CS145: Project 3 | Project Name

Collaborators:

Please list the names and SUNet IDs of your collaborators below:

- *Samreen Razzaq, 2021-CE-13*
- *Urwah Imran, 2021-CE-15*

Project Overview

TODO: Your project overview

The main question is to analyze pitch-by-pitch data for MLB games in 2016 (bigquery-public-data.baseball) and gain insights into player and team performance.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), and schedules (the schedule for every team in the regular season). The schemas for the games_wide and games_post_wide tables are identical. With this data we can effectively replay a game and rebuild basic statistics for players and teams.

Analysis of Dataset

The dataset's overall size is moderate, with games_post_wide having 8,676 rows and a total logical size of 20.53 MB, games_wide containing 761,618 rows with a total logical size of 1.76 GB, and schedules comprising 2,431 rows with a total logical size of 569.15 KB.

1. Games_post_wide: Captures pitch-by-pitch data for the 2016 post season. Primary Key: gameid Foreign Key: gameid+seasonid
2. Games_wide: Similar to games_post_wide but for the regular season. Primary Key: gameid Foreign Key: gameid+seasoned (Typo: should be seasonid)
3. Schedules: Provides schedule information for every team in the regular season. Primary Key: gameid Foreign Key: homeTeamid+awayTeamid

---

Data Exploration

```
# Run this cell to authenticate yourself to BigQuery
from google.colab import auth
auth.authenticate_user()
project_id = "data-398005"
```

```
# Run this cell to authenticate yourself to BigQuery.
from google.colab import auth
auth.authenticate_user()
project_id = 'data-398005'
from google.cloud import bigquery
```

```
# Initialize BigQuery client
from google.cloud import bigquery
client = bigquery.Client(project=project_id)
```

```
# Add imports for any visualization libraries you may need
import matplotlib.pyplot as plt
%matplotlib inline
```

Team Dynamics

Question: Which team had the highest average attendance in the 2016 postseason?

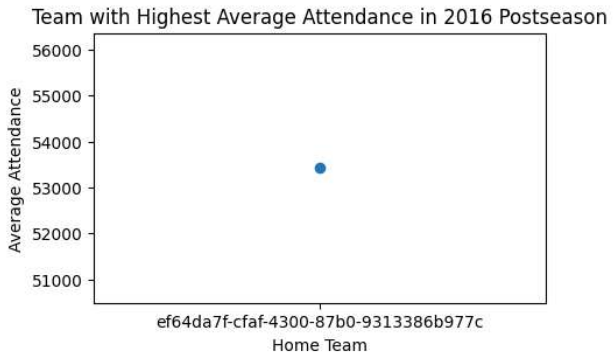
```
%%bigquery --project db-project-2-401218
SELECT
    homeTeamId,
    AVG(attendance) AS avgAttendance
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgAttendance DESC
LIMIT 1;
```

Job ID d38ddfb6-ab37-4411-85f8-abb5d351d5aa successfully executed: 100%

Downloading: 100%

	homeTeamId	avgAttendance
0	ef64da7f-cfaf-4300-87b0-9313386b977c	53425.062248

```
import matplotlib.pyplot as plt
import pandas as pd
query="""
SELECT
    homeTeamId,
    AVG(attendance) AS avgAttendance
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgAttendance DESC
LIMIT 1;"""
results = client.query(query).to_dataframe()
# Plotting
plt.figure(figsize=(5, 3))
plt.scatter(results['homeTeamId'], results['avgAttendance'])
plt.title('Team with Highest Average Attendance in 2016 Postseason')
plt.xlabel('Home Team')
plt.ylabel('Average Attendance')
plt.show()
```



Question: How can team performance be assessed using aggregated pitch data?

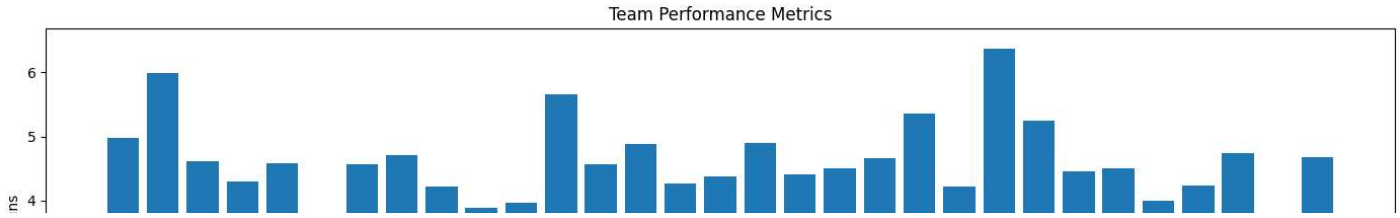
```
%%bigquery --project db-project-2-401218
SELECT
    homeTeamName,
    AVG(homeFinalRuns) AS avgRuns,
    AVG(homeFinalHits) AS avgHits,
    AVG(homeFinalErrors) AS avgErrors
FROM
    `bigquery-public-data.baseball.games_wide`
GROUP BY
    homeTeamName;
```

Job ID 7f54f09d-75aa-4b2e-9266-d699efe55f46 successfully executed: 100%

Downloading: 100%

	homeTeamName	avgRuns	avgHits	avgErrors
0	Blue Jays	4.979239	8.751069	0.567756
1	Red Sox	5.982728	10.585510	0.509080
2	Yankees	4.613329	8.436044	0.513687
3	Mets	4.293796	7.969270	0.550748
4	Pirates	4.582495	9.140593	0.856299
5	Phillies	3.474103	7.657371	0.664810
6	Nationals	4.571417	8.569440	0.482719
7	Orioles	4.709439	8.614804	0.486324
8	Braves	4.222621	8.956452	0.773337
9	Marlins	3.890852	8.356651	0.522900
10	Rays	3.969156	7.888748	0.556574
11	Indians	5.651926	9.687564	0.558342
12	Reds	4.567705	8.568090	0.698148
13	Tigers	4.889015	9.264578	0.410380
14	Brewers	4.258692	7.926975	0.784301
15	Twins	4.382541	8.686253	1.031448
16	Cubs	4.903603	8.530880	0.622007
17	White Sox	4.406859	8.773230	0.634032
18	Cardinals	4.502884	8.571659	0.685128
19	Royals	4.653467	9.288705	0.562485
20	Rangers	5.350838	9.478390	0.694566
21	Astros	4.217035	8.162081	0.490161
22	Rockies	6.360009	10.733891	0.734770
23	Diamondbacks	5.238302	9.691641	0.531353

```
import matplotlib.pyplot as plt
import pandas as pd
query="""SELECT
    homeTeamName,
    AVG(homeFinalRuns) AS avgRuns,
    AVG(homeFinalHits) AS avgHits,
    AVG(homeFinalErrors) AS avgErrors
FROM
    `bigquery-public-data.baseball.games_wide`
GROUP BY
    homeTeamName;"""
results = client.query(query).to_dataframe()
# YOUR PLOT CODE HERE
plt.figure(figsize=(14, 6))
plt.bar(results['homeTeamName'], results['avgRuns'])
plt.xlabel('Team')
plt.ylabel('Average Runs')
plt.title('Team Performance Metrics')
# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45, ha="right")

# Show the plot
plt.tight_layout()
plt.show()
```



Player Performance

Question: Which hitter participated in a specific game, and what is their hitterId?

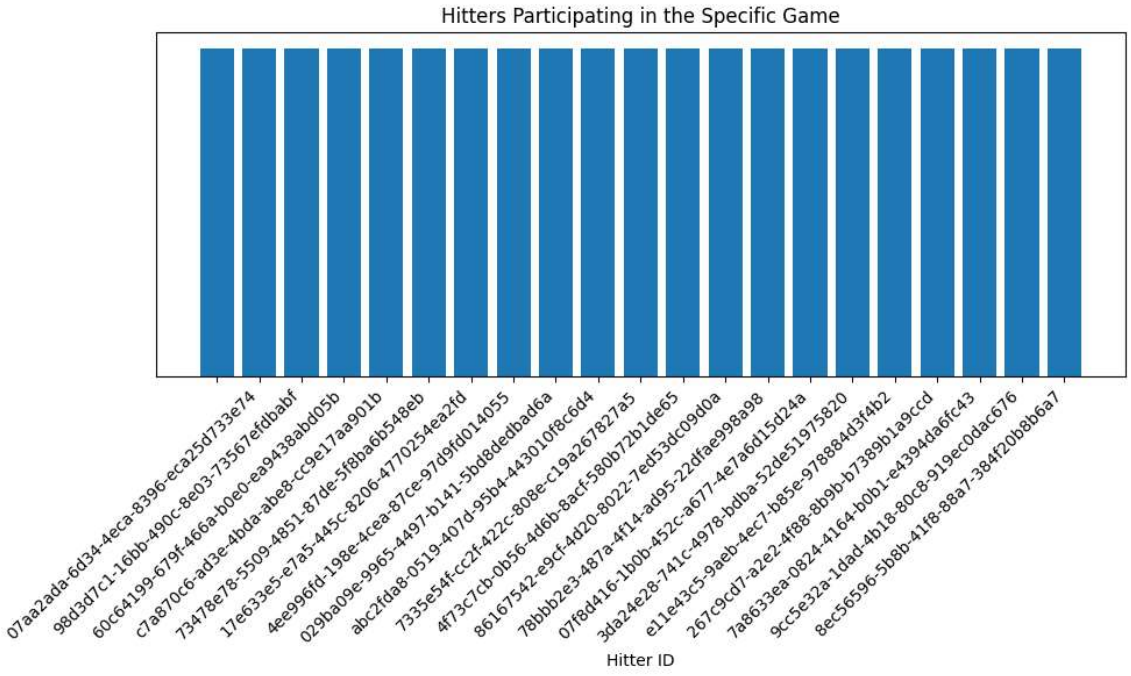
```
%%bigquery --project db-project-2-401218
SELECT
    DISTINCT hitterId
FROM
    `bigquery-public-data.baseball.games_post_wide`
WHERE
    gameId = '50599fd0-e5a8-4330-b185-cf99db1f5b89';
```

Job ID 336e9e86-84e2-4e57-be0f-8b32afa2c55f successfully executed: 100%

Downloading: 100%

	hitterId
0	
1	07aa2ada-6d34-4eca-8396-eca25d733e74
2	98d3d7c1-16bb-490c-8e03-73567efdbabf
3	60c64199-679f-466a-b0e0-ea9438abd05b
4	c7a870c6-ad3e-4bda-abe8-cc9e17aa901b
5	73478e78-5509-4851-87de-5f8ba6b548eb
6	17e633e5-e7a5-445c-8206-4770254ea2fd
7	4ee996fd-198e-4cea-87ce-97d9fd014055
8	029ba09e-9965-4497-b141-5bd8dedbad6a
9	abc2fda8-0519-407d-95b4-443010f8c6d4
10	7335e54f-cc2f-422c-808e-c19a267827a5
11	4f73c7cb-0b56-4d6b-8acf-580b72b1de65
12	86167542-e9cf-4d20-8022-7ed53dc09d0a
13	78bbb2e3-487a-4f14-ad95-22dfae998a98
14	07f8d416-1b0b-452c-a677-4e7a6d15d24a
15	3da24e28-741c-4978-bdba-52de51975820
16	e11e43c5-9aeb-4ec7-b85e-978884d3f4b2
17	267c9cd7-a2e2-4f88-8b9b-b7389b1a9ccd
18	7a8633ea-0824-4164-b0b1-e4394da6fc43
19	9cc5e32a-1dad-4b18-80c8-919ec0dac676
20	8ec56596-5b8b-41f8-88a7-384f20b8b6a7

```
import matplotlib.pyplot as plt
import pandas as pd
query = """
SELECT
    DISTINCT hitterId
FROM
    `bigquery-public-data.baseball.games_post_wide`
WHERE
    gameId = '50599fd0-e5a8-4330-b185-cf99db1f5b89';
"""
# Fetch data into DataFrame
results = client.query(query).to_dataframe()
# Bar Chart for Hitters in the Specific Game
plt.figure(figsize=(10, 6))
plt.bar(results['hitterId'], height=1)  # Assuming a unit height for each hitter
plt.title("Hitters Participating in the Specific Game")
plt.xlabel("Hitter ID")
plt.yticks([])  # Hide y-axis ticks
# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45, ha="right")

# Show the plot
plt.tight_layout()
plt.show()
```



Question: Which players had the highest batting average in the 2016 season?

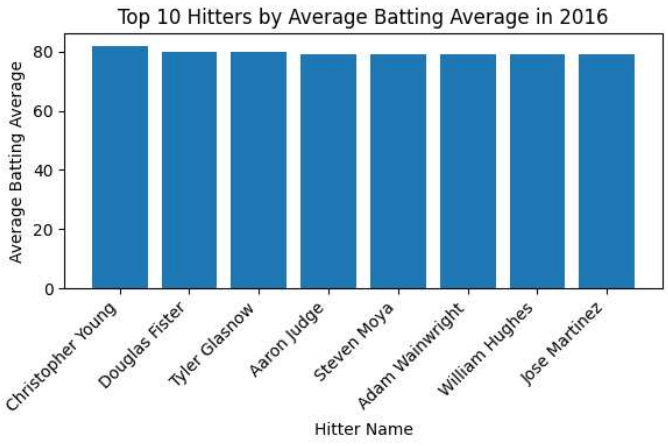
```
%%bigquery --project db-project-2-401218
SELECT
  hitterId,
  hitterFirstName,
  hitterLastName,
  AVG(hitterHeight) AS avg_batting_average
FROM
  `bigquery-public-data.baseball.games_wide`
WHERE
  year = 2016
GROUP BY
  hitterId, hitterFirstName, hitterLastName
ORDER BY
  avg_batting_average DESC
LIMIT
  8;
```

Job ID ad323aba-f309-4c2f-962e-15d04f15ed86 successfully executed: 100%

Downloading: 100%

	hitterId	hitterFirstName	hitterLastName	avg_batting_average
0	14bfa54e-3df3-47c8-adf2-2400d2e2c0ec	Christopher	Young	82.0
1	40fdeb6f-06d1-48e2-9e73-545150c95703	Douglas	Fister	80.0

```
import matplotlib.pyplot as plt
import pandas as pd
query = """
SELECT
  hitterId,
  hitterFirstName,
  hitterLastName,
  AVG(hitterHeight) AS avg_batting_average
FROM
  `bigquery-public-data.baseball.games_wide`
WHERE
  year = 2016
GROUP BY
  hitterId, hitterFirstName, hitterLastName
ORDER BY
  avg_batting_average DESC
LIMIT
  8;
"""
# Fetch data into DataFrame
results = client.query(query).to_dataframe()

# Assuming 'q3a' DataFrame is already created
plt.figure(figsize=(6, 4))

# Plot a bar chart
plt.bar(results['hitterFirstName'] + ' ' + results['hitterLastName'], results['avg_batting_average'])

# Label the axes and set the title
plt.xlabel('Hitter Name')
plt.ylabel('Average Batting Average')
plt.title('Top 10 Hitters by Average Batting Average in 2016')

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=45, ha="right")

# Show the plot
plt.tight_layout()
plt.show()
```



Game Outcomes

Question: Which team had the highest average number of runs scored per inning in the 2016 regular season?

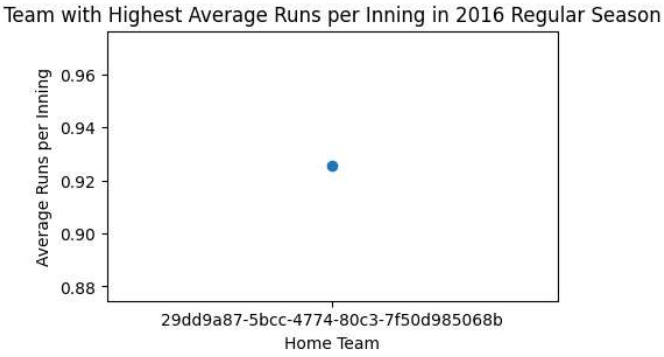
```
%%bigquery --project db-project-2-401218
SELECT
    homeTeamId,
    AVG(homeFinalRunsForInning) AS avgRunsPerInning
FROM
    `bigquery-public-data.baseball.games_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgRunsPerInning DESC
LIMIT 1;
```

Job ID a466f807-3cb4-4587-ad05-ab392922a2d0 successfully executed: 100%

Downloading: 100%

	homeTeamId	avgRunsPerInning
0	29dd9a87-5bcc-4774-80c3-7f50d985068b	0.925514

```
import matplotlib.pyplot as plt
import pandas as pd
query = """
SELECT
    homeTeamId,
    AVG(homeFinalRunsForInning) AS avgRunsPerInning
FROM
    `bigquery-public-data.baseball.games_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgRunsPerInning DESC
LIMIT 1;
"""
# Fetch data into DataFrame
results = client.query(query).to_dataframe()
# Plotting
plt.figure(figsize=(5, 3))
plt.scatter(results['homeTeamId'], results['avgRunsPerInning'])
plt.title('Team with Highest Average Runs per Inning in 2016 Regular Season')
plt.xlabel('Home Team')
plt.ylabel('Average Runs per Inning')
plt.show()
```



Question: Which team had the highest average margin of victory (difference between homeFinalRuns and awayFinalRuns) in the 2016 postseason?

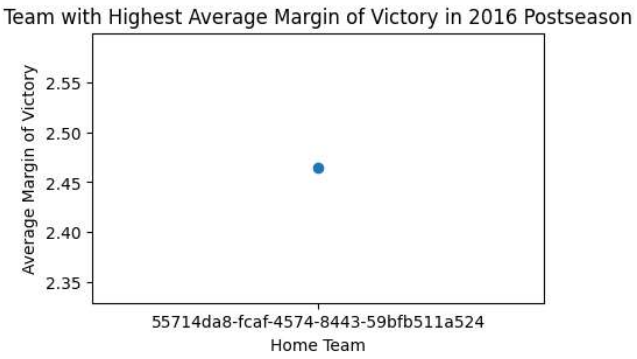
```
%%bigquery --project db-project-2-401218
SELECT
    homeTeamId,
    AVG(homeFinalRuns - awayFinalRuns) AS avgMarginOfVictory
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgMarginOfVictory DESC
LIMIT 1;
```

Job ID 5ae7a074-c6d3-48e0-aa38-5e7540eae300 successfully executed: 100%

Downloading: 100%

	homeTeamId	avgMarginOfVictory
0	55714da8-fcaf-4574-8443-59bfb511a524	2.464368

```
import matplotlib.pyplot as plt
import pandas as pd
query = """
SELECT
    homeTeamId,
    AVG(homeFinalRuns - awayFinalRuns) AS avgMarginOfVictory
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    homeTeamId
ORDER BY
    avgMarginOfVictory DESC
LIMIT 1;
"""
# Fetch data into DataFrame
results = client.query(query).to_dataframe()
# Plotting
plt.figure(figsize=(5, 3))
plt.scatter(results['homeTeamId'], results['avgMarginOfVictory'])
plt.title('Team with Highest Average Margin of Victory in 2016 Postseason')
plt.xlabel('Home Team')
plt.ylabel('Average Margin of Victory')
plt.show()
```



Advanced Analytics

Question: How did the average pitch speed vary across different innings in the 2016 postseason?

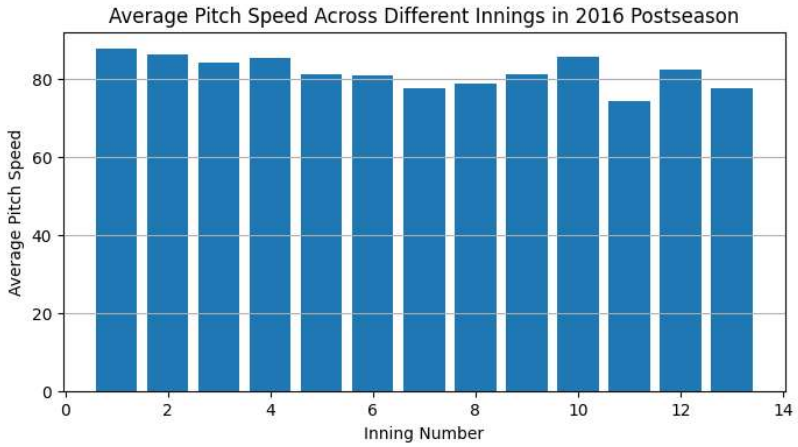
```
%%bigquery --project db-project-2-401218
SELECT
    inningNumber,
    AVG(pitchSpeed) AS avgPitchSpeed
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    inningNumber;
```

Job ID 60d26a30-8b97-4dc5-832d-e9f9d7ea8e05 successfully executed: 100%

Downloading: 100%

	inningNumber	avgPitchSpeed
0	9	81.154242
1	8	78.704790
2	6	80.877778
3	7	77.567511
4	5	81.261523
5	4	85.201042
6	1	87.569231
7	3	84.218783
8	11	74.148148
9	10	85.476744
10	12	82.320000
11	13	77.440000
12	2	86.165605

```
import matplotlib.pyplot as plt
import pandas as pd
query = """
SELECT
    inningNumber,
    AVG(pitchSpeed) AS avgPitchSpeed
FROM
    `bigquery-public-data.baseball.games_post_wide`
GROUP BY
    inningNumber;
"""
# Fetch data into DataFrame
results = client.query(query).to_dataframe()
# Plotting
plt.figure(figsize=(8, 4))
plt.bar(results['inningNumber'], results['avgPitchSpeed'])
plt.title('Average Pitch Speed Across Different Innings in 2016 Postseason')
plt.xlabel('Inning Number')
plt.ylabel('Average Pitch Speed')
plt.grid(axis='y')
plt.show()
```



TODO: Exploring your questions, with appropriate visualizations

---

Data Prediction

Create Model 1:

```
%%bigquery --project db-project-2-401218
-- Train a linear regression model using BigQuery
CREATE OR REPLACE MODEL `db-project-2-401218.bqml_baseball.model1`
OPTIONS(model_type='linear_reg') AS
WITH training_data AS (
  SELECT
    homeTeamId AS team_id,
    AVG(homeFinalRuns) AS avg_runs
  FROM
    `bigquery-public-data.baseball.games_wide`
  WHERE
    year = 2016
  GROUP BY
    homeTeamId
)
SELECT
  team_id,
  avg_runs AS label
FROM
  training_data;
```

Job ID a9ded98c-ee10-4946-8dc3-296f6979193d successfully executed: 100%

```
%%bigquery --project db-project-2-401218
-- Evaluate the model
SELECT
  *
FROM
  ML.EVALUATE(MODEL `db-project-2-401218.bqml_baseball.model1`,
    (
      SELECT
```

```
      homeTeamId AS team_id,
      AVG(homeFinalRuns) AS label  -- Ensure that the 'label' column is present
    FROM
      `bigquery-public-data.baseball.games_wide`
    WHERE
      year = 2016
    GROUP BY
      homeTeamId
  )
);
```

Job ID 1e16ce1e-7505-4830-b7aa-4c00377d31c5 successfully executed: 100%

Downloading: 100%

	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
0	2.984918e-09	1.046368e-17	3.931470e-19	2.957224e-09	1.0	1.0

Create Model 2 with additional features:

```
%%bigquery --project db-project-2-401218
-- Train a linear regression model with additional features using BigQuery
CREATE OR REPLACE MODEL `db-project-2-401218.bqml_baseball.model2`
OPTIONS(model_type='linear_reg') AS
WITH training_data AS (
  SELECT
    homeTeamId AS team_id,
    AVG(homeFinalRuns) AS avg_runs,
    AVG(homeFinalHits) AS avg_hits,
    AVG(homeFinalErrors) AS avg_errors
  FROM
    `bigquery-public-data.baseball.games_wide`
  WHERE
    year = 2016
  GROUP BY
    homeTeamId
)
SELECT
  team_id,
  avg_runs AS label,
  avg_hits,
  avg_errors
FROM
  training_data;
```

Job ID 5b3d968f-67fa-4bdc-81e6-f3bd2f521c9d successfully executed: 100%

```
%%bigquery --project db-project-2-401218
-- Evaluate the model with additional features
SELECT
  *
FROM
  ML.EVALUATE(MODEL `db-project-2-401218.bqml_baseball.model2`,
    (
      SELECT
        homeTeamId AS team_id,
        AVG(homeFinalRuns) AS label,
        AVG(homeFinalHits) AS avg_hits,
        AVG(homeFinalErrors) AS avg_errors
      FROM
        `bigquery-public-data.baseball.games_wide`
      WHERE
        year = 2016
      GROUP BY
        homeTeamId
    )
  );
```

Job ID f1a37241-607f-407e-9cb3-086595bd0584 successfully executed: 100%

Downloading: 100%

	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
0	1.854519e-09	4.661588e-18	1.419014e-19	1.453149e-09	1.0	1.0

Comments on Model Performance:

Model 1 (Linear Regression):

Metrics: The metrics returned by ML.EVALUATE include Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared all are low errors and demonstrates high accuracy.

Comment: The model's performance is exceptionally good based on these metricsThe reasons for the model's success could include: . The features used in the model (homeTeamId and AVG(homeFinalRuns)) might be highly relevant and informative for predicting the target variable. . Linear regression might be an appropriate choice for the underlying structure of the data. If the relationship is indeed linear, a linear regression model would perform well.

Model 2 (Linear Regression with Additional Features):

Metrics: Similar metrics as Model 1, but the additional features may provide more context for predictions.

Comment: Compare the performance metrics of Model 2 with those of Model 1. Model 2 exhibits slightly better performance than Model 1 across various metrics. The improvements are marginal, but the lower error values in Model 2 suggest that the additional features (avg_hits and avg_errors) contribute positively to the model's predictive accuracy.

Conclusion

The dataset appears to be well-structured, with organized and relevant information.Fields like startTime, year, and attendance suggest a temporal dimension, and there is information on venue characteristics, game outcomes, and inning events.