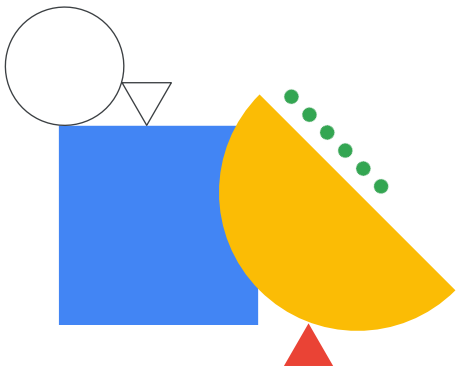
Google Cloud

Enriching your Data Warehouse with JOINs



Agenda

01 Merge Historical Data Tables with UNION

02 Introduce Table Wildcards for Easy Merges

03 Review Data Schemas: Linking Data Across Multiple Tables

Demo: Joining weather datasets

04 JOIN Examples and Pitfalls

Lab: Troubleshooting and Solving Data Join Pitfalls

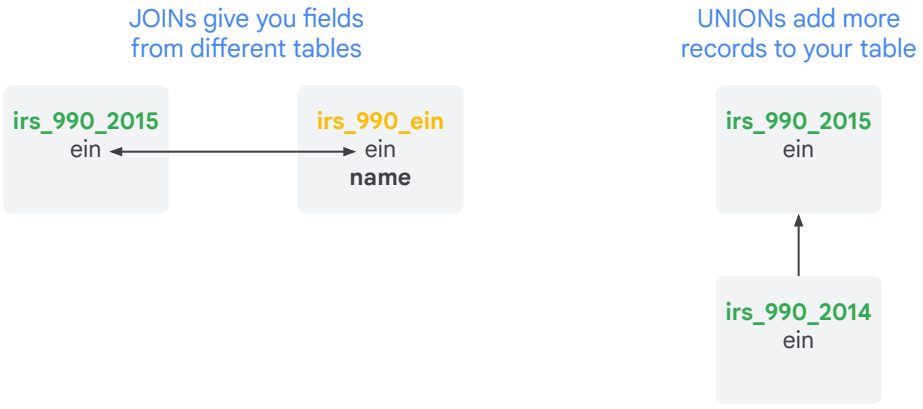
One of the most popular topics in SQL is how mash-up multiple data sources together in a single query to answer more complex insights.

In this module we will tackle how to append additional historical data vertically through unions as well as how to join together different datasets horizontally through SQL joins.

Let's walkthrough the basics and I'll highlight some common pitfalls along the way.

Merge Historical Data Tables with UNION

Enriching your dataset through JOINs and UNIONs

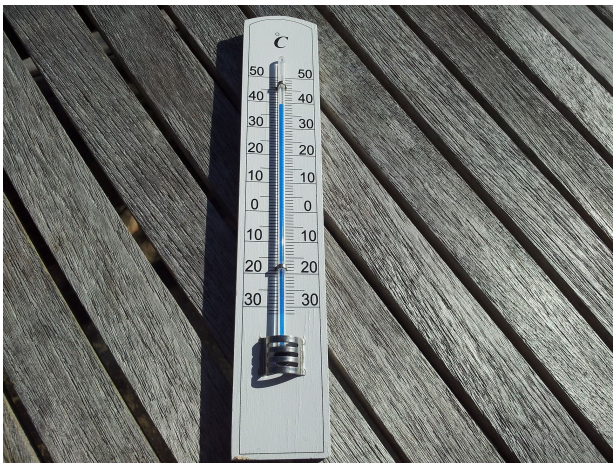


JOINs enrich your dataset by potentially adding fields (horizontally)

UNIONs append more data to your table (vertically)

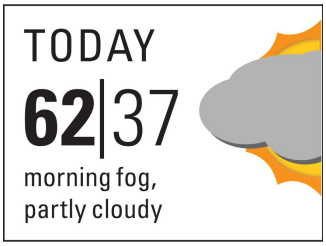
Walkthrough example

Joining and merging *temperature* and *weather station* data



Two types of tables in the **NOAA weather dataset**

Daily temperature readings



Weather recording station locations

Victoria, Australia

Wake Island Harbor

There are two table types: Daily temperature readings and the physical station locations which recorded them.

Two types of tables in the NOAA weather dataset

Daily temperature readings

... current

Weather station location details

Results | Explanation | Job Information

Row	usaf	wban	name	country	state	call	lat	lon	elev	begin	end
1	912450	41606	WAKE ISLAND AIRFLD	WQ	PC	PWAK	19.283	166.65	+0003.7	19451231	20100731
2	912460	41606	WAKE ISLAND AIRFIELD	WQ	UM	PWAK	19.283	166.65	+0007.0	20100801	20170805
3	999999	41606	WAKE ISLAND	WQ	PC	PWAK	19.283	166.65	+0003.7	19491031	19721231
4	912450	99999	WAKE ISLAND AIRFLD	WQ			19.283	166.65	+0004.0	20000101	20100818
5	997387	99999	WAKE ISLAND	WQ			19.28	166.62	+0005.0	20050517	20170804

Table | JSON

We have a separate **table for all daily weather temperatures since 1929**. That's a lot of tables for us to query and combine (don't worry, it won't be so bad)

Our **weather station location details** (lat, long, state, station name) is stored in a single lookup table. Key fields like Country and State are not present in the Daily Temperature table (because of a concept called normalization that we will come to later) but we can look these fields up by joining the tables together.

But, before we can link and join the two tables together, we need to first figure out what linking field they have in common.

What is our unique identifier for weather stations? Is it USAF (US Air Force Station ID) or WBAN (WEATHER BUREAU ARMY NAVY)? Well, let's investigate ….

What is our **unique identifier** for a weather station?

```
#standardSQL
# Is usaf unique over time?
  SELECT
    COUNT(usaf) AS total_count,
    COUNT(DISTINCT usaf) AS distinct_count
FROM
`bigquery-public-data.noaa_gsod.stations`;
```

Row	total_count	distinct_count
1	30016	✗ 26453

No

Results | Explanation | Job Information

Row	usaf	wban	name	country	state	call	lat	lon	elev	begin	end
1	912450	41606	WAKE ISLAND AIRFLD	WQ	PC	PWAK	19.283	166.65	+0003.7	19451231	20100731
2	912460	41606	WAKE ISLAND AIRFIELD	WQ	UM	PWAK	19.283	166.65	+0007.0	20100801	20170805
3	999999	41606	WAKE ISLAND	WQ	PC	PWAK	19.283	166.65	+0003.7	19491031	19721231
4	912450	99999	WAKE ISLAND AIRFLD	WQ			19.283	166.65	+0004.0	20000101	20100818
5	997387	99999	WAKE ISLAND	WQ			19.28	166.62	+0005.0	20050517	20170804

Table | JSON

Before we can link the two tables together, we need to find our unique row identifier...

What is our unique identifier for weather stations? Is it USAF (U.S. Air Force) number?

No, as we see from the above query, **USAF is not unique**. One station could possibly have re-used this ID over time or one station could have multiple recording devices.

Find the duplicate usaf records use this example query:

```
SELECT *
FROM (
  SELECT
     *,
     ROW_NUMBER()
        OVER (PARTITION BY usaf)
        AS station_history_change
  FROM `bigquery-public-data.noaa_gsod.stations`
)
WHERE station_history_change > 1
ORDER BY usaf, station_history_change
```

We need to use a **combination key**

```
#standardSQL
# Is usaf wban combo unique over time?
  SELECT
    COUNT(CONCAT(usaf,wban)) AS total_stations,
    COUNT(DISTINCT CONCAT(usaf,wban)) AS distinct_stations
FROM
`bigquery-public-data.noaa_gsod.stations`;
```

Row	total_stations	distinct_stations
1	30016	30016

Yes

Results | Explanation | Job Information

Row	usaf	wban	name	country	state	call	lat	lon	elev	begin	end
1	912450	41606	WAKE ISLAND AIRFLD	WQ	PC	PWAK	19.283	166.65	+0003.7	19451231	20100731
2	912460	41606	WAKE ISLAND AIRFIELD	WQ	UM	PWAK	19.283	166.65	+0007.0	20100801	20170805
3	999999	41606	WAKE ISLAND	WQ	PC	PWAK	19.283	166.65	+0003.7	19491031	19721231
4	912450	99999	WAKE ISLAND AIRFLD	WQ			19.283	166.65	+0004.0	20000101	20100818
5	997387	99999	WAKE ISLAND	WQ			19.28	166.62	+0005.0	20050517	20170804

Table | JSON

Since it's clear that wban by itself is not unique, what about the combination of the two?

Yes! If we CONCATENATE the two fields we get a **combined unique key** showing 30,016 stations.

Join and Union your data for enriched insights

Daily temperature readings

... current

How are we going to JOIN so many tables?

Can't we combine the temperature readings across years somehow?

Introducing UNION for vertically merging your data

Daily temperature readings

gsod1929

gsod1930

gsod1929

stn	wban	temp	year
030050	99999	49	1929
030050	99999	45.7	1929
030050	99999	48.2	1929

UNION

gsod1930

stn	wban	temp	year
030050	99999	49	1929
030050	99999	45.7	1929
030050	99999	48.2	1929

=

gsod1929 UNION gsod1930

stn	wban	temp	year
030050	99999	49	**1929**
030050	99999	45.7	**1929**
030050	99999	48.2	**1929**
037770	99999	50.7	1930
030910	99999	56	1930
038560	99999	53.2	1930

Union Distinct vs Union All = Union Distinct will deduplicate whereas Union All will include all values

Introducing UNION for vertically merging your data

gsod1929

gsod1930

```
#standardSQL
SELECT
  stn,
  wban,
  temp,
  year
FROM

`bigquery-public-data.noaa_gsod.gsod1929`
  UNION DISTINCT
(SELECT stn, wban, temp, year FROM
`bigquery-public-data.noaa_gsod.gsod1930`)
```

gsod1929 UNION gsod1930

stn	wban	temp	year
030050	99999	49	1929
030050	99999	45.7	1929
030050	99999	48.2	1929
037770	99999	50.7	1930
030910	99999	56	1930
038560	99999	53.2	1930

UNION DISTINCT removes duplicates whereas UNION ALL keeps every record

Union Distinct vs Union All = Union Distinct will deduplicate whereas Union All will include all values

Wait a minute....

- **gsod1929**
- gsod1930
- gsod1931
- gsod1932
- gsod1933
- gsod1934
- gsod1935
- gsod1936
- gsod1937
- gsod1938
- gsod1939

```
#standardSQL
SELECT
  stn,
  wban,
  temp,
  year
FROM
  `bigquery-public-data.noaa_gsod.gsod1929`
  UNION DISTINCT
(SELECT stn,wban,temp,year FROM
`bigquery-public-data.noaa_gsod.gsod1930`)
  UNION DISTINCT
(SELECT stn,wban,temp,year FROM
`bigquery-public-data.noaa_gsod.gsod1931`)
  UNION DISTINCT
(SELECT stn,wban,temp,year FROM
`bigquery-public-data.noaa_gsod.gsod1932`)
# This is getting out of hand...
```

... I don't want to type 100 Unions

Typing all those UNIONs by hand seems tedious...

Introduce Table Wildcards for Easy Merges

Make your UNIONs easier with the **table wildcard ***

```
#standardSQL
SELECT
  stn,
  wban,
  temp,
  year
FROM

`bigquery-public-data.noaa_gsod.gsod1929`
  UNION DISTINCT
`bigquery-public-data.noaa_gsod.gsod1930`
  UNION DISTINCT
`bigquery-public-data.noaa_gsod.gsod1931`
  UNION DISTINCT
`bigquery-public-data.noaa_gsod.gsod1932`
# This is getting out of hand...
```

```
#standardSQL
SELECT
  stn,
  wban,
  temp,
  year
FROM

`bigquery-public-data.noaa_gsod.gsod*`
# All gsod tables
```


Use a UNION table wildcard
https://cloud.google.com/bigquery/docs/wildcard-tables

Filtering with a table wildcard * and _TABLE_SUFFIX_

Use _TABLE_SUFFIX to filter out tables included

Be as granular as you can

- e.g. .gsod2* instead of .gsod* if you only care about the year 2000 onward

```
#standardSQL
SELECT
  stn,
  wban,
  temp,
  year
FROM

`bigquery-public-data.noaa_gsod.gsod*`

# All gsod tables after 1950
WHERE _TABLE_SUFFIX > '1950'
```


Include a collection of tables and then filter them with _TABLE_SUFFIX_
https://cloud.google.com/bigquery/docs/wildcard-tables

Filtering with a **table wildcard** * and **_TABLE_SUFFIX_**

- Use table wildcard * vs writing many UNIONs
- Use _TABLE_SUFFIX to filter out tables wildcard included
- Use _TABLE_SUFFIX in your SELECT statements with CONCAT( )

Avoid **union pitfalls** like brittle schemas

- Duplicate records among tables (Use UNION DISTINCT vs UNION ALL)
- Changing schemas and field names over time
- Mismatched count of columns in your UNION

Unions in SQL require careful handling of schemas between tables

Review of what we've done so far

FROM `bigquery-public-data.noaa_gsod.gsod*`

stn	wban	temp	year
030050	99999	49	1929
030050	99999	45.7	1929
030050	99999	48.2	1929
...	...	...	...
037770	99999	50.7	2017
030910	99999	56	2017
038560	99999	53.2	2017

We are merging all historical gsod tables into one UNION'd table through a **table wildcard.**

How do we **enrich** our temperature data with station details?

FROM `bigquery-public-data.noaa_gsod.gsod*`

stn	wban	temp	year	name	state	country
030050	99999	49	1929			
030050	99999	45.7	1929			
030050	99999	48.2	1929		??	
...	...	...	...			
037770	99999	50.7	2017			
030910	99999	56	2017			
038560	99999	53.2	2017			

... by **JOIN**ing with data in other tables

Review Data Schemas: Linking Data Across Multiple Tables

What is a JOIN?

Combine **data from separate tables** that share a common element **into one table**

```
#standardSQL
SELECT
  a.stn,
  a.wban,
  a.temp,
  a.year,
  b.name,
  b.state,
  b.country
FROM
  `bigquery-public-data.noaa_gsod.gsod*` AS a
JOIN
  `bigquery-public-data.noaa_gsod.stations` AS b
ON
  a.stn=b.usaf
  AND a.wban=b.wban

WHERE
  # Filter data
  state IS NOT NULL
  AND country='US'
  AND _TABLE_SUFFIX > '2015'
```

What is a JOIN?

Fields from Temperature Tables

Fields from Station Details table

Join Type

Join Condition

```
#standardSQL
SELECT
  a.stn,
  a.wban,
  a.temp,
  a.year,
  b.name,
  b.state,
  b.country
FROM
  `bigquery-public-data.noaa_gsod.gsod*` AS a
JOIN
  `bigquery-public-data.noaa_gsod.stations` AS b
ON
  a.stn=b.usaf
  AND a.wban=b.wban

WHERE
  # Filter data
  state IS NOT NULL
  AND country='US'
  AND _TABLE_SUFFIX > '2015'
```


Aliases are optional in the SELECT statement if the field names are unambiguous between the tables

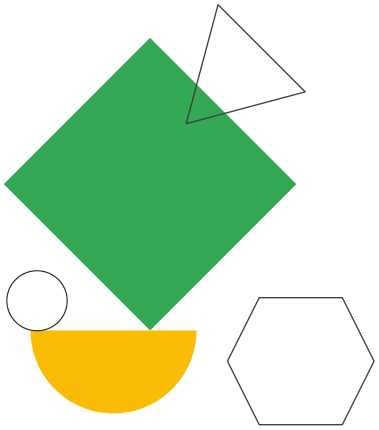
JOINS can have multiple linking fields to establish uniqueness like the one shown here

The default JOIN is an INNER join which means the records must exist in both tables for results to be shown. Let's cover the basic join types now.

Demo

Joining weather datasets

Get insights from multiple tables



Refer to
https://github.com/GoogleCloudPlatform/training-data-analyst/tree/master/courses/data-to-insights/demos/joining-weather-stations.sql

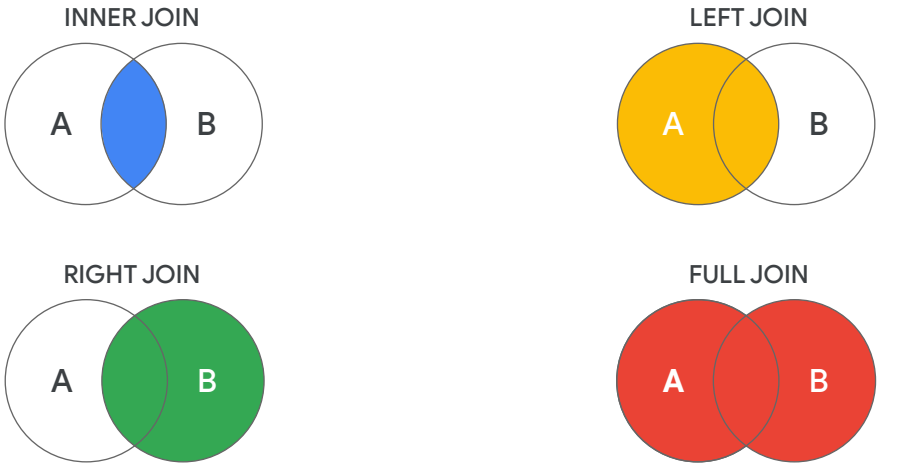
JOIN Examples and Pitfalls

Different types of Joins

INNER JOIN	Returns rows from multiple tables where join condition is met
LEFT JOIN	Returns all rows from the left table and matched rows from the right table
RIGHT JOIN	Returns all rows from the right table and matched rows from the left table
OUTER JOIN	Returns all rows from all tables and unmatched rows are displayed as NULL

BigQuery Join types:
https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#join-types

Joins represented via Venn diagram



Also there is a CROSS JOIN which applies the cross product of all records from each table.

Pitfall: Joining on non-unique fields explodes your dataset

- Doing a many-to-many JOIN could result in more rows than either of your initial tables
- This is a primary reason for exceeding your resource cap in BigQuery (unintentionally high compute)
- Know your dataset and the relationships between your tables before joining

Pitfall: Joining on non-unique fields explodes your dataset

New Query

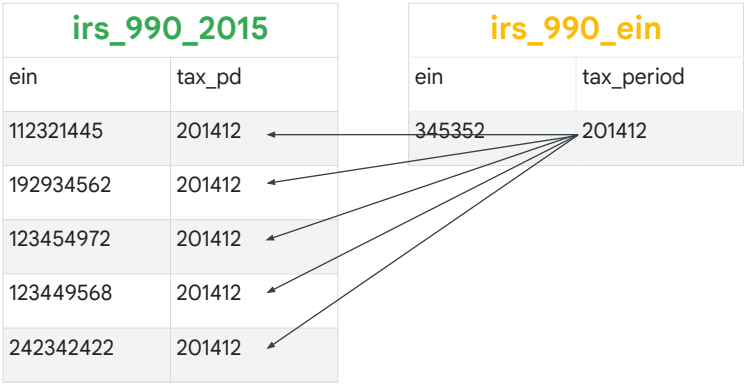
```
1  SELECT filing.ein, tax_pd
2  FROM `bigquery-public-data.irs_990.irs_990_2015` filing
3  LEFT JOIN `bigquery-public-data.irs_990.irs_990_ein` org
4  ON filing.tax_pd = org.tax_period
5
6
```

Standard SQL Dialect

Cancel Query | Save Query | Save View | Format Query | Show Options | Query running (1,033.6s)...

Woah, what happened here?

Pitfall: Joining on non-unique fields explodes your dataset



irs_990_2015

ein	tax_pd
112321445	201412
192934562	201412
123454972	201412
123449568	201412
242342422	201412

irs_990_ein

ein	tax_period
345352	201412

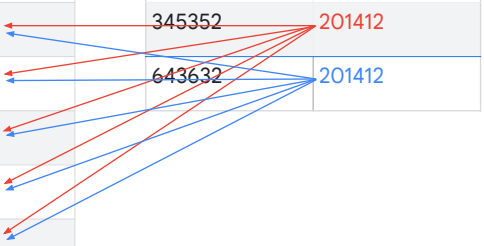
Pitfall: Creating an Unintentional Cross Join

irs_990_2015

ein	tax_pd
112321445	201412
192934562	201412
123454972	201412
123449568	201412
242342422	201412

irs_990_ein

ein	tax_period
345352	201412
643632	201412



Pitfall: Cross Joins multiply your data

irs_990_2015

ein	tax_pd
112321445	201412
192934562	201412
123454972	201412
123449568	201412
242342422	201412

irs_990_ein

ein	tax_period
345352	201412
643632	201412

ein	tax_pd
112321445	**201412**
112321445	**201412**
192934562	201412
192934562	201412
123454972	**201412**
123454972	**201412**
123449568	201412
123449568	201412
242342422	**201412**
242342422	**201412**

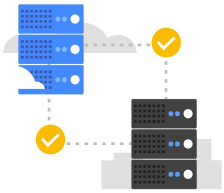
BigQuery CROSS JOIN
https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#cross-join

Pitfall: Understand your data model and relationships

- Understand your data relationship before joining 1:1, N:1, 1:N, N:N
- Use CONCAT( ) to create composite key fields if no unique fields exist or join on more than one field
- Ensure your key fields are distinct (deduplicate)

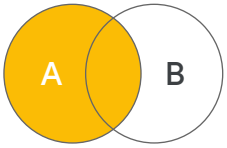
Summary: Mashup your datasets with Joins and Unions

Finding the unique record identifier(s) in table is critical.



Spend time exploring the data relationship model between tables.

Use UNION wildcards and _TABLE_SUFFIX_ to quickly add records to a consolidated table.



Use JOINs to enrich data across multiple tables.

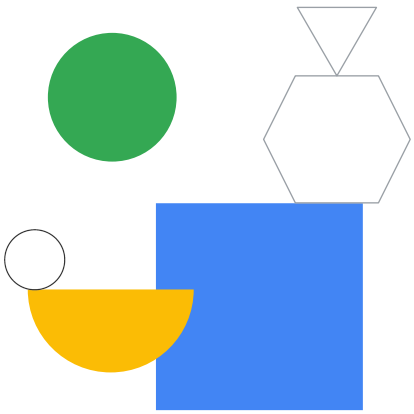
Understanding when and how to use joins and unions in SQL is a concept that is easy to pickup but takes a while to truly master. The best advice I can give you when starting is to really understand how your data tables are supposed to be related to each other (customer to orders, supplier to inventory) and being able to verify if that is actually true though SQL. Remember: all data is dirty and it's your job to investigate and interrogate it before potentially polluting your larger dataset with joins and unions.

Once you understand the relationships between your tables, use unions to append records to a consolidated table and joins to enrich your results with data from multiple sources.

Let's practice these concepts and pitfalls in our next lab.

Lab Intro

Troubleshooting and Solving
Data Join Pitfalls



Lab objectives

01 Use BigQuery to explore a dataset

02 Troubleshoot duplicate rows in a dataset

03 Create joins between data tables

04 Practice when to use each join type