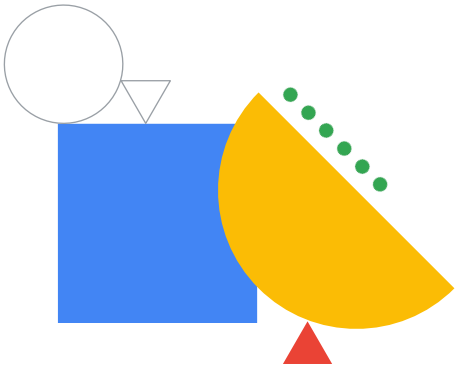
Google Cloud

Exploring your Public Dataset with SQL



Agenda

01 Common Data Exploration Techniques

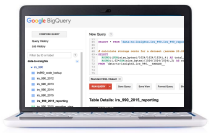
02 Use SQL to Query Public Datasets

- Demo: Exploring Ecommerce Data with SQL
- Lab: Explore your Ecommerce Dataset with SQL in BigQuery

In this module we will compare data exploration techniques and focus on writing SQL in BigQuery on our course dataset.

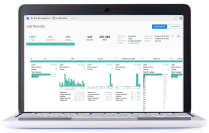
Common Data Exploration Techniques

You can explore data with SQL or with UI tools like Dataprep and Google Data Studio



SQL + Web UI

- Flexible, fast, and familiar
- Requires SQL knowledge



Data Preparation Tools

- GUI for exploring columns and rows
- Fast summary statistics

Visualization Tools

- Visually shape and re-shape quickly
- See data a different way

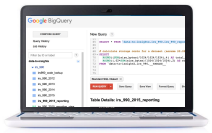
The three primary options for exploring a dataset are:

- Write some SQL and use the UI
- Use a Data Preparation or BI Tool
- Visualize your raw data with another Tool

SQL syntax hasn't changed all that significantly since the 1980s and, out of the three listed, is the most core to your skillset as a data analyst.

As for data preparation and visualization tools, we will also cover those later in this course.

Writing SQL is a fast and familiar way to explore datasets



SQL + Web UI

- Flexible, fast, and familiar
- Requires SQL knowledge

Data Preparation Tools

- GUI for exploring columns and rows
- Fast summary statistics

Visualization Tools

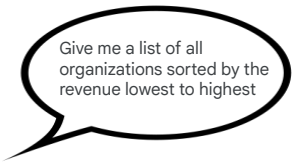
- Visually shape and re-shape quickly
- See data a different way

Let’s focus on exploring through SQL first.

Steps to explore data through SQL

01

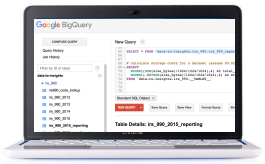
Ask good questions



Question

02

Know your data



Dataset

03

Write good SQL

```
SELECT
  name,
  revenue
FROM dataset
ORDER BY revenue;
```

SQL - Structured Query Language

Exploring a dataset through SQL is more than just writing good code. You need to know what destination you're heading towards and the general layout of your data. Good data analysts will explore how the dataset is structured even before writing a single line of code.

Questions:
What type of data am I interested in?
Financial Nonprofit Organizations

What specifically about that data?
Organization revenue

How do I want to receive the results?
Sorted by highest revenue first

Dataset:
What datasets do I have available to me? Do I need to find and upload my own?
How is the data structured? Are there multiple tables? Important fields?
How much data is there to explore?

SQL:
How do I translate my question into a SQL query?
Is my data clean?

What fields should I focus on?
Do I need to perform any aggregations?
Do I need data from multiple tables?

Use SQL to Query Public Datasets

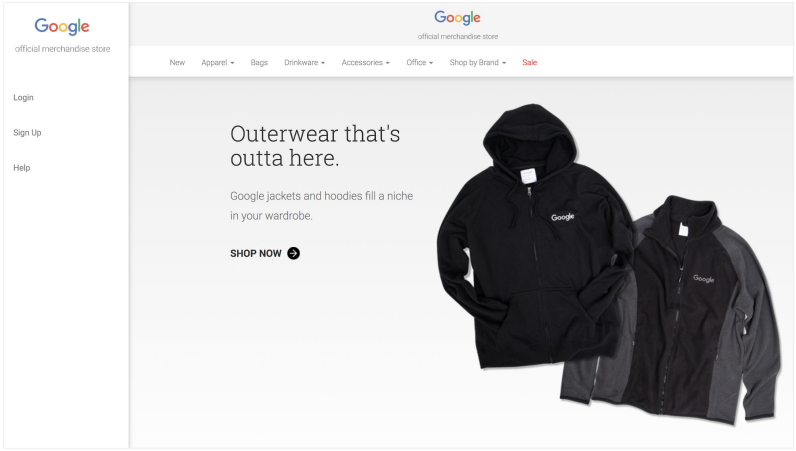
BigQuery hosts 150+ public datasets for SQL practice

Public datasets include flights, taxi cab logs, weather recordings, and many more.

Example SQL code is provided for practice.

Your course dataset is millions of records of ecommerce session data

Google's merchandise store analytics are available to query in BigQuery.



Google Analytics

You are managing an online storefront. What do you want to know about your visitors?

Website analytics:

1. ???

You are managing an online storefront. What do you want to know about your visitors?

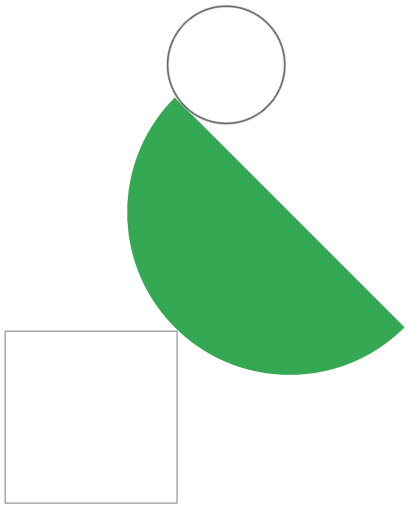
Website analytics:

1. Popular products
2. Popular pages
3. Time on site
4. Traffic source
5. Transactions
6. Device (mobile)
7. and more!

Demo

Exploring Ecommerce Data with SQL

SQL syntax review + BigQuery tips and tricks



Refer to
https://github.com/GoogleCloudPlatform/training-data-analyst/tree/master/courses/data-to-insights/demos/explore-data-with-sql.sql

What is wrong with the below query?

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
all_sessions
LIMIT 10
```


What's missing in the above query?

Be sure to include the dataset name

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
<dataset-name>.all_sessions
LIMIT 10
```


What's missing in the above query?

Use backticks ` around project names in SQL only if they contain hyphens

```
SELECT column
FROM
`project-id.dataset.table`
```


Use backticks `` and not brackets [ ] for table names.

If you omit project-name, BigQuery will assume the project is your current one

```
SELECT column
FROM dataset.table
```


Use backticks `` and not brackets [ ] for table names.

Read BigQuery error messages for helpful tips

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
all_sessions
LIMIT 10
```

Error: Table name "all_sessions" cannot be resolved: dataset name is missing.

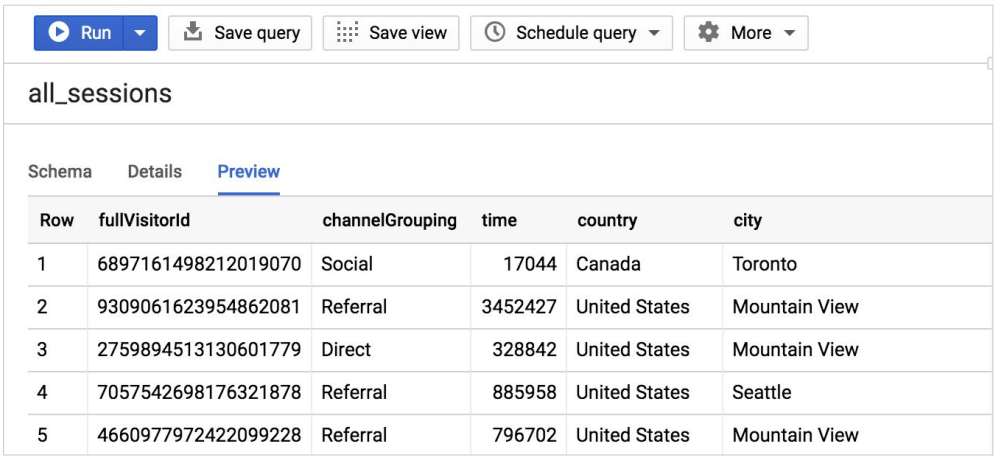
What's missing in the above query?

Avoid using SELECT * to explore data. Try the built-in Preview tab in BigQuery

```
SELECT
*
FROM
`data-to-insights.ecommerce.all_sessions`
```


If you must use SELECT *, be sure to include a limit.

Quickly explore rows using the built-in Preview tab



Row	fullVisitorId	channelGrouping	time	country	city
1	6897161498212019070	Social	17044	Canada	Toronto
2	9309061623954862081	Referral	3452427	United States	Mountain View
3	2759894513130601779	Direct	328842	United States	Mountain View
4	7057542698176321878	Referral	885958	United States	Seattle
5	4660977972422099228	Referral	796702	United States	Mountain View

If you must use SELECT *, be sure to include a limit.

BigQuery charges for data processed + storage

BigQuery job types:

- **Query** - charged by bytes processed
- **Load Data** - free
- **Extract** - free
- **Copy** - free

This query will process 702.6MB when run. 

Note: storing data in BigQuery is a separate cost

Traditional data warehouse costs include:
Hardware
Licensing
Maintenance

BigQuery:
Available as a fully-managed “NoOps” service
You save on hardware, software, maintenance costs

You get 1 TB per month of query processing at no cost

Refer to this page for the latest pricing calculator:
https://cloud.google.com/products/calculator/

Traditional data warehouse costs include:
Hardware
Licensing
Maintenance

BigQuery:
Available as a fully-managed “NoOps” service
You save on hardware, software, maintenance costs

Recap: Avoid selecting columns and rows you don't need

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
`data-to-insights.ecommerce.all_sessions`
LIMIT 10
```


If you must use SELECT *, be sure to include a limit

Note: If you run the exact same query twice, you will get the advantage of query cache*

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
`data-to-insights.ecommerce.all_sessions`
LIMIT 10
```

Query complete (0.1 sec elapsed, cached)

If you must use SELECT *, be sure to include a limit

*Unless you have non-deterministic elements

```
SELECT
  current_timestamp(),
  fullVisitorId,
  country,
  timeOnSite
FROM
`data-to-insights.ecommerce.all_sessions`
LIMIT 10
```


If you must use SELECT *, be sure to include a limit

Use ORDER BY and LIMIT to get the top 10 visitors who spent time on our website ...

```
SELECT
  fullVisitorId,
  country,
  timeOnSite
FROM
`data-to-insights.ecommerce.all_sessions`
ORDER BY timeOnSite DESC
LIMIT 10
```


If you must use SELECT *, be sure to include a limit

... and deduplicate rows with DISTINCT

Row	fullVisitorId	country	timeOnSite
1	0824839726118485274	United States	19017
2	0824839726118485274	United States	19017
3	0824839726118485274	United States	19017
4	0824839726118485274	United States	19017
5	0824839726118485274	United States	19017
6	0824839726118485274	United States	19017
7	0824839726118485274	United States	19017
8	0824839726118485274	United States	19017
9	0824839726118485274	United States	19017
10	0824839726118485274	United States	19017

Row	fullVisitorId	country	timeOnSite
1	0824839726118485274	United States	19017
2	2706961341001088633	United States	15047
3	9894955795481014038	Venezuela	15020
4	596895434219823695	Venezuela	14279
5	4742180546650265795	Panama	12853
6	9264804092676520813	Venezuela	12466
7	6957245643416321514	United States	12136
8	1957458976293878100	United States	11848
9	8826538902252293768	United States	11316
10	7498695963354635199	United States	11275

If you must use SELECT *, be sure to include a limit

Let's create a calculated fields for session duration in minutes (instead of seconds)

```
SELECT DISTINCT
  fullVisitorId,
  country,
  timeOnSite / 60 AS session_time_minutes
FROM
`data-to-insights.ecommerce.all_sessions`
ORDER BY session_time_minutes DESC
LIMIT 10
```

Row	fullVisitorId	country	session_time_minutes
1	0824839726118485274	United States	316.95
2	2706961341001088633	United States	250.78333333333333
3	9894955795481014038	Venezuela	250.33333333333334
4	596895434219823695	Venezuela	237.98333333333332
5	4742180546650265795	Panama	214.21666666666667
6	9264804092676520813	Venezuela	207.76666666666668
7	6957245643416321514	United States	202.26666666666668
8	1957458976293878100	United States	197.46666666666667
9	8826538902252293768	United States	188.6
10	7498695963354635199	United States	187.91666666666666

If you must use SELECT *, be sure to include a limit

Let's use a ROUND function to clean up the results

```
SELECT DISTINCT
  fullVisitorId,
  country,
  ROUND(timeOnSite / 60,2) AS session_time_minutes
FROM
`data-to-insights.ecommerce.all_sessions`
ORDER BY session_time_minutes DESC
LIMIT 10
```

Row	fullVisitorId	country	session_time_minutes
1	0824839726118485274	United States	316.95
2	2706961341001088633	United States	250.78
3	9894955795481014038	Venezuela	250.33
4	596895434219823695	Venezuela	237.98
5	4742180546650265795	Panama	214.22
6	9264804092676520813	Venezuela	207.77
7	6957245643416321514	United States	202.27
8	1957458976293878100	United States	197.47
9	8826538902252293768	United States	188.6
10	7498695963354635199	United States	187.92

If you must use SELECT *, be sure to include a limit

SQL functions perform actions on inputs

```
ROUND(<field>,<decimals>)
Function = Performs an Action
Parameters = Inputs you provide
```

Use the right function for the right job

- **String Manipulation Functions - FORMAT( ) ...**
- Aggregation Functions
- Data Type Conversion Functions
- Date Functions
- Statistical Functions
- Analytic Functions
- User-defined Functions

BigQuery Functions Reference

We will be introducing these to you in context over the rest of this course

Data Type Conversion

Aggregation = perform calculations over a set of values (like SUM, COUNT, MIN, MAX)

String Manipulation = make every letter uppercase, pull the left 5 characters, format

Statistical = standard deviation, variance, and more

Analytic = perform aggregations over a subset or window of data

User-defined = write your own function in SQL or even Javascript

Let's filter for all sessions greater than 4 hours (240 minutes)

```
SELECT DISTINCT
  fullVisitorId,
  country,
  ROUND(timeOnSite / 60,2) AS
session_time_minutes
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE session_time_minutes > 240
ORDER BY session_time_minutes DESC
LIMIT 10

… Error: session_time_minutes undefined
```


Newly defined aliases in the SELECT statement cannot be used yet for filtering but are allowed in ORDER BY, GROUP BY, HAVING

If you do not want to repeat the calculation in the WHERE clause, consider starting with a sub-query and then filtering (we will discuss later)

In SQL, you cannot use an aliased field in the WHERE clause

```
SELECT DISTINCT
  fullVisitorId,
  country,
  ROUND(timeOnSite / 60,2) AS
session_time_minutes
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE ROUND(timeOnSite / 60,2) > 240
ORDER BY session_time_minutes DESC
LIMIT 10
```

Row	fullVisitorId	country	session_time_minutes
1	0824839726118485274	United States	316.95
2	2706961341001088633	United States	250.78
3	9894955795481014038	Venezuela	250.33

When SQL reads the table from disk, it filters for the columns you want returned (where clause). At the time of pulling the data from disk our aliased columns in the select statement are not interpreted and are thus not available.

In SQL, you can use aliased fields elsewhere (like ORDER BY)

```
SELECT DISTINCT
  fullVisitorId,
  country,
  ROUND(timeOnSite / 60,2) AS
session_time_minutes
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE ROUND(timeOnSite / 60,2) > 240
ORDER BY session_time_minutes DESC
LIMIT 10
```

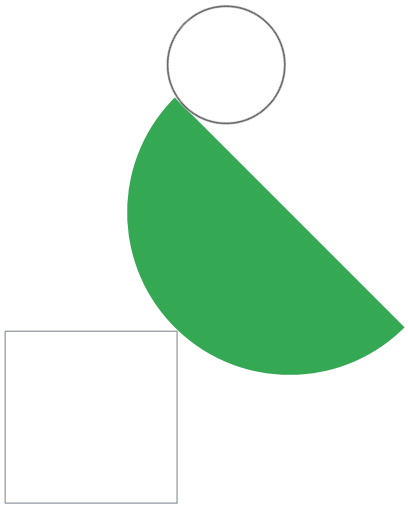
Row	fullVisitorId	country	session_time_minutes
1	0824839726118485274	United States	316.95
2	2706961341001088633	United States	250.78
3	9894955795481014038	Venezuela	250.33

After the data is pulled however, you can use aliased fields in ORDER BY

Demo

Exploring Ecommerce Data with SQL

Demo using hotkeys



Refer to
https://github.com/GoogleCloudPlatform/training-data-analyst/tree/master/courses/data-to-insights/demos/explore-data-with-sql.sql

Demo: explore-data-with-sql.sql

Look for section on “Demo using hotkeys”

Demo: Use BigQuery UI to quickly explore schemas

In your query, hold ctrl or command (mac) to **highlight table names**

Click a **`table_name`**

View the **Schema**

Click on **Column names** to automatically add into your query

Next, Click on **Preview** to see a sample of data values

Refer to
https://github.com/GoogleCloudPlatform/training-data-analyst/tree/master/courses/data-to-insights/demos/explore-data-with-sql.sql

Demo: explore-data-with-sql.sql

Look for section on “Demo using hotkeys”

Use the right function for the right job

- String Manipulation Functions - FORMAT( ) ...
- **Aggregation Functions - SUM( ) COUNT( ) AVG( ) MAX( ) ...**
- Data Type Conversion Functions
- Date Functions
- Statistical Functions
- Analytic Functions
- User-defined Functions

BigQuery Functions Reference

We will be introducing these to you in context over the rest of this course

Data Type Conversion

Aggregation = perform calculations over a set of values (like SUM, COUNT, MIN, MAX)

String Manipulation = make every letter uppercase, pull the left 5 characters, format

Statistical = standard deviation, variance, and more

Analytic = perform aggregations over a subset or window of data

User-defined = write your own function recipe in SQL or even Javascript

Perform calculations over values with aggregation

```
SELECT
  COUNT(DISTINCT fullVisitorId) AS unique_users
FROM
`data-to-insights.ecommerce.all_sessions`
```

Row	unique_users
1	389934

Any non-aggregated fields must be in GROUP BY

```
SELECT
  COUNT(DISTINCT fullVisitorId) AS unique_users,
  country
FROM
`data-to-insights.ecommerce.all_sessions`
GROUP BY country
ORDER BY unique_users DESC
LIMIT 5
```

Row	unique_users	country
1	205574	United States
2	22799	India
3	18140	United Kingdom
4	14774	Canada
5	8691	Germany

Investigate uniqueness with COUNT(DISTINCT field)

```
SELECT
  COUNT(DISTINCT fullVisitorId) AS unique_users,
  COUNT(fullVisitorId) AS users
FROM
`data-to-insights.ecommerce.all_sessions`
```

Row	unique_users	users
1	389934	21493109

What this tells us is the level of detail for the all_sessions table is not at the user level but at a greater level of detail (like maybe the user - pageview or user - product view level).

Filter for duplicates with COUNT and HAVING

```
SELECT
  fullVisitorId,
  COUNT(fullVisitorId) AS records
FROM
`data-to-insights.ecommerce.all_sessions`
GROUP BY fullVisitorId
HAVING records > 1
LIMIT 10
```

Row	fullVisitorId	records
1	8919336618754256169	1146
2	4605997863482509872	276
3	7228946486690765003	264
4	912127983342148918	151
5	7419028556980464579	68
6	832152661091318994	105
7	192907985600193802	504
8	0267830135658533597	278
9	0794270160070827977	171
10	6949496197162068722	556

Looks like some EINs have 7 records in the 2015 tax filings table. This is unusual.

Can we count just the total EINs with duplicates (e.g. exclude those with an ein_count of only 1)?

Insight: fullVisitorId can be duplicative because of product views

```
SELECT
  fullVisitorId,
  date,
  time,
  pageviews,
  pageTitle,
  v2ProductName
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE fullVisitorId = '8919336618754256169'
ORDER BY date, time
LIMIT 100
```

Row	fullVisitorId	date	time	pageviews	pageTitle	v2ProductName
1	8919336618754256169	20160822	0	24	Apparel	Google Women's V-Neck Tee Charcoal
2	8919336618754256169	20160822	0	24	Apparel	Google Women's Scoop Neck Tee Black
3	8919336618754256169	20160822	0	24	Apparel	Android Women's Long Sleeve Blended Cardigan Grey
4	8919336618754256169	20160822	0	24	Apparel	Google Toddler Raglan Shirt Blue Heather/Navy
5	8919336618754256169	20160822	0	24	Apparel	Google Men's Short Sleeve Badge Tee Charcoal
6	8919336618754256169	20160822	0	24	Apparel	Google Women's Short Sleeve Performance Tee Pewter
7	8919336618754256169	20160822	0	24	Apparel	Google Women's Short Sleeve Performance Tee Black
8	8919336618754256169	20160822	0	24	Apparel	Google Women's Short Sleeve Performance Tee Navy
9	8919336618754256169	20160822	0	24	Apparel	Android Men's Vintage Tank
10	8919336618754256169	20160822	0	24	Apparel	YouTube Onesie Heather

Use the right function for the right job

- String Manipulation Functions - FORMAT( ) ...
- Aggregation Functions - SUM( ) COUNT( ) AVG( ) MAX( ) ...
- **Data Type Conversion Functions - CAST( ) ...**
- **Date Functions - PARSE_DATETIME( ) ...**
- Statistical Functions
- Analytic Functions
- User-defined Functions

BigQuery Functions Reference

We will be introducing these to you in context over the rest of this course

Data Type Conversion

Aggregation = perform calculations over a set of values (like SUM, COUNT, MIN, MAX)

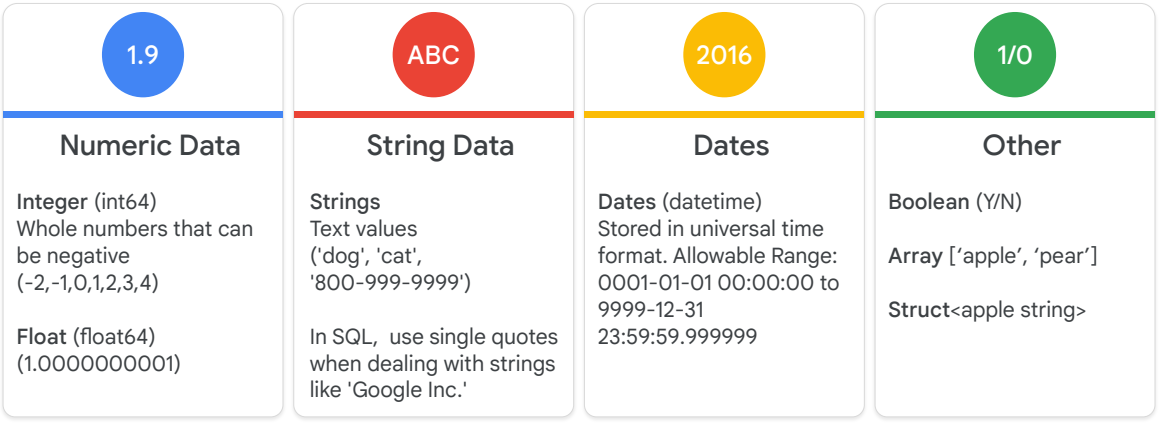
String Manipulation = make every letter uppercase, pull the left 5 characters, format

Statistical = standard deviation, variance, and more

Analytic = perform aggregations over a subset or window of data

User-defined = write your own function recipe in SQL or even Javascript

Comparing BigQuery data types



BigQuery Functions Reference

BigQuery data types
https://cloud.google.com/bigquery/docs/reference/standard-sql/data-types

Using CAST to convert between data types

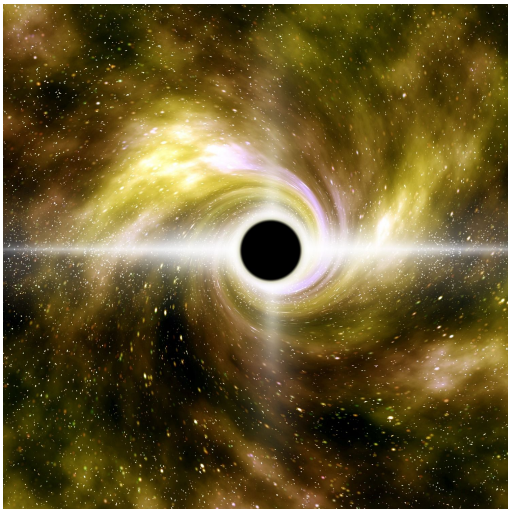
- `SELECT CAST("12345" AS INT64)`
  - `12345`
- `SELECT CAST("2017-08-01" AS DATE)`
  - `2017-08-01`
- `SELECT CAST(1112223333 AS STRING)`
  - `"1112223333"`
- `SELECT SAFE_CAST("apple" AS INT64)`
  - `NULL`

Casting allows you to **treat** one data type as another. This is particularly useful if a Function expects to see an input in a specific data type. Example, our PARSE_TIMESTAMP( ) expects a string input which is why we had to convert tax_period to a string first.

Although rare, you may run across safe_cast if you're unsure if a column may contain "apple" in addition to "123" and "456". If you tried cast( ) normally, this would error out. Now it just produces a NULL for that value.

What is a NULL value?

- NULLs are valid values
- NULL is the absence of data or an empty set
- NULL is not the same as “” or a valid blank string value



Much like black holes are the absence of light, NULLs are the absence of data

NULLs may or may not be included in aggregations. By default NULLs are respected.

We will cover equivalency of NULLs when we discuss data cleanup. You cannot simply do IF VALUE = NULL since NULLS can never be equivalent to anything, not even NULL = NULL.

Filter all records for only those with transaction IDs

```
SELECT DISTINCT
  fullVisitorId,
  date,
  time,
  pageviews,
  pageTitle,
  v2ProductName,
  transactionId
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE transactionId IS NOT NULL
ORDER BY date, time
LIMIT 100
```


Let’s get a list of all our orders

Issue: One transaction can have many products (differing level of data granularity)

Row	fullVisitorId	date	time	pageviews	pageTitle	v2ProductName	transactionId
1	4631129802514106099	20160801	144555	15	Checkout Confirmation	24 oz YouTube Sergeant Stripe Bottle	ORD20160801423
2	6027268712782791947	20160801	210118	21	Checkout Confirmation	Google Men's 100% Cotton Short Sleeve Hero Tee White	ORD20160801430
3	6027268712782791947	20160801	210118	21	Checkout Confirmation	Google Men's 100% Cotton Short Sleeve Hero Tee Red	ORD20160801430
4	5563168194966233133	20160801	259524	17	Checkout Confirmation	Google Infant Zip Hood Pink	ORD20160801436
5	5563168194966233133	20160801	259524	17	Checkout Confirmation	Google Baby Essentials Set	ORD20160801436
6	1468560120795000800	20160801	282300	17	Checkout Confirmation	Deluge Waterproof Backpack	ORD20160801443
7	1468560120795000800	20160801	282300	17	Checkout Confirmation	PaperMate Ink Joy Retractable Pen	ORD20160801443
8	1468560120795000800	20160801	282300	17	Checkout Confirmation	Google Men's 100% Cotton Short Sleeve Hero Tee Black	ORD20160801443

Solution 1: Use a string aggregation function to combine all products ordered:

```
SELECT
  transactionId,
  (totalTransactionRevenue / 1000000) AS revenue,
  STRING_AGG(v2ProductName) AS product_list
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE transactionId IS NOT NULL
GROUP BY transactionId, totalTransactionRevenue
ORDER BY revenue DESC
LIMIT 100
```

Solution 1: Products have been rolled up into a single comma separated field

Row	transactionId	revenue	product_list
1	ORD201704052324	47082.06	Leatherette Journal,Google 5-Panel Cap,Google Luggage Tag,YouTube Twill Cap,Google Sunglasses,Red Spiral Google Notebook,20 oz Stai
2	ORD201704182260	32153.82	Android Baby Esssentials Set,Android Youth Short Sleeve T-shirt Aqua,Google Women's Short Sleeve Hero Tee Sky Blue,Android Lifted Men's Short Sleeve Tee Blue,YouTube Youth Short Sleeve Tee Red,Google Youth Short Sleeve Tee Red,Google Youth Short Sleeve Tee Red,Google Toddler Short Sleeve Tee White,Android Toddler Short Sleeve T-shirt Pink,Android Youth Short Sleeve T-shirt Pewter,YouTube Youth Short Sleeve Tee Red,Android Toddler Short Sleeve T-shirt Pewter,Google Youth Short Sleeve T-shirt Royal Blue,Google Toddler Short Sleeve Tee Red,Android Youth Short Sleeve T-shirt Aqua,Android Lifted Men's Short Sleeve Tee Blue,Android Baby Esssentials Set,Android Toddler Short Sleeve T-shirt Pink,Google Youth Short Sleeve T-shirt Royal Blue,Google Youth Short Sleeve T-shirt Royal Blue,YouTube Youth Short Sleeve Tee Red,Google Men's Zip Hoodie,Google Men's Zip Hoodie,Google Toddler Short Sleeve T-shirt Royal Blue,Android Toddler Short Sleeve T-shirt Pink,Google Youth Short Sleeve Tee Red,Google Spiral Journal with Pe…
3	ORD201707182786	25251.26	Google Luggage Tag,YouTube Twill Cap,Google Twill Cap,YouTube Hard Cover Journal,Sport Bag,Google Water Resistant Bluetooth Speake
4	ORD201702142258	17859.5	Google Men's 100% Cotton Short Sleeve Hero Tee White,Google Wool Heather Cap Heather/Navy,Recycled Mouse Pad,Yoga Mat

Solution 2: Use an array aggregation function to combine all products ordered:

```
SELECT
  transactionId,
  (totalTransactionRevenue / 1000000) AS revenue,
  ARRAY_AGG(v2ProductName) AS product_list
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE transactionId IS NOT NULL
GROUP BY transactionId, totalTransactionRevenue
ORDER BY revenue DESC
LIMIT 100
```

Solution 2: All products are now elements in the product_list array (still just one row!)

Row	transactionId	revenue	product_list
1	ORD201704052324	47082.06	Google 22 oz Water Bottle
			Colored Pencil Set
			Leatherette Journal
			Google Sunglasses
			Google Luggage Tag
			YouTube Twill Cap
			Android Luggage Tag
			YouTube Luggage Tag
			Google Sunglasses
			20 oz Stainless Steel Insulated Tumbler

Arrays give you flexibility for data at differing levels of granularity... we'll revisit them in great detail later

Row	transactionId	revenue	distinct_prroducts_ordered	product_quantity	product_list
1	ORD201704052324	47082.06	15	200	Google 22 oz Water Bottle
				400	Colored Pencil Set
				100	Google Luggage Tag
				144	YouTube Twill Cap
				600	Leatherette Journal
				500	Google 5-Panel Cap
				750	Google Sunglasses
				100	Android Luggage Tag
				200	20 oz Stainless Steel Insulated Tumbler
				1000	YouTube Custom Decals


```
SELECT
  transactionId,
  (totalTransactionRevenue / 1000000) AS revenue,
  ARRAY_LENGTH(ARRAY_AGG(DISTINCT v2ProductName)) AS
distinct_prroducts_ordered,
  ARRAY_AGG(productQuantity) AS product_quantity,
  ARRAY_AGG(v2ProductName) AS product_list
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE transactionId IS NOT NULL
GROUP BY transactionId, totalTransactionRevenue
ORDER BY revenue DESC
LIMIT 100
```

Parsing string values with string functions

- `CONCAT("12345","678")`
  - `"12345678"`
- `ENDS_WITH("Apple","e")`
  - `true`
- `LOWER("Apple")`
  - `"apple"`
- `REGEXP_CONTAINS("Lunchbox",r"^*box$")`
  - `true`

RegEx function examples
https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#regexp_contains

Finding all products with ‘shirt’ the name

```
SELECT DISTINCT
  v2ProductName
FROM
`data-to-insights.ecommerce.all_sessions`
WHERE LOWER(v2ProductName) LIKE '%shirt%'
LIMIT 100
```

Row	v2ProductName
1	BLM Sweatshirt
2	Google Toddler Raglan Shirt Blue Heather/Navy
3	BLM Sweatshirt (Pre-Order)
4	Google Toddler Short Sleeve T-shirt Green
5	Google Women's Vintage T-Shirt Black
6	Google Youth Short Sleeve T-shirt Green
7	Android Toddler Short Sleeve T-shirt Pink
8	Android Youth Short Sleeve T-shirt Pewter

Using a String Function on name to turn everything lowercase and then finding the word ‘shirt’ anywhere in the name

Why the LOWER( )? If we’re unsure whether the field is uppercase, lowercase, or a combination of the two.

When writing complex queries, consider breaking apart the logic using WITH clauses

```
WITH product_views AS (
     # all products
     SELECT
        COUNT(DISTINCT fullVisitorId) AS visitors,
        v2ProductName
     FROM
     `data-to-insights.ecommerce.all_sessions`
     GROUP BY v2ProductName
)
# popular shirts
SELECT * FROM product_views
WHERE LOWER(v2ProductName) LIKE '%shirt%'
AND visitors > 10000
ORDER BY visitors DESC
```

WITH clauses are effectively sub-queries and can be changed together.

If you are continually using a certain WITH clause, consider promoting it to a view or table (covered soon).

WITH product_views AS (
all products
SELECT
  COUNT(DISTINCT fullVisitorId) AS visitors,
  v2ProductName
FROM
`data-to-insights.ecommerce.all_sessions`
GROUP BY v2ProductName
)

popular shirts
SELECT * FROM product_views
WHERE LOWER(v2ProductName) LIKE '%shirt%'
AND visitors > 10000
ORDER BY visitors DESC

Use the right function for the right job

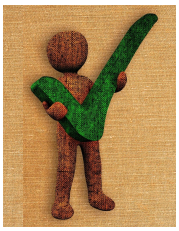
- String Manipulation Functions - FORMAT( ) ...
- Aggregation Functions - SUM( ) COUNT( ) AVG( ) MAX( ) ...
- Data Type Conversion Functions - CAST( ) ...
- Date Functions - PARSE_DATETIME( ) ...
- **Statistical Functions**
- **Analytic Functions**
- **User-defined Functions**

We will return to these three in a future module, need to build up our fundamentals first.

Summary: Explore your dataset with BigQuery and SQL

Explore large datasets quickly with SQL in the BigQuery UI

Comment your code and use #standardSQL for best results



Fix common SQL syntax errors by using the validator

Practice your SQL skills on over 50+ Public Datasets with example queries

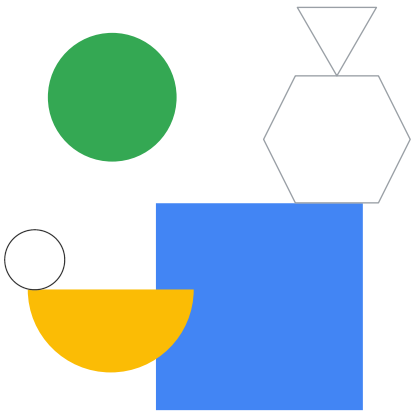
Looking back, this is the first module where we really dive into some of the core capabilities of BigQuery. As you have seen exploring massive datasets with SQL is what the tool does best and what you will need to master as a data analyst. Be sure to use standard sql mode and make best friends with the validator (it will also help you price out the bytes you will process -- which we'll cover later). Lastly, if you're looking for other datasets to explore, complete with example SQL queries, search for BigQuery Public Datasets to see the full list.

Let's test your knowledge with an interactive lab where we've written some pretty ugly and broken queries for you to fix on our IRS dataset.

Links:
https://cloud.google.com/public-datasets/

Lab Intro

Explore your Ecommerce Dataset with SQL in BigQuery



Lab objectives

01 Access an ecommerce dataset

02 Look at the dataset metadata

03 Remove duplicate entries

04 Write and execute queries