Module 3

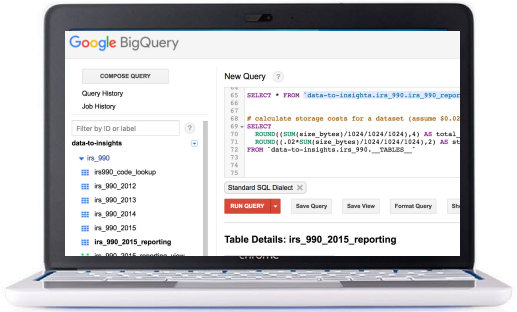
Exploring your Data with SQL

In this module we will:

- **Compare Common Data Exploration Techniques**
- Learn How to Code High Quality Standard SQL
- Explore Google BigQuery Public Datasets

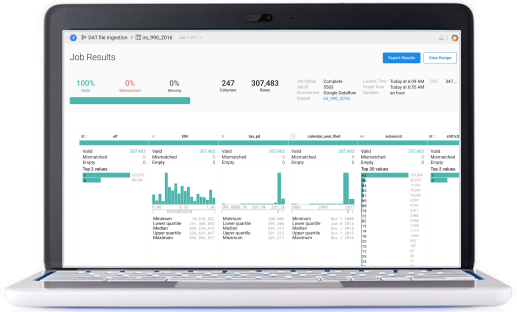
Options for exploring a dataset are complimentary

SQL + Web UI



- Flexible, Fast, and Familiar
- Requires SQL knowledge

Data Preparation Tools



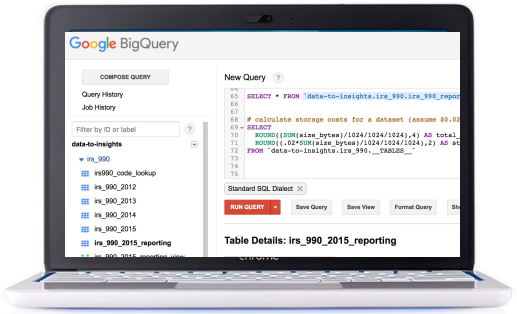
- GUI for Exploring Columns and Rows
- Fast Summary Statistics

Visualization Tools

- Visually Shape and Re-Shape Quickly
- See Data a Different Way

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

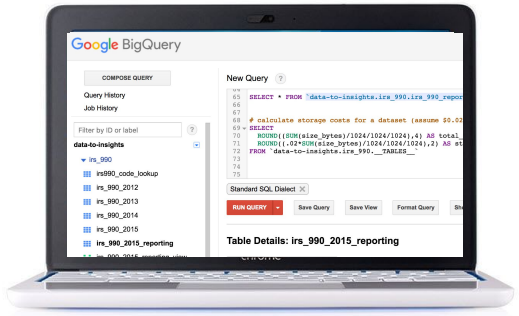
Google Cloud

Steps to Explore Data through SQL

Question

Give me a list of all organizations sorted by the revenue lowest to highest

Dataset



SQL
Structured Query Language

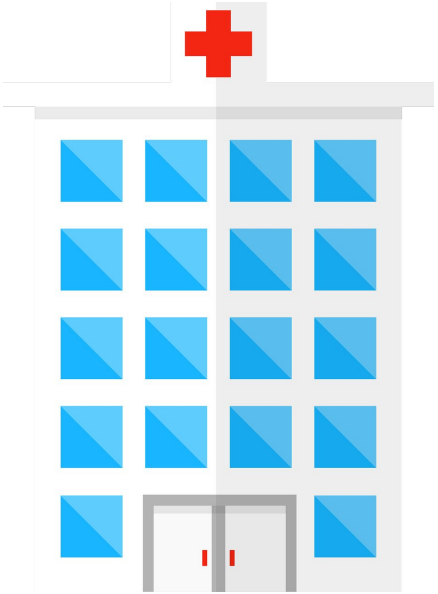
```
SELECT
  name,
  revenue
FROM dataset
ORDER BY revenue;
```

Ask Good Questions

Know Your Data

Write Good SQL

Brainstorm Questions for our Nonprofit Dataset



Open this example form:
https://goo.gl/5ZWCko

What are potential insights about nonprofit organizations?

- Revenue? Expenses? Employees?
- Trends over time?

Module 3

Exploring your Data with SQL

In this module we will:

- Compare Common Data Exploration Techniques
- **Learn How to Code High Quality Standard SQL**
- **Explore Google BigQuery Public Datasets**

Google Cloud

Enable standard SQL in your queries

We will be using Standard SQL:

- Add `#standardSQL` as a comment in first line of your query
- Show Options → Disable Legacy SQL
- Keep this Reference Handy: https://cloud.google.com/bigquery/docs/reference/standard-sql/

Use Run Selected to only run a portion of your query

Highlight and then Ctrl or Command (mac) + E

New Query

```
1  # standardSQL
2  SELECT
3    language,
4    wikimedia_project,
5    title,
6    SUM(views) AS views
7  FROM
8    `bigquery-samples.wikipedia_benchmark.Wiki10B`
9  WHERE
10   title LIKE '%Google%'
11 GROUP BY
12   wikimedia_project,
13   title,
14   language;
15
16 SELECT 'just run this line - ctrl or command (mac) + E';
```

Standard SQL Dialect

RUN QUERY | Save Query | Save View | Format Query | Show Options

Run Selected

Use backticks ` around table names in standard SQL

```
#standardSQL
SELECT column
FROM `project.dataset.table`
```

What is wrong with the below query?

```
#standardSQL
SELECT totrevenue
FROM `irs_990_2015`
```

Read BigQuery error messages for helpful tips

```
#standardSQL
SELECT totrevenue
FROM `irs_990_2015`
```

Error: Table name "irs_990_2015" cannot be resolved: dataset name is missing.

What are a few things wrong with this query?

```
#standardSQL
SELECT totrevenue
FROM
`bigquery-public-data.irs_990.irs_990_2015`
```

How many Total Revenue records does this return for 2015 filings?

Add a LIMIT clause to limit results

```
#standardSQL
SELECT totrevenue
FROM `bigquery-public-data.
irs_990.irs_990_2015`
LIMIT 10
```

... added a LIMIT clause....

Results | Explanation | Job Information

Row	totrevenue
1	5707740
2	10161249
3	14678254
4	1944403
5	7062794
6	285763404
7	1490277
8	2176491
9	422950
10	161055

Table | JSON

Add an ORDER BY clause to sort results

```
#standardSQL
SELECT totrevenue
FROM `bigquery-public-data.
irs_990.irs_990_2015`
ORDER BY totrevenue DESC
LIMIT 10
```

LIMIT is the last clause in your query. order of the clauses matters greatly in SQL.

Results | Explanation | Job Information

Row	totrevenue
1	45409123226
2	20796549014
3	11091388129
4	10098163008
5	9890722789
6	9475129863
7	9021585970
8	8655129029
9	7523260077
10	6740015230

Table | JSON

Add a FORMAT function to format results

```
#standardSQL
SELECT FORMAT("%'d",totrevenue)
FROM
`bigquery-public-data.irs_990.irs_990_2015`
ORDER BY totrevenue DESC
LIMIT 10
```

... format the presentation of the column result ...

SQL Functions Perform Actions on Inputs

```
FORMAT("%'d",totrevenue)
```

Function = Performs an **Action**

Parameters = **Inputs** you provide

```
SELECT FORMAT("%'d",1000)
#Returns 1,000
```

Result: Add a FORMAT Function to **Format Results**

Results | Explanation | Job Information

Row	f0_	
1	45,409,123,226	
2	20,796,549,014	
3	11,091,388,129	
4	10,098,163,008	
5	9,890,722,789	
6	9,475,129,863	
7	9,021,585,970	
8	8,655,129,029	
9	7,523,260,077	
10	6,740,015,230	

Table | JSON

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

Add a FORMAT function to format results

Results | Explanation | Job Information

Row	f0_
1	45,409,123,226
2	20,796,549,014
3	11,091,388,129
4	10,098,163,008
5	9,890,722,789
6	9,475,129,863
7	9,021,585,970
8	8,655,129,029
9	7,523,260,077
10	6,740,015,230

Table | JSON

wait... what happened to our column name?

Create aliases using the AS keyword

Results | Explanation | Job Information

Row	revenue
1	45,409,123,226
2	20,796,549,014
3	11,091,388,129
4	10,098,163,008
5	9,890,722,789
6	9,475,129,863
7	9,021,585,970
8	8,655,129,029
9	7,523,260,077
10	6,740,015,230

Table | JSON

```
SELECT totrevenue AS revenue
```

Pitfall: Unintentionally affecting the data type

Results | Explanation

Row	revenue
1	45,409,123,226
2	20,796,549,014
3	11,091,388,129
4	10,098,163,008
5	9,890,722,789
6	9,475,129,863
7	9,021,585,970
8	8,655,129,029
9	7,523,260,077
10	6,740,015,230

Table | JSON

Beware of stylistic formatting in SQL.

Your output is now treated like a string. This makes math operations on this calculated field more difficult.

It's best to save stylistic elements for your visualization tool

Aliases do not exist yet when filtering in WHERE

```
#standardSQL
SELECT
   (totrevenue - totfuncexpns) AS income
FROM `bigquery-public-data.irs_990.irs_990_2015`
WHERE income > 0  ← Does not exist yet, will error in WHERE clause
ORDER BY income DESC ← Does exist, allowed in ORDER BY, GROUP BY, HAVING
LIMIT 10
```

… **Error: income undefined** …

Add new fields in SELECT clause to return more data

```
#standardSQL
SELECT
   totrevenue AS revenue
   ein,
   operateschools170cd AS is_school,
FROM
`bigquery-public-data.irs_990.irs_990_2015`
ORDER BY totrevenue DESC
LIMIT 10
```

Can you spot the errors?
Hint: It's one character

Add new fields in SELECT clause to return more data

```
#standardSQL
SELECT
   totrevenue AS revenue,
   ein,
   operateschools170cd AS is_school
FROM
`bigquery-public-data.irs_990.irs_990_2015`
ORDER BY totrevenue DESC
LIMIT 10
```

Correctly shown with commas

Add new fields in SELECT clause to return more data

Results | Explanation | Job Information

Row	revenue	ein	is_school
1	45409123226	941340523	N
2	20796549014	941105628	N
3	11091388129	900656139	N
4	10098163008	208295721	N
5	9890722789	900424876	N
6	9475129863	390123480	N
7	9021585970	390123480	N
8	8655129029	941196203	N
9	7523260077	912153073	N
10	6740015230	42103580	Y

Table | JSON

EIN (Employer Identification Number) is the unique identifier for that charity

Is_school is a flag field indicating whether that charity is a school

Pitfall: Using **SELECT *** to Return All Columns

Avoid using * (star) to return all columns.

Selecting only the columns you need greatly increases query speed and helps with readability.

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

Filter rows using the WHERE clause

```
#standardSQL
SELECT
   totrevenue AS revenue,
   ein,
   operateschools170cd AS is_school
FROM
`bigquery-public-data.irs_990.irs_990_2015`
WHERE operateschools170cd = 'Y'
ORDER BY totrevenue DESC
LIMIT 10
```

Why didn't we write this as is_school = 'Y'?

Filter rows using the WHERE clause

Results | Explanation | Job Information

Row	revenue	ein	is_school
1	6740015230	42103580	Y
2	6000839000	231352685	Y
3	5717023246	941156365	Y
4	5569004000	520595110	Y
5	5133788413	135562308	Y
6	4623485966	951642394	Y
7	4560196033	416011702	Y
8	4477633568	60646973	Y
9	4471027733	135598093	Y
10	4368738915	150532082	Y

Table | JSON

only return school filings

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

Perform calculations over values with aggregation

```
#standardSQL
SELECT
   SUM(totrevenue) AS total_2015_revenue,
   AVG(totrevenue) AS avg_revenue,
   COUNT(ein) AS nonprofits,
   COUNT(DISTINCT ein) AS nonprofits_distinct,
   MAX(noemplyeesw3cnt) AS num_employees
FROM
`bigquery-public-data.irs_990.irs_990_2015`
```

Perform calculations over values with aggregation

Results | Explanation | Job Information

Row	total_2015_revenue	avg_revenue	nonprofits	nonprofits_distinct	num_employees
1	2344355088288	7952843.417467928	294782	275077	787050

Table | JSON

More SQL Aggregation Functions

Embed functions inside of other functions

```
#standardSQL
SELECT
   SUM(totrevenue) AS total_2015_revenue,
   ROUND(AVG(totrevenue),2) AS avg_revenue,
   COUNT(ein) AS nonprofits,
   COUNT(DISTINCT ein) AS nonprofits_distinct,
   MAX(noemplyeesw3cnt) AS num_employees
FROM
`bigquery-public-data.irs_990.irs_990_2015`
```

Embed functions inside of other functions

Results | Explanation | Job Information

Row	total_2015_revenue	avg_revenue	nonprofits	nonprofits_distinct	num_employees
1	2344355088288	7952843.42	294782	275077	787050

Table | JSON

Investigate uniqueness with COUNT(DISTINCT field)

Results | Explanation | Job Information

Row	total_2015_revenue	avg_revenue	nonprofits	nonprofits_distinct	num_employees
1	2344355088288	7952843.42	294782	275077	787050

Table | JSON

Create aggregation groupings with GROUP BY

```
#standardSQL
SELECT
  ein, # not aggregated
  COUNT(ein) AS ein_count # aggregated
FROM
`bigquery-public-data.irs_990.irs_990_2015`
GROUP BY ein
ORDER BY ein_count DESC
```

Pitfall: Forgetting to **Group Non-Aggregated Fields**

Do not forget to **use a GROUP BY** if you are using a mix of aggregated and non-aggregated fields

```
SELECT
   company,
   SUM(revenue) AS total
FROM table
GROUP BY company
```

Locate duplicate records with COUNT + GROUP BY

Results | Explanation | Job Information

Row	ein	ein_count
1	680612926	7
2	830345294	7
3	256412267	7
4	364710020	7
5	362235151	7
6	611749317	7
7	20430344	7
8	582535749	7
9	262946183	7
10	262152334	7
11	20786314	7
12	161387890	7
13	208367574	7

Table | JSON

There are many charities that have more than one record for the tax filing year 2015. This is highly unusual.

Next let's count how often this happens in total

Filter aggregations with HAVING clause

```
#standardSQL
SELECT
  ein,
  COUNT(ein) AS ein_count
FROM
`bigquery-public-data.irs_990.irs_990_2015`
GROUP BY ein
HAVING ein_count > 1
ORDER BY ein_count DESC
```

Notice that we can use the alias here for filtering

17,994 EINs with more than one filing in 2015

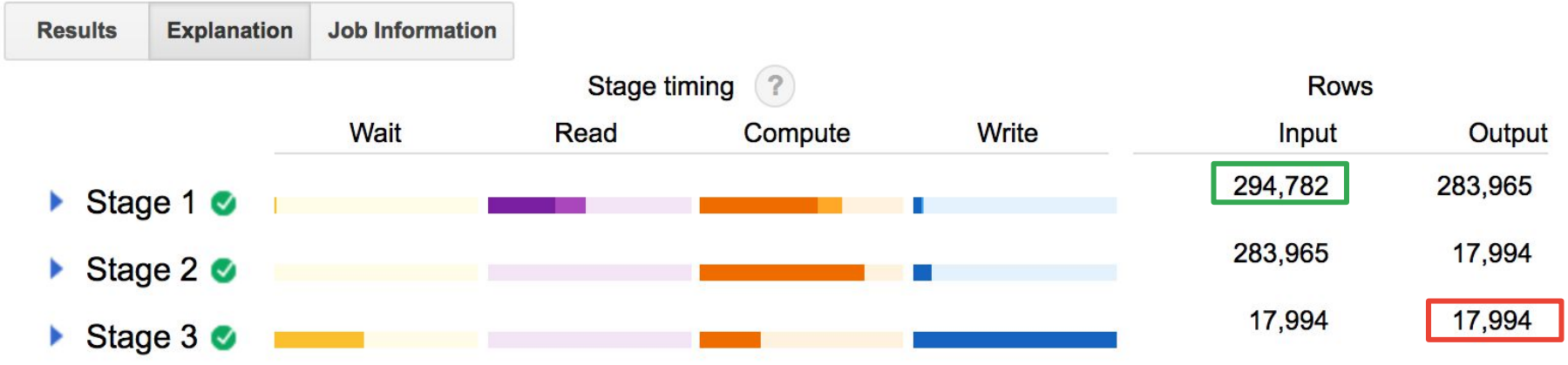
Results | Explanation | Job Information

Row	ein	ein_count
1	262946183	7
2	352123470	7
3	363645106	7
4	341565721	7
5	651192755	7
6	611749317	7
7	208367574	7
8	261511165	7
9	364710020	7
10	20430344	7
11	591800795	7
12	256412267	7
13	582535749	7

Table | JSON

First < Prev Rows 1 - 13 of 17994

Track input and output counts with Explanation tab



Explore further by filtering on one nonprofit

Results | Explanation | Job Information

Row	ein	ein_count
1	262946183	7
2	352123470	7
3	363645106	7
4	341565721	7
5	651192755	7
6	611749317	7
7	208367574	7
8	261511165	7
9	364710020	7
10	20430344	7
11	591800795	7
12	256412267	7
13	582535749	7

Table | JSON

First < Prev Rows 1 - 13 of 17994

Explore further by filtering on one nonprofit

```
#standardSQL
SELECT *
FROM
`bigquery-public-data.irs_990.irs_990_2015`
WHERE ein = 262152334
```

Data quality Insight: Multiple tax periods filed in 2015

Results | Explanation | Job Information

Row	elf	ein	tax_pd	subseccd	s501c3or4947a1cd
1	P	262152334	200912	3	N
2	P	262152334	201112	3	N
3	P	262152334	201312	3	N
4	P	262152334	200812	3	N
5	P	262152334	201412	3	N
6	P	262152334	201212	3	N
7	P	262152334	201012	3	N

Table | JSON

Question:
What could we do if we only wanted the latest 2015 filing (for tax period 2014)?

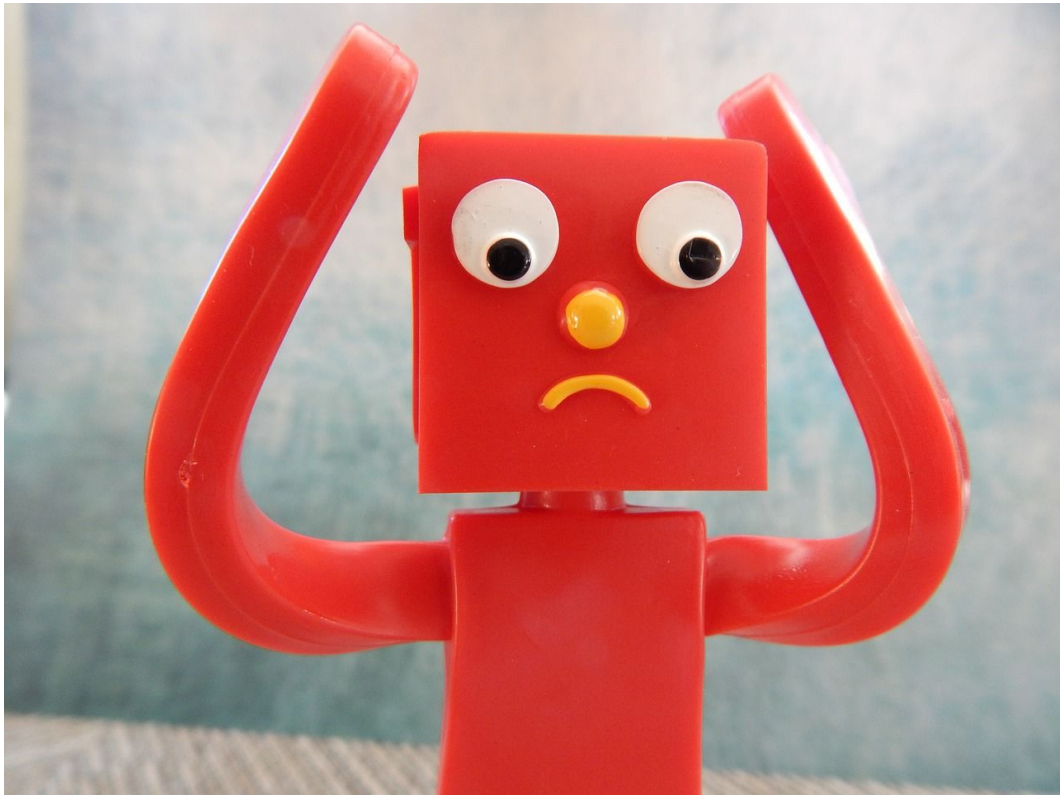
There are multiple right answers.

Example Solution: parse, convert, filter for 2014

```
#standardSQL
SELECT
  ein,
  tax_pd,
  PARSE_DATE('%Y%m', CAST(tax_pd AS STRING)) AS tax_period
FROM `bigquery-public-data.irs_990.irs_990_2015`
WHERE
  EXTRACT(YEAR FROM
     PARSE_DATE('%Y%m', CAST(tax_pd AS STRING))
    ) = 2014
LIMIT 10
```

Tax filings are for the year prior
(2015 means for the period 2014)

Parsing and converting dates can be hard



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

Comparing BigQuery data types

Numeric Data

Integer (int64)
Whole numbers that can be negative (-2,-1,0,1,2,3,4)

Float (float64)
(1.0000000001)

String Data

Strings
Text values ('dog', 'cat', '800-999-9999')

In SQL, use single quotes when dealing with strings like 'Google Inc.'

Dates

Dates (datetime)
Stored in universal time format. Allowable Range: 0001-01-01 00:00:00 to 9999-12-31 23:59:59.999999

Other

Boolean (Y/N)

Array [‘apple’, ‘pear’]

Struct<apple string>

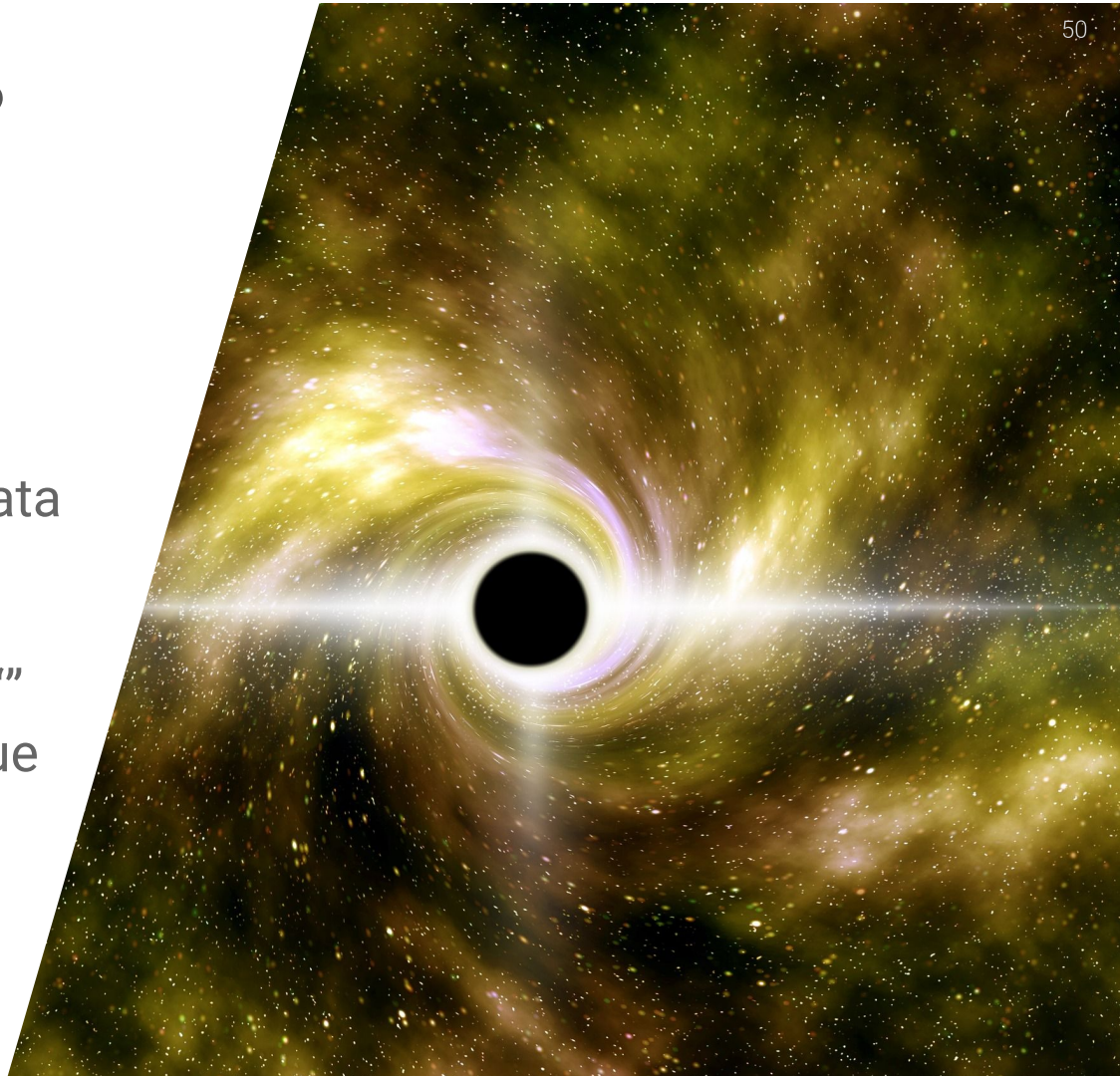
BigQuery Data Types Reference

Using CAST to convert between data types

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

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



Handle NULL values with extreme care

```
#standardSQL
SELECT
  ein,
  street,
  city,
  state,
  zip
FROM
  `bigquery-public-data.irs_990.irs_990_ein`
WHERE
  state IS NULL
LIMIT 10;
```

NULL values cannot be equated to and thus have a special IS NULL or IS NOT NULL test

Valid NULLs for countries not using state field

Results | Explanation | Job Information

Row	ein	street	city	state	zip
1	980121082	APO AE	ITALY	null	00000-0000
2	371658596	33081 AVIANO AB	AB PN	null	00000-0000
3	371658596	33081 AVIANO AB	AB PN	null	00000-0000
4	660563303	KINGSHILL 00851	BRITISH VIRGIN ISLANDS	null	00000-0000
5	200612879	LONDON VICTORIA	UNITED KINGDOM	null	00000-0000
6	453028683	BRIGANTINE 08203	ALGERIA	null	00000-0000
7	300254488	BONNE TERRE MO 63628	ALBANIA	null	00000-0000
8	980683729	TORONTO ONTARIO M3C 3P8	CANADA	null	00000-0000
9	981177602	WHITEHORSE YUKON Y1A 6R4	CANADA	null	00000-0000
10	200318819	HEGELALLEE 8 POTSDAM D-13367	GERMANY	null	00000-0000

Table | JSON

Not all countries will use the state field

YYYY-MM-DD is the expected format for dates

Date Function	Result
CURRENT_DATE([time_zone])	Today's date
EXTRACT(year FROM your_date_field)	Year of date (try changing Year to Month, Day, Week etc.)
DATE_ADD(DATE "2008-12-25", INTERVAL 5 DAY)	Adds an interval of time
SELECT DATE_SUB(DATE "2008-12-25", INTERVAL 5 DAY)	Subtracts an interval of time
DATE_DIFF(DATE "2010-07-07", DATE "2008-12-25", DAY)	Subtracts two dates and returns the interval
DATE_TRUNC(DATE '2008-12-25', MONTH)	Truncates the date (e.g. 2008-12-01)
FORMAT_DATETIME( )	Formats an existing date to a different date string format
PARSE_DATETIME( )	Example: Turn "12/01/2017" to a proper date "2017-12-01"

Parsing string values with string functions

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

Searching for nonprofits with ‘help’ in their name

```
#standardSQL
SELECT
  ein,
  name
FROM
  `bigquery-public-data.irs_990.irs_990_ein`
WHERE
  LOWER(name) LIKE '%help%'
LIMIT 10;
```

Searching for nonprofits with ‘help’ in their name

Row	ein	name
1	201515765	PHELPS CHAMBER OF COMMERCE
2	166052064	PHELPS CEMETERY CORPORATION
3	470407568	PHELPS COUNTY FARM BUREAU INC
4	473394369	HILLPOINT HELPING HANDS
5	471000644	SENIOR HELP ALLIANCE
6	471180492	HELPING IMPROVE KIDS ENVIRONMENTS INC
7	161669678	HIS HELPING HANDS MINISTRIES INCORPORATED
8	472894724	LIL-BIT OF HELP
9	471542706	CALVARY CHAPEL HELPS INC
10	464812552	ONE HAND HELPING ANOTHER

Searching for nonprofits starting with ‘help’

```
#standardSQL
SELECT
  ein,
  name
FROM
  `bigquery-public-data.irs_990.irs_990_ein`
WHERE
   LOWER(name) LIKE 'help%'
LIMIT 10;
```

Note the absence of the beginning wildcard %

Searching for nonprofits starting with ‘help’

Results | Explanation | Job Information

Row	ein	name
1	471180492	HELPING IMPROVE KIDS ENVIRONMENTS INC
2	260903752	HELPING HANDS OUTREACH MINISTRIESINC
3	770640371	HELPER MINISTRIES INTERNATIONAL INC
4	731689263	HELP THE HARVEST - HAWAII
5	810867903	HELPING HEROES THRIVE FOUNDATION
6	474894747	HELPING YOU GROW SUCCESSFULLY
7	800743503	HELPING HANDS OUTREACH
8	811838250	HELP FOR ME YOUTH AND ADULT SERVICES
9	811638595	HELPING HANDS OF CUMBERLAND COUNTY
10	454896030	HELPING HANDS FOUNDATION OF SURRY COUNTY INC

Table | JSON

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

Lab 2
Troubleshoot Common SQL Errors with BigQuery

Troubleshoot Common SQL Errors with BigQuery

In this lab you will be faced with several broken SQL queries that are returning common errors or incorrect results in Google BigQuery.

You will practice your SQL debugging skills to fix these queries.

Identifying and fixing bugs and documenting your code are fundamental best practices

Google Cloud