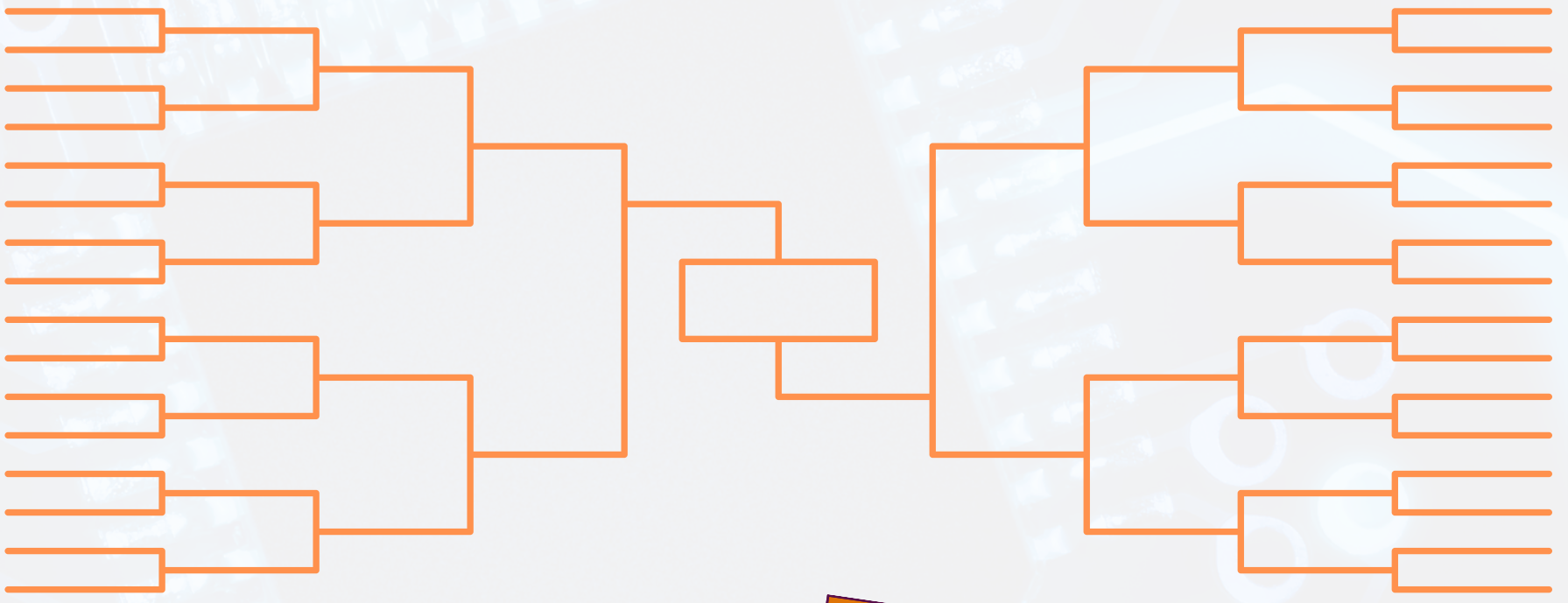
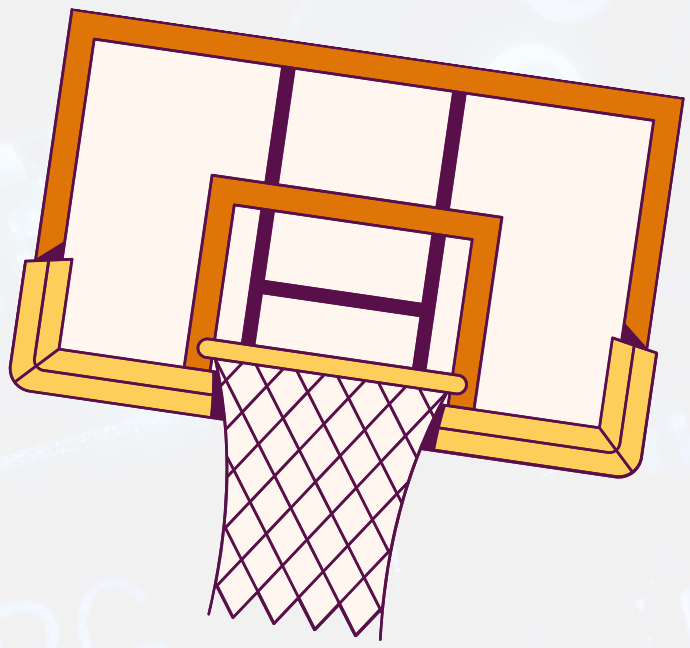
13 Essential SQL Commands Cheat Sheet

This a cheat sheet for the most common SQL commands I use as a data analyst.

The example dataset shows game results for NCAA basketball tournaments.





1. SELECT

Select the columns you want.

```sql
SELECT
  season
  , round
  , win_seed
  , win_school_ncaa
  , win_pts
  , lose_seed
  , lose_school_ncaa
  , lose_pts
  , num_ot
FROM `examples-378922.ncaa.basketball`
```

Row	season	round	win_seed	win_school_ncaa	win_pts	lose_seed	lose_school_ncaa	lose_pts	num_ot
1	1989	64	11	Evansville	94	06	Oregon St.	90	1
2	1997	32	14	Chattanooga	75	06	Illinois	63	0
3	1997	64	14	Chattanooga	73	03	Georgia	70	0
4	1994	64	04	Temple	61	13	Drexel	39	0
5	1985	64	08	Temple	60	09	Virginia Tech	57	0
6	1991	64	10	Temple	80	07	Purdue	63	0
7	1988	32	01	Temple	74	08	Georgetown	53	0
8	1999	16	06	Temple	77	10	Purdue	55	0

2. WHERE

Filter your data based on any criteria.

```
SELECT
  *
FROM `examples-378922.ncaa.basketball`
WHERE season BETWEEN 1990 and 1999
  AND round <= 16
```

3. ORDER BY

Sort your data any way you want. ⬇⬆

```
SELECT
  *
FROM `examples-378922.ncaa.basketball`
ORDER BY season, round desc
```

Row	season	round	days_from_epoch	game_date	day	win_seed	win_region	win_market
1	1985	64	5551	1985-03-14	Thursday	08	W	Temple
2	1985	64	5552	1985-03-15	Friday	02	X	Virginia Common
3	1985	64	5551	1985-03-14	Thursday	01	W	Georgetown
4	1985	64	5552	1985-03-15	Friday	11	Y	Boston College
5	1985	64	5552	1985-03-15	Friday	11	X	UTEP
6	1985	64	5552	1985-03-15	Friday	07	W	Syracuse
7	1985	64	5551	1985-03-14	Thursday	04	X	UNLV
8	1985	64	5551	1985-03-14	Thursday	11	Z	Auburn
9	1985	64	5552	1985-03-15	Friday	02	Y	Memphis
10	1985	64	5552	1985-03-15	Friday	07	Y	UAB
11	1985	64	5551	1985-03-14	Thursday	01	Y	Oklahoma
12	1985	64	5551	1985-03-14	Thursday	04	Y	Ohio State
13	1985	64	5552	1985-03-15	Friday	06	W	Georgia

4. GROUP BY

Aggregate data based on a calculation.

```
SELECT
  season
  , count(*) as num_games
  , sum(num_ot) as total_overtimes
  , max(win_pts - lose_pts) as biggest_victory
FROM `examples-378922.ncaa.basketball`
GROUP BY season
```

Row	season	num_games	total_overtimes	biggest_victory
1	1985	63	3	25
2	1986	63	5	49
3	1987	63	5	34
4	1988	63	2	40
5	1989	63	3	43
6	1990	63	5	35
7	1991	63	4	41

5. HAVING

Similar to WHERE but for aggregate functions .

```sql
SELECT
  season
  , count(*) as num_games
  , sum(num_ot) as total_overtimes
  , max(win_pts - lose_pts) as biggest_victory
FROM `examples-378922.ncaa.basketball`
GROUP BY season
HAVING max(win_pts - lose_pts) > 40
```

season	num_games	total_overtimes	biggest_victory
1989	63	3	43
1991	63	4	41
1999	63	3	41
2013	67	1	47
1986	63	5	49
2011	67	6	42
2001	64	2	43
1993	63	4	45

6. CASE WHEN

Add If/Else logic to a query.

```
SELECT
  season
  , win_school_ncaa
  , win_pts
  , lose_school_ncaa
  , lose_pts
  , win_pts - lose_pts as win_margin
  , case
      when win_pts - lose_pts >= 20
        then 'blowout'
      when win_pts - lose_pts <= 3
        then 'close game'
      else 'normal' end as result_type
FROM `examples-378922.ncaa.basketball`
```

season	win_school_ncaa	win_pts	lose_school_ncaa	lose_pts	win_margin	result_type
1996	Georgia Tech	103	Boston College	89	14	normal
1995	UCLA	102	UConn	96	6	normal
1990	UNLV	102	Little Rock	72	30	blowout
2009	Missouri	102	Memphis	91	11	normal
2004	UAB	102	Washington	100	2	close game
1988	Iowa	102	Florida St.	98	4	normal
1994	Kansas	102	Chattanooga	73	29	blowout
2012	Kentucky	102	Indiana	90	12	normal
1989	NC State	102	Iowa	96	6	normal
1990	Texas	102	Xavier	89	13	normal
2011	North Carolina	102	LIU Brooklyn	87	15	normal
1989	Michigan	102	Virginia	65	37	blowout
1992	Michigan	102	ETSU	90	12	normal

7. IN

Filter by a list of items.

```sql
SELECT
  *
FROM `examples-378922.ncaa.basketball`
WHERE win_school_ncaa in ('Kansas','Michigan','Duke')
```

8. LIKE

Returns true if data matches a text pattern.

```
SELECT
  DISTINCT win_school_ncaa
FROM `examples-378922.ncaa.basketball`
WHERE win_school_ncaa like '%Texas%'
```

Row	win_school_ncaa
1	Texas A&M
2	Texas
3	Texas Tech

9. LEFT JOIN

Join together data from multiple tables.

```sql
-- Get Winning Team Colors
  SELECT
    b.game_date
    , b.win_school_ncaa
    ,c.color
  FROM `examples-378922.ncaa.basketball` b
  LEFT JOIN `bigquery-public-data.ncaa_basketball.team_colors` c
    on b.win_team_id = c.id
```

Row	game_date	win_school_ncaa	color
1	1985-03-14	Temple	#9e1b34
2	1985-03-14	Georgetown	#00275d
3	1985-03-14	UNLV	#cc092f
4	1985-03-14	Auburn	#f58025
5	1985-03-14	Oklahoma	#a80532
6	1985-03-14	Ohio St.	#bb0000
7	1985-03-14	Louisiana Tech	#002f8b
8	1985-03-14	Kansas	#006ab5

10. UNION ALL

Append data together.

```sql
-- Combine Regular Season & Tournament Games
-- **Columns need to be the same.

-- Regular Season
SELECT
  season
  , name as win_team
  , points_game as win_team_pts
  , opp_name as lose_team
  , opp_points_game as lose_team_pts
FROM `bigquery-public-data.ncaa_basketball.mbb_historical_teams_games`
WHERE win = true

UNION ALL

-- Tournament
SELECT
  season
  , win_name
  , win_pts
  , lose_name
  , lose_pts
FROM `bigquery-public-data.ncaa_basketball.mbb_historical_tournament_games`
```

11. CTEs

Write clean SQL.

```sql
with winning_margin as (
  SELECT
    *
    , win_pts - lose_pts as win_margin
  FROM `examples-378922.ncaa.basketball`
)

SELECT *
FROM winning_margin
WHERE win_margin > 20
```

12. WINDOW FUNCTIONS

Perform calculations across rows.

```
SELECT
    game_date
    , win_name
    , lose_name
    , win_pts + lose_pts as total_points
    , ROW_NUMBER() OVER(PARTITION BY win_name ORDER BY game_date) AS team_game_num
    , SUM(win_pts+lose_pts) OVER (ORDER BY game_date, win_team_id) AS running_sum_pts
  FROM `examples-378922.ncaa.basketball`
```

Row	game_date	win_name	lose_name	total_points	team_game_num	running_sum_pts
1	1985-03-14	Tigers	Boilermakers	117	1	117
2	1985-03-14	Wildcats	Huskies	124	1	241
3	1985-03-14	Mustangs	Monarchs	153	1	394
4	1985-03-14	Redbirds	Trojans	113	1	507
5	1985-03-14	Ramblers	Gaels	117	1	624
6	1985-03-14	Rebels	Aztecs	165	1	789
7	1985-03-14	Fighting Irish	Beavers	149	1	938
8	1985-03-14	Buckeyes	Cyclones	139	1	1077
9	1985-03-14	Owls	Hokies	117	1	1194
10	1985-03-14	Hoyas	Mountain Hawks	111	1	1305

13. DATE FUNCTIONS

Work with dates and times.

```
SELECT
  game_date
  , extract(day from game_date) as day_of_month
  , current_date() as today
  , date_add(current_date(), interval 3 day) as today_plus_3
  , date_trunc(game_date, month) as first_of_month
  , last_day(game_date) as end_of_month
  , date_diff(last_day(game_date),game_date, day) as days_left_in_month
FROM `examples-378922.ncaa.basketball`
```

Row	game_date	day_of_m	today	today_plus_3	first_of_month	end_of_month	days_left_i
1	1989-03-17	17	2023-02-26	2023-03-01	1989-03-01	1989-03-31	14
2	1997-03-16	16	2023-02-26	2023-03-01	1997-03-01	1997-03-31	15
3	1997-03-14	14	2023-02-26	2023-03-01	1997-03-01	1997-03-31	17
4	1994-03-18	18	2023-02-26	2023-03-01	1994-03-01	1994-03-31	13
5	1985-03-14	14	2023-02-26	2023-03-01	1985-03-01	1985-03-31	17
6	1991-03-14	14	2023-02-26	2023-03-01	1991-03-01	1991-03-31	17
7	1988-03-20	20	2023-02-26	2023-03-01	1988-03-01	1988-03-31	11
8	1999-03-19	19	2023-02-26	2023-03-01	1999-03-01	1999-03-31	12
9	1999-03-14	14	2023-02-26	2023-03-01	1999-03-01	1999-03-31	17
10	1988-03-24	24	2023-02-26	2023-03-01	1988-03-01	1988-03-31	7

SUMMARY

1. SELECT
2. WHERE
3. ORDER BY
4. GROUP BY
5. HAVING
6. CASE WHEN
7. IN
8. LIKE
9. LEFT JOIN
10. UNION ALL
11. CTEs
12. WINDOW FUNCTIONS
13. DATE FUNCTIONS

Like this cheat sheet?

You'll **love** my BigQuery Jumpstart course.

You learn how to use BigQuery and **access hundreds** of free public datasets.

Link in bio.