

CS145: Data Management and Data Systems  
Stanford University, Fall 2020

**Project 1: Exploring NCAA Basketball Data**  
10% of Course Grade

**Due Date: Friday, October 2nd, 11:59PM**

## Overview

Welcome to CS145! Throughout the course you will be using Google's BigQuery platform to gain hands-on practice with real-world data systems.

BigQuery is Google's service for big data. Throughout the three class projects, we will be using BigQuery's basic SQL querying interface, its interface with Colaboratory<sup>1</sup>, and its built-in machine learning features.

Google has published many datasets on BigQuery -- these range from StackOverflow statistics to real-time air quality data. In this first part of the course project you will be using BigQuery's SQL interface to answer questions about the NCAA Basketball Dataset. To find the dataset in BigQuery, follow the instructions in the [Getting Started with BigQuery](#) support document.

Please note:

- **This is a solo project.** You may discuss ideas at a high-level with other students, but all work should be your own.
- Please include the names and SUNet IDs of any students you collaborate with.
- Please read the “**Submission Instructions**” section before you start so you understand the deliverables of the project.

## Task A: Getting Set Up

Before proceeding, make sure you have read and understood the [Getting Started with BigQuery](#) support document (available on the course website) which describes how to get up and running with your BigQuery account, how to manage your course credit, etcetera.

---

<sup>1</sup>Colaboratory is a free document collaboration tool built on top of [Jupyter](#). You can think of it as a Jupyter notebook, except that you are able to collaborate on it with multiple people.

**Note:** This is a *very important step* as we will be unable to give extra Google Cloud credit to students that use up all of their credit. If you have any questions about Google Cloud, your account, or your credits, check Piazza for similar questions or make a post yourself.

## Task B: Familiarize yourself with the NCAA Basketball Dataset

Now that you've oriented yourself in BigQuery, your second task is to examine the schemas and the descriptions of the NCAA Basketball dataset tables and understand the data that you will be working with.

You may try running some simple queries over the tables to get a feel for them, or use BigQuery's "Preview" tab to see what the data looks like.

mascots [QUERY TABLE](#)

---

Schema Details Preview

Row	id	market	name	mascot	mascot_name
1	2959bd24-7007-41ae-a3a3-abdf26888cfc	Tulsa	Golden Hurricane	Hurricane	Captain Cane
2	ad4bc983-8d2e-4e6f-a8f9-80840a786c64	Arizona State	Sun Devils	Devil	Sparky
3	f2d01b77-0f5d-4574-9e49-2a3eaf822e44	Drexel	Dragons	Dragon	Mario the Magnificent
4	b47d10b8-a2a5-47df-a2f9-7bd0b9d51beb	Bradley	Braves	Gargoyle	Kaboom!
5	0113eea0-c943-4fff-9780-ae0fb099e7ef	Canisius	Golden Griffins	Griffin	Petey
6	bdc2561d-f603-4fab-a262-f1d2af462277	Michigan	Wolverines	None	<i>null</i>
7	dfe0d93f-94a5-47fb-b7aa-f74786e09acb	Illinois-Chicago	Flames	Dragon	Sparky

Some notes:

- `_sr` stands for "Sportradar", which is a company that collects sports data, down to the x/y coordinates of events (shot attempted, rebound, turnover, foul).
- The historical data makes a distinction between tournament games and regular season games. Please make sure you're using the right table!
- "pbp" means play-by-play, which is very granular data about each event that happens in the game

## Task C: Querying!

Now that you've gotten comfortable and familiar with BigQuery and its SQL querying interface, let's get to work and answer some questions about the NCAA Basketball dataset.

We intend for part of this assignment to be about how to translate a question in plain English to a schema - in other words, we want you to read the tables and explore the data and think about which tables and columns are necessary in answering the question we're asking. This skill is

both necessary for the remainder of the projects, and is exactly how real world data querying and analysis works!

Your queries should be fairly efficient -- they should each take at most ten seconds to execute on BigQuery, and most of them will be finished in less than ~4 seconds. If any of your queries are taking much longer than that, you've probably written them in a particularly inefficient way; please try rewriting them, and see the course staff if you need help.

Please also check to make sure you're not querying more than a couple GBs of data - we've specifically chosen this dataset so that no one needs to exhaust their credits completing the assignment. All the queries you write should fit within the 1TB of free querying you're allotted for the month.

You can save your queries for each question from the BigQuery interface directly, or you can keep track of your queries in **project1\_submission.py** (see Submission Instructions Section). Remember that you can use BigQuery's "Query History" tab to inspect previous queries you've run.

**Note:** When querying in BigQuery, **table names should be wrapped in backticks (`)**. For example, instead of saying:

```
x      SELECT * FROM bigquery-public-data.ncaa_basketball.mascots
say:
      SELECT * FROM `bigquery-public-data.ncaa_basketball.mascots`
✓
```

## Questions:

We will provide answers for these questions so that you can check your work. **Please make sure your output from BigQuery matches these answers**, both in terms of values and ordering. Read instructions carefully; if we ask for rounded answers, we may deduct points for not rounding.

**Note:** While matching these answers is a good sanity check, it does not guarantee a perfect score. The datasets we will use to grade your assignment may not perfectly match the datasets on BigQuery; therefore, make sure that your queries are generalizable to other datasets (given that schemas are identical).

We reserve the right to deduct points from your project if your queries are hard-coded in some way or are not generalizable to other tables.

**For the following questions, unless otherwise specified, a game can be either a tournament game or a regular season game.**

Write standard SQL queries to answer the following questions:

1. (1 point) What is the name and capacity of Stanford's NCAA basketball team venue?

**Answer:**

Row	venue_name	venue_capacity
1	Maples Pavilion	7392

2. (1 point) How many games were played at Maples Pavilion in the 2013 season?

**Answer:**

Row	games_at_maples_pavilion
1	16

3. (1 point) Hexadecimal colors codes are a way of representing color on a computer. Hex color codes are of form #AABBCC, where AA, BB, and CC are hexadecimal numbers (00, 01, ... , FE, FF) indicating the intensity of red, green, and blue in the color, respectively.

**Hint:** be careful with the case of the colors in the dataset -- some use lower case characters and some use upper case characters. Note that in the expected answer below, the original case from the dataset is kept.

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

**Answer:**

Row	market	color
1	Idaho State	#ff7800
2	Morehead State	#ffc300
3	North Carolina A&T	#ffb82b
4	Northern Colorado	#ffb500
5	Oklahoma State	#FF6600
6	Pacific	#ff6900
7	South Dakota	#ff2310
8	Syracuse	#ff5113

9	Tennessee-Martin	#ff6900
---	------------------	---------

4. (1 point) 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.

**Answer:**

Depending on which table you use for your query, you may get slightly different values. Any of the following results are acceptable.

Row	number	avg_stanford	avg_opponent
1	71	78.04	64.21

Row	number	avg_stanford	avg_opponent
1	71	78.07	64.13

Row	number	avg_stanford	avg_opponent
1	71	78.07	64.21

5. (2 points) How many players have been on a team based in the same city where they were born? For this question, please only use the player's birth city and state (do not include the player's birth country).

**Answer:**

Row	num_players
1	606

6. (2 points) 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.

**Answer:**

Row	win_name	lose_name	win_pts	lose_pts	margin
1	Jayhawks	Panthers	110	52	58

7. (3 points) In a basketball tournament, teams are ranked from best to worst prior to starting the matches. This ranking is called the “seed” of the team (1 is the best team, and a higher number indicates a worse team). In general, a higher ranked team is expected to beat a lower ranked team.

**Definition:** An **upset** occurs whenever a team with seed  $A$  beats a team with seed  $B$ , and  $A > B$ .

What percentage of historical tournament games are upsets? Round to two decimal places. For example, if 50.2489% of games are upsets, your query should return 50.25.

**Answer:**

Row	upset_percentage
1	27.26

8. (3 points) 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. Put the team name that comes alphabetically first in each pair on the leftmost column, and order the rows alphabetically by the first column.

**Answer:**

Row	teamA	teamB	state
1	Bearcats	Norse	KY
2	Cougars	Red Raiders	TX
3	Razorbacks	Red Wolves	AR

9. **Definition:** A **geographical location**  $L$  is a unique tuple (city, state, country).

**Definition:** A geographical location  $L$  “makes” points for a team  $T$  whenever a player that was born in  $L$  scores points for  $T$ .

(3 points) What three geographical locations made the most points for Stanford’s team in seasons 2013 through 2017, and how many points did they make?

Restrictions:

- For the purposes of this query, avoid using the “*birth\_place*” column.

**Answer:**

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

10. (4 points) 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? **Note:** These players did not all have to score 15+ points in the first half of the *same* 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.

**Answer:**

Row	team_market	num_players
1	Kentucky	14
2	Oregon	14
3	UCLA	14
4	Duke	13
5	Marquette	13

11. **Definition:** Team  $X$  is a **top performer** on season  $Y$  if no other team had more wins than  $X$  in the same season. This includes teams with either null or non-null markets.

(4 points) What **five** teams (identify them here by their “markets”) were top performers in the most seasons between 1900 and 2000 (inclusive), and how many times were they top performers? Output the team markets and the number of times each team was a top performer. If there are ties in the final output, break them by giving a higher ranking to team markets that come first alphabetically. Ignore teams with NULL markets only in the final output.

**Answer:**

Row	team_market	top_performer_count
1	University of California, Los	6

	Angeles	
2	University of Kentucky	6
3	Texas Southern University	5
4	University of Pennsylvania	5
5	Western Kentucky University	5

## Submission Instructions

Once you have written queries that answer all questions and conform to the given result schemas, you're ready to submit.

To submit:

1. Copy all the queries you wrote for Task C into the `project1_submission.py` file (available on the course website), pasting all of your queries into the corresponding places.
2. If you collaborated with others to generate your queries, add their names and SUNet IDs to the comment at the top of the `project1_submission.py` file.
3. Submit this Python file on Gradescope. **In order to be correctly graded, the file must be named `project1_submission.py`.**

You may resubmit as many times as you like; however, only the latest submission and timestamp will be saved, and we will use your latest submission for grading your work and determining any late penalties that may apply. Submissions via email will not be accepted!

### IMPORTANT SUBMISSION NOTES:

When you submit to Gradescope, we will run a syntax checker that will make sure that your SQL runs OK. It should run immediately and return whether the query ran OK or if there were errors - please **make sure that you get a positive result from this test** in your final submission.

**You will not see a final grade until after the project deadline.** The answers are provided above for the questions so that you may check your work yourself. It is your responsibility to ensure that your final submission is free from Python or SQL syntax errors and that you follow all instructions in this section.

We reserve the right to deduct points from your project if you do not follow the submission instructions, or if you have syntax errors in your queries.



## FAQ

### Question:

I'm getting syntax errors when I submit to Gradescope, but I don't see these syntax errors when I run on BigQuery.

### Answer:

Some things to check:

- Were queries copied correctly to the submission file?
- Are you using Standard SQL on BigQuery?
- Did you use backticks around table names?

Otherwise, this may be due to our autograder, which does not run with the same SQL that BigQuery does. For example, our autograder does not support window functions (OVER, PARTITION BY, etc.) and will throw an error. **If you are sure that your query works correctly (and is generalizable) with BigQuery SQL, then don't worry**; we'll be doing a manual pass through submissions, and as long as your query is correct using Standard SQL syntax then you will get full credit.

### Question:

Do I have to match the column names given in the solutions? For example, in question 7, do I have to name the column "upset\_percentage"?

### Answer:

You can name your columns whatever you want, as long as the content matches both in ordering and in values.