&emsp;

<br/>

**AVERAGE TEMPERATURE**

* SELECT AVG(temperature)   
  FROM `skillful-coast-340323.demos.weather\_nyc`   
  WHERE date BETWEEN '2020-06-01' AND '2020-06-30'

**ALIAS**

Basic format for an AS query:

* SELECT column\_name(s)   
  FROM table\_name AS alias\_name;

Notice that AS is preceded by the table name and followed by the new nickname. It is a similar approach to aliasing a column:

* SELECT column\_name AS alias\_name  
  FROM table\_name;

If using AS results in an error when running a query because the SQL database you are working with doesn't support it, you can leave it out. In the previous examples, the alternate syntax for aliasing a table or column would be:

* FROM table\_name alias\_name
* SELECT column\_name alias\_name

**Arithmetic**

SQL can perform arithmetic for you. Just select what you want to +, -, \*, / . Don’t use spaces

SELECT

station\_name,

ridership\_2013,

ridership\_2014,

ridership\_2014-ridership\_2013 AS change\_2014\_raw

FROM `bigquery-public-data.new\_york\_subway.subway\_ridership\_2013\_present`

SELECT

station\_name,

ridership\_2013,

ridership\_2014,

ridership\_2015,

ridership\_2016,

(ridership\_2013+ridership\_2014+ridership\_2015+ridership\_2016) / 4 AS average

FROM `bigquery-public-data.new\_york\_subway.subway\_ridership\_2013\_present`

You can also calculate percentages within your data:

SELECT  
 Region

Small\_bags  
 Total\_bags

(small\_bags / total\_bags)\*100 AS small\_bags\_percent  
FROM avocado\_data.avocado\_prices

WHERE total\_bags <> 0

You can also use != 0 in place of <>0 or use the SAFE\_DIVIDE function

**BETWEEN**

* SELECT  
   Date, purchase\_price  
  FROM customer\_data.purchases  
  WHERE  
   Date BETWEEN ‘2020-12-01’ AND ‘2020-12-20’

**CAST**

* SELECT  
   CAST(purchase\_price AS FLOAT64)  
  FROM customer.data\_purchase  
  ORDER BY purchase\_price DESC
* SELECT  
   CAST(date AS date) AS date\_only  
  FROM customer\_data.purchases

The above statement changes SQL from recognizing the dates as datetime (2020-12-12T0:00:00) to only date (2020-12-12)

* CAST(expression AS typename) Where expression is the data to be converted and typename is the data type to be returned.

**Converting a number to a string:**

* SELECT CAST(MyCount AS String) FROM MyTable

In the above SQL statement, the following occurs: SELECT indicates that you will be selecting data from a table. CAST indicates that you will be converting the data you select to a different data type. AS comes before and identifies the data type which you are casting to. STRING indicates that you are converting the data to a string. FROM indicates which table you are selecting the data from

**Converting String to a number:**

* SELECT CAST(MyVarcharCol AS INT) FROM MyTable

CAST indicates that you will be converting the data you select to a different data type. AS comes before and identifies the data type which you are casting to. INT indicates that you are converting the data to an integer

**Convert date to a string:**

* SELECT CAST(MyDate AS STRING) FROM MyTable

**Converting a date to a datetime**: Datetime values have the format of YYYY-MM-DD hh: mm: ss format

* SELECT CAST (MyDate AS DATETIME) FROM MyTable

**The SAFE\_CAST function**: Using the CAST function in a query that fails returns an error in BigQuery. It returns null instead of error.

* SELECT SAFE\_CAST (MyDate AS STRING) FROM MyTable

**CONCAT**

* SELECT  
   CONCAT(product\_code, product\_color) AS new\_product\_code  
  FROM customer\_purchase.data  
  WHERE  
   Product = ‘couch’
* SELECT usertype  
   CONCAT(start\_station\_name, “ to “, end\_station\_name) AS route,  
   COUNT (\*) as num\_trips,  
  ROUND(AVG(cast(tripduration as int64)/60,2) AS duration  
  FROM big-query-public-data.ny  
  GROUP BY start\_station\_name, end\_station\_name, usertype  
  ORDER BY num\_trips DESC  
  LIMIT 10

Reminder: Make sure to use a backtick (`) instead of an apostrophe (') in the FROM statement.

About: ROUND(AVG(cast(tripduration as int64)/60,2) AS duration  
Big query stores numbers in a 64-bit memory system, which is why there's a 64 after integer in this case. we'll divide it by the number seconds in a minute (60) and tell it how far we want it to round, two decimal places(2).

**COALESCE**

* SELECT  
   COALESCE(product, product\_code) AS product\_info  
  FROM customer\_data.purchase

Return non-null values in a list

**COUNT/ COUNT DISTINCT**

COUNT returns the number of rows in a specified range. COUNT DISTINCT does the same, but it will not count repeating values. Use it after the SELECT line.

* SELECT   
   COUNT(warehouse.state) AS num\_states

Warehouse.state is the column and num\_states is the new column you are creating to return the count.

**CASE**

The CASE statement goes through one or more conditions and returns a value as soon as a condition is met

* SELECT  
   Customer\_id  
   CASE   
   WHEN first\_name = ‘Tnoy’ THEN ‘TONY’  
   ELSE first\_name  
   END AS cleaned\_name  
  FROM customer.data

**DISTINCT**

* SELECT DISTINCT fuel\_type  
  FROM cars.car\_info;
* SELECT DISTINCT name  
  FROM playlist  
  ORDER BY playlist\_id

**EXTRACT**

This is for if you want to use data from only one part of a column of cells- such as the date from a date format that includes more than the year. This seems like it is useful for if you arent planning on cleaning or manipulating your data before using it.

SELECT

EXTRACT (YEAR FROM STARTTIME) AS year,

COUNT (\*) AS number\_of\_rides

FROM

Address.address

GROUP BY Year

ORDER BY year

SELECT

ProductId,

SUM (Quantity) AS unitssold,

ROUND (MAX (UnitPrice), 2) AS UnitPrice,

EXTRACT (YEAR FROM DATE) AS year,

EXTRACT (MONTH FROM DATE) AS month

FROM `skillful-coast-340323.sales.sales`

GROUP BY

year, month, ProductId

ORDER BY

year, month, ProductId

LIMIT 1000

The ROUND (MAX (..), #) seems to need to be run because you cannot run UnitPrice on its own “SELECT list expression references column UnitPrice which is neither grouped nor aggregated at [4:5]”. Trying to run quantity on its own as a column got “SELECT list expression references column Quantity which is neither grouped nor aggregated at [3:5]”

**JOIN**

General join syntax:

SELECT

--table columns are inserted here

Table\_name1.column\_name

Table\_name2.column\_name

FROM

Table\_name1

JOIN

Table\_name2

ON table\_name1.column\_name = table\_name2.column\_name

(the column name is the key, be it primary key or foreign key that they share in common)

* SELECT   
   Customers.customer\_name,  
   Orders.product\_id,  
   Orders.ship\_date  
  FROM  
   Customers  
  INNER JOIN  
   Orders  
  ON customers.customer\_id = orders.customer\_id
* SELECT

employees.name AS employee\_name,

employees.role AS employee\_role,

departments.name AS department\_name

FROM employee\_data.employees

INNER JOIN

employee\_data.departments ON

employees.department\_id = departments.department\_id

* SELECT

employees.name AS employee\_name,

employees.role AS employee\_role,

departments.name AS department\_name

FROM employee\_data.employees

FULL OUTER JOIN

employee\_data.departments ON

employees.department\_id = departments.department\_id

SELECT  
`bigquery-public-data.world\_bank\_intl\_education.international\_education`.country\_name,

`bigquery-public-data.world\_bank\_intl\_education.country\_summary`.country\_code,

`bigquery-public-data.world\_bank\_intl\_education.international\_education`.value,

`bigquery-public-data.world\_bank\_intl\_education.country\_summary`.short\_name,

FROM

`bigquery-public-data.world\_bank\_intl\_education.international\_education`

INNER JOIN

`bigquery-public-data.world\_bank\_intl\_education.country\_summary`

ON `bigquery-public-data.world\_bank\_intl\_education.country\_summary`.country\_code = `bigquery-public-data.world\_bank\_intl\_education.international\_education`.country\_code

The same thing but using alias’ :

SELECT

edu.country\_name,

summary.country\_code,

edu.value

FROM `bigquery-public-data.world\_bank\_intl\_education.international\_education` AS edu

INNER JOIN

`bigquery-public-data.world\_bank\_intl\_education.country\_summary` AS summary

ON edu.country\_code = summary.country\_code

SELECT

AVG(edu.value) average\_value, summary.region

FROM

`bigquery-public-data.world\_bank\_intl\_education.international\_education` AS edu

INNER JOIN

`bigquery-public-data.world\_bank\_intl\_education.country\_summary` AS summary

ON edu.country\_code = summary.country\_code

WHERE summary.region IS NOT null

GROUP BY summary.region

ORDER BY average\_value DESC

^Average\_value is a new column

* SELECT

seasons.market AS university,

seasons.name AS team\_name,

seasons.wins,

seasons.losses,

seasons.ties,

mascots.mascot AS team\_mascot

FROM

`bigquery-public-data.ncaa\_basketball.mbb\_historical\_teams\_seasons` AS seasons

LEFT JOIN

`bigquery-public-data.ncaa\_basketball.mascots` AS mascots

ON

seasons.team\_id = mascots.id

WHERE

seasons.season = 1984

AND seasons.division = 1

ORDER BY

seasons.market

**LENGTH**

* SELECT length (title) AS letters\_in\_title, album\_id  
  FROM album  
  WHERE letters\_in\_title < 4

The function **LENGTH(title) < 4** will return any album names that are less than 4 characters long. The complete query is **SELECT \* FROM album WHERE LENGTH(title) < 4**. The LENGTH function counts the number of characters a string contains.

TRIM

**MIN/MAX**

* SELECT  
   MIN(length) AS min\_length,  
   MAX(length) AS max\_length  
  FROM cars.car\_info;

**Modulo**

An operator (%) that returns the remainder when one number is divided by another.

**Order By**

* SELECT \*  
  FROM movie\_data.movies  
  ORDER BY Release\_date DESC
* SELECT \*   
  FROM movies.data  
  WHERE Genre = ‘Comedy’  
  AND Revenue > 30000000  
  ORDER BY Release\_date DESC
* SELECT total  
  FROM invoice  
  WHERE billing\_city = "Chicago"

ORDER BY total ASC

* SELECT County\_of\_Residence   
  FROM `bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality`   
  ORDER BY Births ASC   
  LIMIT 10
* SELECT County\_of\_Residence   
  FROM `bigquery-public-data.sdoh\_cdc\_wonder\_natality.county\_natality`   
  WHERE year = '2018-01-01'  
  ORDER BY Births DESC   
  LIMIT 10

The year had to be in ‘ ‘ for it to work, and since it is in date formate, it would not work as simply 2018.

**SELECT**

The meteorologists who you’re working with have asked you to get the temperature, wind speed, and precipitation for stations La Guardia and JFK, for every day in 2020, in descending order by date, and ascending order by Station ID. Use the following query to request this information:

* SELECT stn, date,  
   IF(temp=9999.9, NULL, temp) AS temperature,  
   IF(wdsp="999.9", NULL, CAST(wdsp AS Float64)) AS wind\_speed,  
   IF(prcp=99.99, 0, prcp) AS precipitation  
  FROM `bigquery-public-data.noaa\_gsod.gsod2020`  
  WHERE stn="725030" -- La Guardia  
   OR stn="744860" -- JFK  
  ORDER BY date DESC, stn ASC

**-- Use the IF function to replace 9999.9 values, which the dataset description explains is the default value when temperature is missing, with NULLs instead.**

**-- Use the IF function to replace 999.9 values, which the dataset description explains is the default value when wind speed is missing, with NULLs instead. -- Use the IF function to replace 99.99 values, which the dataset description explains is the default value when precipitation is missing, with NULLs instead.**

**SUBSTR**

* SELECT customer\_id,  
   SUBSTR(country,1,3) AS new\_country  
  FROM customer  
  ORDER BY country

The statement **SUBSTR(country, 1, 3) AS new\_country** will retrieve the first 3 letters of each state name and store the result in a new column as *new\_country*. The complete query is **SELECT customer\_id, SUBSTR(country, 1, 3) AS new\_country FROM customer ORDER BY country**. The SUBSTR function extracts a substring from a string. This function instructs the database to return 3 characters of each country, starting with the first character.

* SELECT Invoice\_id,  
   SUBSTR(billing\_city,1,4) AS new\_city  
  FROM invoice  
  ORDER BY billing\_city

Billing city= city, 1=starting position in string, 4=how many you return

**SUBQUERIES**

SELECT  
 Station\_id,  
 Name,  
 Number\_of\_rides AS number\_of\_rides\_starting\_at\_station  
FROM  
 (  
 SELECT  
 Start\_station\_id,  
 COUNT (\*) number\_of\_rides  
 FROM  
 Bigquery-public-data.new\_york\_.citybike\_trips  
 GROUP BY  
 Start\_station\_id  
)  
AS station\_num\_trips  
INNER JOIN  
Bigquery-public-data.new\_york.citibike\_stations ON station\_id = start\_station\_id  
ORDER BY  
 Number\_of\_rides DESC

SELECT

Warehouse.warehouse\_id,

CONCAT (warehouse.state, ‘; ’, warehouse.warehouse\_alias) AS warehouse\_name,  
COUNT orders.order\_id AS number\_of\_orders,  
 (  
 SELECT   
 COUNT(\*)

FROM

Warehouse.orders\_orders Orders)

AS total\_orders

FROM warehouse\_orders.Warehouse Warehouse

LEFT JOIN warehouse\_orders.Orders Orders

ON orders.warehouse\_id = Warehouse.warehouse\_id  
GROUP BY  
 Warehouse.id, warehouse\_name

**UPDATE**

* UPDATE cars.car\_info  
   SET num\_of\_doors = "four"  
  WHERE make = "dodge" AND fuel\_type = "gas" AND body\_style = "sedan";

**WHERE**

* SELECT \*  
  FROM cars.car\_info   
  WHERE num\_of\_doors IS NULL;
* SELECT \*   
  FROM movies.data  
  WHERE Genre = ‘Comedy’

Because the genre is a string, you need to put the ‘ ‘ around the string name. Capitalizations matter

* SELECT CustomerId  
  FROM invoices  
  WHERE BillingCountry = 'Germany' AND Total > 5

**WITH**

With allows you to create temporary tables and query from them right within your query. You can also use SELECT INTO and CREATE TEMP TABLE

WITH trips\_over\_one\_hour AS (

SELECT \*

FROM `bigquery-public-data.new\_york\_citibike.citibike\_trips`

WHERE tripduration >= 60

)

SELECT COUNT (\*) AS cnt

FROM trips\_over\_one\_hour

In order to get this one to work (as I was writing myself) I had to take the ‘ ‘ from around FROM name,and make sure I had my commas in the right place. It took me a long time.

WITH

longest\_used\_bike AS (

SELECT

bikeid,

SUM(duration\_minutes) AS trip\_duration

FROM

bigquery-public-data.austin\_bikeshare.bikeshare\_trips

GROUP BY

bikeid

ORDER BY

trip\_duration DESC

LIMIT 1

)

## find station at which longest bike leaves most often

SELECT

trips.start\_station\_id,

COUNT(\*) AS trip\_ct

FROM

longest\_used\_bike AS longest

INNER JOIN

bigquery-public-data.austin\_bikeshare.bikeshare\_trips AS trips

ON longest.bikeid = trips.bikeid

GROUP BY

trips.start\_station\_id

ORDER BY

trip\_ct DESC

LIMIT 1