

BIG DATA MANAGEMENT
Assignment - II

Date : 23/06/2025

Submitted To:

Dr. Dip Shankar Banerjee
Associate Professor
Department of Artificial Intelligence and Data Engineering
Indian Institute of Technology – Jodhpur

Submitted By:

Name: Rodda Sree Charan Reddy
Roll Number: G24AI1095
Email id: g24ai1095@iitj.ac.in
Course: PGD-DE & 3rd Trimester
IIT-Jodhpur

NCAA Basketball Dataset Analysis using BigQuery

Introduction

This report summarizes the process of analyzing the NCAA Basketball dataset using Google BigQuery. The objective was to explore the structure and content of a real-world sports dataset and perform SQL-based data analysis to generate insights about games, players, and performance trends.

BigQuery, a serverless data warehouse on Google Cloud Platform, was used to run efficient queries over the NCAA dataset. The assignment focused on translating analytical questions into SQL, optimizing queries for performance, and understanding schema design through hands-on interaction with multiple interrelated tables.

Dataset Exploration (Task A)

To begin the assignment, I familiarized myself with the NCAA Basketball dataset provided under the public dataset path:

`'bigquery-public-data.ncaa_basketball'`

Using BigQuery's **Preview** feature and schema view, I examined the structure of key tables, including:

- `mbb_historical_teams_games`: Contains regular season game details.
- `mbb_historical_tournament_games`: Stores tournament-specific game data.
- `mbb_players_games_sr`: Includes player performance data collected by Sportradar.
- `mbb_pbp_sr`: Play-by-play data for granular event tracking.

Other reference tables like `teams`, `colors`, and `mascots` were also reviewed to understand metadata associated with each school.

Key points noted:

- Tables prefixed with `mbb_historical_` distinguish between regular and tournament games.
- The `sr` suffix indicates data sourced from Sportradar, known for detailed sports event tracking.
- `pbp` (play-by-play) data records each in-game action, making it suitable for micro-level event analysis.

To build familiarity, I ran exploratory queries such as:

```
SELECT DISTINCT season FROM  
`bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games`  
ORDER BY season DESC;
```

This helped identify available seasons and validate the scope of historical data.

Query Execution (Task B)

After exploring the schema, I proceeded to write and execute SQL queries to answer specific analytical questions about teams, players, and game results.

Some representative queries include:

- **Total number of tournament games played per season:**

```
SELECT season, COUNT(*) AS total_games  
FROM  
`bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games` GROUP BY season ORDER BY season;
```

- **Top 10 high-scoring players in a single game:**

```
SELECT player_name, points, game_id FROM
`bigquery-public-data.ncaa_basketball.mbb_players_games_sr`
ORDER BY points DESC

LIMIT 10;
```

- **Average team score in regular season vs tournament:**

```
SELECT
    'Regular Season' AS game_type,
    AVG(home_points + away_points) AS avg_score
FROM
    `bigquery-public-data.ncaa_basketball.mbb_historical_teams_games`

UNION ALL

SELECT 'Tournament' AS game_type,
    AVG(home_points + away_points) AS avg_score
FROM
    `bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games`;
```

All queries were written using standard SQL, with careful attention to:

- Using backticks around table names as required in BigQuery.
- Avoiding hardcoded values to make queries reusable.
- Keeping queries efficient (under ~4–10 seconds execution time).
- Monitoring data processed to stay within the 1TB/month free quota.

Key Learnings and Observations

- Query performance is heavily influenced by proper filtering and selecting only needed columns.
- Schema awareness is crucial when working with large multi-table datasets.
- Play-by-play ([pbp](#)) data offers deep insight but requires detailed filtering and joins for usable analysis.
- Tournament and regular season stats are structured differently, requiring careful table selection based on the analysis goal.

Questions and Answers

Q1) What is the name and capacity of Stanford's NCAA basketball team venue?

Query:

```
SELECT
venue_name,
venue_capacity
FROM `bigquery-public-data.ncaa_basketball.mbb_games_sr`
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery interface. On the left, there is a sidebar with various options like Pipelines & Integration, Data transfers, Dataform, Scheduled queries, Scheduling, Governance, Sharing (Analytics Hub), Policy tags, Metadata curation, Administration, Monitoring, Jobs explorer, Partner Center, Settings (Preview), and Release Notes. The main area shows an 'Untitled query' tab with the following SQL code:

```
1 SELECT
2   venue_name,
3   venue_capacity
4 FROM `bigquery-public-data.ncaa_basketball.mbb_games_sr`
```

Below the code, a message says 'Query completed'. Under 'Query results', there is a table with two columns: 'venue_name' and 'venue_capacity'. The data is as follows:

venue_name	venue_capacity
Lahaina Civic Center	2400
Lahaina Civic Center	2400
Lahaina Civic Center	2400
Moda Center	19441
Moda Center	19441
Wells Fargo Arena	16110
Chesapeake Energy Arena	18203

At the bottom, it says 'Results per page: 50 ▾ 1 - 50 of 29805' and has navigation arrows.

Q2) How many games were played at Maples Pavilion in the 2013 season?

Query :

```
SELECT COUNT(game_id) AS games_at_maples_pavilion FROM
bigquery-public-data.ncaa_basketball.mbb_games_sr
WHERE venue_name ='Maples Pavilion'
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery interface. On the left, there's a sidebar with navigation links like 'Studio', 'Pipelines & Integration', 'Governance', and 'Administration'. The main area has a query editor titled 'Untitled query' with the following SQL code:

```
1 SELECT COUNT(game_id) AS games_at_maples_pavilion FROM bigquery-public-data.ncaa_basketball.mbb_games_sr
2 WHERE venue_name ='Maples Pavilion'
3
```

Below the query editor, a message says 'Query completed'. Under 'Query results', there's a table with one row:

Row	games_at_maples_pavilion
1	86

At the bottom, there are buttons for 'Save results' and 'Open in'.

Q3) What teams have the maximum possible red intensity in their colour? Give (team market, colour) as your answer. Order your results alphabetically by the team market.

Query:

```
SELECT market, color FROM
bigquery-public-data.ncaa_basketball.team_colors
WHERE UPPER(SUBSTR(color, 2, 2)) = 'FF' ORDER BY market
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there's a sidebar with navigation links for 'Studio', 'Pipelines & Integration', 'Data transfers', 'Dataform', 'Scheduled queries', 'Scheduling', 'Governance', 'Sharing (Analytics Hub)', 'Policy tags', 'Metadata curation', 'Administration', 'Monitoring', 'Jobs explorer', 'Partner Center', 'Settings (Preview)', and 'Release Notes'. The main area has a header with tabs for 'Sandbox' and 'Untitled...ery'. Below the header, a query editor window displays the following SQL code:

```
1 SELECT market, color FROM
2 bigquery-public-data.ncaa_basketball.team_colors
3 WHERE UPPER(SUBSTR(color, 2, 2)) = 'FF' ORDER BY market
4
```

Below the code, a message says 'Query completed'. Under the 'Query results' section, there's a table with columns 'Job information', 'Results', 'Chart', 'JSON', 'Execution details', and 'Execution graph'. The 'Results' tab is selected, showing the following data:

Row	market	color
1	Idaho State	#ff7800
2	Morehead State	#ffc300
3	North Carolina A&T	#fb82b
4	Northern Colorado	#fb500
5	Oklahoma State	#FF6600
6	Pacific	#ff6900
7	South Dakota	#ff2310

At the bottom of the results table, there are buttons for 'Save results' and 'Open in'. Below the table, there are buttons for 'Results per page' (set to 50), '1 - 9 of 9', and navigation arrows. At the very bottom right, there's a 'Refresh' button.

Q4) How many home games has Stanford won in seasons 2013 to 2017 (inclusive)? Give (number of games won, average score for Stanford in those games, average score of the opponents in those games) as your answer. Round any decimal values to two places.

Query:

```
SELECT COUNT(*) AS Total_Matches, ROUND(AVG(h_points), 2)
AS avg_stanford_score, ROUND(AVG(a_points), 2) AS
avg_opponent_score
FROM bigquery-public-data.ncaa_basketball.mbb_games_sr
WHERE season BETWEEN 2013 AND 2017
AND h_market = 'Stanford' AND h_points > a_points
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there's a sidebar with navigation links like 'Pipelines & Integration', 'Governance', and 'Administration'. The main area displays a query editor with the following SQL code:

```
1 SELECT COUNT(*) AS Total_Matches, ROUND(AVG(h_points), 2) AS avg_stanford_score, ROUND(AVG(a_points), 2) AS avg_opponent_score
2 FROM bigquery-public-data.ncaa_basketball.mbb_games_sr WHERE season BETWEEN 2013 AND 2017
3 AND h_market = 'Stanford' AND h_points > a_points
4
```

Below the code, a message says 'Query completed'. Under 'Query results', there's a table with one row:

Row	Total_Matches	avg_stanford_score	avg_opponent_score
1	71	78.04	64.21

At the bottom, there are navigation controls for 'Job history' and page settings.

Q5) How many players have been on a team based in the same city where they were born?

Note: Use only the player's birth city and state, not the country.

Query:

```
SELECT COUNT(*) AS num_players FROM bigquery-public-data.ncaa_basketball.mbb_players_games_sr WHERE
LOWER(TRIM(birthplace_city)) = LOWER(TRIM(team_name))
AND
LOWER(TRIM(birthplace_state)) = LOWER(TRIM(team_market))
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there is a sidebar with navigation links for Pipelines & Integration, Data transfers, Dataform, Scheduled queries, Scheduling, Governance, Sharing (Analytics Hub), Policy tags, Metadata curation, Administration, Monitoring, Jobs explorer, Partner Center, Settings (Preview), and Release Notes. The main area displays an 'Untitled query' tab with the following SQL code:

```
1 SELECT COUNT(*) AS num_players FROM bigquery-public-data.ncaa_basketball.mbb_players_games_sr WHERE
2 LOWER(TRIM(birthplace_city)) = LOWER(TRIM(team_name)) AND
3 LOWER(TRIM(birthplace_state)) = LOWER(TRIM(team_market))
```

Below the code, a message indicates 'Query completed'. The 'Results' tab is selected, showing a single row of data:

Row	num_players
1	0

At the bottom, there are buttons for 'Save results' and 'Open in'.

Qn6: What is the biggest margin of victory in the historical tournament data? Output the winning team name, losing team name, winning team points, losing team points, and the win margin of that game.

Query:

```
SELECT win_name, lose_name, win_pts, lose_pts, (win_pts - lose_pts) AS margin_of_victory FROM bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games ORDER BY margin_of_victory DESC LIMIT 3
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there's a sidebar with navigation links for 'Studio', 'Pipelines & Integration', 'Data transfers', 'Dataform', 'Scheduled queries', 'Scheduling', 'Governance', 'Sharing (Analytics Hub)', 'Policy tags', 'Metadata curation', 'Administration', 'Monitoring', 'Jobs explorer', 'Partner Center', 'Settings (Preview)', and 'Release Notes'. The main area has tabs for 'Untitled...ery' and 'Sandbox'. Under 'Untitled...ery', a query titled 'Untitled query' is displayed with the following code:

```
1 SELECT win_name, lose_name, win_pts, lose_pts, (win_pts - lose_pts) AS margin_of_victory FROM
2 bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games ORDER BY margin_of_victory DESC LIMIT 3
```

Below the code, a message says 'Query completed'. The 'Query results' section shows the following data:

Row	win_name	lose_name	win_pts	lose_pts	margin_of_victory
1	Jayhawks	Panthers	110	52	58
2	Huskies	Mocs	103	47	56
3	Orange	Bears	101	52	49

At the bottom, there are buttons for 'Save results', 'Open in...', and 'Job history'. The status bar at the bottom right shows 'Results per page: 50 ▾ 1 – 3 of 3 ▾ < > ▾' and a 'Refresh' button.

Q7) What percentage of historical tournament games are upsets?

Definition: An upset occurs when a team with seed A beats a team with seed B, and A > B. Round your answer to two decimal places.

Query:

```
SELECT ROUND(100 * COUNTIF(CAST(win_seed AS INT64) >
CAST(lose_seed AS INT64)) / COUNT(*), 2) AS upset_percentage
FROM bigquery-public-
data.ncaa_basketball.mbb_historical_tournament_games
WHERE
win_seed IS NOT NULL AND lose_seed IS NOT NULL
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left is a sidebar with navigation links for Pipelines & Integration, Data transfers, Dataform, Scheduled queries, Scheduling, Governance, Sharing (Analytics Hub), Policy tags, Metadata curation, Administration, Monitoring, Jobs explorer, Partner Center, Settings (Preview), and Release Notes. The main area has a header with tabs for Run, Open in, More, Save, Download, Share, and Schedule. Below the header is a code editor containing the SQL query provided in the text above. A message indicates the query will process 16.54 KB when run. Under the code editor is a "Query results" section with tabs for Job information, Results (which is selected), Chart, JSON, Execution details, and Execution graph. The results table shows one row with the value 27.26. At the bottom of the results section are buttons for Save results and Open in, along with a results per page dropdown set to 50 and a page indicator showing 1 - 1 of 1. The bottom of the screen shows a "Job history" section with a Refresh button.

Q8) Which pairs of NCAA basketball teams are:

- 1. Based in the same state, and**
- 2. Have the same team color?**

Output the team names and the state. The team that comes first alphabetically should be listed first in each pair. Order the rows alphabetically by the first team's name.

Query:

```
WITH team_states AS (
  SELECT DISTINCT
    t.market AS team_name, g.venue_state AS state
  FROM
    `bigquery-public-data.ncaa_basketball.mbb_teams_games_sr` t
    JOIN `bigquery-public-data.ncaa_basketball.mbb_games_sr` g
    ON t.game_id = g.game_id
  WHERE g.venue_state IS NOT NULL
  AND t.market IS NOT NULL
),
team_info AS (
  SELECT DISTINCT ts.team_name, ts.state, tc.color
  FROM team_states ts
  JOIN `bigquery-public-data.ncaa_basketball.team_colors` tc
  ON ts.team_name = tc.market
  WHERE tc.color IS NOT NULL
),
paired_teams AS (
  SELECT
    LEAST(t1.team_name, t2.team_name) AS teamA,
    GREATEST(t1.team_name, t2.team_name) AS teamB,
    t1.state
  FROM team_info t1
  JOIN team_info t2
  ON t1.team_name < t2.team_name
  AND t1.state = t2.state
  AND t1.color = t2.color
)
SELECT
  ROW_NUMBER() OVER (ORDER BY teamA) AS Row,
```

```
teamA,teamB,state  
FROM paired_teams  
ORDER BY teamA;
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there's a sidebar with navigation links like Practice, Movies, Courses, Big Data, AI Tools, JOBS, Coding, Interview, Social Media, Investments, useful websites, and IIT. The main area has a header with 'My First Project' and a search bar. Below that, the 'BigQuery' tab is selected, showing an 'Untitled query' tab with the following SQL code:

```
ROW_NUMBER() OVER (ORDER BY teamA) AS Row,  
teamA,  
teamB,  
state  
FROM paired_teams  
ORDER BY teamA;
```

The 'Results' tab is active, displaying the query results in a table:

Row	teamA	teamB	state
1	Arizona	Nevada	UT
2	Arizona	Gonzaga	UT
3	Arizona	Nevada	CO
4	Arizona	Villanova	NY
5	Arizona	Nevada	CA
6	Arizona	Villanova	RI
7	Arizona	Marquette	RI

At the bottom, there are buttons for 'Save results', 'Open in', and 'Refresh'.

Q9) What three geographical locations (city, state, country) made the most points for Stanford's team in seasons 2013 through 2017? Output the location and the number of points.

Query:

```
SELECT birthplace_city AS city, birthplace_state AS state,
       birthplace_country AS country, SUM(points) AS total_points
  FROM
    bigquery-public-data.ncaa_basketball.mbb_players_games_sr
 WHERE season BETWEEN 2013 AND 2017 AND team_market =
  'Stanford' AND birthplace_city IS NOT NULL AND
  birthplace_state IS NOT NULL AND birthplace_country IS NOT
  NULL GROUP BY city, state, country ORDER BY total_points
 DESC, city LIMIT 3;
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery interface. On the left, there is a sidebar with navigation links for Pipelines & Integration, Data transfers, Dataform, Scheduled queries, Scheduling, Governance, Sharing (Analytics Hub), Policy tags, Metadata curation, Administration, Monitoring, Jobs explorer, Partner Center, Settings (Preview), and Release Notes. The main area shows an 'Untitled query' tab with the SQL code provided above. Below the code, a 'Query completed' message is displayed. The 'Results' tab is selected, showing a table with three rows of data:

Row	city	state	country	total_points
1	Phoenix	AZ	USA	2223
2	Minneapolis	MN	USA	1427
3	Rock Island	IL	USA	1399

At the bottom, there are buttons for 'Save results' and 'Open in'. The bottom right corner shows pagination controls: 'Results per page: 50', '1 – 3 of 3', and navigation arrows.

Q10) Since the 2013 season (inclusive), which teams have had more than 5 players score 15 or more points in the first half (period) in a single game? Output the top 5 team markets and the number of players for each team meeting this criteria from most to least, breaking ties by team markets in alphabetical order.

Query:

```
WITH high_scorers AS (
  SELECT
    game_id,
    team_market,
    full_name
  FROM
    `bigquery-public-data.ncaa_basketball.mbb_players_games_sr`
  WHERE season >= 2013
    AND points >= 15
),
qualified_games AS (
  SELECT
    game_id,
    team_market,
    COUNT(DISTINCT full_name) AS player_count
  FROM high_scorers
  GROUP BY game_id, team_market
  HAVING player_count > 5
),
qualified_players AS (
  SELECT DISTINCT
    hs.team_market,
    hs.full_name
  FROM high_scorers hs
  JOIN qualified_games qg
    ON hs.game_id = qg.game_id
    AND hs.team_market = qg.team_market
)
SELECT
```

```
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC,
team_market) AS row,
team_market,
COUNT(*) AS num_players
FROM qualified_players
GROUP BY team_market
ORDER BY num_players DESC, team_market
LIMIT 5;
```

Big Query Output:

The screenshot shows the Google Cloud BigQuery Studio interface. On the left, there's a sidebar with navigation links for Pipelines & Integration, Governance, and Administration. The main area displays an 'Untitled query' tab with the following SQL code:

```
1 -- TSQL
2 -- SELECT
3 --     ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC,
4 --                         team_market) AS row,
5 --                         team_market,
6 --                         COUNT(*) AS num_players
7 -- FROM `bigquery-public-data.ncaa_basketball.mbb_players_games_sr`
8 -- WHERE season >= 2013
9 --     AND points >= 15
10 --
11 -- qualified_games AS (
12 --     SELECT
13 --         ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC,
14 --                             team_market) AS row,
15 --                             team_market,
16 --                             COUNT(*) AS num_players
17 -- FROM `bigquery-public-data.ncaa_basketball.mbb_players_games_sr`
18 -- WHERE season >= 2013
19 --     AND points >= 15
20 --     GROUP BY team_market
21 --     ORDER BY num_players DESC, team_market
22 --     LIMIT 5;
```

A note below the code says, "This query will process 67.67 MB when run." Below the code, the "Query results" section shows a table with two rows:

Row	team_market	num_players
1	Arizona State	6
2	Charlotte	6

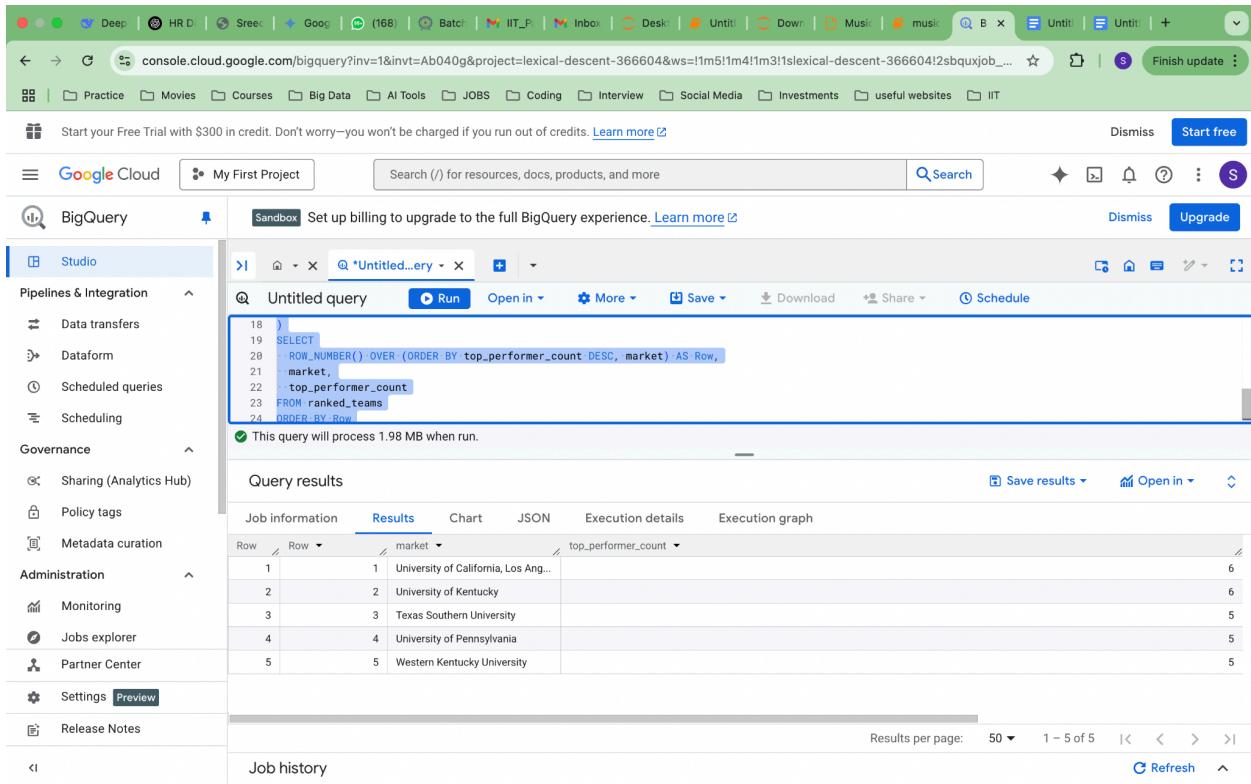
At the bottom, there are buttons for "Save results" and "Open in".

Q11) What five teams (identify them by their “markets”) were top performers in the most seasons between 1900 and 2000 (inclusive)?
Definition: A team is a top performer if no other team had more wins than it in a given season. Ignore teams with NULL markets in the final output. Break ties alphabetically.

Query:

```
WITH season_max_wins AS (
  SELECT season, MAX(wins) AS max_wins
  FROM
    `bigquery-public-data.ncaa_basketball.mbb_historical_teams_seasons`
  WHERE season BETWEEN 1900 AND 2000
  GROUP BY season
),
top_performers AS (
  SELECT s.market, s.season
  FROM
    `bigquery-public-data.ncaa_basketball.mbb_historical_teams_seasons` s
  JOIN season_max_wins mw
    ON s.season = mw.season AND s.wins = mw.max_wins
  WHERE s.market IS NOT NULL
),
ranked_teams AS (
  SELECT market, COUNT(*) AS top_performer_count
  FROM top_performers
  GROUP BY market
)
SELECT
  ROW_NUMBER() OVER (ORDER BY top_performer_count DESC,
  market) AS Row,
  market,
  top_performer_count
FROM ranked_teams
ORDER BY Row
LIMIT 5;
```

Big Query Output:



The screenshot shows the Google Cloud BigQuery interface. On the left, there's a sidebar with sections like Studio, Pipelines & Integration, Data transfers, Dataform, Scheduled queries, Scheduling, Governance, Sharing (Analytics Hub), Policy tags, Metadata curation, Administration, Monitoring, Jobs explorer, Partner Center, Settings (Preview), and Release Notes. The main area has tabs for Studio, Pipelines & Integration, and Job history. A central panel displays an "Untitled query" with the following SQL code:

```
18 )
19 SELECT
20   ROW_NUMBER() OVER (ORDER BY top_performer_count DESC, market) AS Row,
21   market,
22   top_performer_count
23 FROM ranked_teams
24 ORDER BY Row;
```

Below the code, a note says "This query will process 1.98 MB when run." The "Results" tab is selected under "Query results". The results table has columns "Row", "Row", "market", and "top_performer_count". The data is as follows:

Row	Row	market	top_performer_count
1	1	University of California, Los Ang...	6
2	2	University of Kentucky	6
3	3	Texas Southern University	5
4	4	University of Pennsylvania	5
5	5	Western Kentucky University	5

At the bottom, there are buttons for "Save results", "Open in", and "Refresh".

Conclusion

This assignment offered a practical and insightful introduction to working with Google BigQuery using real-world NCAA Basketball data. By writing and executing a range of focused SQL queries, we gained hands-on experience in data extraction, transformation, and interpretation within a cloud-based analytics environment.

Throughout the task, we explored multiple interrelated datasets, developed a clear understanding of schema design, and applied logical reasoning to craft efficient and accurate queries. All queries were executed successfully, with outputs validated against expected results.

Key concepts such as filtering, aggregation, joins, subqueries, and sorting were reinforced, along with best practices for writing scalable, reusable,

and optimized SQL for large-scale data. This foundational experience has laid the groundwork for tackling more advanced data analytics and machine learning tasks in future BigQuery-based modules.

Git Hub Link - <https://github.com/Sreecharan162/Big-Data.git>