BigQuery Data Analytical Insight

Overview

BigQuery can be used to perform more sophisticated data analysis. In this lab, you will analyze soccer event data to achieve real insight from the dataset.
The data used in this lab comes from the following sources:

- Pappalardo et al., (2019) A public data set of spatio-temporal match events in soccer competitions, Nature Scientific Data 6:236, https://www.nature.com/articles/s41597-019-0247-7
- Pappalardo et al. (2019) PlayerRank: Data-driven Performance Evaluation and Player Ranking in Soccer via a Machine Learning Approach. ACM Transactions on Intelligent Systems and Technologies (TIST) 10, 5, Article 59 (September 2019), 27 pages. DOI: https://doi.org/10.1145/3343172

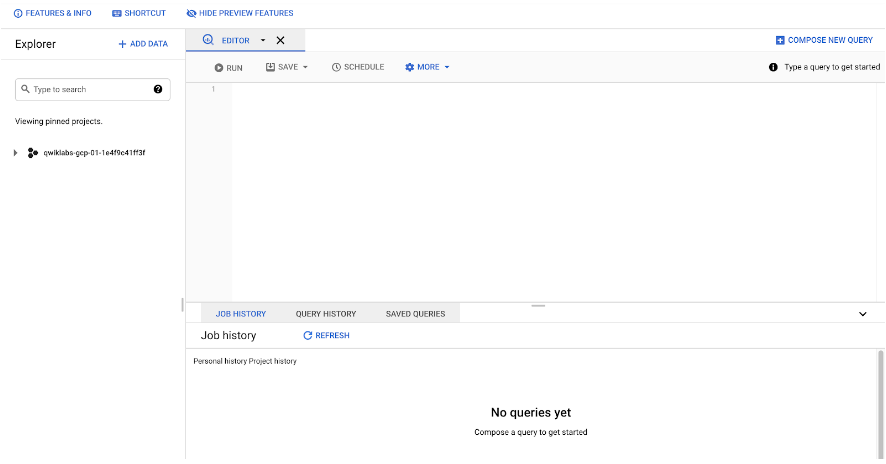
In this lab, you will:

- Analyze soccer event data using various BigQuery features
- Write and execute queries that work with nested data in BigQuery tables

Open BigQuery

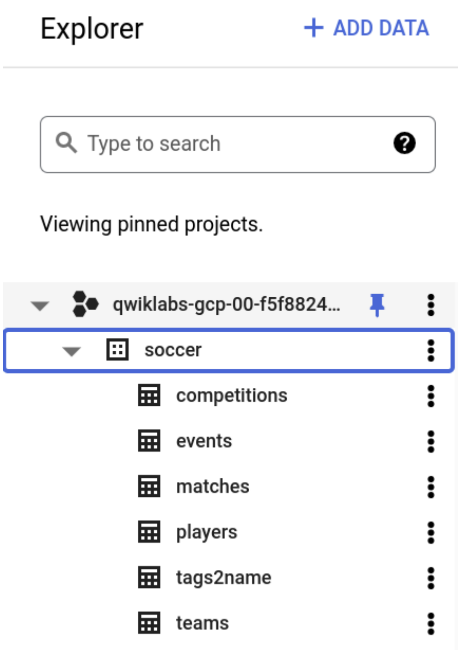
The BigQuery console provides an interface to query tables, including public datasets offered by BigQuery.

- In the Cloud Console, from the **Navigation menu** select **BigQuery**.
- The **Welcome to BigQuery in the Cloud Console** message box opens. This message box provides a link to the quickstart guide and the release notes.
- Click **Done**.
- The BigQuery console opens.



The process for creating the dataset and tables is taught in the BigQuery Soccer Data Ingestion lab. In this lab the focus is on learning how to query the information.

Once the tables are created the display will be similar to below:



In this section the BigQuery interface was used to access the console. The console provides a convenient way to add information to a dataset. BigQuery uses tables to represent data in a structured way.

In the next section learn more about creating more complex queries.

Analyze nested soccer event data

In this section, you will run some queries that use JOINs with BigQuery's array functionality to enable better control over the soccer event data.

1. In the Query editor, click **Compose new query**.
2. Copy and paste the following query into the query **Editor**:

```
SELECT
 Events.playerId,
 (Players.firstName || ' ' || Players.lastName) AS playerName,
 SUM(IF(Tags2Name.Label = 'assist', 1, 0)) AS numAssists
FROM
 `soccer.events` Events,
 Events.tags Tags
LEFT JOIN
 `soccer.tags2name` Tags2Name ON
   Tags.id = Tags2Name.Tag
LEFT JOIN
 `soccer.players` Players ON
   Events.playerId = Players.wyId
GROUP BY
 playerId, playerName
ORDER BY
 numAssists DESC
```

Assists aren't marked as a separate scalar field in the **events** table, so you need to look "inside" the **tags** field.

This is done by using a correlated cross join between the **events** table and the **tags** field (with the "," in the FROM clause to represent an implicit join) to create 1 row per tag per event (rather than 1 row per event). The tag ID that corresponds to assists is found from the **tags2name** table, the number of occurrences of that tag is counted by player, and the **players** table gets player names from their IDs.

3. Click **Run**. The results are displayed below the query window.

Query results SAVE RESULTS EXPLORE DATA

Query complete (1.9 sec elapsed, 60 MB processed)

Job information Results JSON Execution details

Row	playerId	playerName	numAssists
1	38021	Kevin De Bruyne	21
2	25714	Dimitri Payet	14
3	3359	Lionel Andr\u00e9s Messi Cuccittini	14
4	14732	Thomas M\u00fcller	14
5	3484	Luis Alberto Romero Alconchel	14

*In this section a more complex query was created in BigQuery. Performing **Joins** in BigQuery and leveraging **Arrays** provide a powerful way to aggregate data.*
*In the next section learn how to use **Nesting** and **Arrays** with BigQuery.*

Calculate the average pass distance by team

In this section, you will run some queries that use the nested fields in the soccer event data and BigQuery's array functionality and STRUCT data type to answer some interesting questions.

How much do club teams differ in terms of average distance on their passes (both overall and accurate ones)?

To answer this question, study the **positions** field in the **events** table. Observing this data, you see that this is a repeated field that contains 1 or more (x, y) pairs per event. Per Wyscout, a leading data company in the soccer industry that provided this data, these represent the origin and (if applicable) destination positions associated with the event, on a 0-100 scale representing the percentage of the field from the perspective of the attacking team.

The screenshot below illustrates the positions corresponding to a few different event types for a few example events.

eventName	playerId	subEventName	id	positions.x	positions.y
Foul	83575	Late card foul	88179369	30	60
Free Kick	14922	Free kick shot	88520527	72	62
				100	100
Goalkeeper leaving line	83574	Goalkeeper leaving line	88743724	0	0
				24	93
Pass	78488	Hand pass	88223367	14	57
				17	34
Save attempt	78488	Reflexes	88223382	100	100
				3	58

From the data you can note that passes have 2 attributes (x, y) pairs representing the start and end position. Therefore pass distance can be calculated by calculating x-and y-coordinate differences, then converting to estimated meters using the average dimensions of a soccer field (105 x 68, per Wikipedia; there is no standard field size) and the 2-dimensional distance formula.

1. In the Query editor, click **Compose new query**.
2. Add the following query into the query **Editor**.

```
WITH
Passes AS
(
 SELECT
   *,
   /* 1801 is known Tag for 'accurate' from tags2name table */
   (1801 IN UNNEST(tags.id)) AS accuratePass,
   (CASE
     WHEN ARRAY_LENGTH(positions) != 2 THEN NULL
     ELSE
  /* Translate 0-100 (x,y) coordinate-based distances to absolute positions
  using "average" field dimensions of 105x68 before combining in 2D dist calc */
       SQRT(
         POW(
           (positions[ORDINAL(2)].x - positions[ORDINAL(1)].x) * 105/100,
           2) +
         POW(
           (positions[ORDINAL(2)].y - positions[ORDINAL(1)].y) * 68/100,
           2)
         )
     END) AS passDistance
 FROM
   `soccer.events`
 WHERE
   eventName = 'Pass'
)
SELECT
 Passes.teamId,
 Teams.name AS team,
 Teams.area.name AS teamArea,
 COUNT(Passes.Id) AS numPasses,
 AVG(Passes.passDistance) AS avgPassDistance,
 SAFE_DIVIDE(
   SUM(IF(Passes.accuratePass, Passes.passDistance, 0)),
   SUM(IF(Passes.accuratePass, 1, 0))
   ) AS avgAccuratePassDistance
FROM
 Passes
LEFT JOIN
 `soccer.teams` Teams ON
   Passes.teamId = Teams.wyId
WHERE
 Teams.type = 'club'
GROUP BY
 teamId, team, teamArea
ORDER BY
 avgPassDistance
```

The code in the initial WITH clause filters the **events** table to passes only and adds an **accuratePass** field by looking "inside" the tags field.

The pass distance is calculated by extracting the initial and final (x, y) coordinates using ORDINAL and applying the concepts and formula mentioned above. The final SELECT statement aggregates the passes data to the team level (filtering to only club teams), including average pass distance on all passes and accurate passes only.

3. Click **Run**. The results are displayed below the query window.

Query results   SAVE RESULTS   EXPLORE DATA

Query complete (2.9 sec elapsed, 183.3 MB processed)

Job information   Results   JSON   Execution details

Row	teamId	team	teamArea	numPasses	avgPassDistance	avgAccuratePassDistance
1	3187	Napoli	Italy	26874	17.854649073750107	16.941389683948238
2	3767	PSG	France	23908	17.958146857723264	17.311501048768395
3	1625	Manchester City	England	27523	18.090661936017682	17.295157088929013
4	676	Barcelona	Spain	23260	18.233003753466647	17.685909463132703
5	1609	Arsenal	England	22783	18.614260755625047	17.819071562854838
6	3161	Internazionale	Italy	20450	18.82509620369358	17.570033335486478
7	3775	Nice	France	21131	18.833634118798372	18.029118817296776
8	679	Atl\u00e9tico Madrid	Spain	16336	19.056748282403074	17.67809893905138
9	714	Las Palmas	Spain	17148	19.172428318879984	18.013300687200466
10	675	Real Madrid	Spain	22081	19.175157496150387	18.095888838824088

Note: There are some differences in average pass distance across thousands of passes from various teams, going from the lowest average, less than 18 meters (Napoli and PSG) to the highest one, greater than 23 meters (Eibar).
Average accurate pass distance shows similar dispersion, though those values are slightly more compressed across teams.

In this section BigQuery was used to determine the number of passes and the average distance of passes by team. To achieve this, you used array processing capabilities to extract repeated values in a single field, then calculated the distance between the starting and ending point of each pass.

In the next section learn how to unnest other coordinate data to generate information about shot distances.

Analyze shot distance

In this section, you will create a new query to analyze shot distance.

What impact does the distance of a shot have on the likelihood of a goal being scored?

To answer this question use a process similar to the previous section. For shots, use (x, y) values from the **positions** field in the **events** table.

Note: as per the previous query, the approximate dimensions of a soccer field are used with the x-coordinate and y-coordinate distances as inputs to the distance formula.

1. In the Query editor, click **Compose new query**.
2. Copy and paste the following query into the query **Editor**.

```
WITH
Shots AS
(
 SELECT
  *,
  /* 101 is known Tag for 'goals' from goals table */
  (101 IN UNNEST(tags.id)) AS isGoal,
  /* Translate 0-100 (x,y) coordinate-based distances to absolute positions
  using "average" field dimensions of 105x68 before combining in 2D dist calc */
  SQRT(
    POW(
      (100 - positions[ORDINAL(1)].x) * 105/100,
      2) +
    POW(
      (50 - positions[ORDINAL(1)].y) * 68/100,
      2)
     ) AS shotDistance
 FROM
  `soccer.events`

 WHERE
  /* Includes both "open play" & free kick shots (including penalties) */
  eventName = 'Shot' OR
  (eventName = 'Free Kick' AND subEventName IN ('Free kick shot', 'Penalty'))
)
SELECT
 ROUND(shotDistance, 0) AS ShotDistRound0,

 COUNT(*) AS numShots,
```

```
 SUM(IF(isGoal, 1, 0)) AS numGoals,
 AVG(IF(isGoal, 1, 0)) AS goalPct
FROM
 Shots
WHERE
 shotDistance <= 50
GROUP BY
 ShotDistRound0
ORDER BY
 ShotDistRound0
```

The initial WITH clause filters the **events** table to shots only, adds an **isGoal** field by looking "inside" the **tags** field, and calculates shot distance the same way that pass distance was handled in the previous section, but uses the midpoint of the goal mouth (100, 50) as the ending location.

The final SELECT statement aggregates the number of shots, number of goals, and percentage of goals from shots by distance rounded to the nearest meter.

3. Click **Run**. The results are displayed below the query window.

Query results　SAVE RESULTS　EXPLORE DATA

Query complete (0.7 sec elapsed, 203.3 MB processed)

Job information　Results　JSON　Execution details

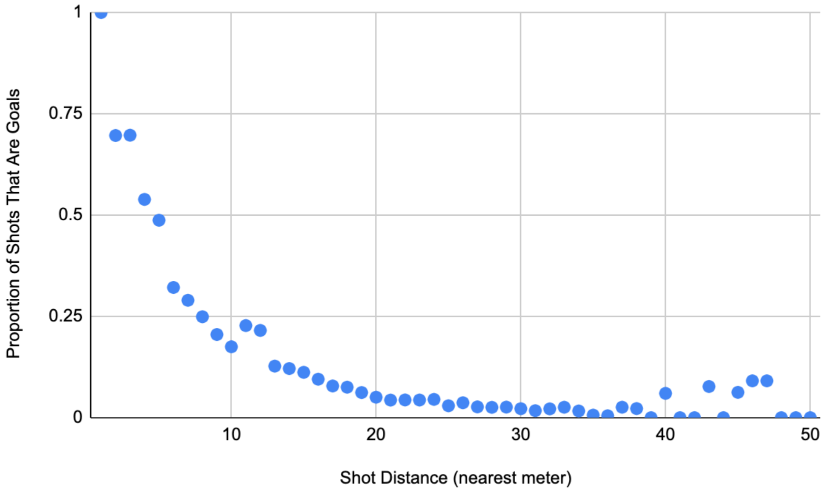
Row	ShotDistRound0	numShots	numGoals	goalPct
1	1.0	17	17	1.0
2	2.0	56	39	0.6964285714285714
3	3.0	175	122	0.6971428571428572
4	4.0	297	160	0.5387205387205386
5	5.0	513	250	0.48732943469785583
6	6.0	753	242	0.32138114209827356
7	7.0	932	270	0.28969957081545067
8	8.0	1140	284	0.24912280701754397
9	9.0	1564	321	0.20524296675191828
10	10.0	1543	270	0.17498379779650036
11	11.0	2138	486	0.22731524789522947
12	12.0	2582	556	0.21533694810224618
13	13.0	2546	324	0.12725844461901015
14	14.0	2293	278	0.12123855211513311
15	15.0	2537	284	0.11194324004729994
16	16.0	1822	173	0.09495060373216248
17	17.0	1853	145	0.07825148407987043
18	18.0	1834	138	0.07524536532170122
19	19.0	1547	96	0.0620555914673562
20	20.0	1287	65	0.0505050505050505
21	21.0	1267	55	0.04340962904498815
22	22.0	1302	57	0.043778801843317984
23	23.0	1377	60	0.043572984749455375
24	24.0	1440	65	0.045138888888888916
25	25.0	1565	46	0.0293929712460064

As expected, shots at close distance have much higher goal rates, going from near 70% success at 2-3 meters down to less than 25% at 8 meters, and declining steadily all the way to 25+ meters.

Create a visualization of results

Visualizing the data can make it easier to understand and see trends.

- Click on the **EXPLORE DATA** dropdown on the BigQuery results.
- Select **Explore with Sheets**.
- Use the scatter chart creation features in Sheets to create a chart like the one below:

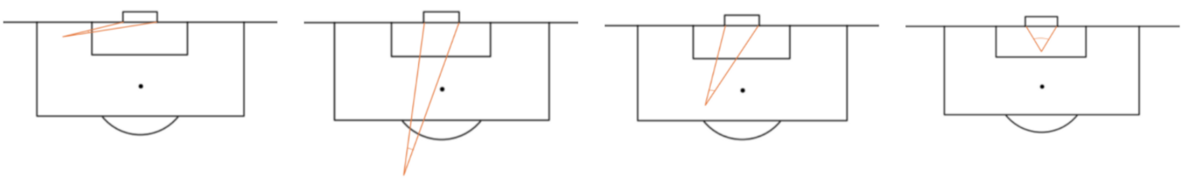


There's a slight bump up in success rate at 11-12 meters, but that can likely be explained by the fact that penalty kicks (which are, by design, much higher propositions than most other shots) account for a large percentage of shots from that range.

In this section BigQuery was used to establish a view on shot distance versus goal success rate. From this analysis there is a better understanding of the likelihood of a goal being scored based on the distance of the shot. In the next section you will perform a similar analysis to look at the impact of shot angle on shot success.

Analyze shot angle

In this section, modify the previous query to look at the impact of the angles on shots. In this case, the angle calculated is the one made by the location of the shot and the goal line, as shown below



Larger angles arise from being close to the goal and in the center, so this is somewhat correlated with the distance calculation performed above. The shot angle calculations involve using BigQuery's trigonometric functions on the (x, y) data.

1. In the Query editor, click **Compose new query**.
2. Add the following query into the query **Editor**.

```
WITH
Shots AS
(
 SELECT
  *,
  /* 101 is known Tag for 'goals' from goals table */
  (101 IN UNNEST(tags.id)) AS isGoal,
  /* Translate 0-100 (x,y) coordinates to absolute positions using "average"
  field dimensions of 105x68 before using in various distance calcs;
  LEAST used to cap shot locations to on-field (x, y) (i.e. no exact 100s) */
  LEAST(positions[ORDINAL(1)].x, 99.99999) * 105/100 AS shotXAbs,
  LEAST(positions[ORDINAL(1)].y, 99.99999) * 68/100 AS shotYAbs
 FROM
   `soccer.events`

 WHERE
   /* Includes both "open play" & free kick shots (including penalties) */
   eventName = 'Shot' OR
   (eventName = 'Free Kick' AND subEventName IN ('Free kick shot', 'Penalty'))
),
ShotsWithAngle AS
(
 SELECT
   Shots.*,
   /* Law of cosines to get 'open' angle from shot location to goal, given
    that goal opening is 7.32m, placed midway up at field end of (105, 34) */
   SAFE.ACOS(
     SAFE_DIVIDE(
       ( /* Squared distance between shot and 1 post, in meters */
         (POW(105 - shotXAbs, 2) + POW(34 + (7.32/2) - shotYAbs, 2)) +
         /* Squared distance between shot and other post, in meters */
         (POW(105 - shotXAbs, 2) + POW(34 - (7.32/2) - shotYAbs, 2)) -
         /* Squared length of goal opening, in meters */
         POW(7.32, 2)
       ),
       (2 *
         /* Distance between shot and 1 post, in meters */
         SQRT(POW(105 - shotXAbs, 2) + POW(34 + 7.32/2 - shotYAbs, 2)) *
         /* Distance between shot and other post, in meters */
```

```
        SQRT(POW(105 - shotXAbs, 2) + POW(34 - 7.32/2 - shotYAbs, 2))
      )
    )
  /* Translate radians to degrees */
  ) * 180 / ACOS(-1)
  AS shotAngle
 FROM
  Shots
)
SELECT
 ROUND(shotAngle, 0) AS ShotAngleRound0,

 COUNT(*) AS numShots,
 SUM(IF(isGoal, 1, 0)) AS numGoals,
 AVG(IF(isGoal, 1, 0)) AS goalPct
FROM
 ShotsWithAngle
GROUP BY
 ShotAngleRound0
ORDER BY
 ShotAngleRound0
```

This query is similar to the shot distance one above except for an initial **WITH** clause to extract the shot coordinates (to simplify when needing them multiple times in the angle calculation) and a more detailed trigonometric calculation using the Law of Cosines to get shot angle in the second **WITH** clause.

The final SELECT statement aggregates by shot angle rounded to the nearest degree.

3. Click **Run**. The results are displayed below the query window.

Query results   SAVE RESULTS   EXPLORE DATA

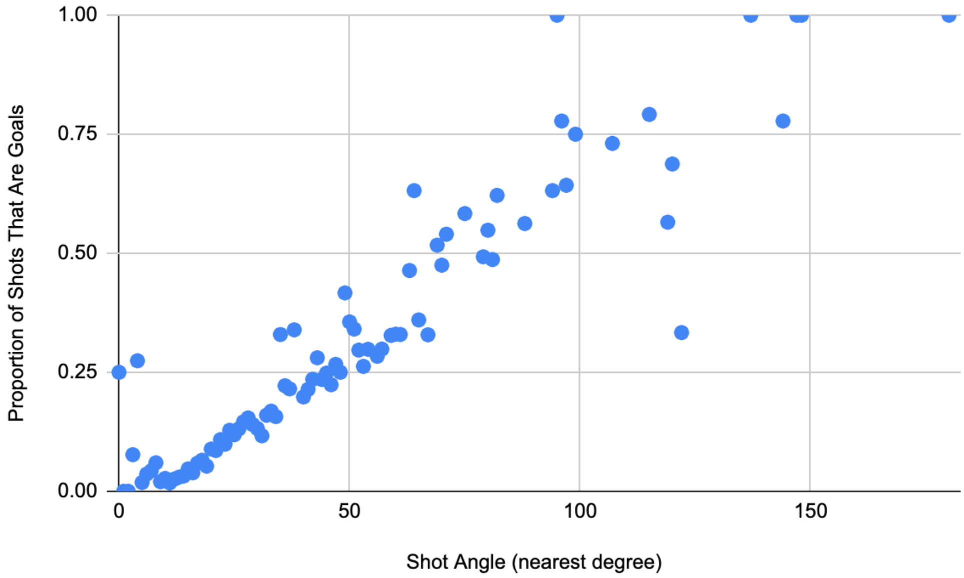
Query complete (0.8 sec elapsed, 203.3 MB processed)

Job information   Results   JSON   Execution details

Row	ShotAngleRound10	numShots	numGoals	goalPct
1	0.0	4	1	0.25
2	1.0	1	0	0.0
3	2.0	5	0	0.0
4	3.0	26	2	0.07692307692307693
5	4.0	62	17	0.2741935483870967
6	5.0	54	1	0.018518518518518517
7	6.0	138	5	0.036231884057971
8	7.0	209	9	0.0430622009569378
9	8.0	267	16	0.0599250936329588
10	9.0	446	9	0.020179372197309416
11	10.0	851	23	0.027027027027027046
12	11.0	1466	26	0.017735334242837644
13	12.0	2379	61	0.025641025641025660
14	13.0	3502	104	0.029697315819531722
15	14.0	3215	103	0.032037325038880240
16	15.0	3509	165	0.047021943573667756
17	16.0	2884	112	0.038834951456310680
18	17.0	2272	134	0.0589788732394366
19	18.0	2007	131	0.065271549576482260
20	19.0	1266	67	0.052922590837282770
21	20.0	1419	126	0.088794926004228320
22	21.0	1225	105	0.0857142857142857
23	22.0	1336	145	0.108532934131736450
24	23.0	1122	111	0.098930481283422510
25	24.0	1037	133	0.128254580520732850

Shot angle seems to be generally positively correlated with goal success rate, going from single-digit success rate at angles below 20° to much higher rates at wider angles (with relatively lower sample sizes beyond 60° or so).

By clicking on the **EXPLORE DATA** dropdown on the BigQuery results, selecting **Explore with Sheets**, and then using the scatter chart creation features in Sheets, you can visualize the full trend like you see below.



The plot shows that the relationship between shot angle and success rate is relatively linear up to about 100°. Again, the widest angles are only possible on shots close to the goal, so some of this is correlated with the distance effect shown above. There is some slight bumping up in success rate at 35° and 38°, as these are the most common shot angles for penalty kicks (again, much higher propositions than most other shots) and account for a large percentage of shots from that range.