Midterm Review: Integration of NFL Statistics Into Google BigQuery

Minhyuk Kang
Parker Jensen

Presentation Agenda

01

Dataset Introduction

What data did we collect for our dataset?

02

Data Enrichment

How did we fill the holes in our data?

03

Data Compliance

How have we structured our data so that it can be queried properly?

Dataset Introduction

What is Our Dataset?

For our solution, we chose to model several pieces of data related to the NFL.

We used four datasets to create our data warehouse. Three of them are dataset pulled from Kaggle, a website owned by Google that hosts user submitted datasets.

The other is extracted from an official document released by the Baltimore Ravens PR team, tracking statistics for the 2024 season.

What Are Our Tables?

Our dataset consists of Nine Tables:

nfl_stadiums - Models football stadiums entities.

spreadspoke_scores - Models individual football games over a large period of time

team_conference - Models the conference and division of individual teams.

team_identification - Models the name, abbreviated name, and id of a team.

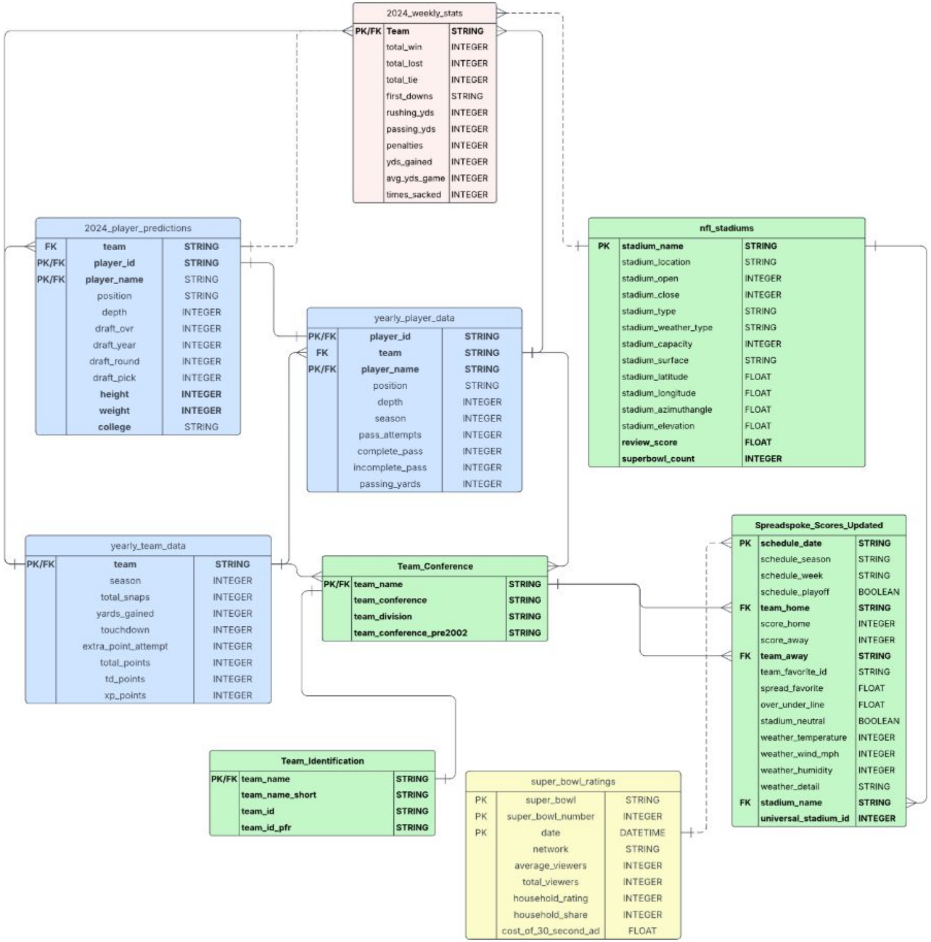
yearly_team_data - Models aggregate data for a team over a season, like total touchdowns, total points, etc.

yearly_player_data - Models aggregate data for a player over a season, like touchdowns, yards gained, etc.

2024_player_predictions - Models yearly predictions for player statistics on sports betting websites.

Super_bowl_ratings - Models individual super bowl games, focusing on the cost of advertisement, viewers, and television network hosting.

2024_weekly_stats - Models weekly stats for individual players in 2024. Extracted from a PDF file release by Baltimore Ravens PR.



	2024_weekly_stats	
PK/FK	Team	STRING
	total_win	INTEGER
	total_lost	INTEGER
	total_tie	INTEGER
	first_downs	STRING
	rushing_yds	INTEGER
	passing_yds	INTEGER
	penalties	INTEGER
	yds_gained	INTEGER
	avg_yds_game	INTEGER
	times_sacked	INTEGER

	2024_player_predictions	
FK	team	STRING
PK/FK	player_id	STRING
PK/FK	player_name	STRING
	position	STRING
	depth	INTEGER
	draft_ovr	INTEGER
	draft_year	INTEGER
	draft_round	INTEGER
	draft_pick	INTEGER
	height	INTEGER
	weight	INTEGER
	college	STRING

	yearly_player_data	
PK/FK	player_id	STRING
FK	team	STRING
PK/FK	player_name	STRING
	position	STRING
	depth	INTEGER
	season	INTEGER
	pass_attempts	INTEGER
	complete_pass	INTEGER
	incomplete_pass	INTEGER
	passing_yards	INTEGER

	nfl_stadiums	
PK	stadium_name	STRING
	stadium_location	STRING
	stadium_open	INTEGER
	stadium_close	INTEGER
	stadium_type	STRING
	stadium_weather_type	STRING
	stadium_capacity	INTEGER
	stadium_surface	STRING
	stadium_latitude	FLOAT
	stadium_longitude	FLOAT
	stadium_azimuthangle	FLOAT
	stadium_elevation	FLOAT
	review_score	FLOAT
	superbowl_count	INTEGER

	yearly_team_data	
PK/FK	team	STRING
	season	INTEGER
	total_snaps	INTEGER
	yards_gained	INTEGER
	touchdown	INTEGER
	extra_point_attempt	INTEGER
	total_points	INTEGER
	td_points	INTEGER
	xp_points	INTEGER

	Team_Conference	
PK/FK	team_name	STRING
	team_conference	STRING
	team_division	STRING
	team_conference_pre2002	STRING

	Spreadspoke_Scores_Updated	
PK	schedule_date	STRING
	schedule_season	STRING
	schedule_week	STRING
	schedule_playoff	BOOLEAN
FK	team_home	STRING
	score_home	INTEGER
	score_away	INTEGER
FK	team_away	STRING
	team_favorite_id	STRING
	spread_favorite	FLOAT
	over_under_line	FLOAT
	stadium_neutral	BOOLEAN
	weather_temperature	INTEGER
	weather_wind_mph	INTEGER
	weather_humidity	INTEGER
	weather_detail	STRING
FK	stadium_name	STRING
	universal_stadium_id	INTEGER

	Team_Identification	
PK/FK	team_name	STRING
	team_name_short	STRING
	team_id	STRING
	team_id_pfr	STRING

	super_bowl_ratings	
PK	super_bowl	STRING
PK	super_bowl_number	INTEGER
PK	date	DATETIME
	network	STRING
	average_viewers	INTEGER
	total_viewers	INTEGER
	household_rating	INTEGER
	household_share	INTEGER
	cost_of_30_second_ad	FLOAT

Our current entity relationship diagram (ERD). Different background shading indicates data from different sources.

Why Choose to Model This Data?

- NFL data has important implications for player careers, often influencing the decisions of team ownership in important contract negotiations.
- Furthermore, it’s helpful in fantasy football decision making, allowing one to construct a better average team
- NFL data has important connections to sports betting markets, where millions of dollars circulate and outcomes can be predicted and bet on based on average cases.
- Overall, NFL data allows us to make more informed decisions about what to do with players whether it be in betting markets, hobby fantasy football, or business decisions.

P

Introduction of Data Challenges

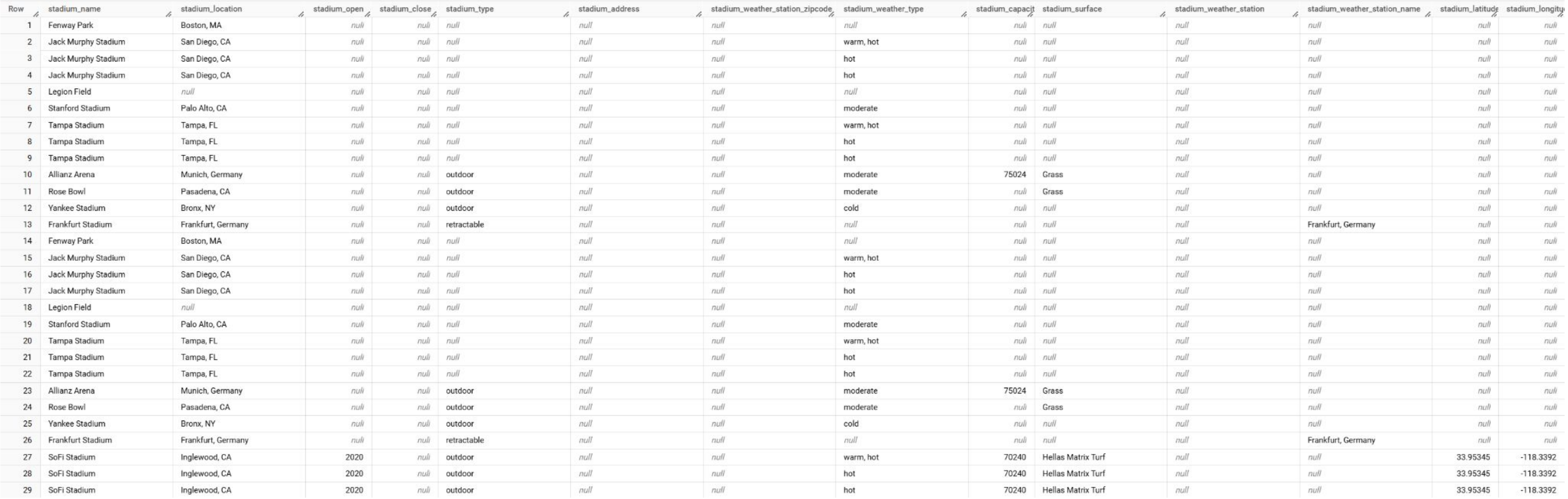
Interesting Challenges/Aspects of our Solution

Our data was often very incomplete, requiring enrichment to be more useful

Many elements of our data are highly repetitive, often interfering with Primary Key enforcement

Data Enrichment

Problem: Our data was often very incomplete, requiring enrichment to be more useful



Row	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_zipcode	stadium_weather_type	stadium_capacit	stadium_surface	stadium_weather_station	stadium_weather_station_name	stadium_latitude	stadium_longitu
1	Fenway Park	Boston, MA	null	null	null	null	null	null	null	null	null	null	null	null
2	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	warm, hot	null	null	null	null	null	null
3	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
4	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
5	Legion Field	null	null	null	null	null	null	null	null	null	null	null	null	null
6	Stanford Stadium	Palo Alto, CA	null	null	null	null	null	moderate	null	null	null	null	null	null
7	Tampa Stadium	Tampa, FL	null	null	null	null	null	warm, hot	null	null	null	null	null	null
8	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
9	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
10	Allianz Arena	Munich, Germany	null	null	outdoor	null	null	moderate	75024	Grass	null	null	null	null
11	Rose Bowl	Pasadena, CA	null	null	outdoor	null	null	moderate	null	Grass	null	null	null	null
12	Yankee Stadium	Bronx, NY	null	null	outdoor	null	null	cold	null	null	null	null	null	null
13	Frankfurt Stadium	Frankfurt, Germany	null	null	retractable	null	null	null	null	null	null	Frankfurt, Germany	null	null
14	Fenway Park	Boston, MA	null	null	null	null	null	null	null	null	null	null	null	null
15	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	warm, hot	null	null	null	null	null	null
16	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
17	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
18	Legion Field	null	null	null	null	null	null	null	null	null	null	null	null	null
19	Stanford Stadium	Palo Alto, CA	null	null	null	null	null	moderate	null	null	null	null	null	null
20	Tampa Stadium	Tampa, FL	null	null	null	null	null	warm, hot	null	null	null	null	null	null
21	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
22	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
23	Allianz Arena	Munich, Germany	null	null	outdoor	null	null	moderate	75024	Grass	null	null	null	null
24	Rose Bowl	Pasadena, CA	null	null	outdoor	null	null	moderate	null	Grass	null	null	null	null
25	Yankee Stadium	Bronx, NY	null	null	outdoor	null	null	cold	null	null	null	null	null	null
26	Frankfurt Stadium	Frankfurt, Germany	null	null	retractable	null	null	null	null	null	null	Frankfurt, Germany	null	null
27	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	warm, hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392
28	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392
29	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392

A view of our table of Sports Stadiums. Many fields are almost entirely empty, when they should not be.

So How Do We Make This Data More Complete?

- It’s possible that we could find another dataset and attempt to integrate the two together.
- The issue is that datasets are so varied and different that one may use an entirely different system for measuring player statistics than another.
- Similarly, it’s often very difficult to actually find data that would help in this situation.
- Manual input would take such a long amount of time, and be so costly that it isn’t feasible either.

Large datasets with simple, public data available that have many holes in them? This is the perfect use case for an LLM.

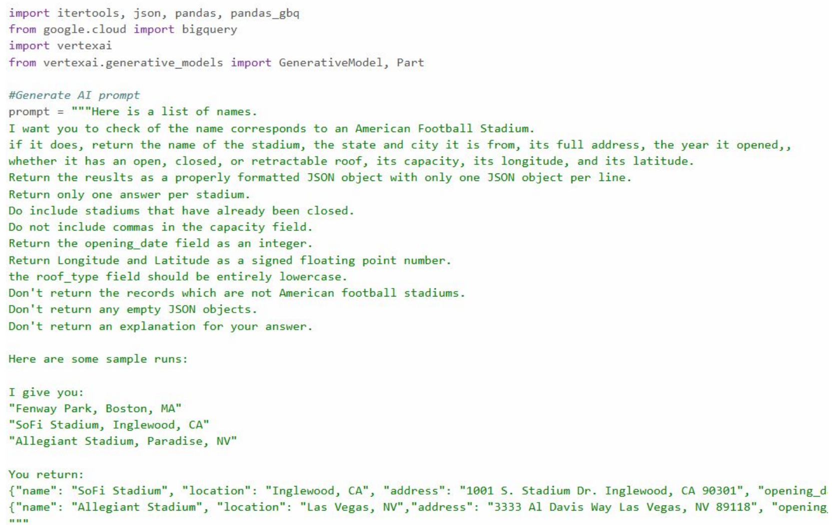
Methodology for LLM Use

- Essentially, what we need to do is give the LLM a list of our primary keys, and have it return the missing or incorrect data in an implementable format.
- To accomplish this, we'll make use of the BigQuery client's ability to parse JSON objects by having the LLM return our data as a collection of JSON objects in one large string.
- Afterward, we'll replace the previous, incorrect/null data with the new, correct data.

Implementation

- First, we import all the necessary packages we need.
- Afterward, we create the LLM's prompt.
- After much refinement, we're very specific in what data we want, and how we want it returned.
- We even give the LLM a few examples to follow, including one where it should not return data.

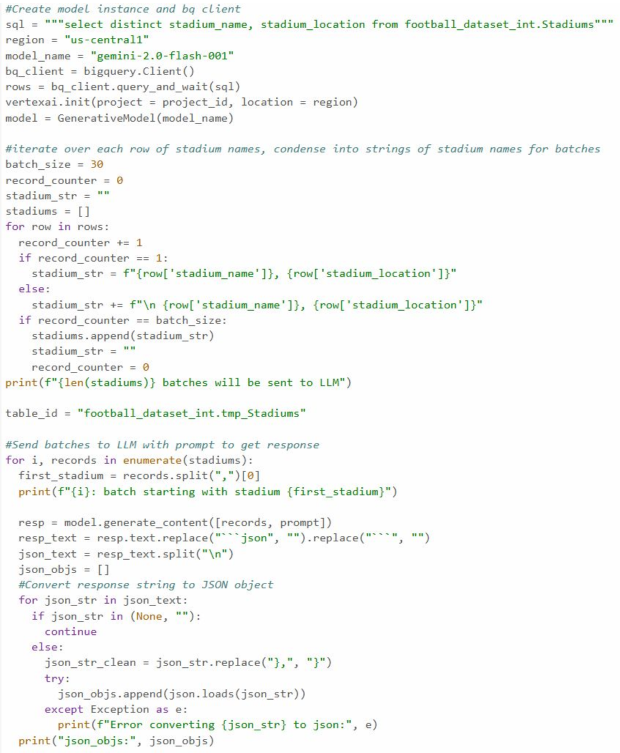
(The cut off portion of of this snippet is an example JSON object)



```
import itertools, json, pandas, pandas_gbq
from google.cloud import bigquery
import vertexai
from vertexai.generative_models import GenerativeModel, Part

#Generate AI prompt
prompt = """Here is a list of names.
I want you to check of the name corresponds to an American Football Stadium.
if it does, return the name of the stadium, the state and city it is from, its full address, the year it opened,,
whether it has an open, closed, or retractable roof, its capacity, its longitude, and its latitude.
Return the reuslts as a properly formatted JSON object with only one JSON object per line.
Return only one answer per stadium.
Do include stadiums that have already been closed.
Do not include commas in the capacity field.
Return the opening_date field as an integer.
Return Longitude and Latitude as a signed floating point number.
the roof_type field should be entirely lowercase.
Don't return the records which are not American football stadiums.
Don't return any empty JSON objects.
Don't return an explanation for your answer.

Here are some sample runs:

I give you:
"Fenway Park, Boston, MA"
"SoFi Stadium, Inglewood, CA"
"Allegiant Stadium, Paradise, NV"

You return:
{"name": "SoFi Stadium", "location": "Inglewood, CA", "address": "1001 S. Stadium Dr. Inglewood, CA 90301", "opening_d
{"name": "Allegiant Stadium", "location": "Las Vegas, NV","address": "3333 Al Davis Way Las Vegas, NV 89118", "opening
"""
```

Implementation

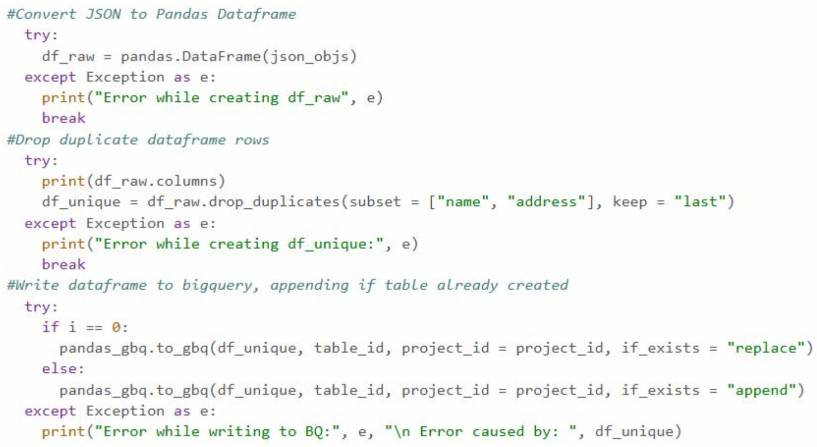
- Afterward, we define all our other variables needed for client use, and create instances of the LLM and BigQuery Client
- We then split the primary key fields into multiple batches to reduce error from the LLM.
- We then send the batches to the LLM for a response, and receive a string of JSON objects in return.
- Finally, we parse all JSON objects and add them to a list to make use of later.



````python
#Create model instance and bq client
sql = """select distinct stadium_name, stadium_location from football_dataset_int.Stadiums"""
region = "us-central1"
model_name = "gemini-2.0-flash-001"
bq_client = bigquery.Client()
rows = bq_client.query_and_wait(sql)
vertexai.init(project = project_id, location = region)
model = GenerativeModel(model_name)

#iterate over each row of stadium names, condense into strings of stadium names for batches
batch_size = 30
record_counter = 0
stadium_str = ""
stadiums = []
for row in rows:
  record_counter += 1
  if record_counter == 1:
    stadium_str = f"{row['stadium_name']}, {row['stadium_location']}"
  else:
    stadium_str += f"\n {row['stadium_name']}, {row['stadium_location']}"
  if record_counter == batch_size:
    stadiums.append(stadium_str)
    stadium_str = ""
    record_counter = 0
print(f"{len(stadiums)} batches will be sent to LLM")

table_id = "football_dataset_int.tmp_Stadiums"

#Send batches to LLM with prompt to get response
for i, records in enumerate(stadiums):
  first_stadium = records.split(",")[0]
  print(f"{i}: batch starting with stadium {first_stadium}")

  resp = model.generate_content([records, prompt])
  resp_text = resp.text.replace("```json", "").replace("```", "")
  json_text = resp_text.split("\n")
  json_objs = []
  #Convert response string to JSON object
  for json_str in json_text:
    if json_str in (None, ""):
      continue
    else:
      json_str_clean = json_str.replace("},", "}")
      try:
        json_objs.append(json.loads(json_str))
      except Exception as e:
        print(f"Error converting {json_str} to json:", e)
  print("json_objs:", json_objs)
````

Implementation

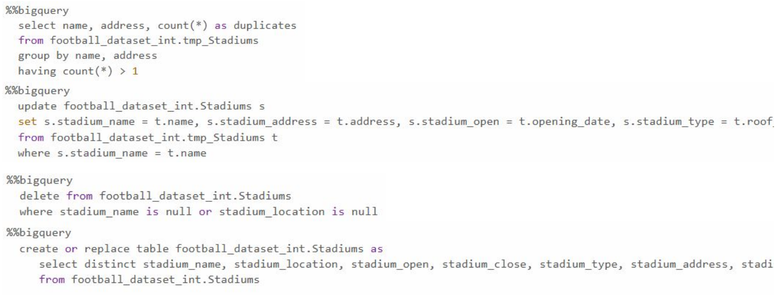
- Then, we create a Pandas (a library in Python that handles data) dataframe by passing the constructor method the list of JSON objects.
- This gives us a table with all of our information. We then remove all duplicate primary key entries.
- Then, we upload this table to BigQuery as a new, temporary table.



```
#Convert JSON to Pandas Dataframe
  try:
    df_raw = pandas.DataFrame(json_objs)
  except Exception as e:
    print("Error while creating df_raw", e)
    break
#Drop duplicate dataframe rows
  try:
    print(df_raw.columns)
    df_unique = df_raw.drop_duplicates(subset = ["name", "address"], keep = "last")
  except Exception as e:
    print("Error while creating df_unique:", e)
    break
#Write dataframe to bigquery, appending if table already created
  try:
    if i == 0:
      pandas_gbq.to_gbq(df_unique, table_id, project_id = project_id, if_exists = "replace")
    else:
      pandas_gbq.to_gbq(df_unique, table_id, project_id = project_id, if_exists = "append")
  except Exception as e:
    print("Error while writing to BQ:", e, "\n Error caused by: ", df_unique)
```

Implementation

- Now, we check for duplicate entries in the temporary table, and update the original table with the new values from the temporary table.
- Then, we remove null and duplicate Primary Key entries from the resulting table

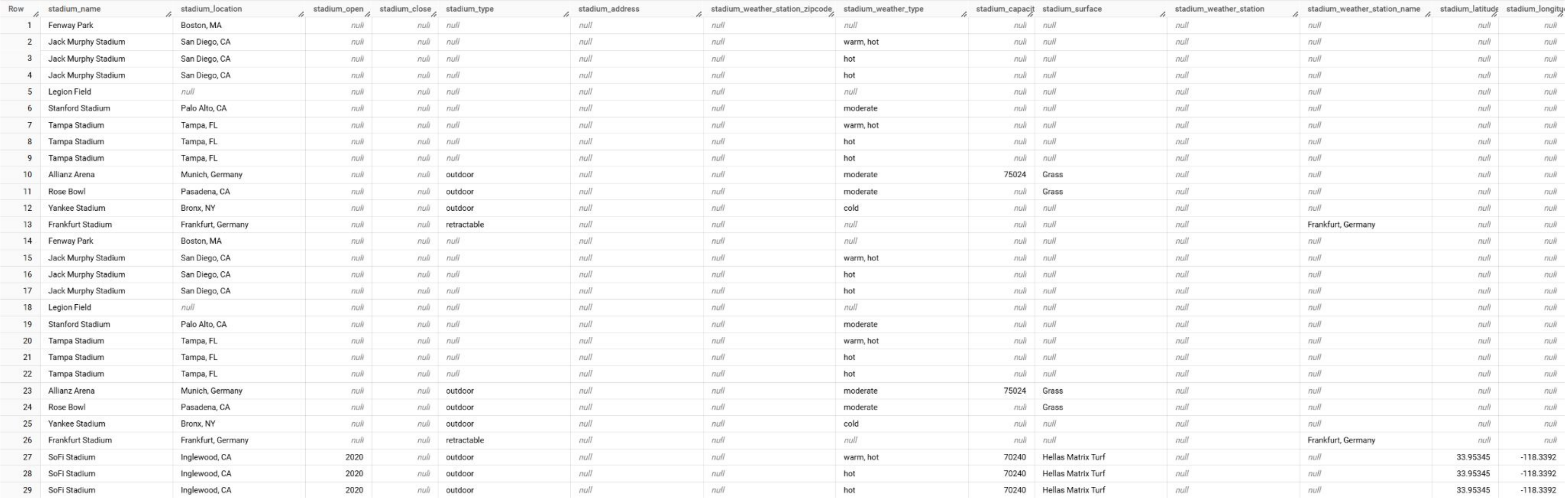


```
%%bigquery
  select name, address, count(*) as duplicates
  from football_dataset_int.tmp_Stadiums
  group by name, address
  having count(*) > 1
%%bigquery
  update football_dataset_int.Stadiums s
  set s.stadium_name = t.name, s.stadium_address = t.address, s.stadium_open = t.opening_date, s.stadium_type = t.roof
  from football_dataset_int.tmp_Stadiums t
  where s.stadium_name = t.name

%%bigquery
  delete from football_dataset_int.Stadiums
  where stadium_name is null or stadium_location is null
%%bigquery
  create or replace table football_dataset_int.Stadiums as
     select distinct stadium_name, stadium_location, stadium_open, stadium_close, stadium_type, stadium_address, stadi
     from football_dataset_int.Stadiums
```

Code Execution

Data Compliance

Problem: The fields of our data that should be primary keys often have multiple entries.



Row	stadium_name	stadium_location	stadium_open	stadium_close	stadium_type	stadium_address	stadium_weather_station_zipcode	stadium_weather_type	stadium_capacit	stadium_surface	stadium_weather_station	stadium_weather_station_name	stadium_latitude	stadium_longitu
1	Fenway Park	Boston, MA	null	null	null	null	null	null	null	null	null	null	null	null
2	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	warm, hot	null	null	null	null	null	null
3	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
4	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
5	Legion Field	null	null	null	null	null	null	null	null	null	null	null	null	null
6	Stanford Stadium	Palo Alto, CA	null	null	null	null	null	moderate	null	null	null	null	null	null
7	Tampa Stadium	Tampa, FL	null	null	null	null	null	warm, hot	null	null	null	null	null	null
8	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
9	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
10	Allianz Arena	Munich, Germany	null	null	outdoor	null	null	moderate	75024	Grass	null	null	null	null
11	Rose Bowl	Pasadena, CA	null	null	outdoor	null	null	moderate	null	Grass	null	null	null	null
12	Yankee Stadium	Bronx, NY	null	null	outdoor	null	null	cold	null	null	null	null	null	null
13	Frankfurt Stadium	Frankfurt, Germany	null	null	retractable	null	null	null	null	null	null	Frankfurt, Germany	null	null
14	Fenway Park	Boston, MA	null	null	null	null	null	null	null	null	null	null	null	null
15	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	warm, hot	null	null	null	null	null	null
16	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
17	Jack Murphy Stadium	San Diego, CA	null	null	null	null	null	hot	null	null	null	null	null	null
18	Legion Field	null	null	null	null	null	null	null	null	null	null	null	null	null
19	Stanford Stadium	Palo Alto, CA	null	null	null	null	null	moderate	null	null	null	null	null	null
20	Tampa Stadium	Tampa, FL	null	null	null	null	null	warm, hot	null	null	null	null	null	null
21	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
22	Tampa Stadium	Tampa, FL	null	null	null	null	null	hot	null	null	null	null	null	null
23	Allianz Arena	Munich, Germany	null	null	outdoor	null	null	moderate	75024	Grass	null	null	null	null
24	Rose Bowl	Pasadena, CA	null	null	outdoor	null	null	moderate	null	Grass	null	null	null	null
25	Yankee Stadium	Bronx, NY	null	null	outdoor	null	null	cold	null	null	null	null	null	null
26	Frankfurt Stadium	Frankfurt, Germany	null	null	retractable	null	null	null	null	null	null	Frankfurt, Germany	null	null
27	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	warm, hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392
28	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392
29	SoFi Stadium	Inglewood, CA	2020	null	outdoor	null	null	hot	70240	Hellas Matrix Turf	null	null	33.95345	-118.3392

A view of our table of Sports Stadiums. Notice how many stadium names are repeated.

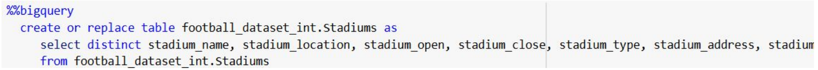
How Do We Make Our Data Queryable?

- The only solution for this problem is to enforce a primary key constraint on the tables, so that we have unique primary key entries.
- The problem here is that BigQuery does not currently have a method of enforcing constraints, like MySQL does.
- So, we have to manually ensure the data properly adheres to a constraint.

BigQuery gives us the tools to do this, it may just be a little more complex than it would be in another program, like MySQL.

Implementation

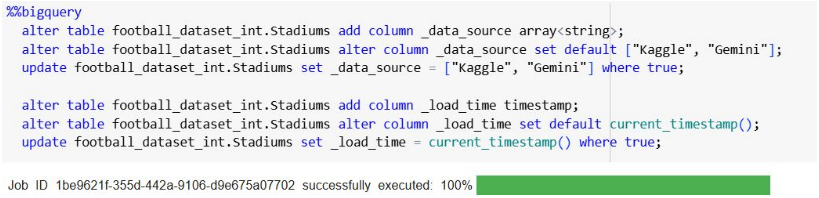
- If we try to add a Primary Key constraint now, BigQuery will raise an exception, so we must first remove duplicate primary key entries
- This is relatively simple. We update the table by querying itself, looking for only distinct values.
- Note that we do not take distinct _load_time values, as these are the only field different across repeated primary key entries. We will add these back in a moment.



```
%%bigquery
  create or replace table football_dataset_int.Stadiums as
     select distinct stadium_name, stadium_location, stadium_open, stadium_close, stadium_type, stadium_address, stadium
     from football_dataset_int.Stadiums
```

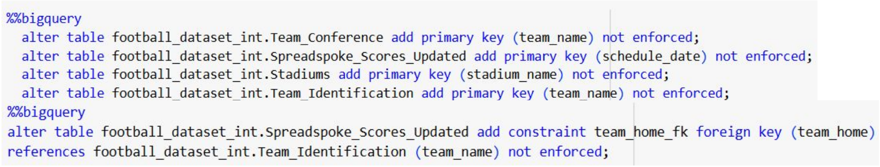
Implementation

- After doing this (and adding new fields not in the original table, not shown here), we add _load_time and _data_source back into the table with their respective default values.
- Then, we add primary key constraints to the tables.
- Finally, we add foreign key constraints to show that some fields may reference fields in other tables.



```
%%bigquery
  alter table football_dataset_int.Stadiums add column _data_source array<string>;
  alter table football_dataset_int.Stadiums alter column _data_source set default ["Kaggle", "Gemini"];
  update football_dataset_int.Stadiums set _data_source = ["Kaggle", "Gemini"] where true;

  alter table football_dataset_int.Stadiums add column _load_time timestamp;
  alter table football_dataset_int.Stadiums alter column _load_time set default current_timestamp();
  update football_dataset_int.Stadiums set _load_time = current_timestamp() where true;
```

```
Job ID 1be9621f-355d-442a-9106-d9e675a07702 successfully executed: 100%
```



```
%%bigquery
  alter table football_dataset_int.Team_Conference add primary key (team_name) not enforced;
  alter table football_dataset_int.Spreadspoke_Scores_Updated add primary key (schedule_date) not enforced;
  alter table football_dataset_int.Stadiums add primary key (stadium_name) not enforced;
  alter table football_dataset_int.Team_Identification add primary key (team_name) not enforced;
%%bigquery
alter table football_dataset_int.Spreadspoke_Scores_Updated add constraint team_home_fk foreign key (team_home)
references football_dataset_int.Team_Identification (team_name) not enforced;
```

Thank you!