

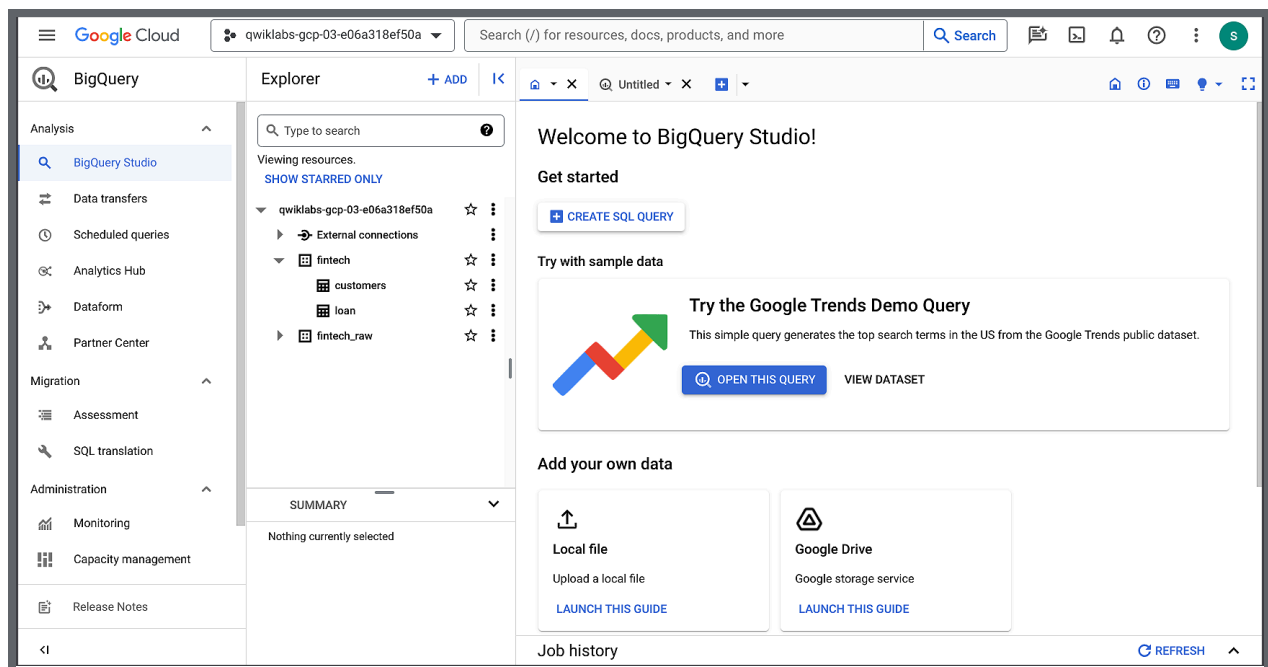
Solution Guide: Collect, process, and store data in BigQuery

The **Collect, process, and store data in Big Query lab** is a portion of the capstone project that puts your data analysis skills to the test; the lab includes a set of tasks and challenges that involve transforming data within a business scenario. Each task in the lab guides you to apply the skills you learned throughout the course, focusing on data collection, processing, and storage within the BigQuery environment. The lab also requires you to tackle two challenges to assess your skills on your own: transform data and create a report.

This solution guide provides the results of each guided task in the lab for you to assess against your own work. It also includes the solution query and results for the two challenges so that you may evaluate your approach, as well as identify potential areas for improvement.

Task 1: Get started with BigQuery

To complete this task, open up the BigQuery environment. Select the project that matches the **Google Cloud project ID** provided during login, and locate the fintech dataset in the **Explorer** pane.



Task 2: Explore the Fintech data

To complete this task, expand the fintech dataset to view the `customers` and `loans` table. Then, click on each name to select the table and review the **Details**, **Preview**, and **Schema** tabs.

The screenshot shows the Google Cloud BigQuery interface. On the left is the navigation menu with categories like Analysis, Migration, and Administration. The central Explorer pane shows the project hierarchy: `qwiklabs-gcp-03-e06a318ef50a` > `fintech` > `customers`. The right pane displays the 'customers' table schema in the 'SCHEMA' tab. Below is a table of the schema details:

Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Description
<code>customer_id</code>	BYTES	NULLABLE	-	-	-	-	-
<code>emp_title</code>	STRING	NULLABLE	-	-	-	-	-
<code>emp_length</code>	STRING	NULLABLE	-	-	-	-	-
<code>home_ownership</code>	STRING	NULLABLE	-	-	-	-	-
<code>annual_inc</code>	FLOAT	NULLABLE	-	-	-	-	-
<code>annual_inc_joint</code>	FLOAT	NULLABLE	-	-	-	-	-
<code>verification_status</code>	STRING	NULLABLE	-	-	-	-	-
<code>zip_code</code>	STRING	NULLABLE	-	-	-	-	-
<code>addr_state</code>	STRING	NULLABLE	-	-	-	-	-
<code>avg_cur_bal</code>	NUMERIC	NULLABLE	-	-	-	-	-
<code>Tot_cur_bal</code>	NUMERIC	NULLABLE	-	-	-	-	-

Buttons at the bottom include 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'. The 'Job history' section is visible at the very bottom.

Task 3: Import a CSV file and create a standard table

To complete this task, run the provided code in the query editor to import a CSV file from Cloud Storage. Then, review the table using the preview tab. A new table called `state_region` will be added to the `fintech` dataset.

The screenshot shows the Google Cloud BigQuery Explorer interface. The left sidebar contains navigation options like Analysis, Migration, and Administration. The main pane displays the 'state_region' table with columns: Row, state, subregion, and region. The table contains 18 rows of data, showing various US states and their corresponding subregions and regions.

Row	state	subregion	region
1	AK	Pacific	West
2	CA	Pacific	West
3	HI	Pacific	West
4	OR	Pacific	West
5	WA	Pacific	West
6	AZ	Mountain	West
7	CO	Mountain	West
8	ID	Mountain	West
9	MT	Mountain	West
10	NM	Mountain	West
11	NV	Mountain	West
12	UT	Mountain	West
13	WY	Mountain	West
14	DC	South Atlantic	South
15	DE	South Atlantic	South
16	FL	South Atlantic	South
17	GA	South Atlantic	South
18	MD	South Atlantic	South

Task 4: Join data from two tables

To complete this task, run **Query B** in the **Query Editor** to join the two tables and review the results in the **Query results** panel.

The screenshot shows the Google Cloud BigQuery Query Editor. The left sidebar is the same as the previous image. The main pane shows a SQL query in 'Untitled 3' that joins the 'loan' table from the 'fintech' dataset with the 'state_region' table from the 'fintech' dataset. The query results are displayed in a table with columns: Row, loan_id, loan_amount, and region. The results show 10 rows of data.

```

1 SELECT
2   lo.loan_id,
3   lo.loan_amount,
4   sr.region
5 FROM fintech.loan lo
6 INNER JOIN fintech.state_region sr
7 ON lo.state = sr.state;
  
```

Row	loan_id	loan_amount	region
1	73458	8725	West
2	80520	9600	West
3	170191	16425	West
4	141363	14075	West
5	19961	4000	West
6	33799	5000	West
7	98488	10000	West
8	47389	6075	West
9	7835	2500	West
10	161268	15875	West

Task 5: Create a table based on the results of a query using CTAS

To complete this task, run the provided query in the **Query Editor** to create a new table named `loan_with_region`. The new table appears in the `fintech` dataset. Then, export the data to Google Sheets to review the `loan_with_region` data.

Note: If the export to Google Sheets fails, an error will appear stating that Google Sheets will not open. Try exporting the data again.

The screenshot shows a Google Sheets interface with a table named 'loan_with_region' containing 270K rows. The table has three columns: 'loan_id', 'loan_amount', and 'region'. The data is displayed in a preview mode.

loan_id	loan_amount	region
211104	22400	West
211045	22400	West
211219	22400	West
211116	22400	West
210970	22400	West
211224	22400	West
210981	22400	West
211051	22400	West
211031	22400	West
210984	22400	West
211181	22400	West
211200	22400	West
210980	22400	West
211238	22400	West
211050	22400	West

Task 6: Work with nested data

To complete this task, use dot notation to query the `purpose` column, which is nested inside of the `application` record. The results of the query will be a table with two columns: `loan_id` and `purpose`.

The screenshot shows the Google Cloud BigQuery interface. The Explorer pane on the left shows the project structure with a dataset named 'fintech' containing a table named 'loan'. The Query Editor on the right shows a query that selects the 'purpose' column from the 'loan' table. The query results are displayed in a table with 14 rows and 3 columns: 'loan_id', 'application', and 'purpose'.

Row	loan_id	application	purpose
1	73458	wedding	wedding
2	81846	wedding	wedding
3	123862	wedding	wedding
4	15001	wedding	wedding
5	61267	wedding	wedding
6	256460	renewable_energy	renewable_energy
7	34182	renewable_energy	renewable_energy
8	137776	renewable_energy	renewable_energy
9	116571	renewable_energy	renewable_energy
10	45239	renewable_energy	renewable_energy
11	98705	wedding	wedding
12	135494	renewable_energy	renewable_energy
13	98709	wedding	wedding
14	3049	wedding	wedding

Task 7 challenge: Deduplicate

To complete this challenge, write a query to create a table named `fintech.loan_purposes` that has a single column named `purpose`. The `purpose` column should include the unique results found in the nested `purpose` column in the `loan` table of the `fintech` dataset.

Solution

You can use a **Create Table as Select (CTAS)** statement to create the table and dot notation to select the `purpose` column that is nested in the application record.

Here's the query:

```
Unset
CREATE TABLE fintech.loan_purposes AS
SELECT DISTINCT application.purpose
FROM fintech.loan;
```

1. Copy and paste the above query into the **Query Editor**.
2. Click **Run**.

As a result of this query, a new table named `loan_purposes` is added to the `fintech` dataset. This table has one column that selects the distinct values from the `purpose` column within the application record of the `loan` table.

Viewing resources.

SHOW STARRED ONLY

- ▼ qwiklabs-gcp-03-09db93772688
 - ▶ External connections
 - ▼ fintech
 - customers
 - loan
 - loan_purposes**
 - loan_with_region
 - state_region
 - ▶ fintech_raw

SCHEMA		DETAILS	PREVIEW
Row	purpose		
1	debt_consolidation		
2	major_purchase		
3	small_business		
4	other		
5	wedding		
6	vacation		
7	renewable_energy		
8	moving		
9	car		
10	medical		
11	home_improvement		
12	house		
13	credit_card		

Task 8 challenge: Answer business questions with a report

To complete this challenge, write a query to create a table called `loan_count_by_year` in the `fintech` dataset that counts loans grouped by `issue_year`.

Solution

You can use a **Create Table as Select (CTAS)** statement to create the table and **COUNT** and **GROUP BY** to count the loans and group them by `issue_year`.

Here's the query:

Unset

```
CREATE TABLE fintech.loan_count_by_year AS
SELECT issue_year, count(loan_id) AS loan_count
FROM fintech.loan
GROUP BY issue_year;
```

1. Copy and paste the above query into the **Query Editor**.
2. Click **Run**.

As a result of this query, a new table named `loan_count_by_year` is added to the `fintech` dataset. This table has two columns: `issue_year` and `loan_count`. The `loan_count` column counts the number of loans by issue year.

Viewing resources.			SCHEMA	DETAILS	PREVIEW
SHOW STARRED ONLY			Row	issue_year	loan_count
▼	qwiklabs-gcp-03-09db93772688	☆ ⋮	1	2013	13460
▶	External connections	⋮	2	2017	44435
▼	fintech	☆ ⋮	3	2018	49333
	customers	☆ ⋮	4	2019	51737
	loan	☆ ⋮	5	2016	43368
	loan_count_by_year	☆ ⋮	6	2015	41919
	loan_purposes	☆ ⋮	7	2012	2594
	loan_with_region	☆ ⋮	8	2014	23453
	state_region	☆ ⋮			
▶	fintech_raw	☆ ⋮			

Resources for more information

Use these readings to help support you as you work through the solution:

- **SQL query terms reading** available in course 1 module 1
- **Guide to BigQuery reading** available in course 2 module 1