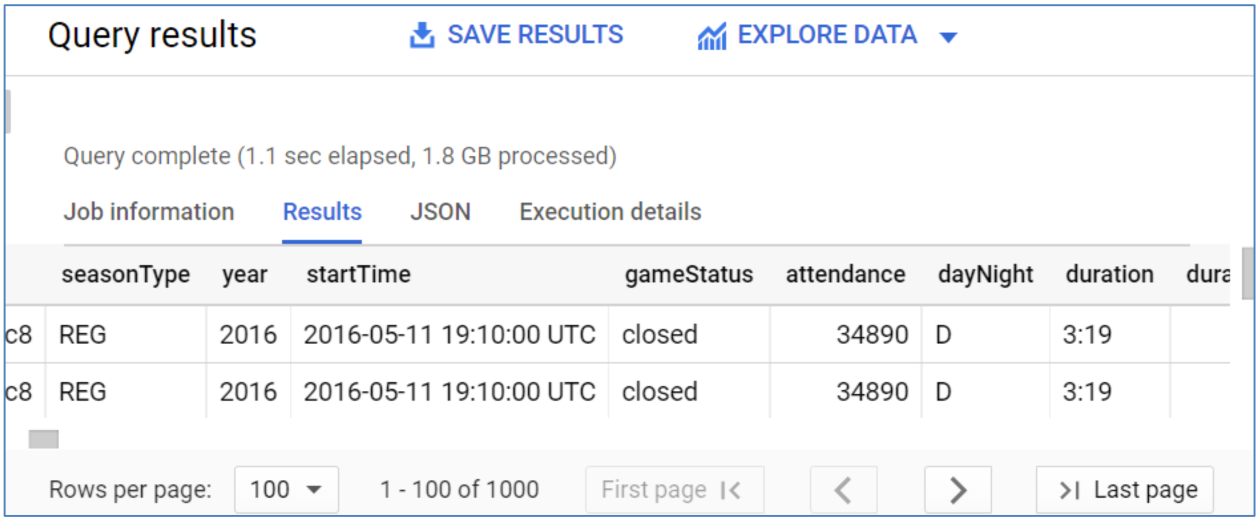
2. Look into a single table and perform (For these queries the output should bring out some understanding about the dataset. And appropriate analysis should follow for each query)
    a. A simple select query

```
SELECT venueCity FROM `bigquery-public-data.baseball.games_wide` LIMIT 1000
```



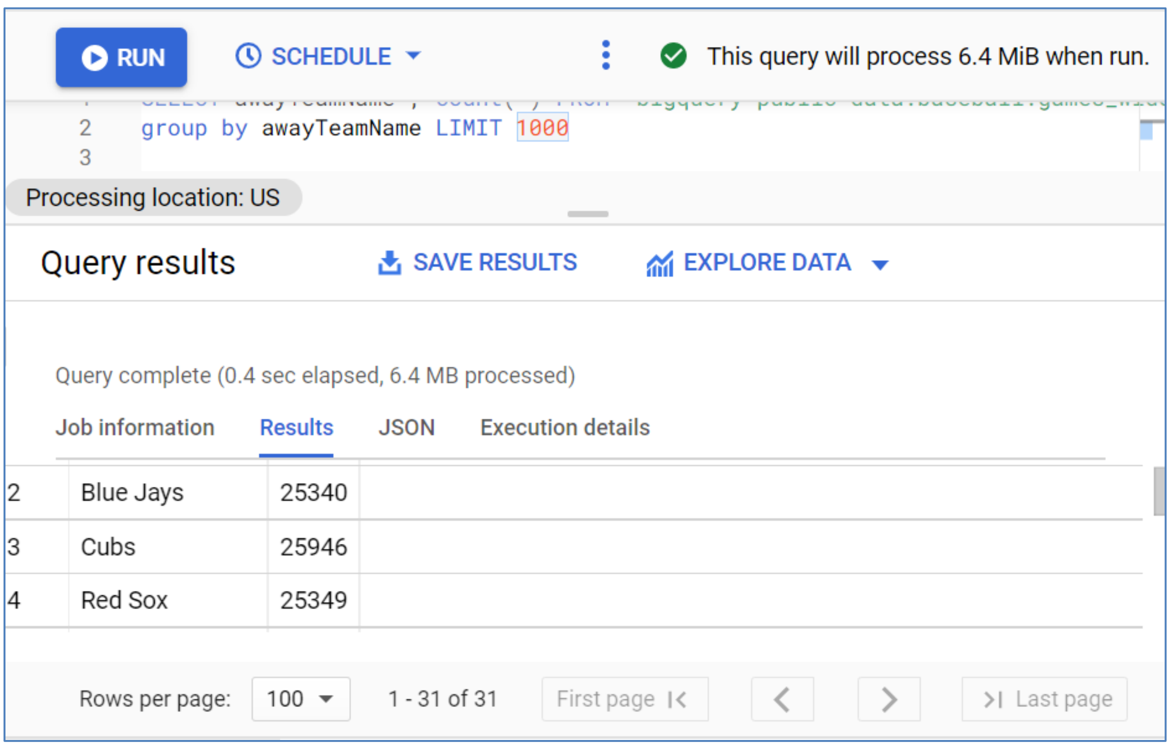
    b. Select query with a filter

```
SELECT * FROM `bigquery-public-
data.baseball.games_wide`  where gameStatus='closed' LIMIT 1000
```



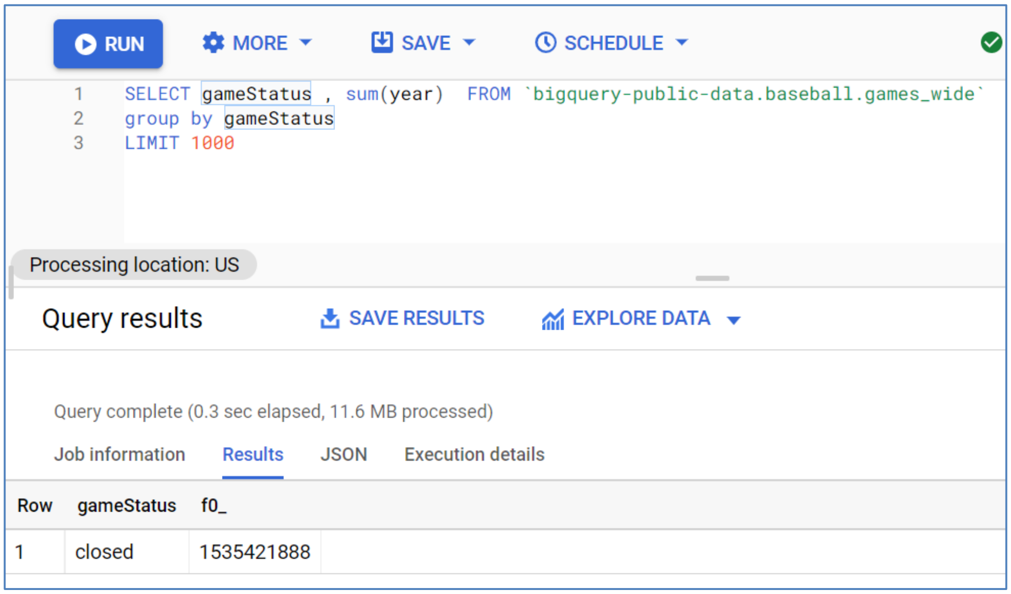
c. Analytical queries

```sql
SELECT awayTeamName , count(*) FROM `bigquery-public-
data.baseball.games_wide`
group by awayTeamName LIMIT 1000
```



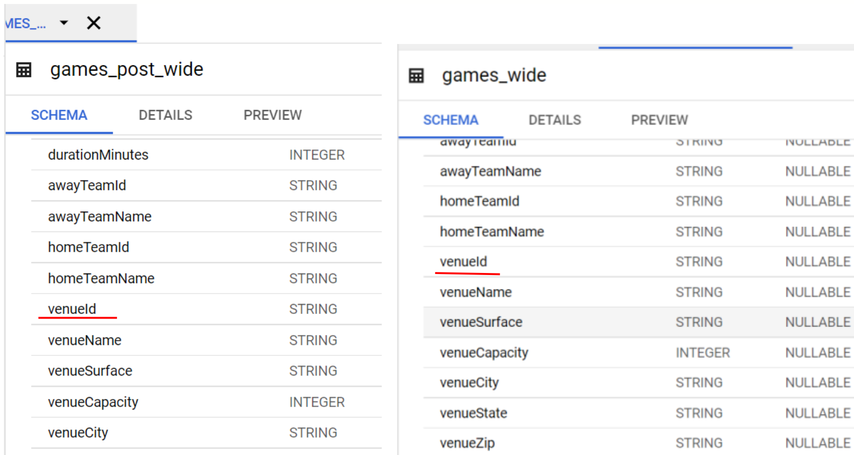
d. Group by game status

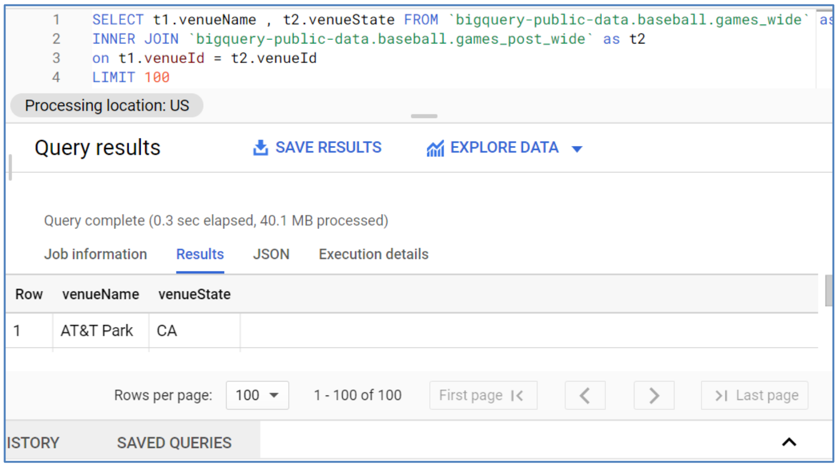
```sql
SELECT gameStatus , sum(year)  FROM `bigquery-public-
data.baseball.games_wide`group by gameStatus LIMIT 1000
```



3. Try to combine two related datasets and extract data from two related datasets.
   Print the Venue name and Venue State from 2 tables namely , games_post_wide and games_wide using an inner join clause on the venue ID

```
SELECT t1.venueName , t2.venueState FROM `bigquery-public-data.baseball.games_wide` as t1
INNER JOIN `bigquery-public-data.baseball.games_post_wide` as t2
on t1.venueId = t2.venueId LIMIT 100
```





Conclusion:-

Bigtable is ideal for storing very large amounts of data in a key-value store and supports high read and write throughput at low latency for fast access to large amounts of data.Bigtable is schema-free where as RDBMS follows a rigid schema based architecture. Cloud Bigtable is a sparsely populated table that can scale to billions of rows and thousands of columns, enabling you to store terabytes or even petabytes.During this experiment , successfully implemented various queries on bigquery-public-data:baseball.games_wide dataset.