Getting table metadata using INFORMATION_SCHEMA

ature is covered by the Pre-GA Offerings Terms (/terms/service-terms#1) of the Google Cloud Platform Terms
e. Pre-GA features may have limited support, and changes to pre-GA features may not be compatible with oth
sions. For more information, see the launch stage descriptions (/products#product-launch-stages).

`INFORMATION_SCHEMA` contains these views for table metadata:

- `TABLES` and `TABLE_OPTIONS` for metadata about tables
- `COLUMNS` and `COLUMN_FIELD_PATHS` for metadata about columns and fields

`TABLES` and `TABLE_OPTIONS` also contain high-level information about views. For detailed information, query `VIEWS` (/bigquery/docs/information-schema-views) instead.

Required permissions

For `TABLES` and `TABLE_OPTIONS`, you must be granted these permissions:

- `bigquery.tables.get`
- `bigquery.tables.list`
- `bigquery.routines.get`
- `bigquery.routines.list`

For `COLUMNS` and `COLUMN_FIELD_PATHS`, you must be granted these permissions:

- `bigquery.tables.get`
- `bigquery.tables.list`

Syntax

Queries against any of these views must have a dataset or region qualifier (/bigquery/docs/information-schema-intro#syntax).

```
turns metadata for tables in a single dataset.
T * FROM myDataset.INFORMATION_SCHEMA.TABLES;

turns metadata for tables in a region.
T * FROM region-us.INFORMATION_SCHEMA.TABLES;
```

TABLES view

When you query the `INFORMATION_SCHEMA.TABLES` view, the query results contain one row for each table or view in a dataset.

The `INFORMATION_SCHEMA.TABLES` view has the following schema:

Column name	Data type	Value
`TABLE_CATALOG`	`STRING`	The name of the project that contains the dataset
`TABLE_SCHEMA`	`STRING`	The name of the dataset that contains the table or view also referred to as the `datasetId`
`TABLE_NAME`	`STRING`	The name of the table or view also referred to as the `tableId`
`TABLE_TYPE`	`STRING`	The table type: • `BASE TABLE`: A BigQuery table (/bigquery/docs/tables-intro) • `VIEW`: A BigQuery view (/bigquery/docs/views-intro) • `EXTERNAL`: A table that references an external data source (/bigquery/external-data-sources)
`IS_INSERTABLE_INTO`	`STRING`	`YES` or `NO` depending on whether the table supports DML INSERT (/bigquery/docs/reference/standard-sql/dml-syntax#insert_statement) statements
`IS_TYPED`	`STRING`	The value is always `NO`
`CREATION_TIME`	`TIMESTAMP`	The table's creation time

Examples

Example 1:

The following example retrieves all columns from the `INFORMATION_SCHEMA.TABLES` view except for `is_typed` which is reserved for future use. The metadata returned is for all tables in `mydataset` in your default project — `myproject`.

`mydataset` contains the following tables:

- `mytable1`: a standard BigQuery table
- `myview1`: a BigQuery view

To run the query against a project other than your default project, add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `myproject`.mydataset.INFORMATION_SCHEMA.TABLES ``.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 * EXCEPT(is_typed)
FROM
 mydataset.INFORMATION_SCHEMA.TABLES
```

   ★ **Note:** `INFORMATION_SCHEMA` view names are case-sensitive.

3. Click **Run**.

The results should look like the following:

```
--------------+---------------+----------------+------------+--------------------+--
able_catalog  | table_schema  |   table_name   | table_type | is_insertable_into |
--------------+---------------+----------------+------------+--------------------+--
yproject      | mydataset     | mytable1       | BASE TABLE | YES                | 2
yproject      | mydataset     | myview1        | VIEW       | NO                 | 2
--------------+---------------+----------------+------------+--------------------+--
```

Example 2:

The following example retrieves all tables of type `BASE TABLE` from the `INFORMATION_SCHEMA.TABLES` view. The `is_typed` column is excluded. The metadata returned is for tables in `mydataset` in your default project — `myproject`.

To run the query against a project other than your default project, add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `myproject`.mydataset.INFORMATION_SCHEMA.TABLES ``.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 * EXCEPT(is_typed)
FROM
 mydataset.INFORMATION_SCHEMA.TABLES
WHERE
 table_type="BASE TABLE"
```

★ **Note:** **`INFORMATION_SCHEMA`** view names are case-sensitive.

3. Click **Run**.

The results should look like the following:

```
--------------+---------------+----------------+------------+--------------------+--
able_catalog  | table_schema  |   table_name   | table_type | is_insertable_into |
--------------+---------------+----------------+------------+--------------------+--
yproject      | mydataset     | mytable1       | BASE TABLE | NO                 | 2
--------------+---------------+----------------+------------+--------------------+--
```

TABLE_OPTIONS view

When you query the `INFORMATION_SCHEMA.TABLE_OPTIONS` view, the query results contain one row for each table or view in a dataset.

The `INFORMATION_SCHEMA.TABLE_OPTIONS` view has the following schema:

Column name	Data type	Value
`TABLE_CATALOG`	`STRING`	The name of the project that contains the dataset
`TABLE_SCHEMA`	`STRING`	The name of the dataset that contains the table or view also referred to as the `datasetId`
`TABLE_NAME`	`STRING`	The name of the table or view also referred to as the `tableId`
`OPTION_NAME`	`STRING`	One of the name values in the options table (#options_table)
`OPTION_TYPE`	`STRING`	One of the data type values in the options table (#options_table)
`OPTION_VALUE`	`STRING`	One of the value options in the options table (#options_table)

Options table

OPTION_NAME	OPTION_TYPE	OPTION_VALUE

`partition_expiration_days`	`FLOAT64`	The default lifetime, in days, of all partitions in a partitioned table
`expiration_timestamp`	`FLOAT64`	The time when this table expires
`kms_key_name`	`STRING`	The name of the Cloud KMS key used to encrypt the table
`friendly_name`	`STRING`	The table's descriptive name
`description`	`STRING`	A description of the table
`labels`	`ARRAY<STRUCT<STRING, STRING>>`	An array of `STRUCT`s that represent the labels on the table
`require_partition_filter`	`BOOL`	Whether queries over the table require a partition filter

Examples

Example 1:

The following example retrieves the default table expiration times for all tables in `mydataset` in your default project (`myproject`) by querying the `INFORMATION_SCHEMA.TABLE_OPTIONS` view.

To run the query against a project other than your default project, add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `myproject`.mydataset.INFORMATION_SCHEMA.TABLE_OPTIONS ``.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 *
```

```
FROM
 mydataset.INFORMATION_SCHEMA.TABLE_OPTIONS
WHERE
 option_name="expiration_timestamp"
```

★ **Note:** **`INFORMATION_SCHEMA`** view names are case-sensitive.

3. Click **Run**.

The results should look like the following:

```
--------------+---------------+------------+----------------------+-------------+---
able_catalog  | table_schema  | table_name |     option_name      | option_type |
--------------+---------------+------------+----------------------+-------------+---
yproject      | mydataset     | mytable1   | expiration_timestamp | TIMESTAMP   | TI
yproject      | mydataset     | mytable2   | expiration_timestamp | TIMESTAMP   | TI
--------------+---------------+------------+----------------------+-------------+---
```

Tables without an expiration time are excluded from the query results.

Example 2:

The following example retrieves metadata about all tables in `mydataset` that contain test data. The query uses the values in the `description` option to find tables that contain "test" anywhere in the description. `mydataset` is in your default project — `myproject`.

To run the query against a project other than your default project, add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `myproject`.mydataset.INFORMATION_SCHEMA.TABLE_OPTIONS ``.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 *
FROM
 mydataset.INFORMATION_SCHEMA.TABLE_OPTIONS
WHERE
 option_name="description" AND option_value LIKE "%test%"
```

★ **Note:** **`INFORMATION_SCHEMA`** view names are case-sensitive.

3. Click **Run**.

The results should look like the following:

```
--------------+---------------+------------+-------------+-------------+------------
able_catalog  | table_schema  | table_name | option_name | option_type | option_valu
--------------+---------------+------------+-------------+-------------+------------
yproject      | mydataset     | mytable1   | description | STRING      | "test data"
yproject      | mydataset     | mytable2   | description | STRING      | "test data"
--------------+---------------+------------+-------------+-------------+------------
```

`COLUMNS` view

When you query the `INFORMATION_SCHEMA.COLUMNS` view, the query results contain one row for each column (field) in a table.

The `INFORMATION_SCHEMA.COLUMNS` view has the following schema:

Column name	Data	Value

	type	
TABLE_CATALOG	STRING	The name of the project that contains the dataset
TABLE_SCHEMA	STRING	The name of the dataset that contains the table also referred to as the datasetId
TABLE_NAME	STRING	The name of the table or view also referred to as the tableId
COLUMN_NAME	STRING	The name of the column
ORDINAL_POSITION	INT64	The 1-indexed offset of the column within the table; if it's a pseudo column such as _PARTITIONTIME or _PARTITIONDATE, the value is NULL
IS_NULLABLE	STRING	YES or NO depending on whether the column's mode allows NULL values
DATA_TYPE	STRING	The column's standard SQL data type (/bigquery/docs/reference/standard-sql/data-types)
IS_GENERATED	STRING	The value is always NEVER
GENERATION_EXPRESSION	STRING	The value is always NULL
IS_STORED	STRING	The value is always NULL
IS_HIDDEN	STRING	YES or NO depending on whether the column is a pseudo column such as _PARTITIONTIME or _PARTITIONDATE
IS_UPDATABLE	STRING	The value is always NULL
IS_SYSTEM_DEFINED	STRING	YES or NO depending on whether the column is a pseudo column such as _PARTITIONTIME or _PARTITIONDATE
IS_PARTITIONING_COLUMN	STRING	YES or NO depending on whether the column is a partitioning column (/bigquery/docs/creating-column-partitions)
CLUSTERING_ORDINAL_POSITION	INT64	The 1-indexed offset of the column within the table's clustering columns; the value is NULL if the table is not a clustered table

Examples

The following example retrieves metadata from the `INFORMATION_SCHEMA.COLUMNS` view for the `population_by_zip_2010` table in the `census_bureau_usa`

(https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=census_bureau_usa&page=dataset)
dataset. This dataset is part of the BigQuery public dataset program (https://cloud.google.com/public-datasets/).

Because the table you're querying is in another project, the `bigquery-public-data` project, you add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `bigquery-public-data`.census_bureau_usa.INFORMATION_SCHEMA.TABLES ``.

The following columns are excluded from the query results because they are currently reserved for future use:

- `IS_GENERATED`
- `GENERATION_EXPRESSION`
- `IS_STORED`
- `IS_UPDATABLE`

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 * EXCEPT(is_generated, generation_expression, is_stored, is_updatable)
FROM
 `bigquery-public-data`.census_bureau_usa.INFORMATION_SCHEMA.COLUMNS
WHERE
 table_name="population_by_zip_2010"
```

★ **Note:** `INFORMATION_SCHEMA` view names are case-sensitive.

3. Click **Run**.

The results should look like the following. For readability, `table_catalog` and `table_schema` are excluded from the results:

```
--------------------+-------------+------------------+-------------+-----------+----
   table_name       | column_name | ordinal_position | is_nullable | data_type | is_
--------------------+-------------+------------------+-------------+-----------+----
ulation_by_zip_2010 | zipcode     |                1 | NO          | STRING    | NO
ulation_by_zip_2010 | geo_id      |                2 | YES         | STRING    | NO
ulation_by_zip_2010 | minimum_age |                3 | YES         | INT64     | NO
ulation_by_zip_2010 | maximum_age |                4 | YES         | INT64     | NO
ulation_by_zip_2010 | gender      |                5 | YES         | STRING    | NO
ulation_by_zip_2010 | population  |                6 | YES         | INT64     | NO
--------------------+-------------+------------------+-------------+-----------+----
```

`COLUMN_FIELD_PATHS` view

Query results contain one row for each column nested (/bigquery/docs/nested-repeated) within a `RECORD` (or `STRUCT`) column.

When you query the `INFORMATION_SCHEMA.COLUMN_FIELD_PATHS` view, the query results contain one row for each column nested (/bigquery/docs/nested-repeated) within a `RECORD` (or `STRUCT`) column.

The `INFORMATION_SCHEMA.COLUMN_FIELD_PATHS` view has the following schema:

Column name	Data type	Value
`TABLE_CATALOG`	`STRING`	The name of the project that contains the dataset
`TABLE_SCHEMA`	`STRING`	The name of the dataset that contains the table also referred to as the **`datasetId`**
`TABLE_NAME`	`STRING`	The name of the table or view also referred to as the **`tableId`**
`COLUMN_NAME`	`STRING`	The name of the column
`FIELD_PATH`	`STRING`	The path to a column nested (/bigquery/docs/nested-repeated) within a `RECORD`

		or `STRUCT` column
`DATA_TYPE`	`STRING`	The column's standard SQL data type (/bigquery/docs/reference/standard-sql/data-types)
`DESCRIPTION`	`STRING`	The column's description

Examples

The following example retrieves metadata from the `INFORMATION_SCHEMA.COLUMN_FIELD_PATHS` view for the `commits` table in the `github_repos` dataset (https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=github_repos&page=dataset). This dataset is part of the BigQuery public dataset program (https://cloud.google.com/public-datasets/).

Because the table you're querying is in another project, the `bigquery-public-data` project, you add the project ID to the dataset in the following format: `` `project_id`.dataset.INFORMATION_SCHEMA.view ``; for example, `` `bigquery-public-data`.github_repos.INFORMATION_SCHEMA.COLUMN_FIELD_PATHS ``.

The `commits` table contains the following nested and nested and repeated columns:

- `author`: nested `RECORD` column
- `committer`: nested `RECORD` column
- `trailer`: nested and repeated `RECORD` column
- `difference`: nested and repeated `RECORD` column

Your query will retrieve metadata about the `author` and `difference` columns.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
SELECT
 *
FROM
 `bigquery-public-data`.github_repos.INFORMATION_SCHEMA.COLUMN_FIELD_PATH
WHERE
 table_name="commits"
 AND column_name="author"
 OR column_name="difference"
```

★ **Note:** **`INFORMATION_SCHEMA`** view names are case-sensitive.

3. Click **Run**.

The results should look like the following. For readability, `table_catalog` and `table_schema` are excluded from the results.

```
----------+-------------+---------------------+-------------------------------------
able_name | column_name |     field_path      |
----------+-------------+---------------------+-------------------------------------
ommits    | author      | author              | STRUCT<name STRING, email STRING, ti
ommits    | author      | author.name         | STRING
ommits    | author      | author.email        | STRING
ommits    | author      | author.time_sec     | INT64
ommits    | author      | author.tz_offset    | INT64
ommits    | author      | author.date         | TIMESTAMP
ommits    | difference  | difference          | ARRAY<STRUCT<old_mode INT64, new_mod
ommits    | difference  | difference.old_mode | INT64
ommits    | difference  | difference.new_mode | INT64
ommits    | difference  | difference.old_path | STRING
ommits    | difference  | difference.new_path | STRING
ommits    | difference  | difference.old_sha1 | STRING
ommits    | difference  | difference.new_sha1 | STRING
ommits    | difference  | difference.old_repo | STRING
ommits    | difference  | difference.new_repo | STRING
----------+-------------+---------------------+-------------------------------------
```

Advanced example

The following advanced example queries the `INFORMATION_SCHEMA.TABLES`, `TABLE_OPTIONS`, and `COLUMNS` views to retrieve metadata about the tables in `mydataset` in your default project — `myproject`. `mydataset` contains 2 tables:

- `mytable1`: Uses the same schema as the `commits` table in the `github_repos` (https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=github_repos&page=dataset) public dataset
- `mytable2`: Uses the same schema as the `population_by_zip_2010` table in the `census_bureau_usa` (https://console.cloud.google.com/bigquery?p=bigquery-public-data&d=census_bureau_usa&page=dataset) public dataset

The results are used by user-defined functions (/bigquery/docs/reference/standard-sql/user-defined-functions) to assemble the DDL (/bigquery/docs/reference/standard-sql/data-definition-language) statements necessary to recreate the tables. You can then use the DDL statements in the query results to recreate the tables in `mydataset`.

To run the query against a project other than your default project, add the project ID to the dataset in the following format: `` `*project_id*`.*dataset*.INFORMATION_SCHEMA.*view* ``; for example, `` `myproject`.mydataset.INFORMATION_SCHEMA.TABLES ``.

To run the query:

Consolebq (#bq)

1. Open the BigQuery web UI in the Cloud Console.

   Go to the Cloud Console (https://console.cloud.google.com/bigquery)

2. Enter the following standard SQL query in the **Query editor** box. `INFORMATION_SCHEMA` requires standard SQL syntax. Standard SQL is the default syntax in the Cloud Console.

```
CREATE TEMP FUNCTION MakePartitionByExpression(
  column_name STRING, data_type STRING
```

```
) AS (
  IF(
    column_name = '_PARTITIONTIME',
    'DATE(_PARTITIONTIME)',
    IF(
      data_type = 'TIMESTAMP',
      CONCAT('DATE(', column_name, ')'),
      column_name
    )
  )
);

CREATE TEMP FUNCTION MakePartitionByClause(
  columns ARRAY<STRUCT<column_name STRING, data_type STRING, is_nullable
) AS (
  IFNULL(
    CONCAT(
      'PARTITION BY ',
      (SELECT MakePartitionByExpression(column_name, data_type)
       FROM UNNEST(columns) WHERE is_partitioning_column = 'YES'),
      '\n'),
    ''
  )
);

CREATE TEMP FUNCTION MakeClusterByClause(
  columns ARRAY<STRUCT<column_name STRING, data_type STRING, is_nullable
) AS (
  IFNULL(
    CONCAT(
      'CLUSTER BY ',
      (SELECT STRING_AGG(column_name, ', ' ORDER BY clustering_ordinal_po
        FROM UNNEST(columns) WHERE clustering_ordinal_position IS NOT NUL
      '\n'
    ),
    ''
  )
);

CREATE TEMP FUNCTION MakeNullable(data_type STRING, is_nullable STRING)
AS (
  IF(not STARTS_WITH(data_type, 'ARRAY<') and is_nullable = 'NO', ' NOT N
);

CREATE TEMP FUNCTION MakeColumnList(
```

```
  columns ARRAY<STRUCT<column_name STRING, data_type STRING, is_nullable
) AS (
  IFNULL(
    CONCAT(
      '(\n',
      (SELECT STRING_AGG(CONCAT('  ', column_name, ' ', data_type,  MakeN
       FROM UNNEST(columns)),
      '\n)\n'
    ),
    ''
  )
);

CREATE TEMP FUNCTION MakeOptionList(
  options ARRAY<STRUCT<option_name STRING, option_value STRING>>
) AS (
  IFNULL(
    CONCAT(
      'OPTIONS (\n',
      (SELECT STRING_AGG(CONCAT('  ', option_name, '=', option_value), ',
      '\n)\n'),
    ''
  )
);

WITH Components AS (
  SELECT
    CONCAT('`', table_catalog, '.', table_schema, '.', table_name, '`') A
    ARRAY_AGG(
      STRUCT(column_name, data_type, is_nullable, is_partitioning_column,
      ORDER BY ordinal_position
    ) AS columns,
    (SELECT ARRAY_AGG(STRUCT(option_name, option_value))
     FROM mydataset.INFORMATION_SCHEMA.TABLE_OPTIONS AS t2
     WHERE t.table_name = t2.table_name) AS options
  FROM mydataset.INFORMATION_SCHEMA.TABLES AS t
  LEFT JOIN mydataset.INFORMATION_SCHEMA.COLUMNS
  USING (table_catalog, table_schema, table_name)
  WHERE table_type = 'BASE TABLE'
  GROUP BY table_catalog, table_schema, t.table_name
)
SELECT
  CONCAT(
    'CREATE OR REPLACE TABLE ',
    table_name,
```

```
    '\n',
    MakeColumnList(columns),
    MakePartitionByClause(columns),
    MakeClusterByClause(columns),
    MakeOptionList(options))
FROM Components
```

The output should look like the following:

```
-------------------------------------------------------------------------------------
                                                                              f0_
-------------------------------------------------------------------------------------
ATE OR REPLACE TABLE `myproject.mydataset.population_by_zip_2010`

ipcode STRING NOT NULL,
eo_id STRING,
inimum_age INT64,
aximum_age INT64,
ender STRING,
opulation INT64

IONS (
xpiration_timestamp=TIMESTAMP "2019-04-17T02:10:32.055Z"

ATE OR REPLACE TABLE `myproject.mydataset.commits`

ommit STRING,
ree STRING,
arent ARRAY<STRING>,
uthor STRUCT<name STRING, email STRING, time_sec INT64, tz_offset INT64, date TIMEST
ommitter STRUCT<name STRING, email STRING, time_sec INT64, tz_offset INT64, date TIM
ubject STRING,
essage STRING,
railer ARRAY<STRUCT<key STRING, value STRING, email STRING>>,
ifference ARRAY<STRUCT<old_mode INT64, new_mode INT64, old_path STRING, new_path STR
ifference_truncated BOOL,
epo_name ARRAY<STRING>,
ncoding STRING

IONS (
xpiration_timestamp=TIMESTAMP "2019-04-17T03:12:03.248Z"
```

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see the Google Developers Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2020-06-30 UTC.