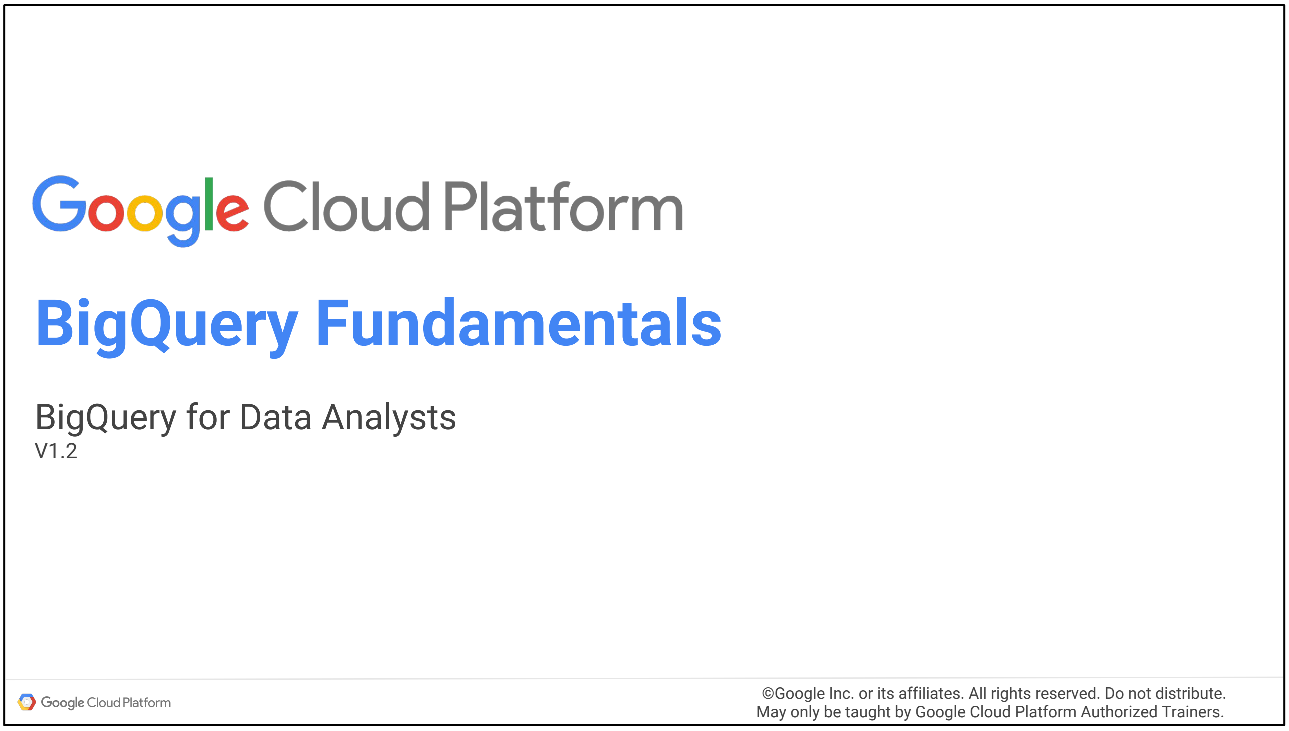


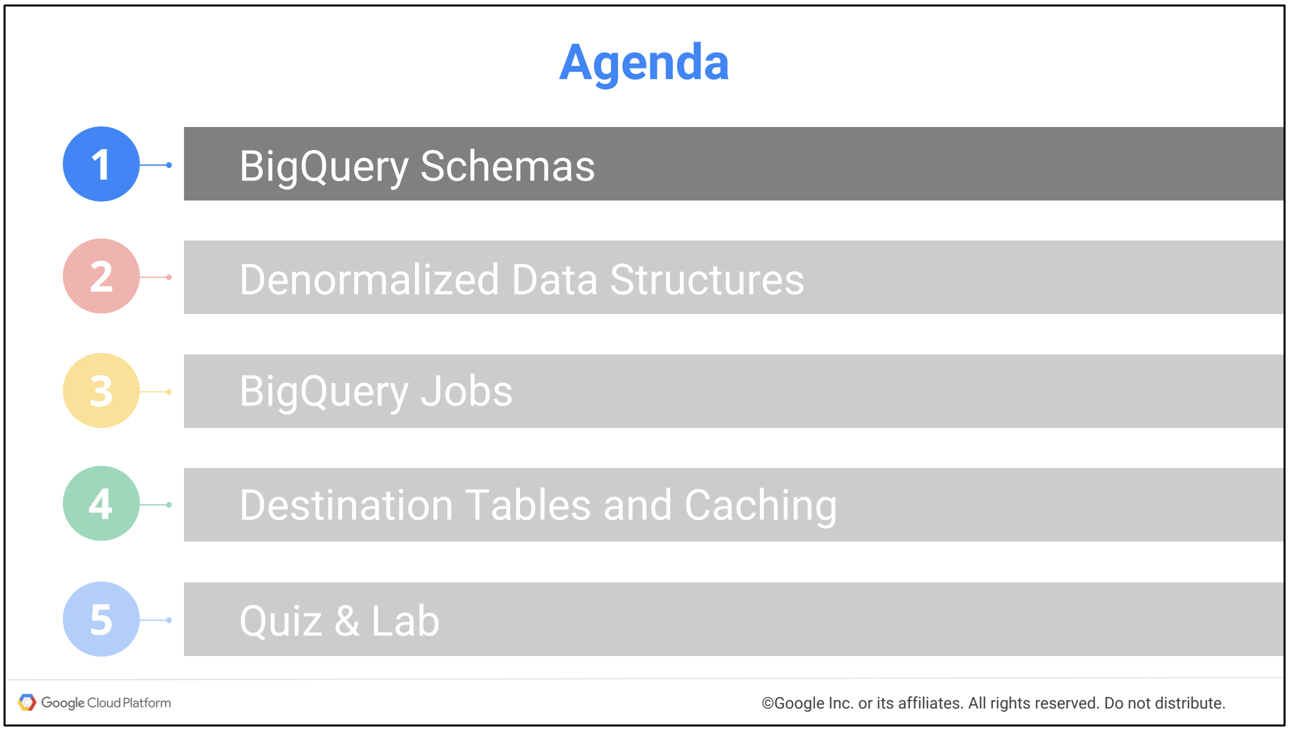
Google Cloud Platform

BigQuery Fundamentals

BigQuery for Data Analysts
V1.2

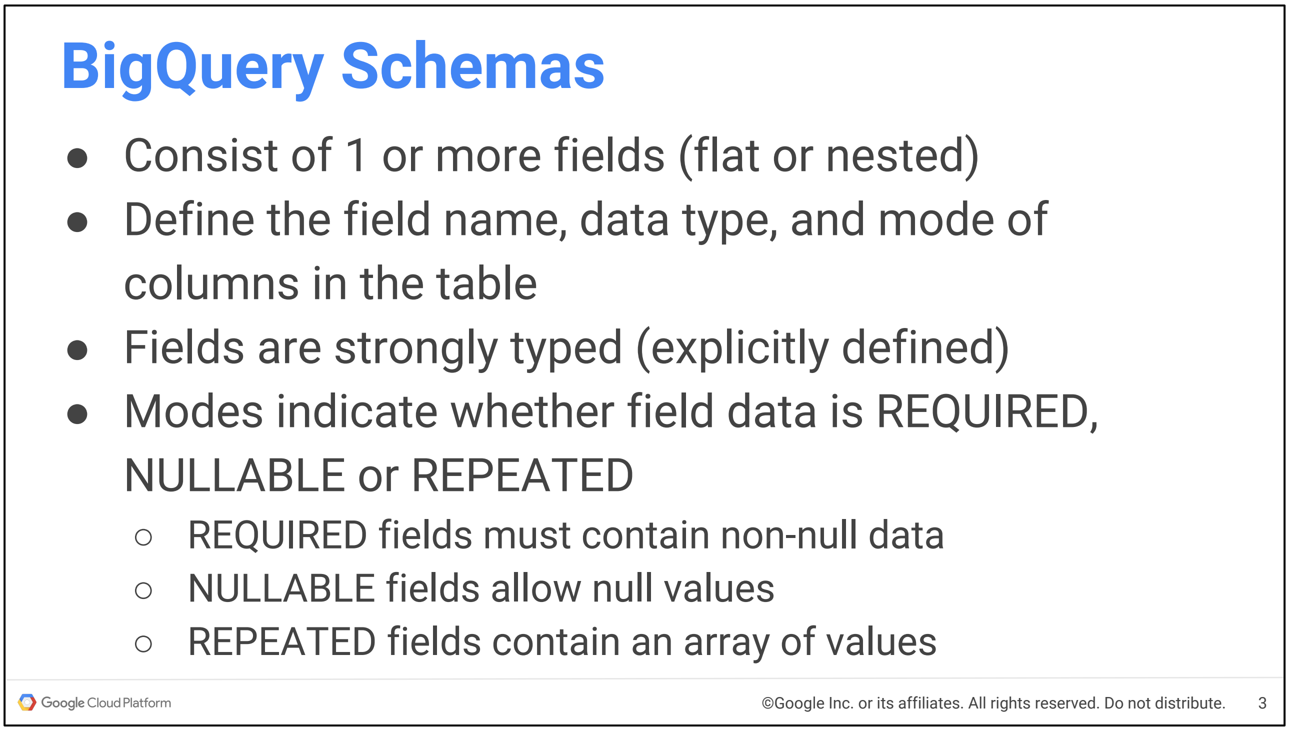
Google Cloud Platform

Approximate timing: 45 minutes



Agenda

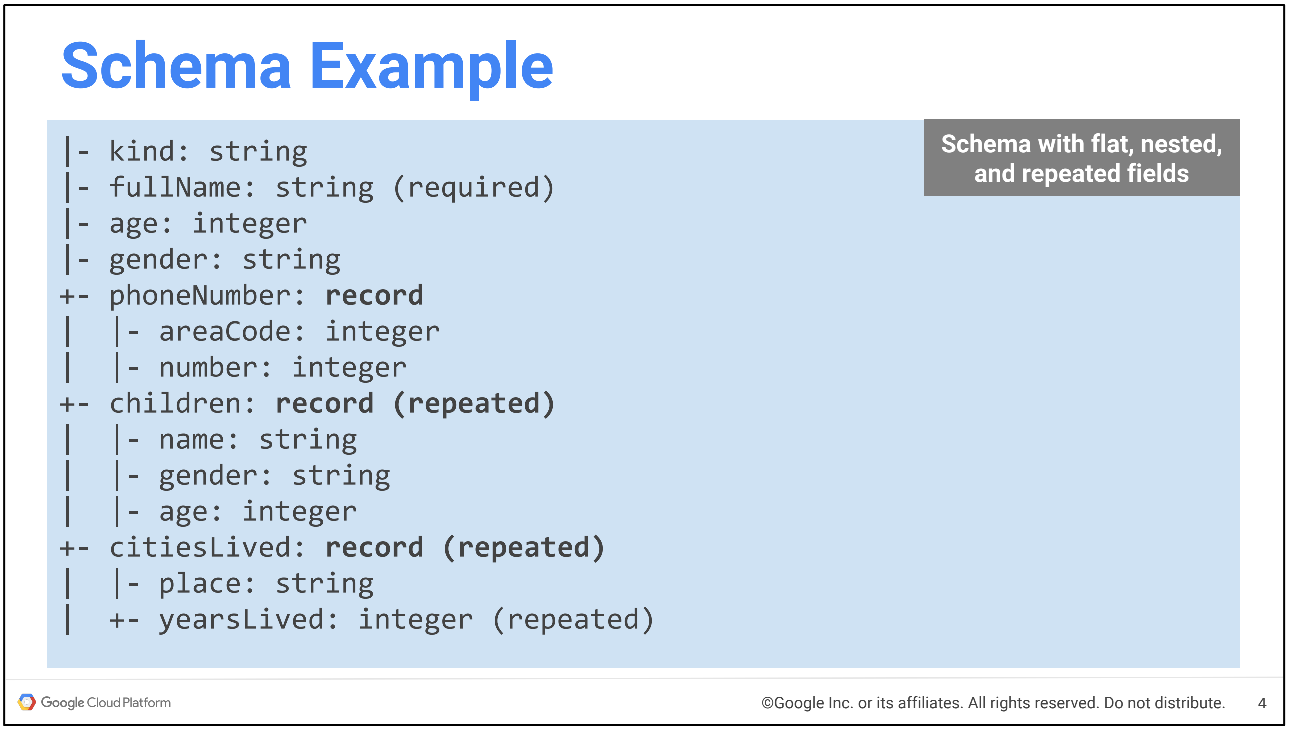
1. BigQuery Schemas
2. Denormalized Data Structures
3. BigQuery Jobs
4. Destination Tables and Caching
5. Quiz & Lab



BigQuery Schemas

- Consist of 1 or more fields (flat or nested)
- Define the field name, data type, and mode of columns in the table
- Fields are strongly typed (explicitly defined)
- Modes indicate whether field data is REQUIRED, NULLABLE or REPEATED
  - REQUIRED fields must contain non-null data
  - NULLABLE fields allow null values
  - REPEATED fields contain an array of values

Notes:
A schema defines the data structures to be used for the tables. A table's schema is made up of 1 or more fields and are strongly-typed columns of values. Each field must be defined with one of the primitive field types of INTEGER, FLOAT, BOOLEAN, STRING, or TIMESTAMP.



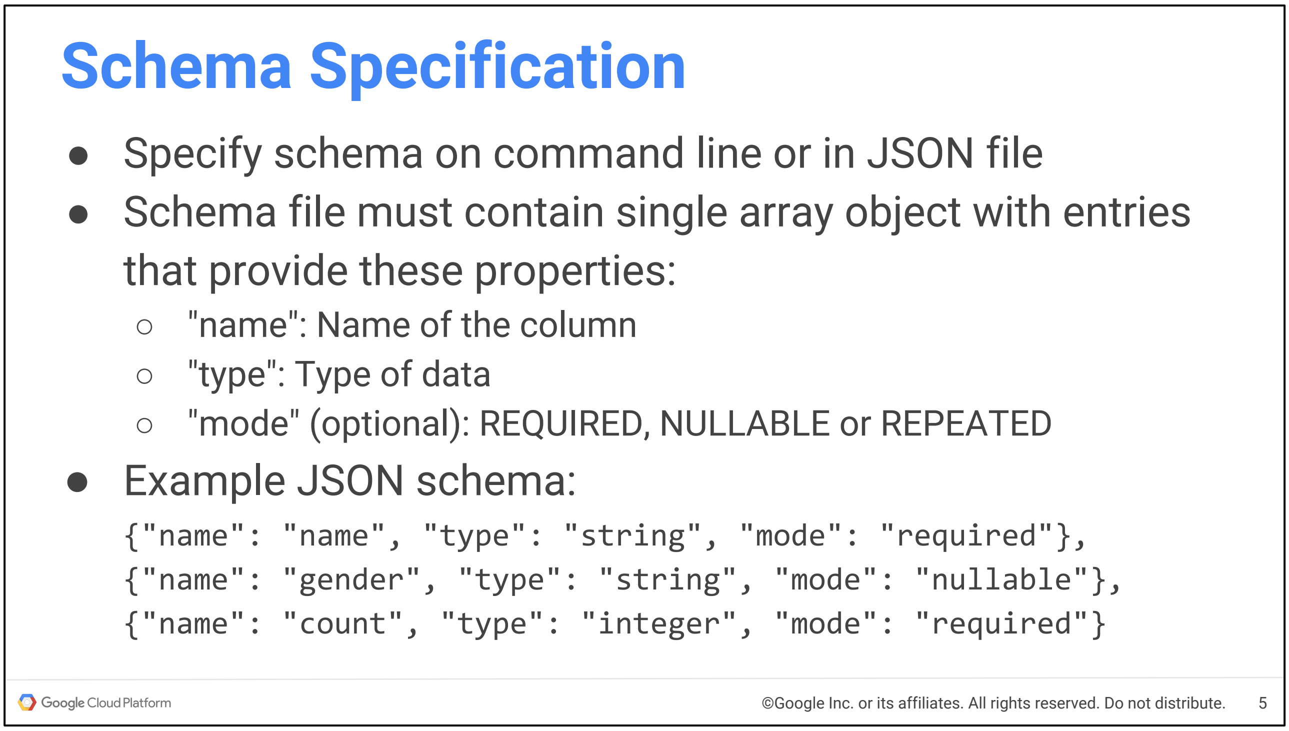
Schema Example

```
|- kind: string
|- fullName: string (required)
|- age: integer
|- gender: string
+- phoneNumber: record
|   |- areaCode: integer
|   |- number: integer
+- children: record (repeated)
|   |- name: string
|   |- gender: string
|   |- age: integer
+- citiesLived: record (repeated)
|   |- place: string
|   +- yearsLived: integer (repeated)
```

Schema with flat, nested, and repeated fields

Google Cloud Platform

Notes:
fullName, age, and gender are flat fields. phoneNumber is a nested field containing the areaCode and number fields (indicated by the record type). children and citiesLived are nested and repeated (indicated by the record (repeated) type).



Schema Specification

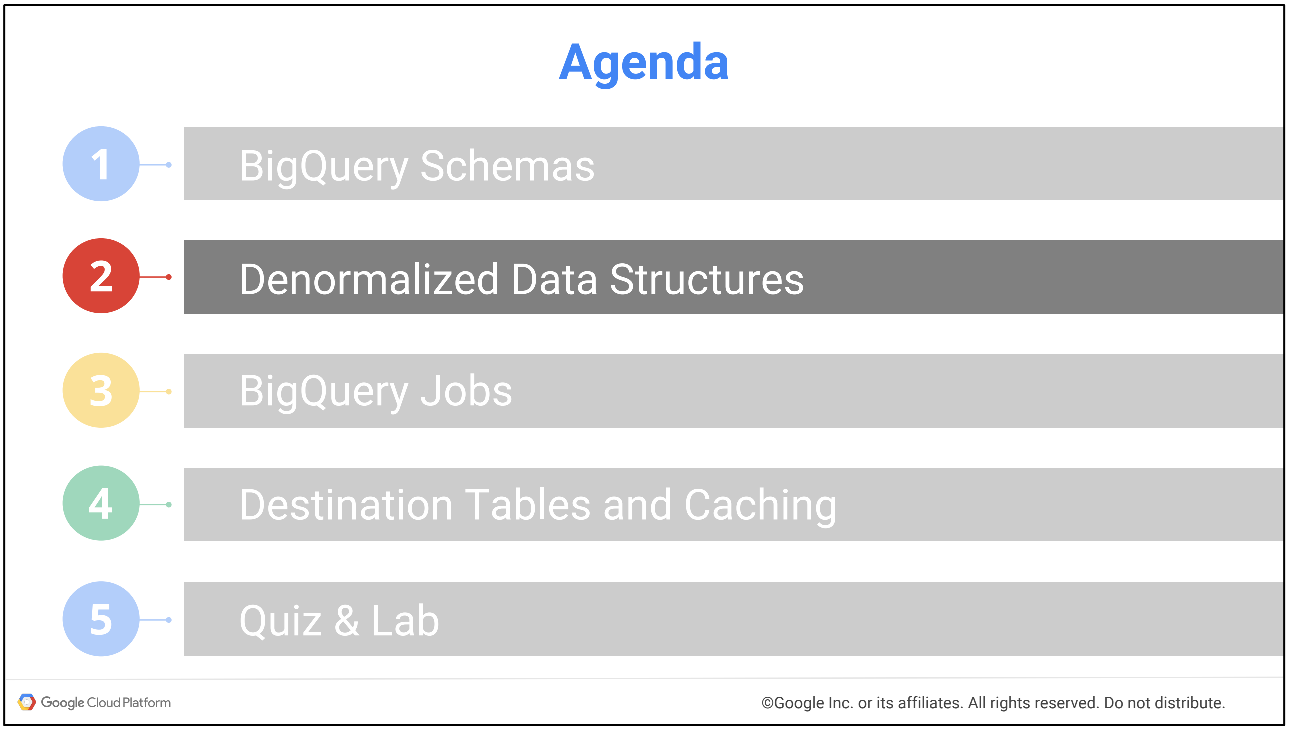
- Specify schema on command line or in JSON file
- Schema file must contain single array object with entries that provide these properties:
  - "name": Name of the column
  - "type": Type of data
  - "mode" (optional): REQUIRED, NULLABLE or REPEATED
- Example JSON schema:

```
{"name": "name", "type": "string", "mode": "required"},
{"name": "gender", "type": "string", "mode": "nullable"},
{"name": "count", "type": "integer", "mode": "required"}
```

Notes:
Once your table is created, you can update the schema by calling the tables.update or tables.patch functions. For example, using the BigQuery command-line tool: bq update -t <*schema*>.

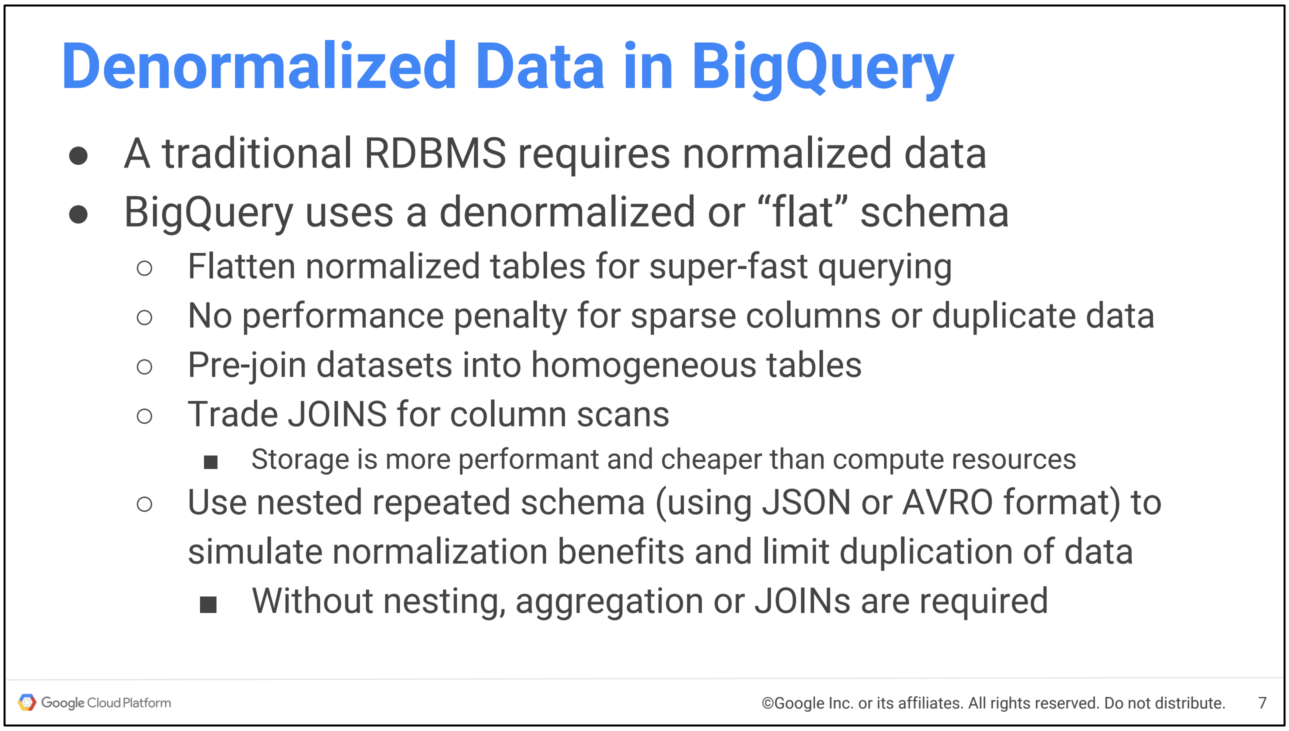
Allowed operations include:

- Adding NULLABLE or REPEATED columns at the end
- Making REQUIRED fields NULLABLE



Agenda

1. BigQuery Schemas
2. **Denormalized Data Structures**
3. BigQuery Jobs
4. Destination Tables and Caching
5. Quiz & Lab



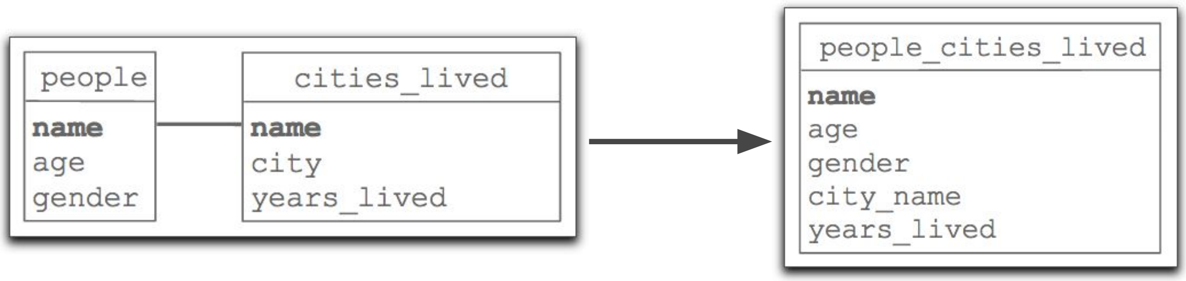
Denormalized Data in BigQuery

- A traditional RDBMS requires normalized data
- BigQuery uses a denormalized or “flat” schema
  - Flatten normalized tables for super-fast querying
  - No performance penalty for sparse columns or duplicate data
  - Pre-join datasets into homogeneous tables
  - Trade JOINS for column scans
    - Storage is more performant and cheaper than compute resources
  - Use nested repeated schema (using JSON or AVRO format) to simulate normalization benefits and limit duplication of data
    - Without nesting, aggregation or JOINs are required

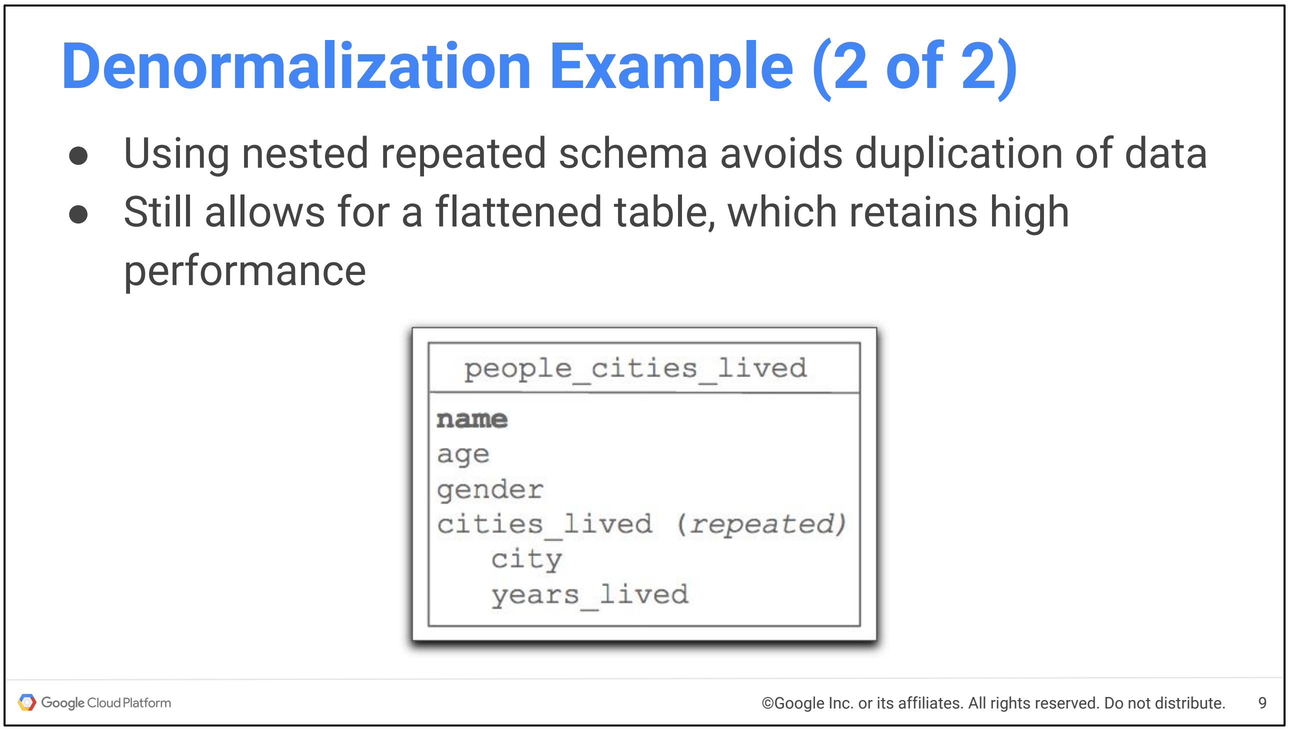
Notes:
BigQuery performs best using a denormalized schema and table structure. Joins to other tables are supported and large JOINs are fast in BigQuery, but JOINs are not be as fast as pure scans. Thus, if performance optimization is important, it's recommended to pre-JOIN and trade JOINs for scans.

Denormalization Example (1 of 2)

- A normalized one-to-many relationship between people and cities_lived is denormalized ("flattened") into 1 table
  - A person may have one to many rows in the table
  - Name, age, and gender may be duplicated

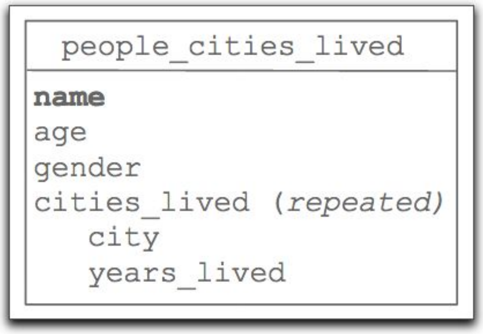


Notes:
This format allows for high performance querying in BigQuery. However, the query may have to account for duplicated data in the first 3 columns. This schema can be loaded in either CSV or JSON format.



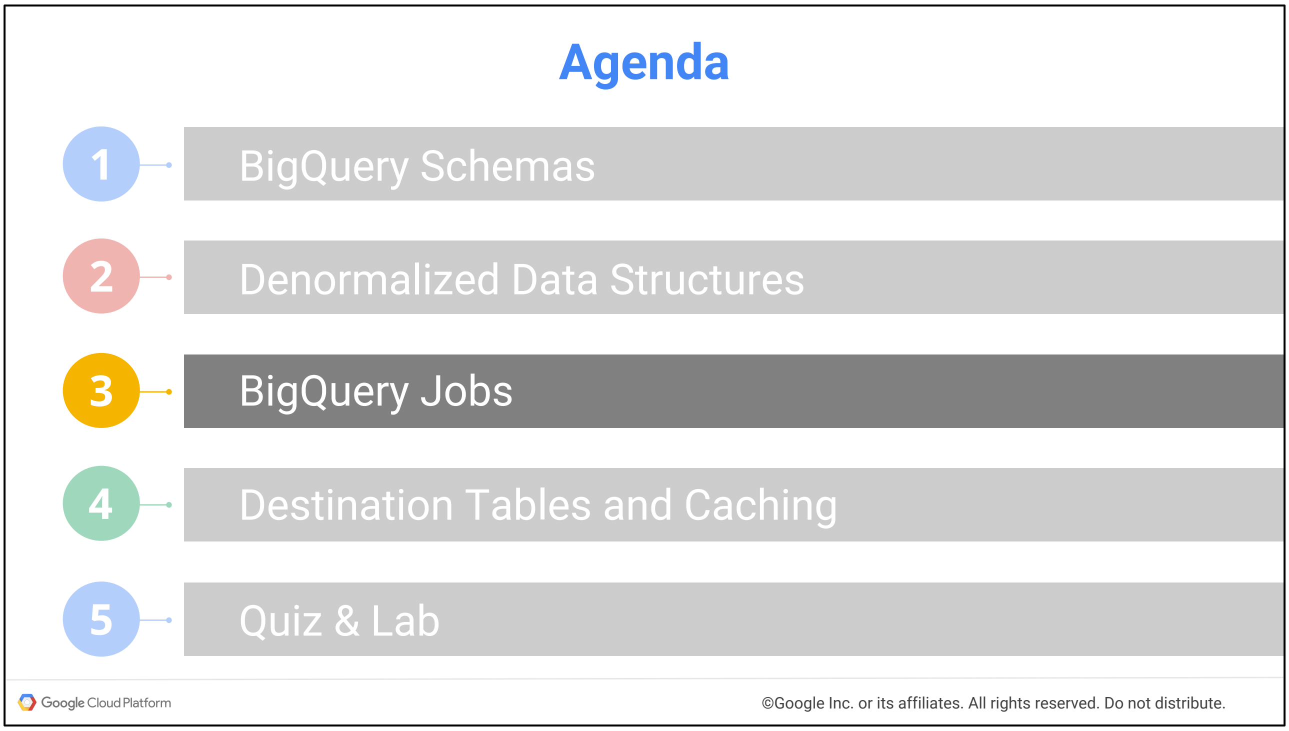
Denormalization Example (2 of 2)

- Using nested repeated schema avoids duplication of data
- Still allows for a flattened table, which retains high performance



```
people_cities_lived
name
age
gender
cities_lived (repeated)
    city
    years_lived
```

Notes:
This is the optimal solution in BigQuery in that it flattens the data for high performance querying, removes duplication, and keeps a natural format for understanding the data record. The nested repeated schema would require the data to be loaded in JSON format.



Agenda

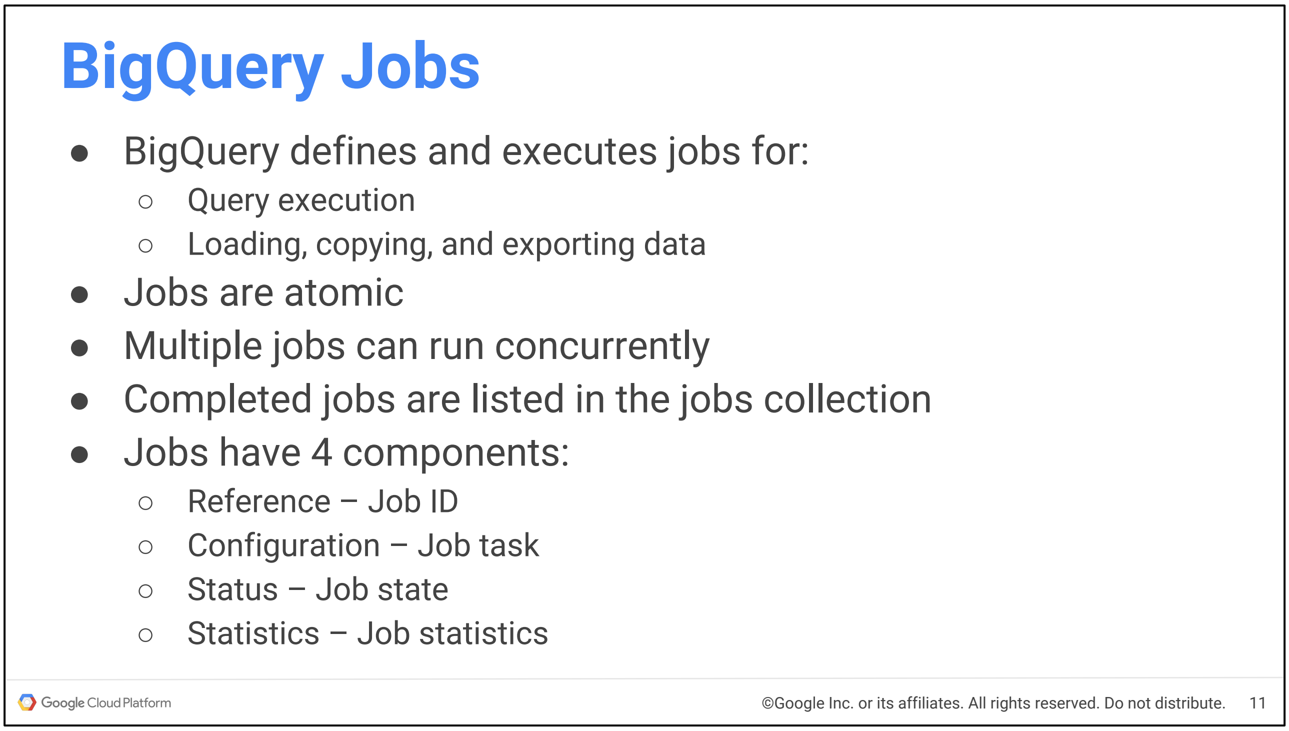
1 BigQuery Schemas

2 Denormalized Data Structures

3 BigQuery Jobs

4 Destination Tables and Caching

5 Quiz & Lab



BigQuery Jobs

- BigQuery defines and executes jobs for:
  - Query execution
  - Loading, copying, and exporting data
- Jobs are atomic
- Multiple jobs can run concurrently
- Completed jobs are listed in the jobs collection
- Jobs have 4 components:
  - Reference – Job ID
  - Configuration – Job task
  - Status – Job state
  - Statistics – Job statistics

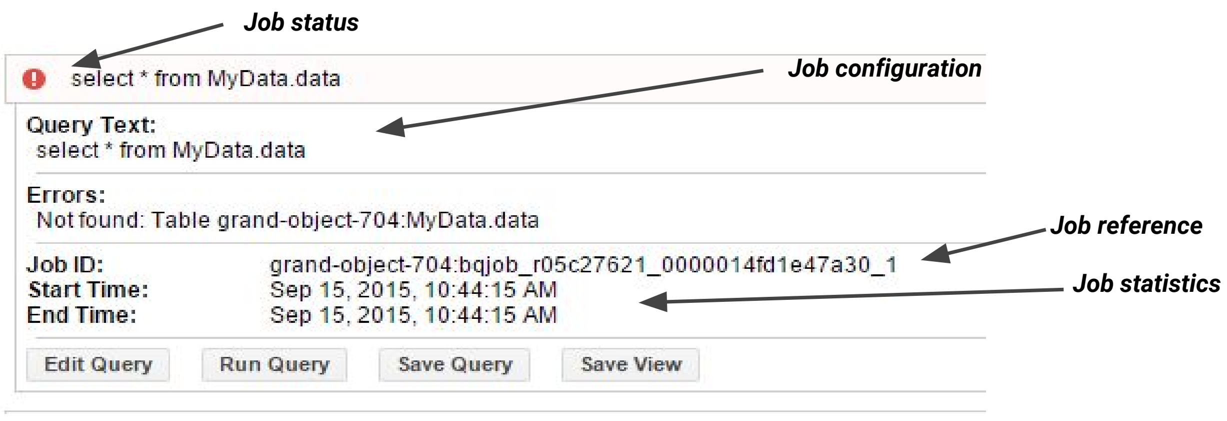
Google Cloud Platform

Notes:
Processing data (be it load, query, or export) creates a job internally in BigQuery. All jobs are asynchronous in nature. Multiple jobs can run concurrently up to the quota limit. All jobs are listed in a jobs collection per project.

Please note that jobs are atomic. Meaning:
- There are no half-way state or race conditions in jobs - once a job is done, state flips over immediately (imagine doing a table copy into a table that exists)
- If a job fails, all of it fails, so you don't need to clean up failed jobs (exception being streaming API that returns a response object with pertinent info)

Example: Job Components

- Job components in the web UI



select * from MyData.data

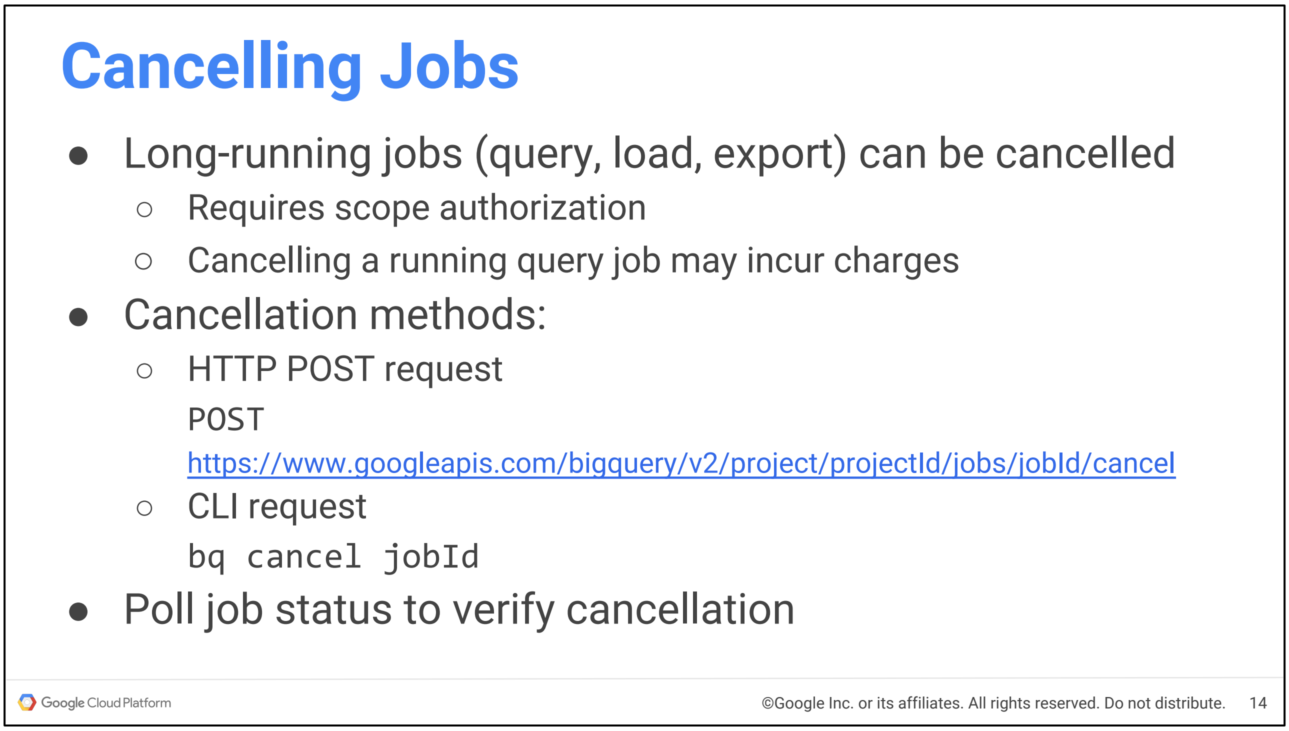
Query Text:
select * from MyData.data

Errors:
Not found: Table grand-object-704:MyData.data

Job ID:	grand-object-704:bqjob_r05c27621_0000014fd1e47a30_1
Start Time:	Sep 15, 2015, 10:44:15 AM
End Time:	Sep 15, 2015, 10:44:15 AM

Job History

- Jobs collection list
  - Stores history of completed jobs by project
  - Availability is only guaranteed for jobs created in past 6 months
  - If needed, contact Google support for automatic deletion of jobs more than 50 days old



Cancelling Jobs

- Long-running jobs (query, load, export) can be cancelled
  - Requires scope authorization
  - Cancelling a running query job may incur charges
- Cancellation methods:
  - HTTP POST request

    POST
    https://www.googleapis.com/bigquery/v2/project/projectId/jobs/jobId/cancel
  - CLI request

    ```
    bq cancel jobId
    ```
- Poll job status to verify cancellation

Notes:
Jobs that perform load, exports, and queries may be canceled. Users must have the required scope authorization in order to cancel jobs. Job canceling can be done either by HTTP post request, the CLI, or the web UI. Cancelling a query job may incur a query charge depending the amount of data processing BigQuery performed at the time of cancel request.

Agenda

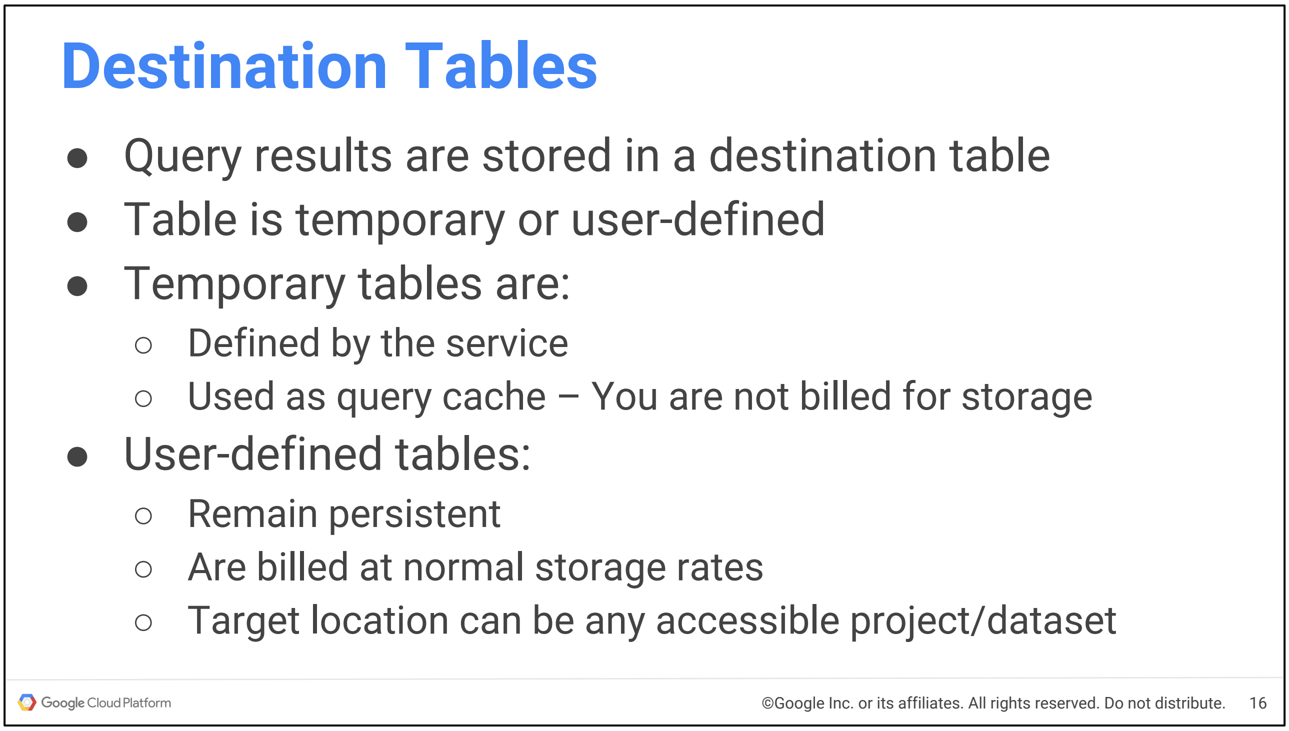
1 BigQuery Schemas

2 Denormalized Data Structures

3 BigQuery Jobs

4 **Destination Tables and Caching**

5 Quiz & Lab

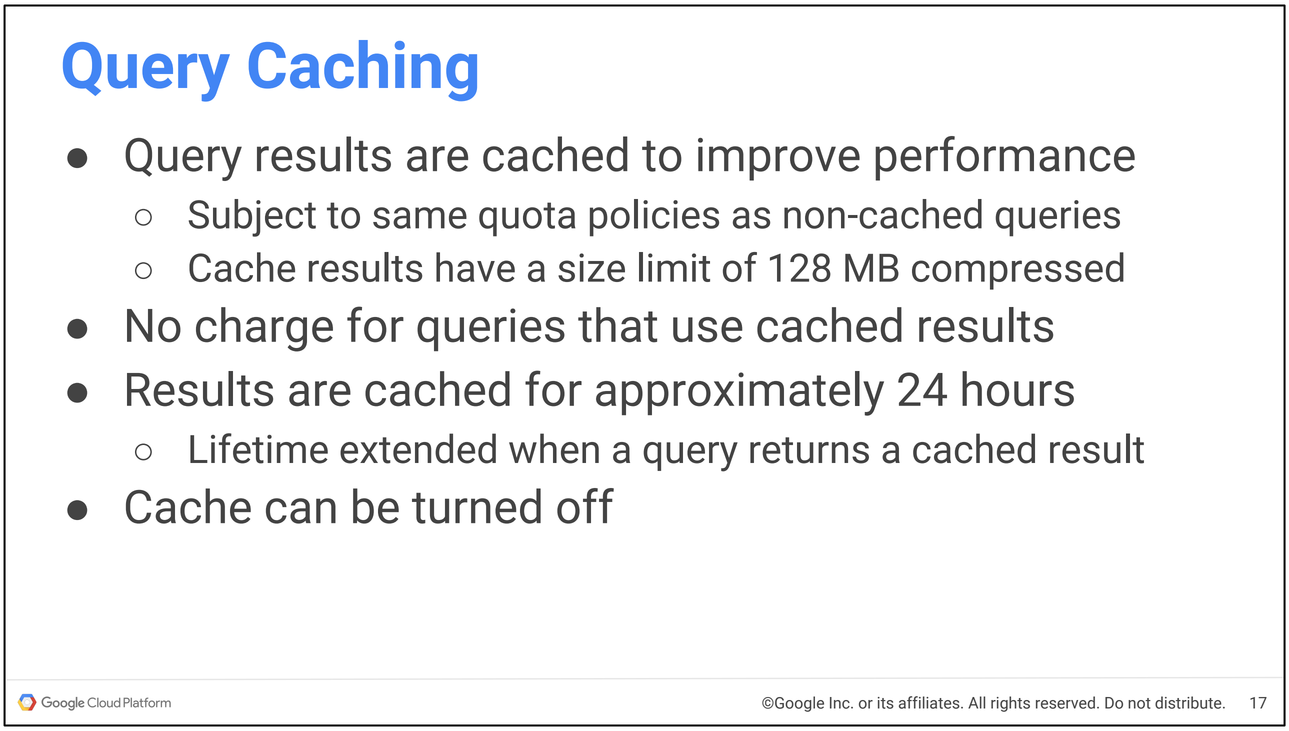


Destination Tables

- Query results are stored in a destination table
- Table is temporary or user-defined
- Temporary tables are:
  - Defined by the service
  - Used as query cache – You are not billed for storage
- User-defined tables:
  - Remain persistent
  - Are billed at normal storage rates
  - Target location can be any accessible project/dataset

Google Cloud Platform

Notes:
A **persistent** table can be a new or existing table in any dataset in which you have WRITE privileges.



Query Caching

- Query results are cached to improve performance
  - Subject to same quota policies as non-cached queries
  - Cache results have a size limit of 128 MB compressed
- No charge for queries that use cached results
- Results are cached for approximately 24 hours
  - Lifetime extended when a query returns a cached result
- Cache can be turned off

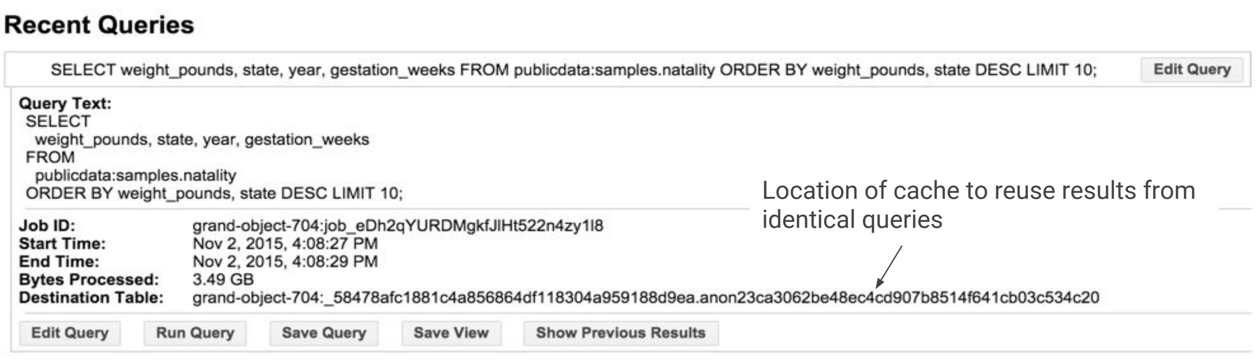
Google Cloud Platform

Notes:
It is important to note that:
- Cache is per-user
- Cache needs predictable queries, so a query that has, say "current_timestamp()" in it, is not cacheable since it changes every time you run it

jobs.getQueryResults is used to page through cached query results in a temporary table. tabledata.list is used when saving the results of an asynchronous query to a permanent table.

Cache Example (1 of 2)

- By default, BigQuery caches the result of this query in a temporary table for future use



Recent Queries

```
SELECT weight_pounds, state, year, gestation_weeks FROM publicdata:samples.natality ORDER BY weight_pounds, state DESC LIMIT 10;
```

Query Text:

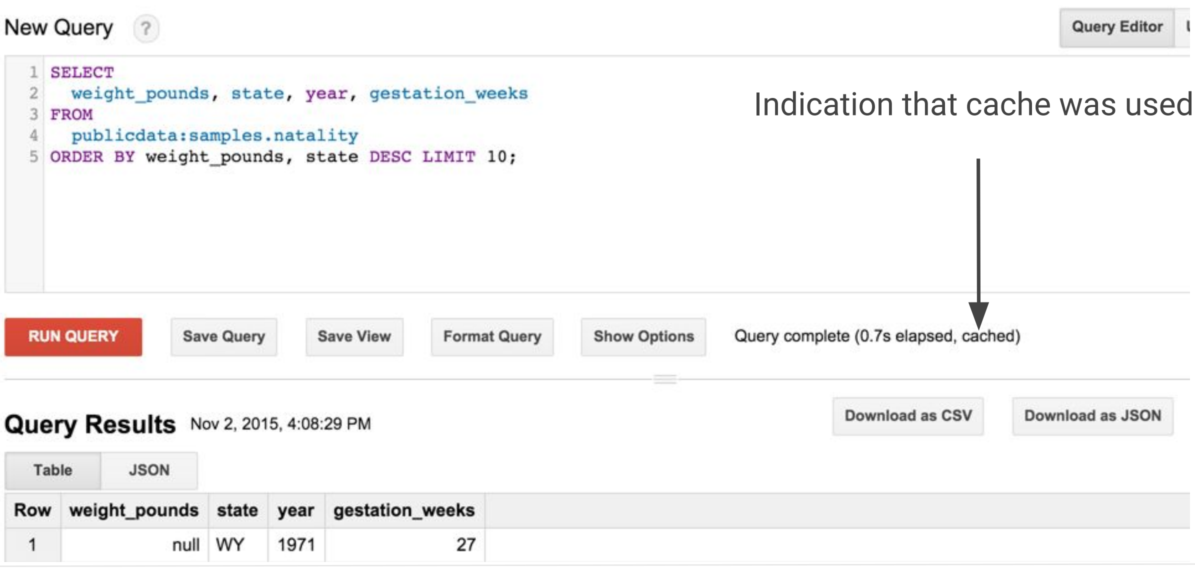
```
SELECT
  weight_pounds, state, year, gestation_weeks
FROM
  publicdata:samples.natality
ORDER BY weight_pounds, state DESC LIMIT 10;
```

Job ID:	grand-object-704:job_eDh2qYURDMgkfJlHt522n4zy1l8
Start Time:	Nov 2, 2015, 4:08:27 PM
End Time:	Nov 2, 2015, 4:08:29 PM
Bytes Processed:	3.49 GB
Destination Table:	grand-object-704:_58478afc1881c4a856864df118304a959188d9ea.anon23ca3062be48ec4cd907b8514f641cb03c534c20

Location of cache to reuse results from identical queries

Notes:
This slide shows that the system created a destination table to store the query results. This table is temporary and remains persistent for 24 hours. BigQuery users are not billed for storage of temporary tables.

Cache Example (2 of 2)



```
SELECT
  weight_pounds, state, year, gestation_weeks
FROM
  publicdata:samples.natality
ORDER BY weight_pounds, state DESC LIMIT 10;
```

Indication that cache was used

Query complete (0.7s elapsed, cached)

Query Results Nov 2, 2015, 4:08:29 PM

Row	weight_pounds	state	year	gestation_weeks
1	null	WY	1971	27

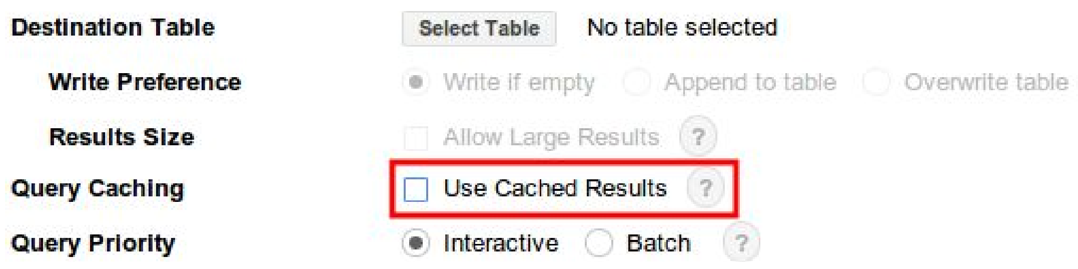
Notes:
If query cache is turned on, BigQuery verifies that the results from a query are identical to the existing results from a prior query job. If verification is successful, it hits the cache, and returns the existing result set, if the cache still persists. The user is not charged for cached queries.

Caveats: Query Caching

- Query caching only works for predictable queries
  - When result set is identical to previous run(s) of a query
- Cache is per-user
  - Not shareable across users
- Cache is invalidated if:
  - Data in underlying table(s) changes
  - Query uses dynamic functions, such as `CURRENT_TIMESTAMP()` or `NOW()`

Disabling Caching

- To disable caching in the web UI:
  - Click 'Enable Options' and deselect 'Use Cached Results'



Destination Table	Select Table No table selected
Write Preference	[✓ Write if empty] Append to table Overwrite table
Results Size	☐ Allow Large Results ?
Query Caching	☐ Use Cached Results ?
Query Priority	[✓ Interactive] Batch ?

- To disable caching using the CLI:
  - Use the `useQueryCache= false` flag

```
$bq -jobproperty useQueryCache= false query “select * from
MyData.Data”
```

Agenda

1. BigQuery Schemas
2. Denormalized Data Structures
3. BigQuery Jobs
4. Destination Tables and Caching
5. Quiz & Lab

Module Review (1 of 2)

Which of the following is true about BigQuery data organization and data structures?
*(select **1** of the available options)*

- ❑ A BigQuery schema is loosely typed
- ❑ A BigQuery schema requires string lengths to be defined
- ❑ There is no limit on the number of tables contained in a dataset
- ❑ Only load and export jobs are asynchronous
- ❑ Canceling jobs eliminates any charges associated with the job

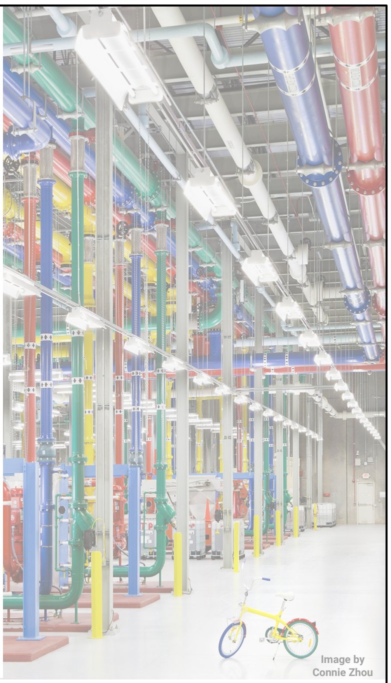
Module Review (2 of 2)

Which of the following statements is true?
*(select **1** of the available options)*

- ❑ Query results are always stored in a temporary table
- ❑ BigQuery performs best with tables that are normalized
- ❑ Query caching is default and cannot be turned off
- ❑ You are not charged for queries that get results from the cache

Lab

Examine BigQuery components and jobs



Resources

- Jobs
  https://cloud.google.com/bigquery/docs/reference/v2/jobs/jobs
- Managing tables
  https://cloud.google.com/bigquery/docs/tables

Module Review Answers (1 of 2)

Which of the following is true about BigQuery data organization and data structures?
*(select **1** of the available options)*

- ❏ A BigQuery schema is loosely typed
- ❏ A BigQuery schema requires string lengths to be defined
- ✓ There is no limit on the number of tables contained in a dataset
- ❏ Only load and export jobs are asynchronous
- ❏ Canceling jobs eliminates any charges associated with the job

Module Review Answers (2 of 2)

Which of the following statements is true?
*(select **1** of the available options)*

- ❏ Query results are always stored in a temporary table
- ❏ BigQuery performs best with tables that are normalized
- ❏ Query caching is default and cannot be turned off
- ✓ You are not charged for queries that get results from the cache

cloud.google.com