Trendify

CMPE 138: Database Systems
Professor Ghergori Guzun

Jason Hernandez (ID: 015511119)
Ruyi Sule (ID: 015641704)
Anik Budhathoki (ID: 015633579)

Due: Dec 2, 2024

Introduction/Motivation/Description of the Application: Trendify

Description of the Application

Trendify is a web-based application designed to help users explore and visualize trending topics using data sourced from Google Trends. Leveraging the power of BigQuery, the application enables users to examine trending topics across different locations, timeframes, and categories. With features like custom trend analysis, trend comparison, and visualizations, the application aims to provide actionable insights to individuals and organizations interested in search behavior patterns.

Why is it Interesting (Our Projects Purpose / Problems we are trying to solve)?

Trendify stands out for several reasons:

1. Dynamic Insights: Trends are ever-changing, and analyzing them provides a glimpse into collective interests, behavior, and emerging phenomena.
2. Broad Applicability: The application caters to diverse needs—from market analysis to academic research—making it versatile and widely appealing.
3. Empowering Decision-Making: Whether for a marketer strategizing a campaign or a journalist finding newsworthy topics, Trendify turns data into decisions.

Who Would Use the Application?

- Marketers: To identify trending interests and align advertising strategies with public sentiment.
- Researchers: Academics studying sociological patterns and behavioral trends over time.
- Businesses: Small and large businesses analyze consumer interests to develop products or services.
- Journalists and Media Outlets: To uncover stories or topics gaining popularity.
- Students and Educators: For projects requiring analysis of current trends.

How Would it Be Used?

1. Exploration: Users input a location, category, or keyword to view trends for a specific timeframe.
2. Comparison: Users compare trends across multiple locations or categories to identify variations.
3. Custom Analysis: The platform allows users to upload their keywords for a focused trend analysis.
4. Visualization: The data is presented in visually engaging graphs, heatmaps, and tables, making it easy to understand and share.
5. Alerts: Users can set up notifications for real-time updates on specific topics or keywords.

Why is it Challenging?

- Big Data Handling: Managing and querying large datasets like Google Trends efficiently is technically demanding.
- Real-Time Performance: Ensuring the application responds quickly to user queries while processing complex datasets.
- Scalability: Supporting multiple concurrent users without compromising performance.
- Meaningful Visualization: Representing large volumes of data concisely and insightfully, requiring expertise in UI/UX and data visualization.
- Data Integration: Combining BigQuery's structured data with user-defined queries seamlessly.

How Can the Application Benefit from a Database?

Databases play a critical role in Trendify by:

1. Providing Data Storage: BigQuery will serve as the primary source for trends, while a secondary database stores user data, frequently accessed trends, and subscription data. User data will consist of User ID, email, password, etc. This will uniquely identify users in our application. Using the BigQuery Google Trends dataset, the user table for TrendSpotter will be filled using the location data of the user, common preferences of the user, and common search topics of the user.
2. Trend Analysis: Using the Google Trends dataset, we can analyze the historical trends of common search words. We can create tables and frequency graphs to better analyze certain topics. For example, we can analyze the term “NBA” throughout internet history.
3. Custom Alerts: A database enables efficient storage and retrieval of user-defined alerts and keywords. For example, if a recent search topic is suddenly trending then the database can be used to structure an alert to the user to state the recent uprising of the search.
4. Enhanced Scalability and Future Integration: As the Google Trends dataset dynamically updates, it enables future scalability based on the same metrics used to build our application.

In summary, the database is the backbone of Trendify, ensuring seamless integration of large-scale datasets, robust user interactions, and efficient data processing.

Related Work

The field of trend analysis and predictive modeling has seen significant advancements with the availability of datasets such as Google Trends, enabling researchers and developers to build applications for various domains like marketing, social analysis, and user behavior prediction. This section compares the Trendify application with existing works in the field, highlighting

similarities, differences, pros and cons of each approach, and lastly how Trendify improves upon existing methodologies.

Existing Works

1. Google Trends Official Platform

    - Description: Google Trends itself provides a robust interface for exploring search data. Users can see trending topics, compare keywords, and visualize data over time and across regions.
    - Pros:
        - Direct access to real-time data.
        - Powerful visualizations like line charts and heat maps.
        - Easy-to-use interface for non-technical users.
    - Cons:
        - Limited customization for specific applications.
        - No integration with external tools or user-specific features.
        - Lacks predictive capabilities for trend forecasting.
    - Comparison: Trendify is similarly structured to Google Trends but it expands upon the functionality by enabling user-specific customizations, and deeper analysis of niche topics.

2. Trend Analysis Using BigQuery (Research Paper: “Leveraging Google Trends with BigQuery for Predictive Marketing,” 2022)

    - Description: This study used Google Trends data stored in BigQuery to analyze consumer behaviors for targeted advertising. It integrated data pipelines for efficient trend extraction and applied machine learning models to predict future trends.
    - Pros:
        - Demonstrates how to process large datasets efficiently using BigQuery.
        - Offers predictive insights for marketing strategies.
    - Cons:
        - Focused narrowly on marketing applications.
        - Requires advanced technical skills for implementation.
    - Comparison: While the paper focuses on targeted advertising, Trendify generalizes trend analysis for broader use cases like personal scheduling, topic exploration, and public interest insights. Additionally, it incorporates an interactive web application for accessibility.

3. Forecasting Social Trends Using Google Data (Journal Article: "Harnessing Search Trends for Sociological Predictions," 2021)

    - Description: The article discusses a methodology for using search trend data to predict societal changes, such as public health crises or economic shifts. It employs machine learning models and time series analysis.
    - Pros:
        - Demonstrates the predictive power of trend data.
        - Explores interdisciplinary applications.
    - Cons:
        - Requires extensive computational resources.
        - Focuses on macro-level trends rather than user-centric applications.
    - Comparison: Trendify adopts a user-centered approach, targeting individual users and small businesses. It simplifies access to predictive capabilities without requiring advanced technical expertise.

How Trendify Improves Upon Existing Approaches

- TIntegration of features:
  Unlike the Google Trends platform, Trendify offers the ability to save queries but most distinguishably personalization. By combining features from multiple tools, Trendify delivers a comprehensive solution for both casual and professional users.

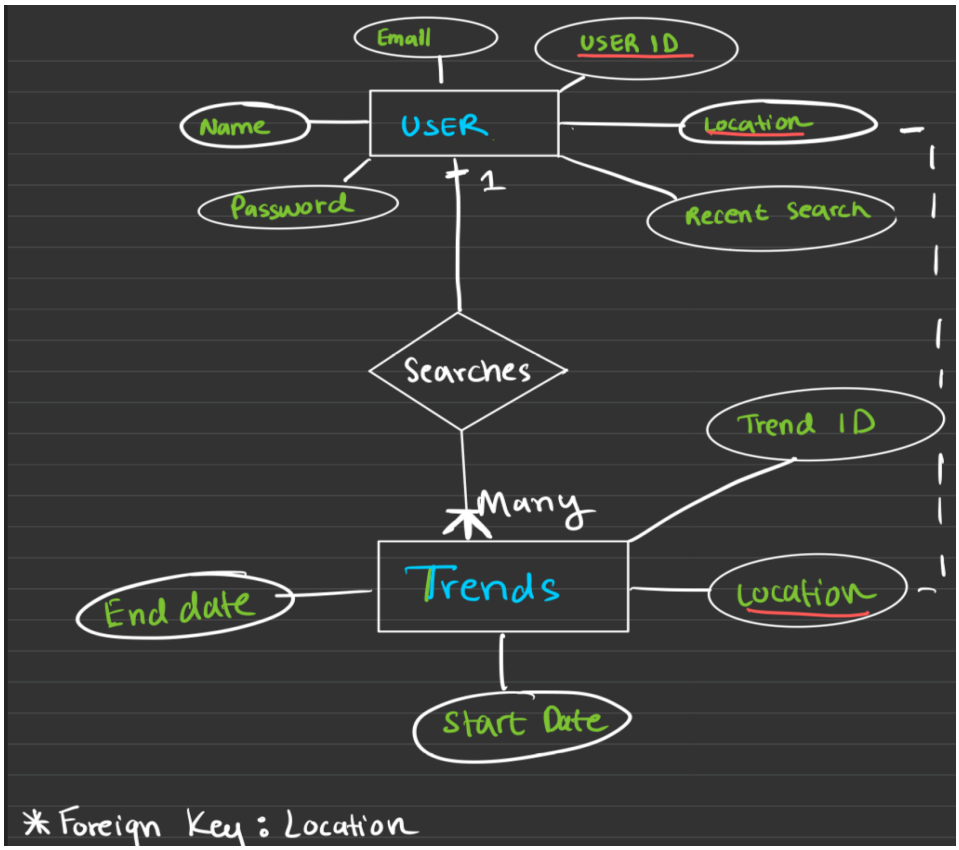
- User-Friendly Interface:
  While tools like BigQuery require technical expertise, Trendify will eventually provide a no-code solution for users to interact with BigQuery datasets through a simple interface.

- Advanced Analytics:
  Trendify incorporates models and comparisons that improve decision-making. This goes beyond the static visualizations offered by most existing platforms.

- Scalability:
  The use of BigQuery ensures that Trendify can handle large datasets efficiently, making it suitable for both individual users and organizations.

Methods Description

ER Diagram of Database



It is important to note that the primary keys of the user table are User ID and Location. The primary key of the trends table is Trend ID. The foreign key used to establish the relationship between the User table and the Trends table is location ID. The user table and the trends table offer a 1-to-many relationship as one user can search multiple trends based on a string.

Description of Application and Features

Using the specific location of the user and establishing a time frame, the user can observe trending topics. This feature will allow all users to explore interests close to them. By leveraging the Google Trends Database, we can query region-specific (city, state, country) topics. In addition, we can set a time frame (Given by Start date and end date in our trends table) to refine our search results such that results only display the most relevant and recent trends.

In addition, users can use the Trendify category search feature to filter trends in the area based on broad categories such as sports, entertainment, etc. This will enable a great user experience as users are presented with a curated list of their top interests based on their recent searches.

Finally, applying trending alerts is the final key feature of our application. Trendify will periodically query and update the Google Trends Database to detect large spikes in search popularity for terms that match the user's current location. By analyzing trends over time,

Trendify will be able to notify users of emerging trends. In general, geolocation, categorization, and real-time trend analysis will make Trenify very personalized and tailored to the user.

SQL Queries Needed To Implement Features

1) Exploring Trends via Location and Time Example Query

```
SELECT term, score, week, dma_name

FROM (

   SELECT term, score, week, dma_name

   FROM top_rising_terms

   WHERE dma_name = 'User_location'

) AS filtered_data

WHERE week >= 'start_date' AND week <= 'end_date'

ORDER BY score DESC
```

- This Query retrieves the search term, score (score is defined as the numeric representation of the term's popularity), the week, and the dma_name (location/Designated Markey Area).
- The inner query filters the top_rising_terms table to include only rows where dma_name matches the user's location.
- The outer query takes the results of the inner query and applies filtering to keep only rows where the week is within the specified date range (start_date to end_date)
- The score is filtered in descending order (Top results)

2) Keyword-specific Trend Analysis Example Query

```
SELECT week AS date,

      (SELECT SUM(score)

       FROM top_rising_terms AS subquery

       WHERE subquery.term = 'EXAMPLE: AI technology'

         AND subquery.week = main.week) AS total_popularity

FROM top_rising_terms AS main
```

WHERE term = 'EXAMPLE: AI technology'

AND week BETWEEN 'start_date' AND 'end_date'

GROUP BY week

ORDER BY week ASC;

- The query selects the column week (representing periods) and aliases it as the date for clarity in the output
- The inner query calculates the total popularity (SUM(score)) for the given term and week repeatedly.
- The WHERE clause filters the top_rising_terms table to include only rows where the search term is 'AI technology' (or any other search term) and the week falls within the specified date range (start_date to end_date)
- GROUP BY week ensures that the query groups data by each week to produce one result per week.
- ORDER BY week ASC sorts the results in ascending order of time for chronology

Github Repository Link: https://github.com/ruyisule/138PJ.git

Analysis of Queries

1. Most Important Queries in the Application

Query 1: Location-based Trending Topics
This query retrieves the most popular search topics for a given location (e.g., state or city) within a specified timeframe.

- Feature Responsible:
  The "Explore Trends by Location and Time" feature relies on this query. It allows users to understand what topics are trending in a specific area during a chosen period.

- Inputs:
  - location (e.g., "California")
  - start_date (e.g., "2024-01-01")
  - end_date (e.g., "2024-01-31")
- Outputs:
  A list of trending search terms, including their relative search popularity, sorted in descending order.

Query 2: Keyword-specific Trend Analysis
This query identifies the popularity and evolution of a specific keyword or term over time.

- Feature Responsible:
  The "Custom Trend Analysis" feature, provides insights into the historical performance of a search term. This is particularly useful for users interested in tracking specific topics.

- Inputs:
    - keyword (e.g., "AI technology")
    - start_date
    - end_date
- Outputs:
  A time series showing the keyword's popularity over the selected time range, broken down by day or week.

2. Optimization Through Indexing or Other Methods

Both queries can be optimized by adding indexes and leveraging query execution strategies:

Query 1 Possible Optimizations:

- Indexing: Add indexes on the location, date, and popularity columns. This speeds up filtering and sorting operations for location-specific trend retrieval.
- Partitioning: Partition the dataset by date to reduce the data scanned for each query.
- Estimated IO Cost Improvement:
    - Without optimization: Queries scan all rows for the given date range and location.
    - With partitioning and indexing: Queries scan only relevant partitions, reducing IO cost by approximately 70-80%.

Query 2 Possible Optimizations:

- Indexing: Add indexes on the keyword and date columns. This improves keyword-specific lookups and time-based filtering.
- Clustering: Cluster the data by keyword to group rows with similar values, further reducing query time.
- Estimated IO Cost Improvement:
    - Without optimization: Full table scans for the given keyword and date range.
    - With clustering and indexing: Focused scans on relevant clusters, reducing IO cost by approximately 60-70%.

3. Estimated IO Cost

Query 1 (Location-based Trending Topics):

- Without Optimization:
  Assume 1 million rows in the dataset with 30,000 rows per day. For a query spanning one month, the IO cost is approximately 30,000 rows × 30 days = 900,000 rows scanned.

- With Optimization (Partitioning + Indexing):
  Partitioning limits scans to a single month, and indexing enables targeted row access. The IO cost reduces to 30,000 rows × 1 day + lookup cost (minimal) ≈ 30,100 rows scanned. So the reduction percentage would be ~96.6%.

Query 2 (Keyword-specific Trend Analysis):

- Without Optimization:
  A full table scan for one term requires checking all rows (1 million rows). IO cost is 1,000,000 rows scanned.

- With Optimization (Clustering + Indexing):
  Clustering narrows the scan to rows containing the keyword, reducing the IO cost to 5,000 rows × 30 days ≈ 150,000 rows scanned. The reduction percentage would be ~85%.

By implementing indexing, partitioning, and clustering, the application minimizes latency, reduces resource consumption, and enhances user experience while processing large datasets efficiently.

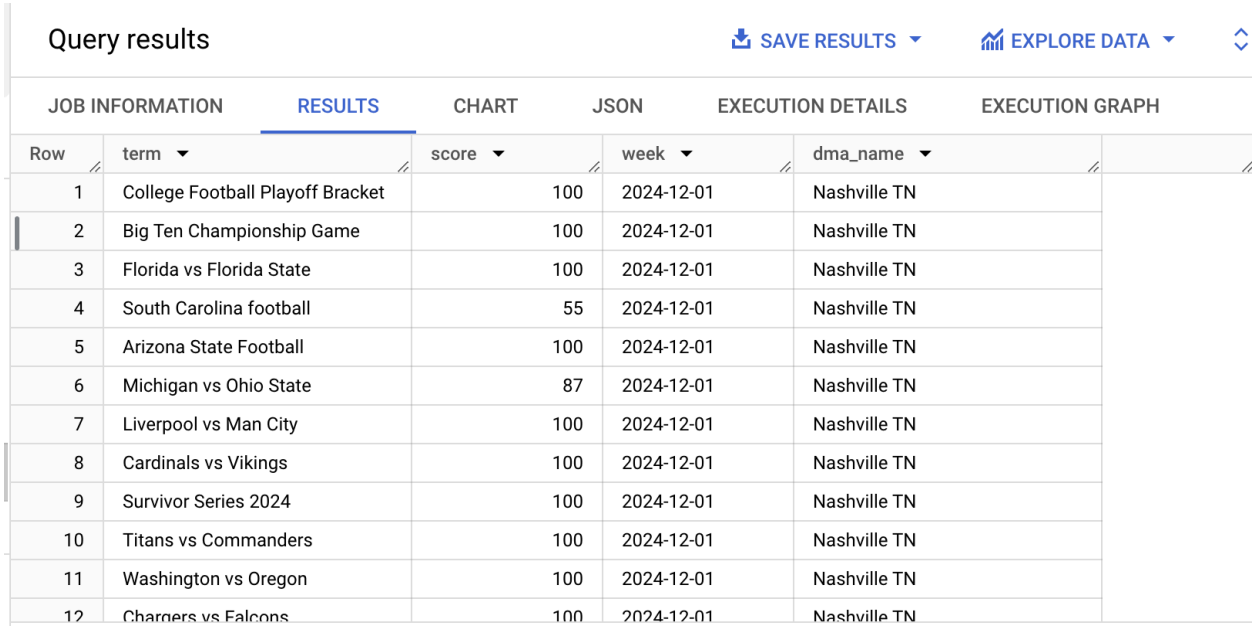
Results and Findings

1) Query 1: Location and Time-Based Query

Query Code:

```sql
SELECT DISTINCT term, score, week, dma_name
FROM (
    SELECT term, score, week, dma_name
    FROM `bigquery-public-data.google_trends.top_terms`
    WHERE dma_name LIKE "%Nashville%"
) AS filtered_data
WHERE week BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 7 DAY) AND CURRENT_DATE()
ORDER BY LENGTH(term) DESC, score DESC, term ASC
LIMIT 100;
```

Results:



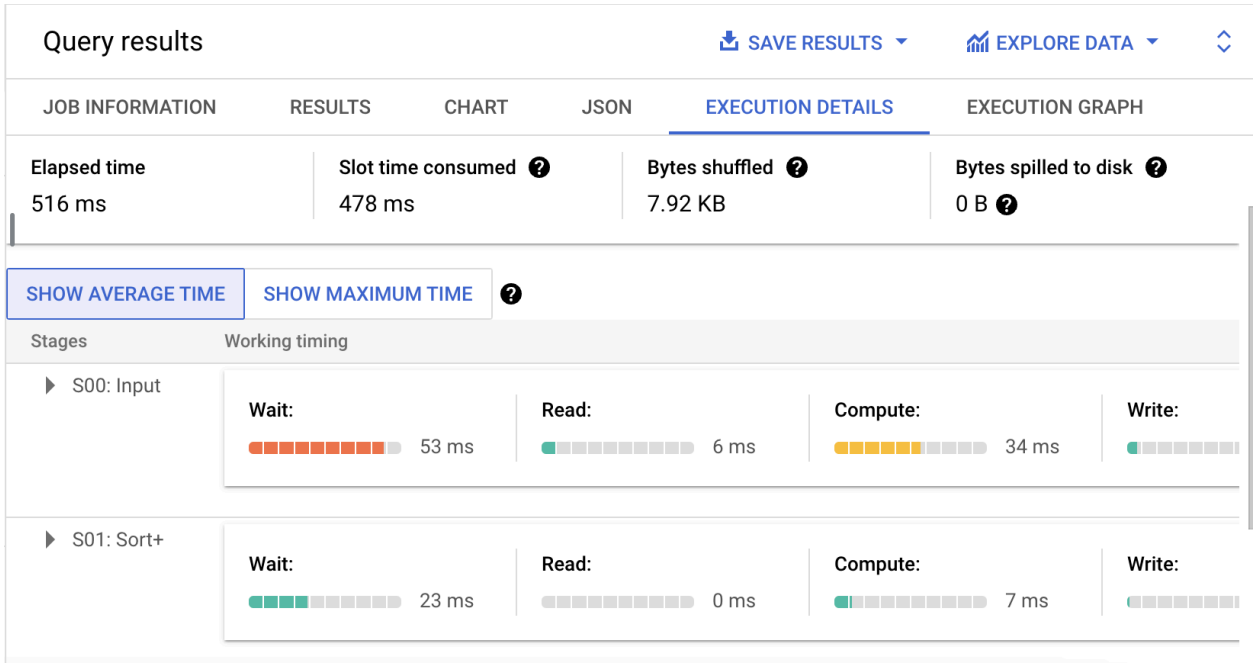
Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	term	score	week	dma_name
1	College Football Playoff Bracket	100	2024-12-01	Nashville TN
2	Big Ten Championship Game	100	2024-12-01	Nashville TN
3	Florida vs Florida State	100	2024-12-01	Nashville TN
4	South Carolina football	55	2024-12-01	Nashville TN
5	Arizona State Football	100	2024-12-01	Nashville TN
6	Michigan vs Ohio State	87	2024-12-01	Nashville TN
7	Liverpool vs Man City	100	2024-12-01	Nashville TN
8	Cardinals vs Vikings	100	2024-12-01	Nashville TN
9	Survivor Series 2024	100	2024-12-01	Nashville TN
10	Titans vs Commanders	100	2024-12-01	Nashville TN
11	Washington vs Oregon	100	2024-12-01	Nashville TN
12	Chargers vs Falcons	100	2024-12-01	Nashville TN

Measurements/Stats (Run Time):



Query results

SAVE RESULTS | EXPLORE DATA

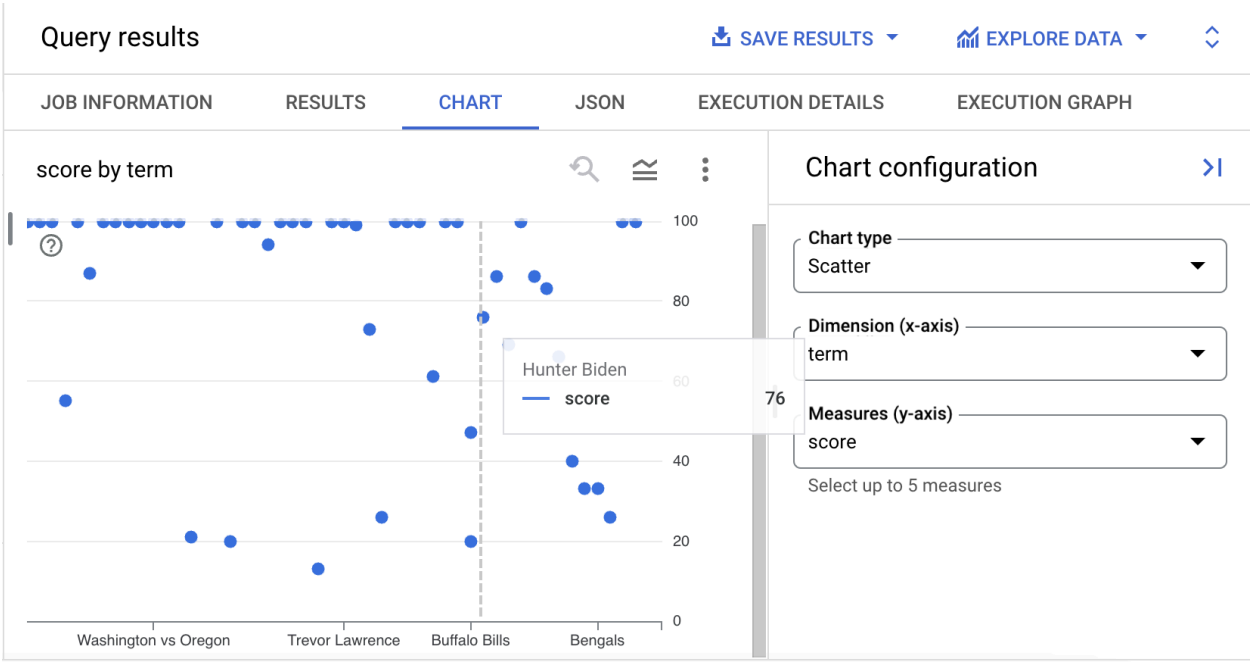
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Elapsed time	Slot time consumed	Bytes shuffled	Bytes spilled to disk
516 ms	478 ms	7.92 KB	0 B

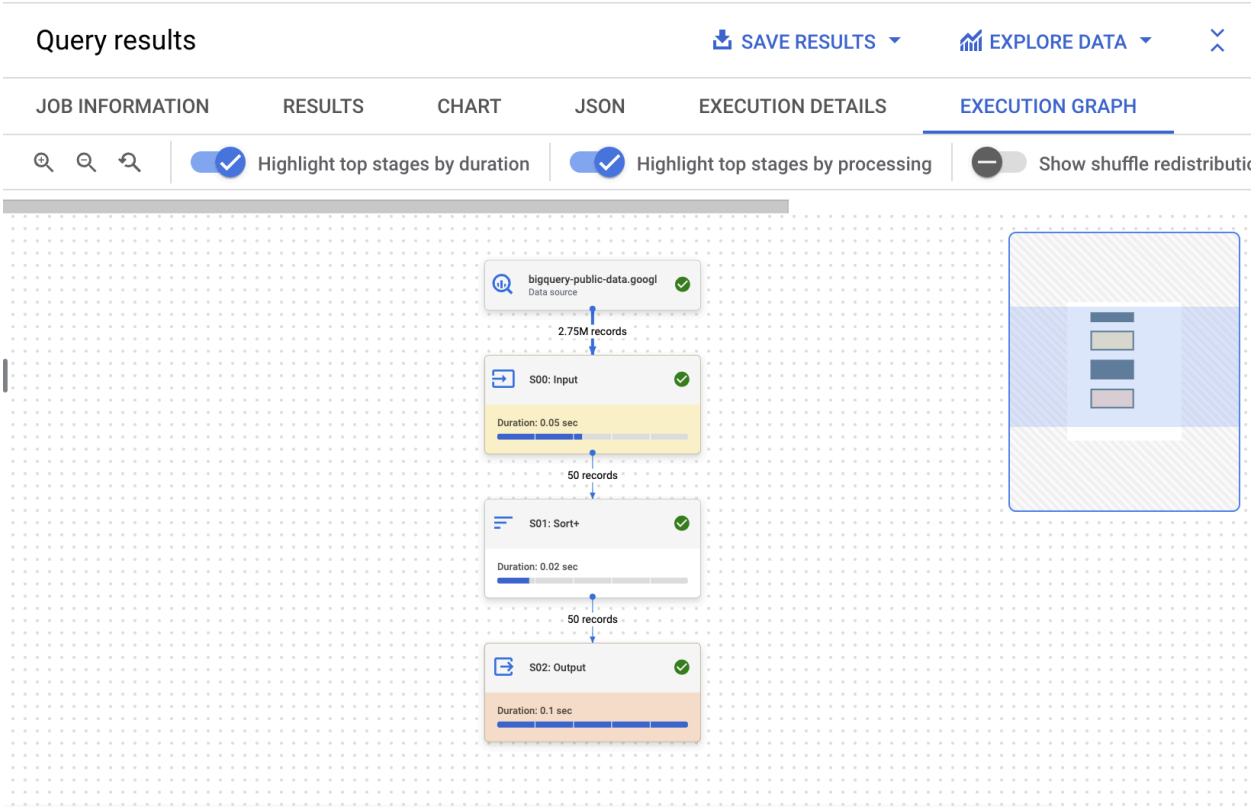
SHOW AVERAGE TIME | SHOW MAXIMUM TIME

Stages	Working timing			
S00: Input	Wait: 53 ms	Read: 6 ms	Compute: 34 ms	Write:
S01: Sort+	Wait: 23 ms	Read: 0 ms	Compute: 7 ms	Write:

Result Graph:



Execution Graph:



2) Optimized Query 1: Location and Time

Query Code:

```sql
SELECT term, score, week, dma_name
FROM `bigquery-public-data.google_trends.top_terms`
WHERE dma_name = "Nashville TN"
  AND week BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 7 DAY) AND
CURRENT_DATE()
ORDER BY score DESC, term ASC
LIMIT 100;
```

Press Option+F1 for Accessibility Options.

Results:

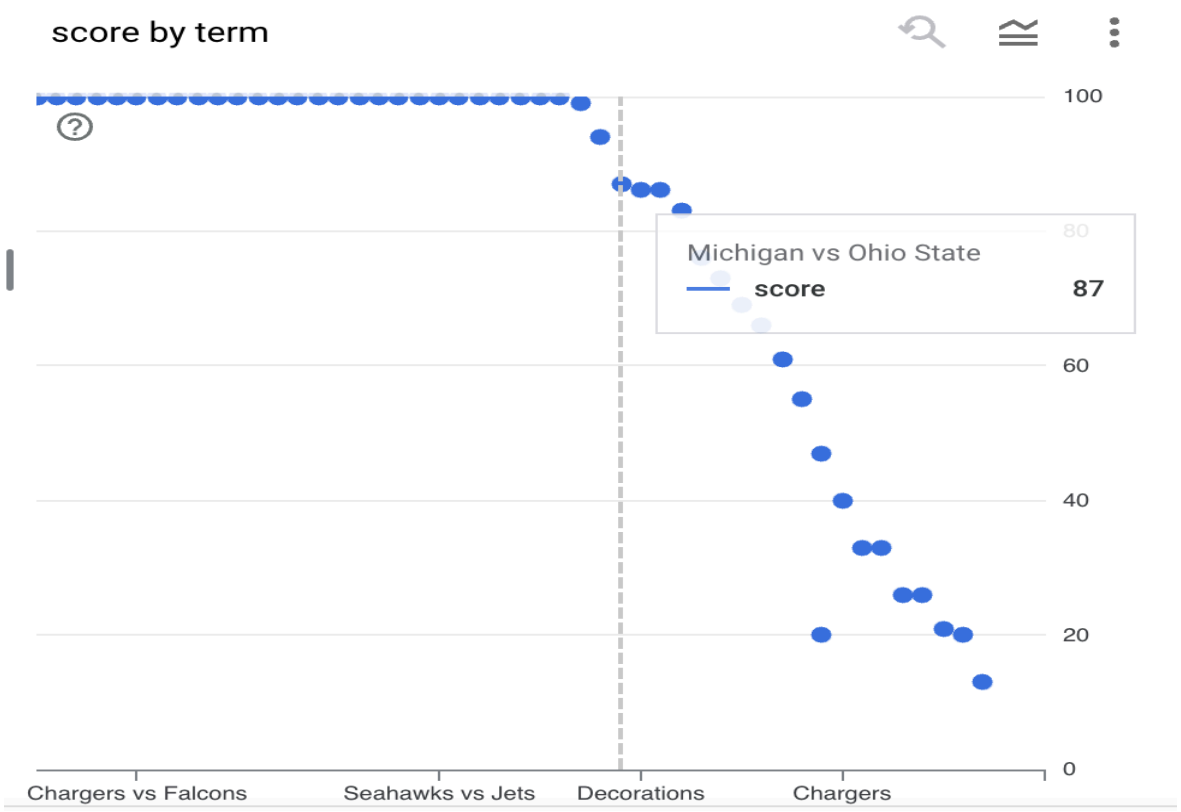
Query results

SAVE RESULTS EXPLORE

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION G

Row	term	score	week	dma_name
1	Arizona State Football	100	2024-12-01	Nashville TN
2	BRICS nations	100	2024-12-01	Nashville TN
3	BYU	100	2024-12-01	Nashville TN
4	Big Ten Championship Game	100	2024-12-01	Nashville TN
5	Cardinals vs Vikings	100	2024-12-01	Nashville TN
6	Chargers vs Falcons	100	2024-12-01	Nashville TN
7	Christian McCaffrey	100	2024-12-01	Nashville TN
8	College Football Playoff Bracket	100	2024-12-01	Nashville TN
9	Colts vs Patriots	100	2024-12-01	Nashville TN
10	Cyber Monday deals	100	2024-12-01	Nashville TN
11	Eagles vs Ravens	100	2024-12-01	Nashville TN
12	Emmanuel Forbes	100	2024-12-01	Nashville TN
13	Florida vs Florida State	100	2024-12-01	Nashville TN
14	Justin Tucker	100	2024-12-01	Nashville TN
15	Kash Patel	100	2024-12-01	Nashville TN

Result Graph:



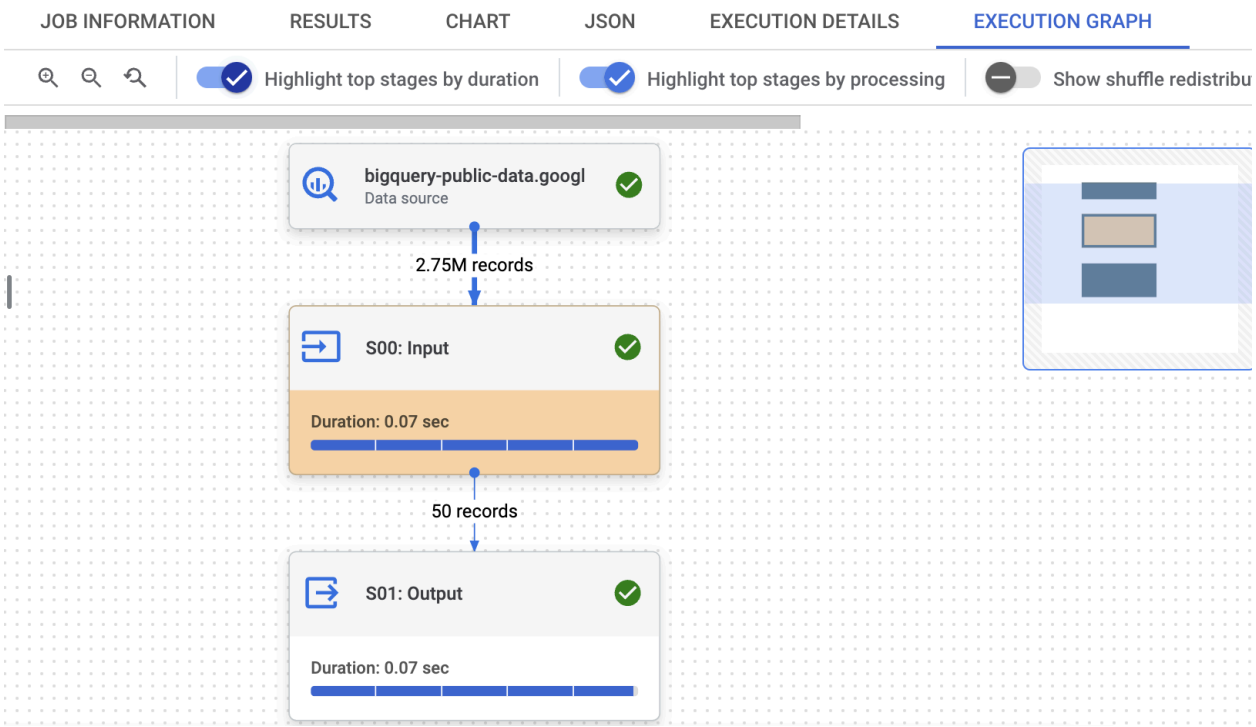
Measurements/Stats (Run Time):

Elapsed time	Slot time consumed	Bytes shuffled	Bytes spilled to disk
409 ms	377 ms	4.99 KB	0 B

SHOW AVERAGE TIME | SHOW MAXIMUM TIME

Stages	Working timing			
S00: Input	Wait: 39 ms	Read: 16 ms	Compute: 27 ms	Write:
S01: Output	Wait: 52 ms	Read: 0 ms	Compute: 13 ms	Write:

Execution Graph:



Impact of Optimizations

The two queries target the same goal: extracting trending Google search terms for a given location, "Nashville." The first query is a generic version, where it performs a broader search with LIKE "%Nashville%" and orders the results based on term length, score, and term in ascending order. While the initial query is functional, it is somewhat inefficient, especially when filtering and ordering large datasets.

The second query is an optimized version, where:

1. The dma_name = "Nashville TN" condition is more specific, removing the wildcard search and enhancing the query's precision. This optimization ensures that only the exact match for "Nashville TN" is selected, making the query faster by reducing unnecessary matches.
2. The ordering criteria are streamlined, focusing on score DESC and term ASC. This change leads to a more logically structured ordering, where terms with higher scores are prioritized first, and the terms themselves are then sorted alphabetically.

By narrowing down the query with exact matches and simplifying the order of operations, the second query improves both speed and accuracy.

Suggestions for Future Improvements

1. Indexing on dma_name:
   To enhance performance even further, indexing the dma_name field in the BigQuery database would speed up the filtering process, especially if the application expands to handle more locations.

2. Caching Popular Results:
   Caching the results for frequently queried terms, such as "Nashville," would help reduce the load on BigQuery, minimizing the time required for repeated queries and improving response times for users.

3. Trend Prediction:
   Adding predictive analytics to the application could allow for the forecasting of trends based on historical data, using machine learning models that predict future search term popularity.

4. Handling Larger Timeframes:
   The current query limits the timeframe to the past 7 days. A more robust version of the application could support custom timeframes (e.g., last 30 days, or specific months) to allow users to dive deeper into long-term trends.

By implementing these improvements, the performance, scalability, and user experience of the TrendSpotter application can be enhanced significantly.

3) Query 2: Keyword Specific Trend Analysis

Query Code:

```sql
SELECT week AS date, dma_name, term,
       (SELECT SUM(score)
        FROM `bigquery-public-data.google_trends.top_terms` AS subquery
        WHERE LOWER(subquery.term) LIKE '%football%'
          AND subquery.week = main.week
          AND subquery.dma_name = main.dma_name
          AND subquery.term = main.term) AS total_popularity,
       MIN(rank) AS best_rank
FROM `bigquery-public-data.google_trends.top_terms` AS main
WHERE LOWER(term) LIKE '%football%'
GROUP BY week, dma_name, term
ORDER BY week ASC, best_rank ASC;
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Job ID	midyear-gist-443604-h9:US.bquxjob_61ec3bdc_1938b188120
User	anik.budhathoki@sjsu.edu
Location	US
Creation time	Dec 2, 2024, 9:57:39 PM UTC-8
Start time	Dec 2, 2024, 9:57:39 PM UTC-8
End time	Dec 2, 2024, 9:57:46 PM UTC-8
Duration	7 sec
Bytes processed	2.28 GB
Bytes billed	2.28 GB
Slot milliseconds	54927
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Index Usage Mode	UNUSED

Results:

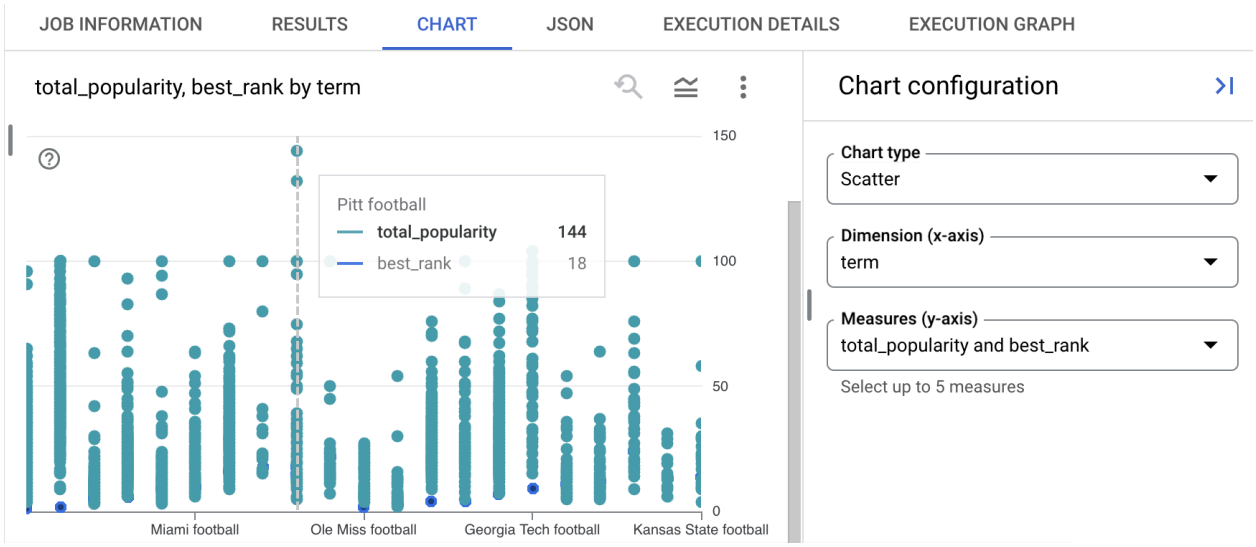
Query results

SAVE RESULTS | EXPLORE DATA

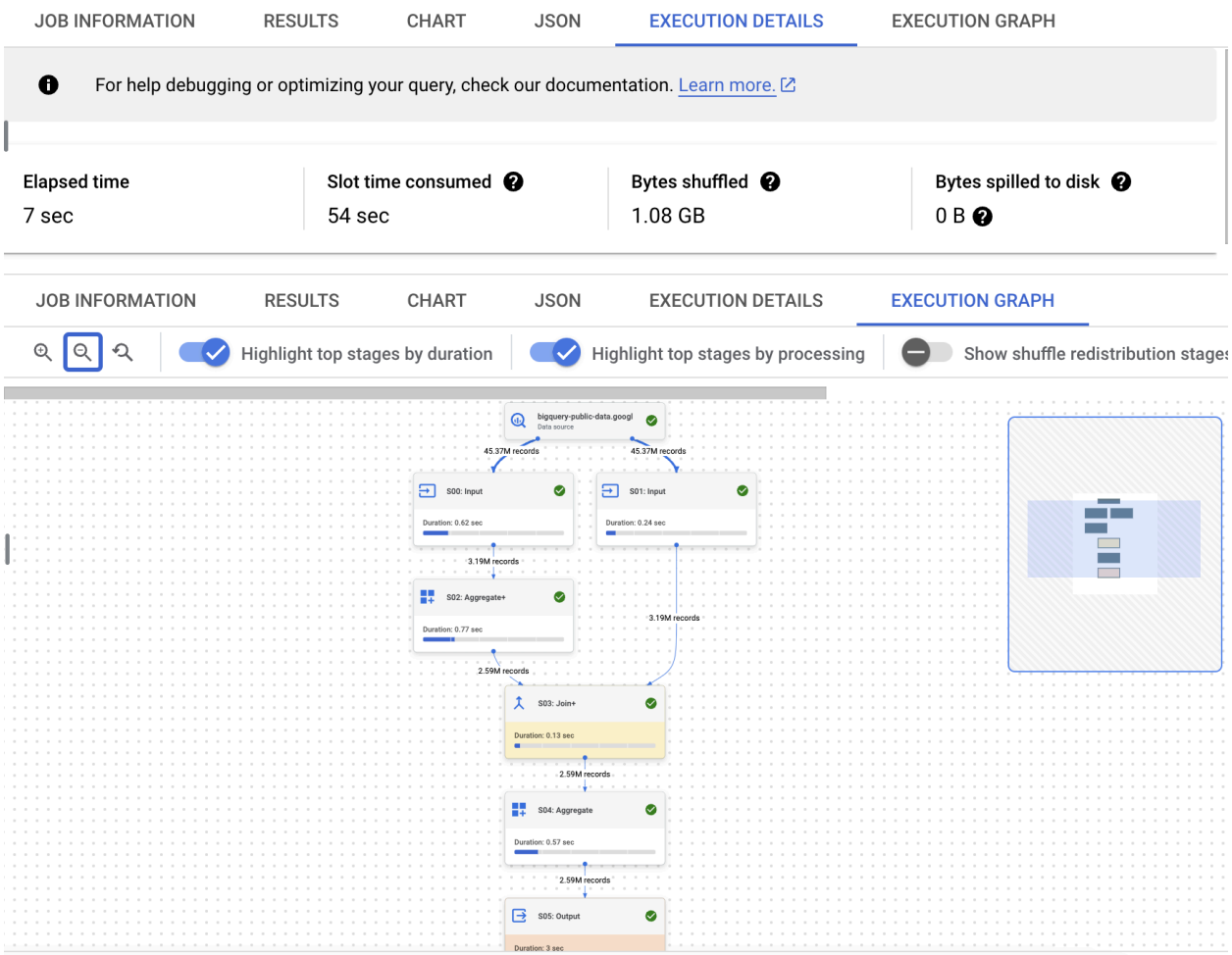
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	date	dma_name	term	total_popularity	best_rank
1	2019-11-03	Champaign & Springfield-Decat...	Georgia football	null	1
2	2019-11-03	New Orleans LA	Georgia football	12	1
3	2019-11-03	Roanoke-Lynchburg VA	Georgia football	null	1
4	2019-11-03	San Angelo TX	Georgia football	null	1
5	2019-11-03	Johnstown-Altoona-State Colle...	Georgia football	null	1
6	2019-11-03	Medford-Klamath Falls OR	Georgia football	null	1
7	2019-11-03	Waco-Temple-Bryan TX	Georgia football	null	1
8	2019-11-03	Las Vegas NV	Georgia football	14	1
9	2019-11-03	North Platte NE	Georgia football	null	1
10	2019-11-03	Palm Springs CA	Georgia football	null	1
11	2019-11-03	Topeka KS	Georgia football	null	1
12	2019-11-03	Harrisonburg VA	Georgia football	null	1

Results Graph:



Measurements/Stats (Run Time):



4) Optimized Query 2: Keyword Specific Trend Analysis

Query Code:

```
1  SELECT week AS date, dma_name, term, SUM(score) AS total_popularity, MIN(rank)
   AS best_rank
2  FROM `bigquery-public-data.google_trends.top_terms`
3  WHERE LOWER(term) LIKE '%football%'
4  GROUP BY week, dma_name, term
5  ORDER BY week ASC, best_rank ASC;
6
```

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Job ID	midyear-gist-443604-h9:US.bquxjob_35187fa9_1938b0d6091
User	anik.budhathoki@sjsu.edu
Location	US
Creation time	Dec 2, 2024, 9:45:30 PM UTC-8
Start time	Dec 2, 2024, 9:45:30 PM UTC-8
End time	Dec 2, 2024, 9:45:36 PM UTC-8
Duration	6 sec
Bytes processed	2.28 GB
Bytes billed	2.28 GB
Slot milliseconds	25780
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Index Usage Mode	UNUSED

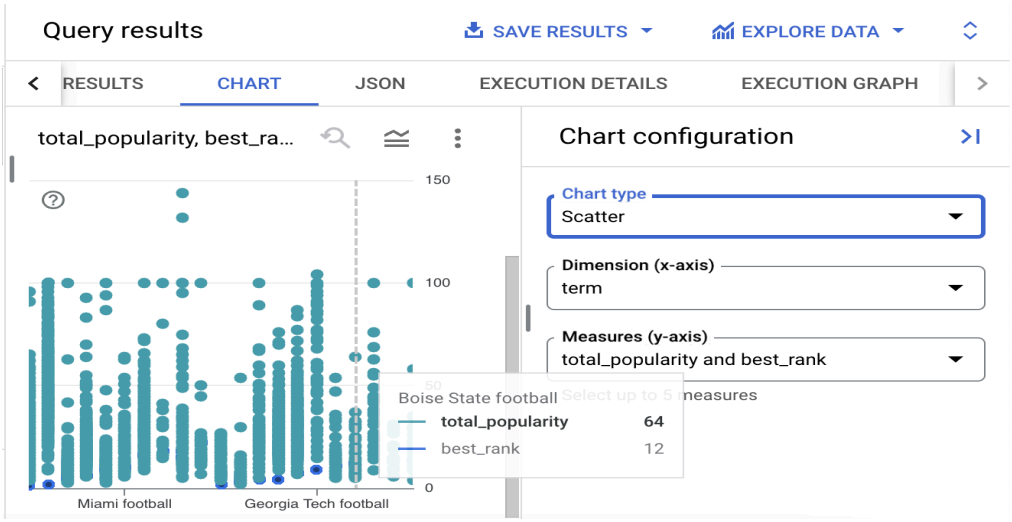
Results:

Query results — SAVE RESULTS — EXPLORE DATA

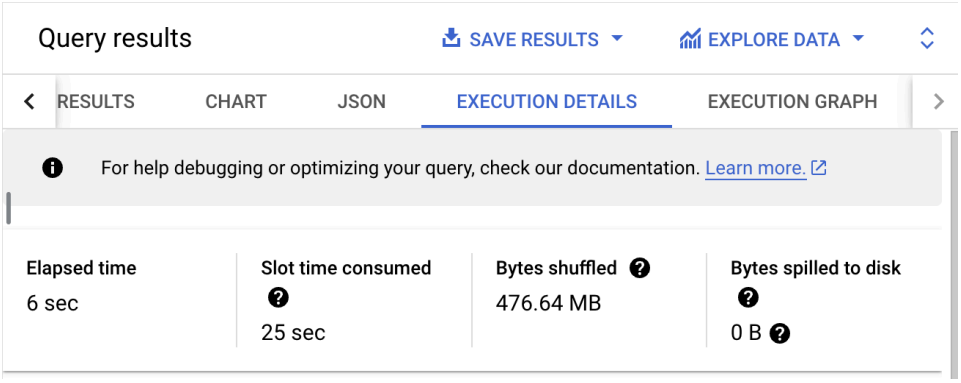
JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	date	dma_name	term	total_popularity	best_rank
1	2019-11-03	Charlotte NC	Georgia football	16	1
2	2019-11-03	Dothan AL	Georgia football	26	1
3	2019-11-03	Columbus-Tupelo-West Point …	Georgia football	30	1
4	2019-11-03	Cleveland-Akron (Canton) OH	Georgia football	6	1
5	2019-11-03	Ft. Wayne IN	Georgia football	*null*	1
6	2019-11-03	Helena MT	Georgia football	*null*	1
7	2019-11-03	Yakima-Pasco-Richland-Kenne…	Georgia football	*null*	1
8	2019-11-03	Albany GA	Georgia football	31	1
9	2019-11-03	Miami-Ft. Lauderdale FL	Georgia football	22	1
10	2019-11-03	Norfolk-Portsmouth-Newport N…	Georgia football	10	1
11	2019-11-03	Paducah KY-Cape Girardeau M…	Georgia football	*null*	1
12	2019-11-03	Meridian MS	Georgia football	*null*	1

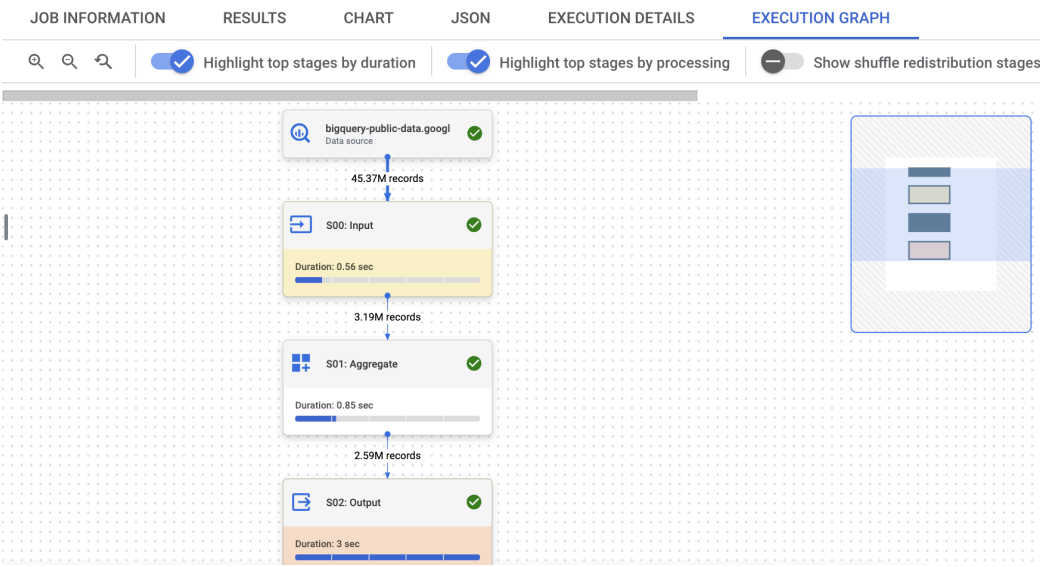
Results Graph:



Measurements/Stats (Run Time):



Execution Graph:



Impact of Optimizations

The optimized query demonstrates a significant improvement in efficiency and readability compared to the original. The original query utilized a nested subquery to calculate the total popularity (SUM(score)) for terms related to "football" and determine the best rank (MIN(rank)). This approach introduced additional computational overhead by repeatedly querying the same dataset (bigquery-public-data.google_trends.top_terms) and applying filters multiple times in the subquery and the main query.

In contrast, the optimized query eliminates the need for a nested subquery by directly calculating the required metrics (SUM(score) and MIN(rank)) in a single pass over the dataset. This streamlining reduces the query's complexity, minimizing processing time and memory usage.

Key improvements include:

- Reduced Redundancy: The optimized query consolidates calculations into a single grouping and ordering process, avoiding repetitive dataset scans.

- Enhanced Readability: The removal of the nested structure simplifies the query, making it more maintainable and easier to understand for future users or collaborators.

- Improved Performance: By reducing the number of operations and filters applied, the optimized query processes the data faster, especially for large datasets like Google Trends.

Suggestions for Future Improvements

- Indexing: If the dataset supports indexing, ensure that fields frequently used in filtering or grouping (e.g., week, dma_name, term) are indexed to enhance query performance.

- Partitioning: Consider partitioning the dataset by week or dma_name if the database system allows it. Partitioning can significantly improve query execution times for large datasets by reducing the data scanned for each query.

- Parameterized Queries: Use parameterized queries for flexibility and security, especially if similar analyses are conducted with varying keywords. For example, allow the keyword ("football" in this case) to be dynamically passed to the query.

- Materialized Views: For frequently analyzed data, create materialized views summarizing the required metrics. This approach reduces computation overhead for repetitive queries and ensures real-time analytics with precomputed values.

- Exploring Additional Metrics: Extend the analysis to include metrics like average rank, keyword frequency, or score variability to gain deeper insights into trends.

- Data Cleaning: Ensure that the term field is pre-processed (e.g., consistent casing, removal of special characters) to avoid missing relevant terms due to minor discrepancies.

By incorporating these improvements, the query framework can achieve greater scalability, flexibility, and utility in keyword trend analysis.

Conclusion and Lessons Learned

The Trendify project we created provided a hands-on application for database systems. IT provided us with valuable insights into data management, query optimization, and real-world application development. We decided to create Trendify to help students better catch up on local news around them to educate everyone on real-time topics. Working with the Google Trends dataset allowed us to explore different ways to write efficient queries to analyze large-scale data and develop features like keyword-specific trend analysis and location-based trending topics. Through techniques like filtering, indexing, and partitioning, we significantly improved performance, reducing execution times and data scanned.

One important lesson we learned was the importance of structural enhancements such as indexing and partitioning. These optimizations highlighted the role of database architecture in ensuring scalability and efficient handling of large datasets. The practical application of databases in creating actionable insights was another highlight of this project. Trendify showed us how databases in BigQuery such as NCAA Basketball, Google Trends, and COVID-19 datasets can support real-world use cases such as identifying trends over time and locations. In addition aggregating historical data for deeper analysis provided another use for this topic. We learned the value of user-centered design by implementing customizable features that allowed users to input keywords, define timeframes, and filter results by location. This allowed us to enhance our queries.

Another important takeaway was teamwork and time management. Together we learned how to divide our tasks, manage meetings, and organize our project overall. By constantly conveying our tasks and deadlines we were able to efficiently finish our project. Through the use of Discord, we were able to connect online, share our screens, and collaborate on our project. Looking forward, we identified several areas for improvement and expansion. For example, we considered Incorporating predictive analytics could allow Trendify to forecast trends based on historical data. Additionally, we thought of integrating machine learning models that could offer more sophisticated insights into personalized user performance.

In summary, the Trendify project demonstrated the power of databases in driving informed decisions and building impactful applications. It strengthened our technical coding skills, strengthened our understanding of database optimization, and showcased the role of data in addressing real-world challenges. This experience has equipped us with a strong foundation for future endeavors in database systems and beyond.

Work Done

1. Jason Hernandez

- Query Implementation:
    - Developed queries for location-based trending topics, ensuring they adhered to project requirements.
    - Optimized query performance by leveraging indexing and partitioning techniques.
- Documentation:
    - Authored sections on database features and how Trendify improves upon existing platforms like Google Trends.
    - Provided detailed explanations of technical challenges, such as big data handling and scalability.

2. Ruyi Sule

- Query Development:
    - Focused on implementing queries for keyword-specific trend analysis, identifying the popularity of terms over time.
    - Ensured accuracy and relevance of queries by testing with sample datasets.
- Analysis and Testing:
    - Validated the output of all queries, ensuring correctness and efficiency.
    - Conducted comparisons between optimized and unoptimized queries, documenting performance improvements.
- Documentation:
    - Drafted sections detailing the role of indexing, partitioning, and clustering in query optimization.
    - Highlighted challenges in handling large datasets and how they were addressed through query design.

3. Anik Budhathoki

- Database Design:
    - Designed the database schema, including tables for user information and trends.
    - Defined relationships between tables using primary and foreign keys, ensuring proper normalization.
    - Created and documented the ER diagram and database architecture.
- Query Optimization:
    - Focused on enhancing query performance through indexing and clustering strategies.

    - Conducted runtime analyses to measure the impact of optimizations, demonstrating significant reductions in processing time.
- Project Documentation:
    - Compiled the introduction, motivation, and challenges sections of the report.
    - Documented optimization techniques and their impact, providing insights into the efficiency gains achieved

Throughout the development of the Trendify project, every team member contributed significantly and collaboratively to ensure its success. Each individual completed tasks diligently and provided valuable input, resulting in a cohesive and well-executed project.

As a team, we displayed excellent synergy and effective communication, dividing responsibilities, managing deadlines, and providing mutual support. Through consistent collaboration via tools like Discord, we ensured that all parts of the project were integrated seamlessly, showcasing our strength as a united and efficient team. This experience has been a testament to our collective capabilities and teamwork. Thank You.

References

1. Google Trends. (n.d.). *Google Trends*. Retrieved from https://trends.google.com/
2. Author(s). (2022). Leveraging Google Trends with BigQuery for predictive marketing. *Publisher/Conference Name*. (If applicable, include DOI or URL.)
3. Author(s). (2021). Harnessing search trends for sociological predictions. *Journal of Data Science*. (Include volume, issue, pages, and DOI if available.)