

# BigQuery Data Definition Model



Welcome to the "BigQuery Data Definition Model" module. This module summarizes the key details of BigQuery's resource hierarchy and data definition model, including how to create datasets and tables in BigQuery. Drawing upon your knowledge of Redshift, this module also provides a high-level overview of the similarities and differences between the Redshift and BigQuery resource hierarchies and primary data types to help you start working with data in BigQuery.

Let's jump in!

## INTRODUCTION

---

### BigQuery Data Definition Model

## RESOURCE HIERARCHY IN REDSHIFT

---

### Lesson introduction

### Resource hierarchy in BigQuery and Redshift

### Data definition models in BigQuery and Redshift

## OVERVIEW OF BIGQUERY RESOURCE HIERARCHY

---

**Lesson introduction**

**Organizing BigQuery resources**

**Google Cloud resource hierarchy**

**References**

## **CREATING DATASETS AND TABLES IN BIGQUERY**

---

**Lesson introduction**

**Dataset location**

**Dataset management**

**Table management**

**References**

## **MAPPING DATA TYPES FROM REDSHIFT TO BIGQUERY**

---

**Lesson introduction**

**Key points about data types**

**Map of data types in BigQuery and Redshift**

# BigQuery Data Definition Model

---

---

**Upon completion of this module, you will be able to:**

- 1 Describe BigQuery's resource hierarchy.
- 2 Explain how Redshift's resource hierarchy differs from BigQuery.
- 3 Map data types from Redshift to BigQuery.
- 4 Create datasets and tables in BigQuery.

# Lesson introduction

---

In this first lesson, you will examine the similarities and differences between the Redshift and BigQuery resource hierarchies and data definition models.



---

Let's get started!

# Resource hierarchy in BigQuery and Redshift

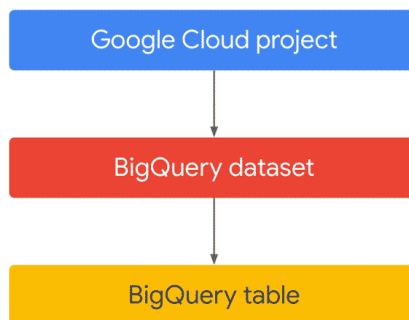
---

Redshift and BigQuery both use a hierarchical model for creating and managing data resources.

However, there are a few key differences in their respective data definition models.

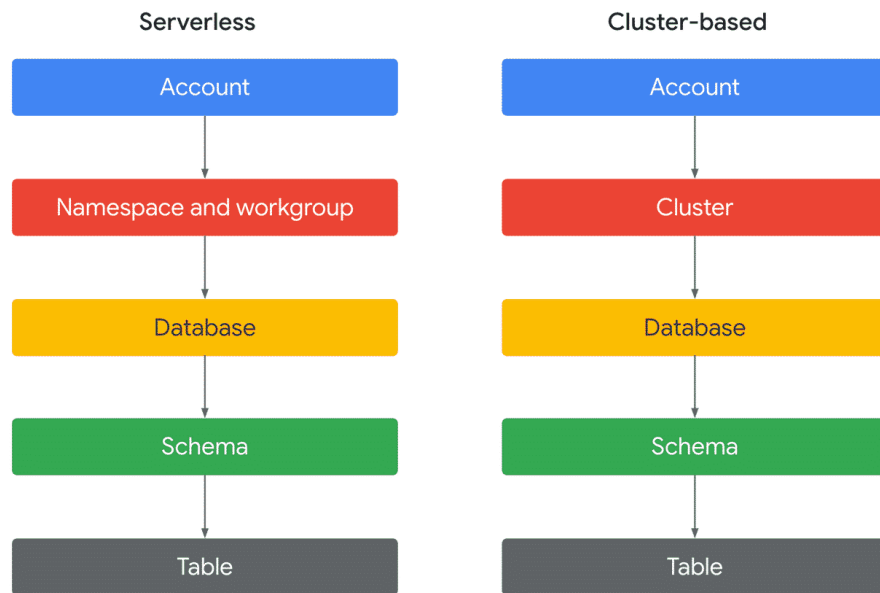
- Redshift has two different resource hierarchy models depending on whether the deployment is serverless or cluster-based; BigQuery is serverless and has only one resource hierarchy model.
- Redshift uses schemas to store tables in a database; BigQuery tables are stored in a dataset created within a Google Cloud project.

## BigQuery Data Resource Hierarchy



## Redshift Data Resource Hierarchy

---



# Data definition models in BigQuery and Redshift

---

Let's explore their data definition models.

Topic	BigQuery	Redshift
<b>Data resource hierarchy</b>	<p>BigQuery inherits the <a href="#">Google Cloud resource hierarchy</a> and adds an additional grouping mechanism called datasets, which are specific to BigQuery.</p> <ol style="list-style-type: none"><li>1. <a href="#">Google Cloud project</a></li><li>2. <a href="#">BigQuery dataset</a></li><li>3. <a href="#">BigQuery table</a></li></ol>	<p>Serverless:</p> <ol style="list-style-type: none"><li>1. Account</li><li>2. <a href="#">Namespace and Workgroup</a></li><li>3. Database</li><li>4. <a href="#">Schema</a></li><li>5. <a href="#">Table</a></li></ol> <p>Cluster-based:</p> <ol style="list-style-type: none"><li>1. Account</li><li>2. <a href="#">Cluster</a></li><li>3. Database</li><li>4. <a href="#">Schema</a></li><li>5. <a href="#">Table</a></li></ol>

Topic	BigQuery	Redshift
Data types	<a href="#">BigQuery data types</a> ( <a href="#">Google Standard SQL</a> ).	<a href="#">Redshift data types</a>

---

Next, let's examine these topics in BigQuery with more detail.

CONTINUE



# Lesson introduction

---

## Introduction

Now that you're familiar with the concept of resource hierarchy models, it's time to examine how BigQuery organizes your data. In this lesson, you will learn about the Google Cloud resource hierarchy, including how policies are inherited, and considerations for working in BigQuery.



---

Let's get started!

# Organizing BigQuery resources

---

**Datasets** are top-level containers that are used to organize and control access to your tables and views in BigQuery. A dataset is contained within a specific **project**. A table or view must belong to a dataset, so you need to create at least one dataset before loading data into BigQuery.

You can use multiple datasets to separate tables pertaining to different analytical domains, and you can use project-level scoping to isolate datasets from each other according to your business needs.

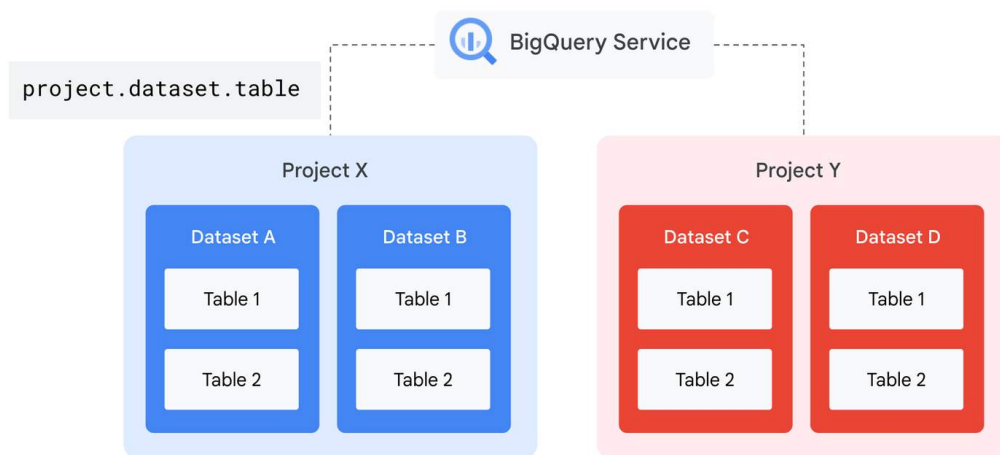
---

## BigQuery dataset billing is associated with projects.

To run a query, you need to be logged in to Google Cloud. When you run a query in your own Google Cloud project on BigQuery data in other projects, **the query charges are billed to your project but the storage costs are not**. For example, if you queried a table that belongs to the `bigquery-public-data` project, the query costs are billed to your project, but the storage costs are billed to the `bigquery-public-data` project which hosts the data.

Access control to run a query is through Identity and Access Management (IAM); different access policies can be set at the Project, Dataset, Table, View, Column or Row levels. To query data in a table or view, you need at least read permissions on the Table or View.

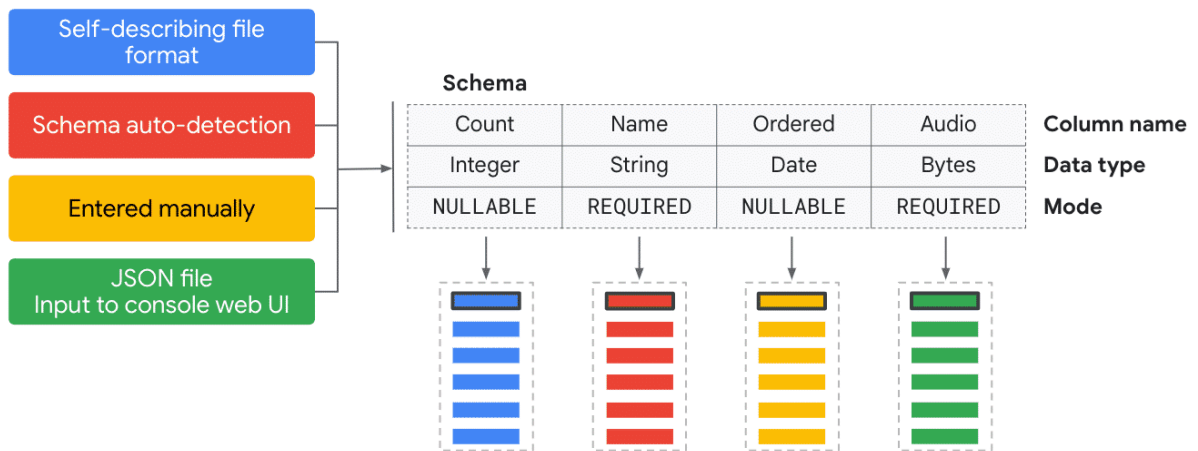
In Google Cloud, users run jobs to query BigQuery data. In order to run a query in a specific project, you need IAM permission to submit a job in that project.



BigQuery tables are stored in datasets created within Google Cloud projects.

---

**In BigQuery, the table schema provides structure to the data such as the names, data types, and specific requirements of the columns.**



You can define the schema manually, by supplying a JSON file, leveraging self-describing file formats, or using auto-detection of the schema when loading new data or querying external data sources.

## Why structure your BigQuery data into projects, datasets, and tables?

These multiple scopes—project, dataset, and table—can help you structure and secure your information logically. You can use multiple datasets to separate tables pertaining to different teams or domains, and you can use project-level scoping to share or isolate datasets according to your business needs.

You can also align projects to billing, use datasets to group and control access to related data, and store data in separate tables based on logical schema considerations.

# Google Cloud resource hierarchy

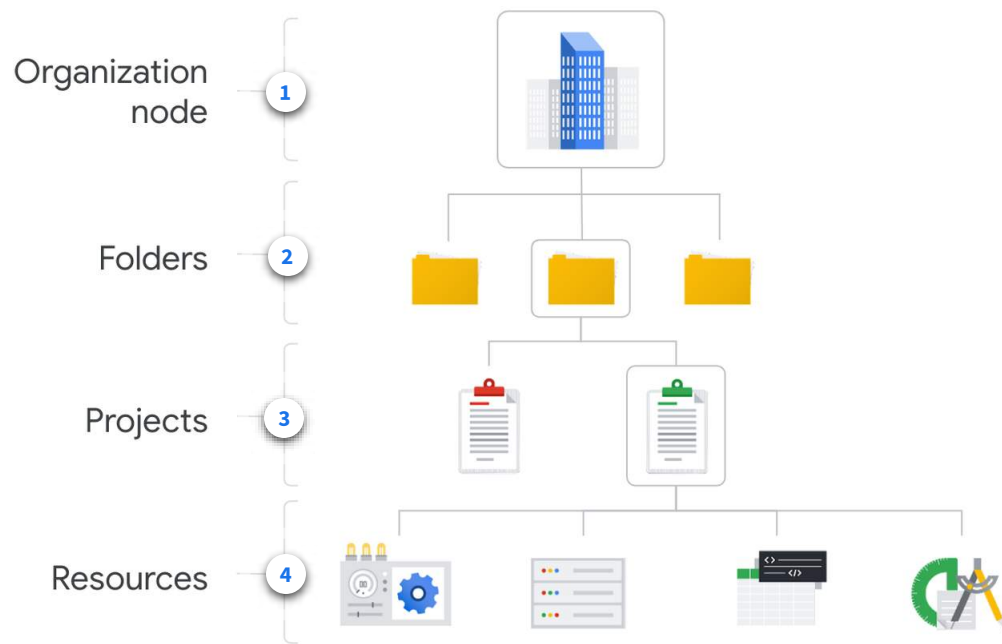
---

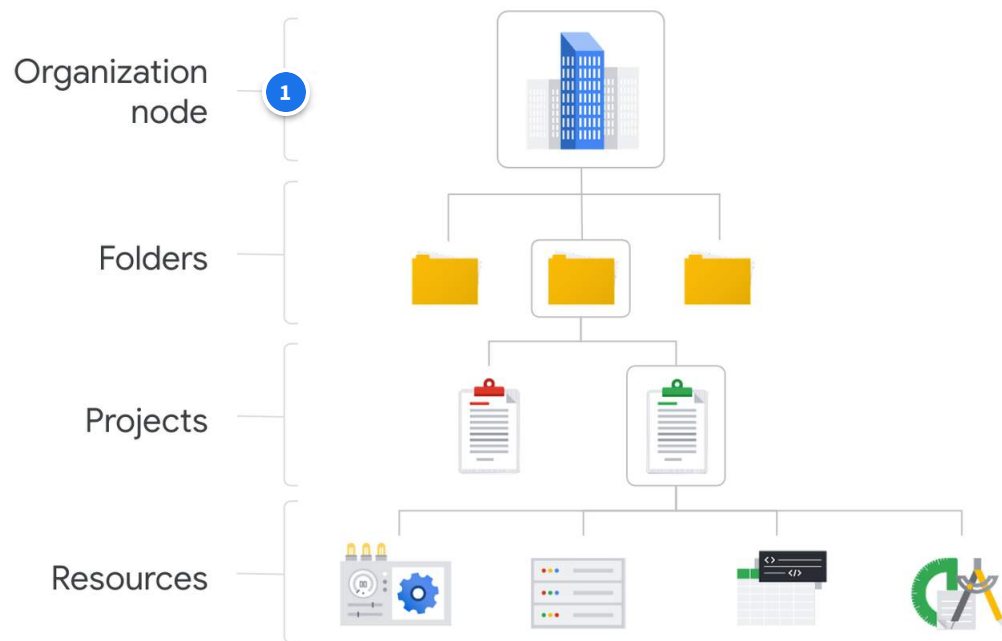
The Google Cloud resource hierarchy allows companies to map their organization's structure onto Google Cloud using an organization resource containing projects that can be further organized into folders. This resource hierarchy provides logical points for setting both organization-level policies and IAM at multiple levels of the hierarchy including folders and projects.

In Google Cloud, all resources (except for the highest resource in a specific hierarchy) have exactly one parent. At the lowest level, service resources are the fundamental components that make up all Google Cloud services such as BigQuery. These lower level resources have project resources as their parents. Both IAM and organization policies are inherited through the hierarchy, and the effective policy for each resource in the hierarchy is the result of policies directly applied on that resource and policies inherited from its ancestors.

Let's explore an example of a Google Cloud resource hierarchy in its complete form.

*Click the numbers in sequence to learn more.*

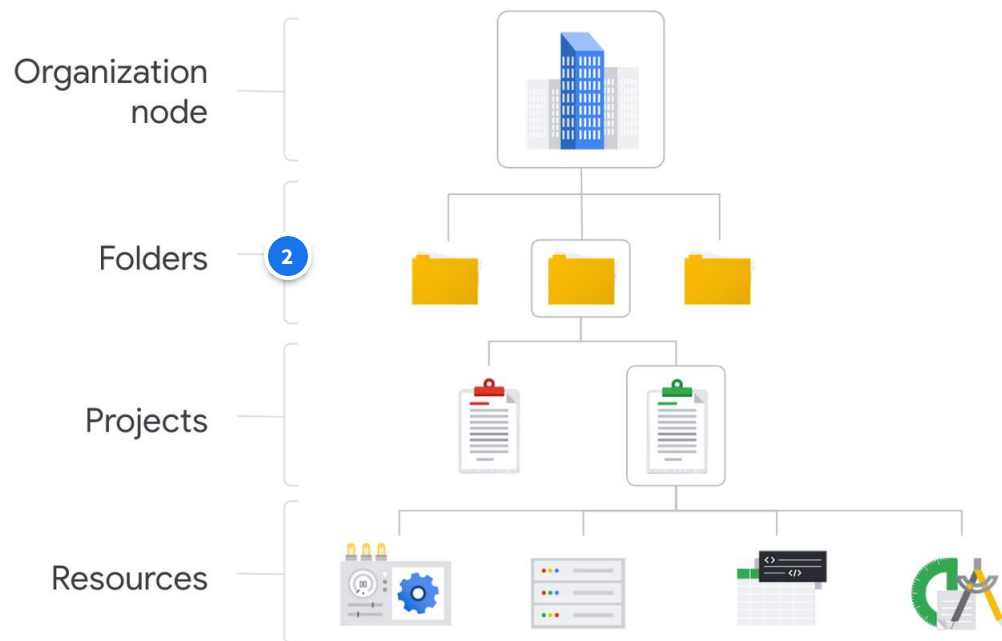




## Organization

The organization resource represents an organization and when present, is the root node in the Google Cloud resource hierarchy. The organization resource is the hierarchical ancestor of folder and project resources.

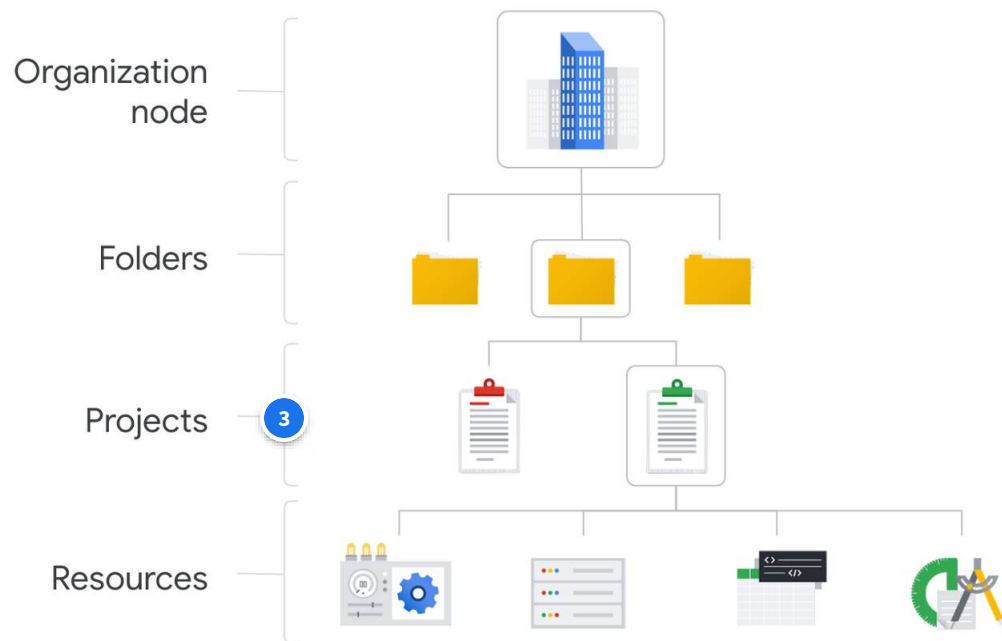
A Google Workspace or Cloud Identity account representing a company is a prerequisite in order to have access to the organization resource. In the Google Cloud context, it provides identity management, recovery mechanism, ownership, and lifecycle management.



## Folders

Each folder can represent a department or a team. It is also possible to have subfolders.

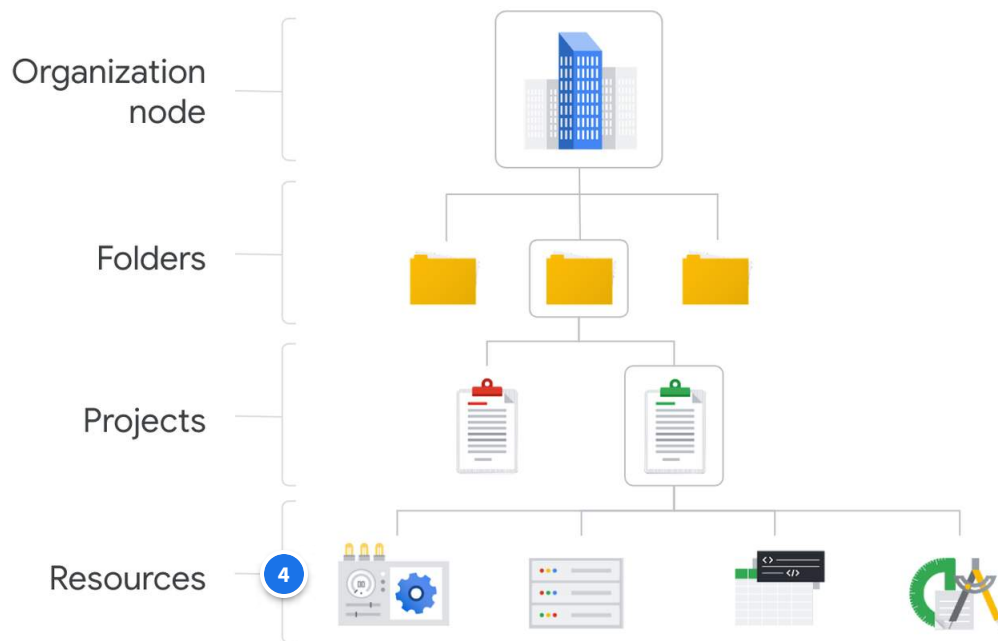




## Projects

Resources get organized into projects. Projects can be organized into folders, or even subfolders.

Project resources at the top of their hierarchy do not have parent resources, but they can be migrated into an organization resource once it has been created for the domain.




## Resources

Resources represent virtual machines, Cloud Storage buckets, Virtual Private Networks (VPCs), tables in BigQuery, or anything else in Google Cloud.

With an organization resource, project resources belong to your organization instead of the employee who created the project. This means that the project resources are no longer deleted when an employee leaves the company. Instead, they will follow the organization resource's lifecycle on Google Cloud. Furthermore, Organization Administrators have central control to view and manage all of your company's project resources, and you can use IAM to easily assign the Organization Administrator role to specific users or groups.

## IAM policies and the resource hierarchy

You can set an IAM policy at the organization level, the folder level, the project level, or (in some cases) the resource level. Resources inherit the policies of the parent resource. If you set a policy at the organization level, it is inherited by all its child folder and project resources, and if you set a policy at the project level, it is inherited by all its child resources.

 **The IAM access control policies applied on the organization resource apply throughout the hierarchy on all resources in the organization.**

The IAM policy hierarchy follows the same path as the Google Cloud resource hierarchy. If you change the resource hierarchy, the policy hierarchy changes as well. For example, moving a project into an organization resource will update the project's IAM policy to inherit from the organization resource's IAM policy.

Similarly, **moving a project resource from one folder resource to another will change the inherited permissions**. Permissions that were inherited by the project resource from the original parent resource will be lost when the project resource is moved to a new folder resource. Permissions set at the destination folder resource will be inherited by the project resource as it is moved.

# References

---

- Learn more about the BigQuery resource hierarchy from the documentation titled ["Organizing BigQuery resources."](#)
- Learn more about the Google Cloud resource hierarchy from the documentation titled ["Resource hierarchy."](#)
- Learn more about the organization resource in the Google Cloud resource hierarchy from the documentation titled ["Creating and managing organizations."](#)
- Learn more about organization policies from the documentation titled ["Introduction to the Organization Policy Service."](#)



---

Google Cloud

---

# Lesson introduction

---

## Introduction

In this lesson, you will learn the basics of how to create and manage datasets and tables in BigQuery.

You will also learn how the storage location of a BigQuery dataset impacts the compute jobs that run on the underlying data.



---

Let's get started!

# Dataset location

---

---

**BigQuery datasets can be regional or multi-regional. You specify a location for storing your BigQuery data when you create a dataset.**

1

**A region** is a specific geographic place, such as London.

2

**A multi-region** is a large geographic area, such as the United States, that contains two or more geographic places.

There are many regions available including several across the Americas, Asia Pacific, and Europe. For a multi-region, there are two available options: data centers in the United States and data centers within the member states of the European Union. You specify a regional or a multi-regional location for storing your BigQuery data when you create a dataset.

---

## Why does location matter?

BigQuery processes your queries in the same regional or multi-regional location as the dataset that contains the tables you're querying. For example, if a query references a table in a dataset stored in the `asia-northeast1` region, the query job will run in that region. If a query does not reference any tables or other resources contained within datasets, and no destination table is provided, the query job will run in the `US` multi-region.

After you create the dataset, the location cannot be changed, but you can copy the dataset to a different location, or manually move (recreate) the dataset in a different location.

**i** The location of BigQuery datasets are also important when querying external data sources across Google Cloud or using BigQuery data within other Google Cloud tools. Cloud Storage, Cloud SQL, Cloud Spanner, Bigtable, Dataproc, and Vertex AI have colocation considerations for BigQuery datasets.



# Dataset management

---

BigQuery provides many options for managing your datasets including creating, copying, updating, and deleting datasets.

---

## Creating datasets

You can create BigQuery datasets in the following ways:

- 1 Using the Google Cloud console.
  - 2 Using a SQL query.
  - 3 Using the `bq mk` command in the `bq` command-line tool.
  - 4 Calling the `datasets.insert` API method.
  - 5 Using the client libraries.
  - 6 Copying an existing dataset.
-

Instructions for each creation option are provided in the documentation titled ["Create datasets"](#).

To create a dataset, you need the IAM permission named `'bigquery.datasets.create'`, which is in many predefined IAM roles for BigQuery, including `'roles/bigquery.user'` and `'roles/bigquery.dataEditor'`.

---

## Managing datasets

After creating a dataset, you can manage the dataset in several ways including copying the dataset to duplicate, rename, or move it, setting dataset properties, and deleting the dataset.

To copy a dataset, you can use the BigQuery Data Transfer Service to transfer (or copy) data from a source to a destination dataset in BigQuery or use Cloud Composer to copy large datasets programmatically.

BigQuery also provides many dataset properties that you can set to manage your data such as default expiration times for tables and partitions, dataset access controls, and dataset descriptions.

---

## BigQuery datasets are subject to certain limitations:

- You can set the geographic location at creation time only. After a dataset has been created, the location becomes immutable and can't be changed.
- All tables that are referenced in a query must be stored in datasets in the same location.

- If you want to copy a table, then the datasets that contain the source table and destination table must reside in the same location.
- Dataset names for each project must be unique.

# Table management

---

BigQuery provides many options for managing your tables, including creating, copying, updating, deleting, and restoring tables.

---

## Creating tables

You can create a table in BigQuery in the following ways:

- 1 Manually using the Google Cloud console or the bq command-line tool bq mk command.
- 2 Programmatically by calling the tables.insert API method.
- 3 By using the client libraries.
- 4 From query results.
- 5 By defining a table that references an external data source.
- 6 When you load data.
- 7 By using a CREATE TABLE data definition language (DDL) statement.

Instructions for each creation option are provided in the documentation titled ["Create tables"](#).

To create a table within a BigQuery dataset, you need the following specific IAM permissions:

`'bigquery.tables.create'`, `'bigquery.tables.updateData'` and `'bigquery.jobs.create'`.

These permissions are included in many predefined IAM roles for BigQuery, including

`'roles/bigquery.user'` and `'roles/bigquery.dataEditor'`.

---

## Managing tables

After creating a table in a BigQuery dataset, you can manage the table in several ways including copying the table to duplicate or move it, renaming the table, setting table properties, deleting the table, and even restoring a deleted table.

There are many options for copying tables, including the Google Cloud console, the `'bq cp'` command, and the data definition language (DDL) `'CREATE TABLE COPY'` statement.

BigQuery also provides many table properties that you can update to manage your data such as names, descriptions, schema definitions, expiration times, and access control for individual tables in a dataset.

---

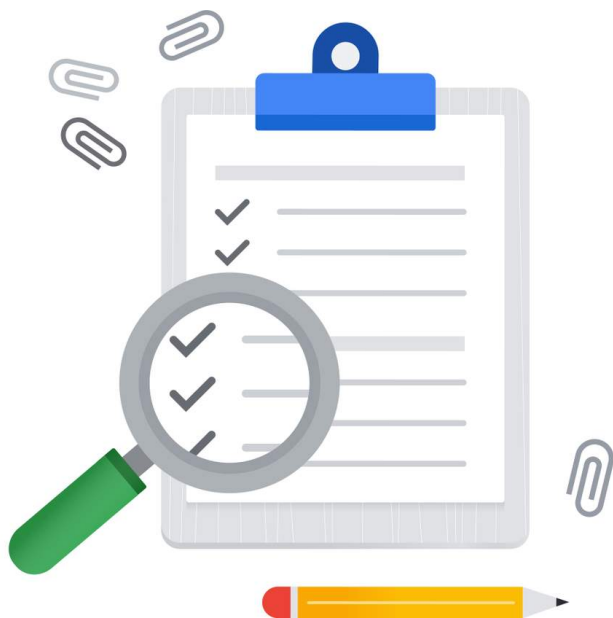
## BigQuery tables are subject to certain limitations:

- Table names must be unique per dataset.
- The Google Cloud console supports copying only one table at a time.
- The Google Cloud console can display up to 50,000 tables for each dataset.
- When you copy tables, the destination dataset must reside in the same location as the table being copied. For example, you cannot copy a table from an EU-based dataset to a US-based dataset.
- When you copy multiple source tables to a destination table by using the bq command-line tool, the API, or the client libraries, all source tables must have identical schemas.
- When you export table data, the only supported destination is Cloud Storage. However, you can use the Google Cloud console to [download query results](#) to a local file, Sheets, or Drive.

# References

---

- Learn more about creating, copying, moving, listing, updating, and managing datasets from the documentation titled ["Introduction to datasets."](#)
- Learn more about factors that you should consider when choosing a location for BigQuery datasets from the documentation titled ["Location considerations."](#)
- Learn more about working with tables from the documentation titled ["Create and use tables."](#)
- Learn more about working with views in BigQuery from the documentation titled ["Create views."](#)



---

Google Cloud

---



# Lesson introduction

---

In this lesson, you will examine data types in Redshift and BigQuery. You'll learn about some of the key similarities and differences, and then explore how data types map between Redshift and BigQuery.



---

Let's get started!

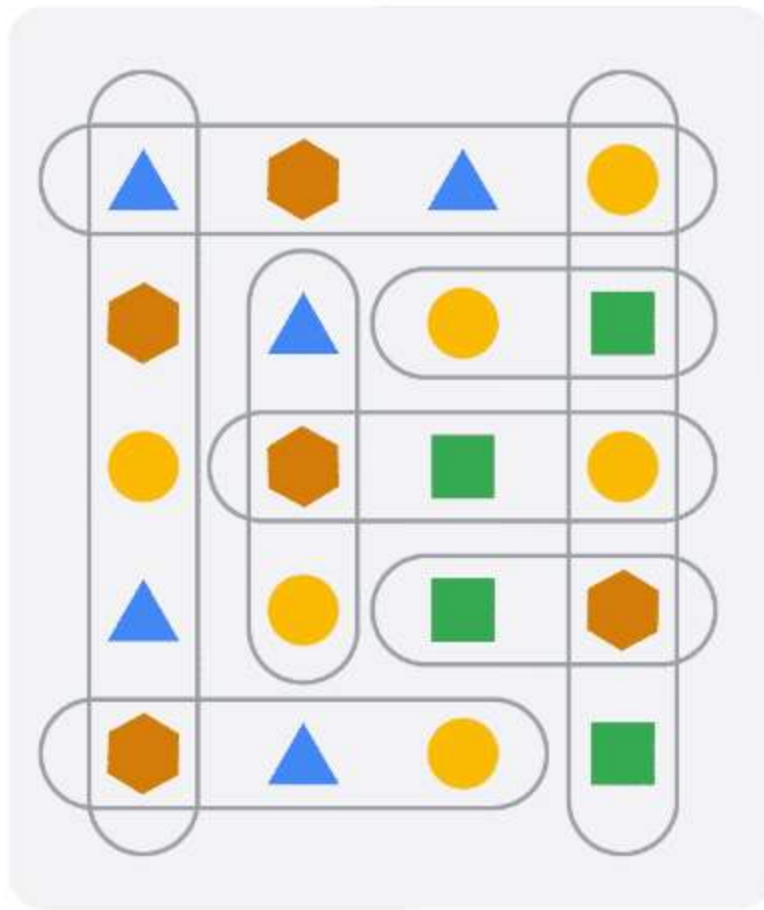
# Key points about data types

---

There are some similarities when working with data types in Redshift and BigQuery.

For example:

- Redshift and BigQuery use dynamic typing when ingesting data without a specified type.
- Both have user-defined data type parameters where you can set limits on the data type.



Let's also consider some of the differences between Redshift and BigQuery.

*Click the + icon to expand and learn more.*

#### Boolean values

- Redshift supports TRUE and FALSE for Boolean values.

- BigQuery supports the NULL value in addition to TRUE and FALSE for Boolean values.

## Column limit —

- Redshift's column limit per table is [1,600](#).
- BigQuery's column limit per table is [10,000](#).

## Schema detection —

- Redshift requires user input for a [data-specific schema](#).
- BigQuery offers [schema auto-detection](#) when loading data or querying an external data source.

## JSON Data —

- Redshift uses a set of schemaless array and structure values to tag data locations.
- BigQuery uses canonicalization to format JSON data such as preserving Booleans, strings, and nulls exactly but not preserving whitespace characters.

## Query interface —

- When working with SUPER data in Redshift, users [utilize PartiQL in a separate query interface](#).

- BigQuery provides a unified query interface to work with all data types including arrays and JSON.

## Nested depth —

- Redshift's nested depth limit is [100](#).
- BigQuery's nested depth limit is [15](#) for the RECORD type and [500](#) for JSON.

# Map of data types in BigQuery and Redshift

---

Here's how [data types map](#) across BigQuery and Redshift.

Type	BigQuery	Redshift
<b>Array type</b>	<a href="#">ARRAY</a> : ordered list of zero or more elements of non-ARRAY values	<a href="#">SUPER</a> datatype stores semistructured data as values including arrays and nested structures. SUPER uses tags that reference distinct entities in the data.
<b>Boolean type</b>	<a href="#">BOOL</a> : NULL, FALSE, TRUE	<a href="#">Boolean Type</a> : true/false
<b>Bytes type</b>	<a href="#">BYTES</a> : variable-length binary data	<a href="#">VARBYTE</a> : variable-length binary value with a fixed limit
<b>Date type</b>	<a href="#">DATE</a> : range from 0001-01-01 to 9999-12-31	<a href="#">Date</a> : Calendar date only
<b>Datetime type</b>	<a href="#">DATETIME</a> : range from 0001-01-01 00:00:00 to 9999-12-31 23:59:59.999999	<a href="#">Datetime</a> includes DATE, TIME, TIMETZ, TIMESTAMP, and TIMESTAMPTZ plus interval literals

Type	BigQuery	Redshift
<b>Geography type</b>	<a href="#">GEOGRAPHY</a> : collection of points, linestrings, and polygons, which is represented as a point set, or a subset of the surface of the Earth.	<a href="#">Geography</a> : two-dimensional (2D), three-dimensional (3DZ), two-dimensional with a measure (3DM), and four-dimensional (4D) geometry primitive data types  <a href="#">Geometry</a> : Cartesian plane data type with spatial reference system identifier
<b>Interval type</b>	<a href="#">INTERVAL</a> : duration or amount of time	<a href="#">Datetime</a> includes interval literals
<b>JSON type</b>	<a href="#">JSON</a> : represents JSON	<a href="#">SUPER</a> : store semistructured data or documents as values

Type	BigQuery	Redshift
<b>Numeric types</b>	<p><a href="#">Numeric types</a></p> <p>INT64 (alias INT, SMALLINT, INTEGER, BIGINT, TINYINT, BYTEINT): integer</p> <p>NUMERIC (alias DECIMAL) and BIGNUMERIC (alias BIGDECIMAL): fixed decimal precision and scale</p> <p>FLOAT64: double precision (approximate) numeric values</p> <p><a href="#">Parameterized decimal types</a></p>	<p><a href="#">Numeric types</a> include integers, decimals and floating point numbers</p> <p>Integer: SMALLINT, INTEGER, and BIGINT</p> <p>Decimal OR Numeric: user-defined precision with up to 38 digits of precision</p>
<b>String types</b>	<p><a href="#">String types</a></p> <p><a href="#">STRING(L): parameterized string type</a></p>	<p><a href="#">Character types</a> include characters, varying characters, text, bpchar and national types</p>
<b>Struct type</b>	<p><a href="#">STRUCT</a>: container of ordered fields each with a type (required) and field name (optional).</p>	<p><a href="#">SUPER</a> datatype stores semistructured data as values including arrays and nested structures. SUPER uses tags that reference distinct entities in the data.</p>



Type	BigQuery	Redshift
Time type	<a href="#">TIME</a> : range from 00:00:00 to 23:59:59.999999	<a href="#">TIME</a> : time of day without time zone  <a href="#">TIMETZ</a> : time of day with time zone
Timestamp type	<a href="#">TIMESTAMP</a> : range from 0001-01-01 00:00:00 to 9999-12-31 23:59:59.999999 UTC  <a href="#">Time zones</a>	<a href="#">TIMESTAMP</a> : date and time  <a href="#">TIMESTAMPTZ</a> : date and time with time zone
Hyper Log Log	BigQuery provides <a href="#">HyperLogLog++ functions</a> , rather than a specific data type.	<a href="#">HLLSKETCH</a> : sparse HyperLogLog format

CONTINUE