# Activity Exemplar: Partition data and create indexes in BigQuery

Here is a completed exemplar along with an explanation of how the exemplar fulfills the expectations for the activity.

## Assessment of Exemplar

Compare the exemplar to your completed activity. Review your work using each of the criteria in the exemplar. What did you do well? Where can you improve? Use your answers to these questions to guide you as you continue to progress through the course.

In the previous activity, you ran SQL code that created tables with partitions and indexes. Partitions and indexes help you create shortcuts to specific rows and divide large datasets into smaller, more manageable tables. By creating partitions and indexes, you can build faster and more efficient databases, making it easier to pull data when you need to analyze or visualize it.

After creating the tables, you ran queries on those tables to compare their performance and demonstrate how useful partitions and indexes can be.

At each step, you took screenshots of the **Details** or **Execution Details** pane to compare to the following exemplar images. This will help you ensure that you completed the activity properly. It will also explain the context of why the tables you created and the queries you ran differ from each other. By the end of this reading, you will understand how this activity demonstrates that partitions and clusters speed up queries and optimize database performance.

Note that the answers for these queries might differ depending on whether you're using the sandbox or free trial/full version of BigQuery. The sandbox version might not read the full dataset, so the table size you receive might not match the results you would get from the query in the full version. This reading explains the results for both the sandbox and the full version so you can check your work regardless of how you're using BigQuery.

## Explore the exemplar

### Table details

This is the **Details** pane for the table you created without partitions or indexes. It simply describes the table size (4.37MB of logical and active bytes) and the number of rows (41,025).

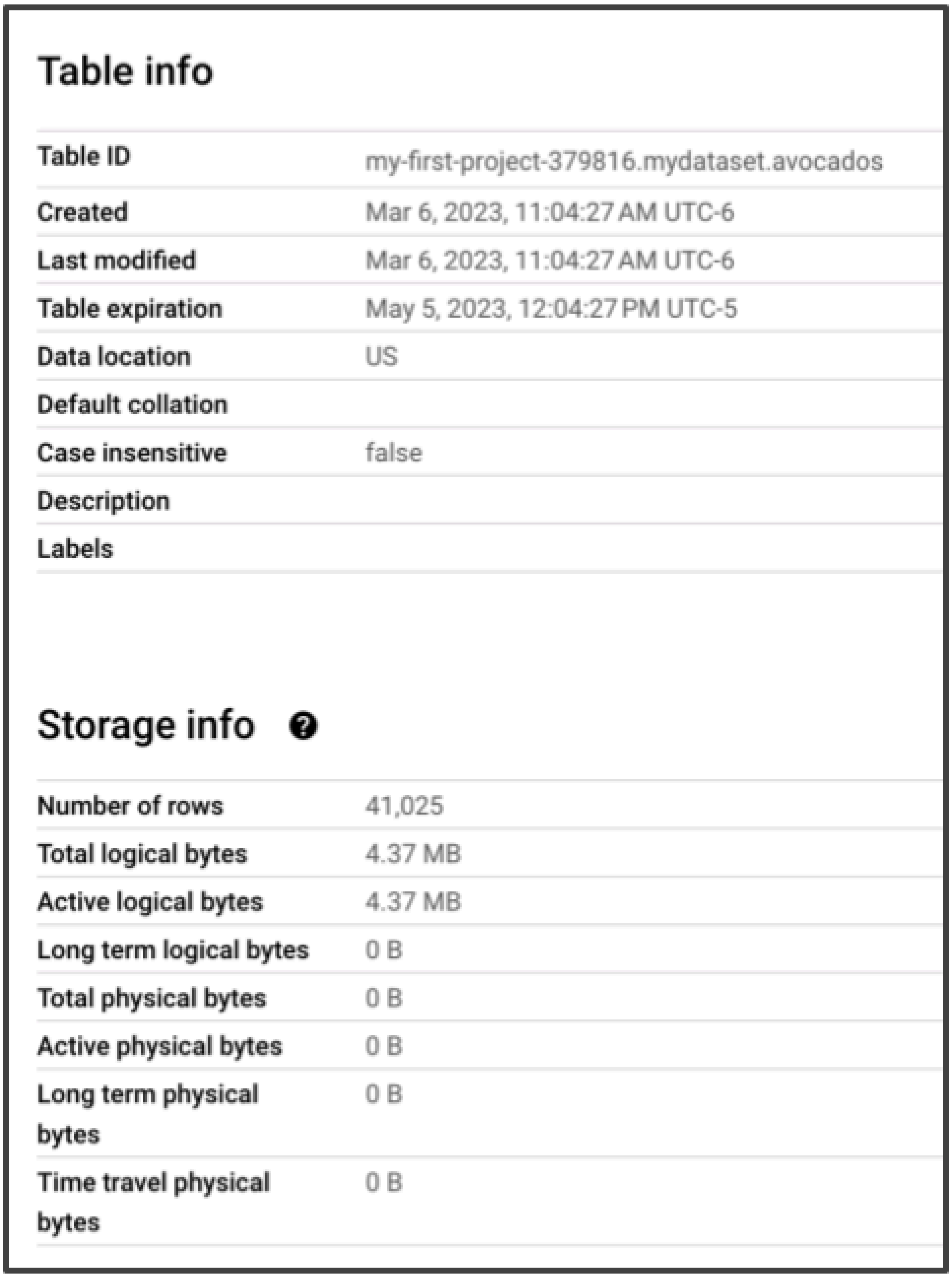


Table info

| | |
|---|---|
| Table ID | my-first-project-379816.mydataset.avocados |
| Created | Mar 6, 2023, 11:04:27 AM UTC-6 |
| Last modified | Mar 6, 2023, 11:04:27 AM UTC-6 |
| Table expiration | May 5, 2023, 12:04:27 PM UTC-5 |
| Data location | US |
| Default collation | |
| Case insensitive | false |
| Description | |
| Labels | |

Storage info

| | |
|---|---|
| Number of rows | 41,025 |
| Total logical bytes | 4.37 MB |
| Active logical bytes | 4.37 MB |
| Long term logical bytes | 0 B |
| Total physical bytes | 0 B |
| Active physical bytes | 0 B |
| Long term physical bytes | 0 B |
| Time travel physical bytes | 0 B |

This is the **Details** pane for the table you created with a partition. It has the same details as the first details pane, but it also includes details about table type (partitioned), as well as the field on which the partition was created (year).

The sandbox limitations mean that this table won’t have a size, but it will still be created with the query. The table size is 0B and there is a section that includes “Table Type: Partitioned,” “Partitioned by: Integer Range,” “Partitioned on field: year,” and “Partition filter: Not required.” The partition range start (2015), end (2022) and interval is also shown. The full version has a table size of 4.37MB, but it has the same additional partition section.

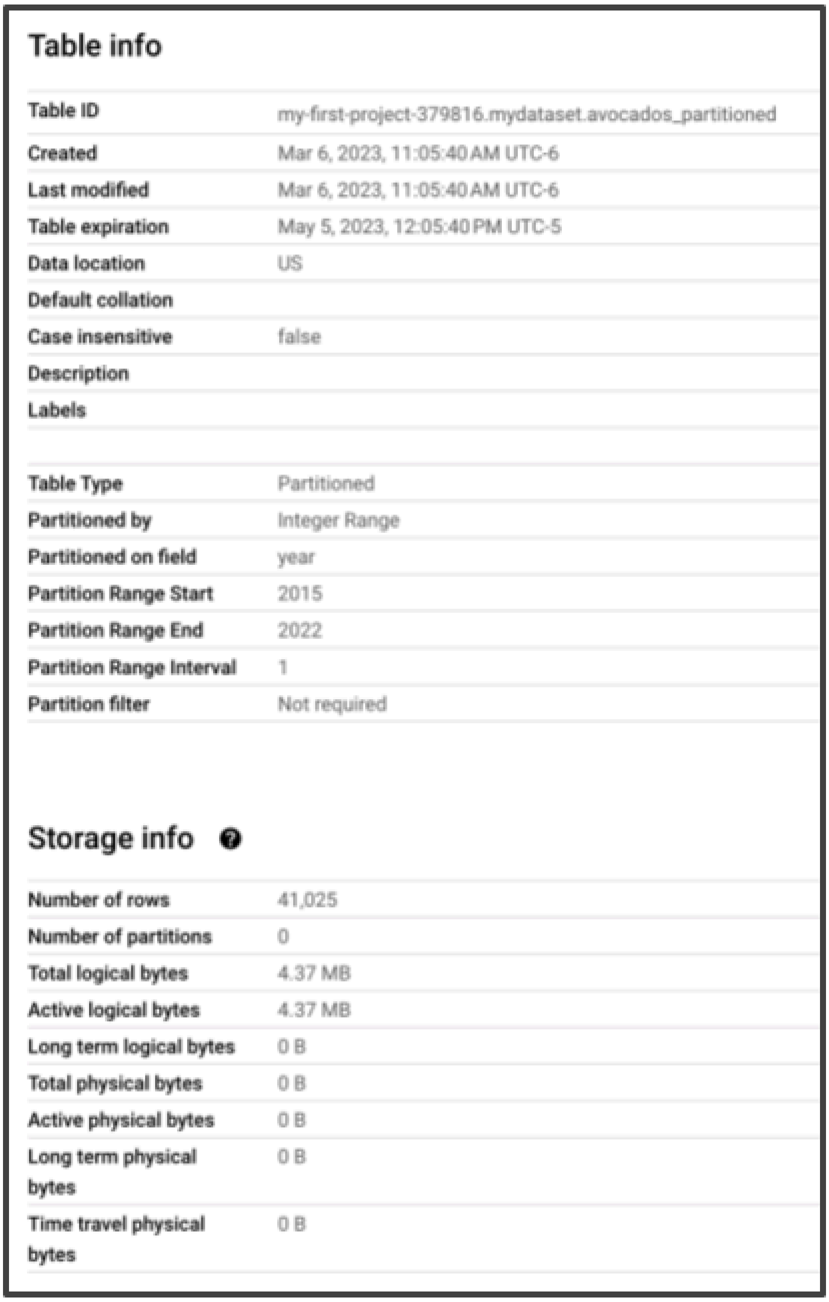


Table info

| | |
|---|---|
| Table ID | my-first-project-379816.mydataset.avocados_partitioned |
| Created | Mar 6, 2023, 11:05:40 AM UTC-6 |
| Last modified | Mar 6, 2023, 11:05:40 AM UTC-6 |
| Table expiration | May 5, 2023, 12:05:40 PM UTC-5 |
| Data location | US |
| Default collation | |
| Case insensitive | false |
| Description | |
| Labels | |

| | |
|---|---|
| Table Type | Partitioned |
| Partitioned by | Integer Range |
| Partitioned on field | year |
| Partition Range Start | 2015 |
| Partition Range End | 2022 |
| Partition Range Interval | 1 |
| Partition filter | Not required |

Storage info

| | |
|---|---|
| Number of rows | 41,025 |
| Number of partitions | 0 |
| Total logical bytes | 4.37 MB |
| Active logical bytes | 4.37 MB |
| Long term logical bytes | 0 B |
| Total physical bytes | 0 B |
| Active physical bytes | 0 B |
| Long term physical bytes | 0 B |
| Time travel physical bytes | 0 B |

This is the **Details** pane for the table you created with a partition and clusters. It has the same details as the two previous images, but also includes that you clustered the data by the **type** column. The sandbox version of the table might not have a table size, but the full version has 4.37MB total logical and active bytes.

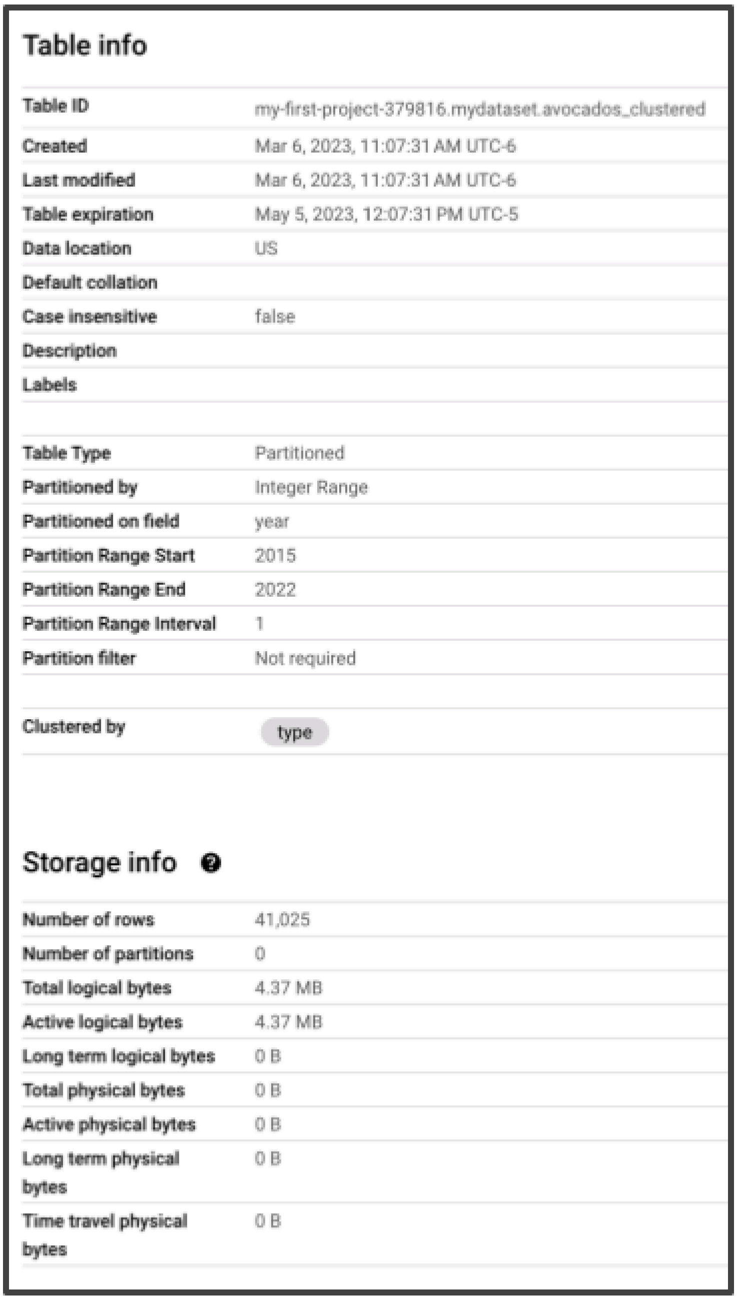


Table info

| | |
|---|---|
| Table ID | my-first-project-379816.mydataset.avocados_clustered |
| Created | Mar 6, 2023, 11:07:31 AM UTC-6 |
| Last modified | Mar 6, 2023, 11:07:31 AM UTC-6 |
| Table expiration | May 5, 2023, 12:07:31 PM UTC-5 |
| Data location | US |
| Default collation | |
| Case insensitive | false |
| Description | |
| Labels | |
| Table Type | Partitioned |
| Partitioned by | Integer Range |
| Partitioned on field | year |
| Partition Range Start | 2015 |
| Partition Range End | 2022 |
| Partition Range Interval | 1 |
| Partition filter | Not required |
| Clustered by | type |

Storage info ❓

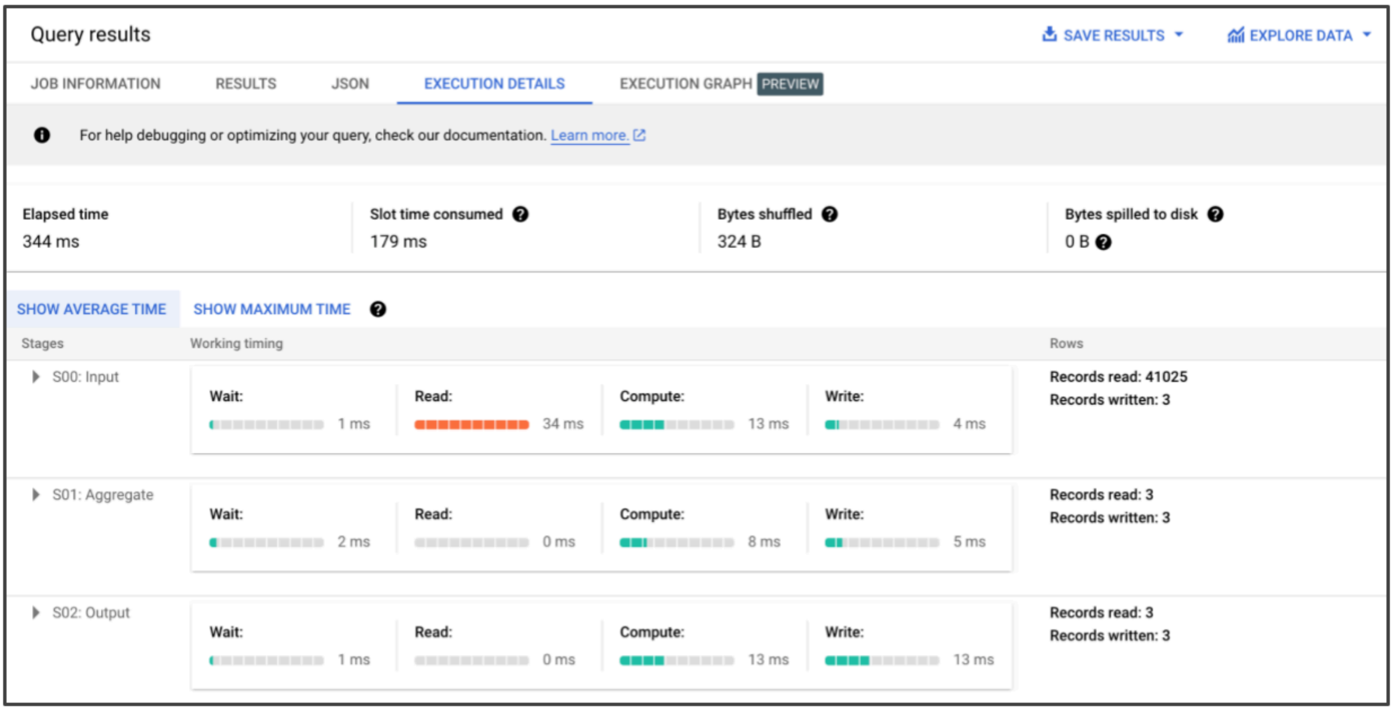
| | |
|---|---|
| Number of rows | 41,025 |
| Number of partitions | 0 |
| Total logical bytes | 4.37 MB |
| Active logical bytes | 4.37 MB |
| Long term logical bytes | 0 B |
| Total physical bytes | 0 B |
| Active physical bytes | 0 B |
| Long term physical bytes | 0 B |
| Time travel physical bytes | 0 B |

Execution details

Then, the **Execution Details** panes compare the query performance for each table. In the sandbox, these details won't appear for queries on the partitioned and partitioned and clustered tables. If you're using the sandbox, take note of the screenshots in the section.

**Note**: The **Working timing** section on your screen might vary in color or duration. Your SQL query might take longer or shorter to run depending on differing BigQuery engine server speeds. Your screen might not match the following screenshot, but the records read and records written should match with the **Rows** section.

This is the **Execution Details** pane for the query on the table you created without partitions or clusters. The number of rows read is the total number of the rows on the table. You'll find this in the S00:Input section, where Records read: 41,025 and Records written: 3.

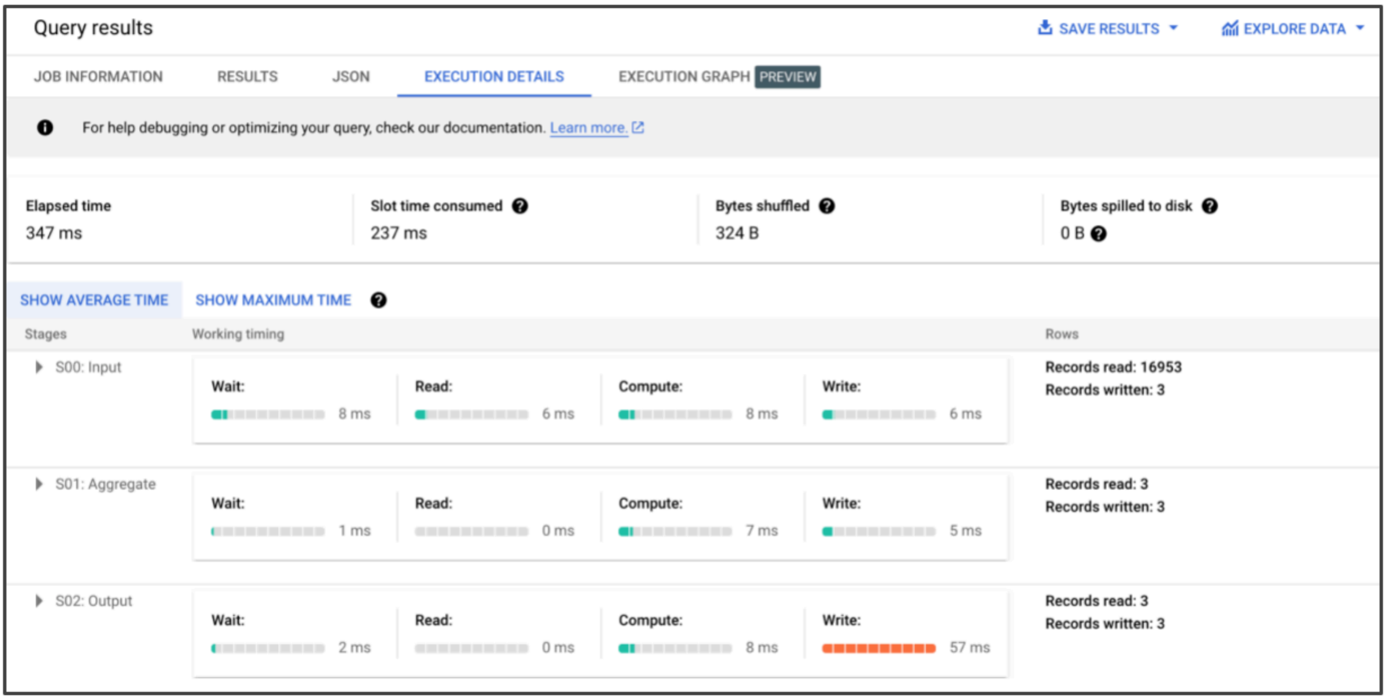


For help debugging or optimizing your query, check our documentation. Learn more.

| Elapsed time | Slot time consumed | Bytes shuffled | Bytes spilled to disk |
|---|---|---|---|
| 344 ms | 179 ms | 324 B | 0 B |

| Stages | Wait | Read | Compute | Write | Rows |
|---|---|---|---|---|---|
| S00: Input | 1 ms | 34 ms | 13 ms | 4 ms | Records read: 41025<br>Records written: 3 |
| S01: Aggregate | 2 ms | 0 ms | 8 ms | 5 ms | Records read: 3<br>Records written: 3 |
| S02: Output | 1 ms | 0 ms | 13 ms | 13 ms | Records read: 3<br>Records written: 3 |

This is the **Execution Details** pane for the query on the table you created partitioned by an **integer range**. You'll notice that the number of records read is less. Now, Records read: 16,953 and Records written: 3. In this query, the database processes only the records from the partitions filtered by the where clause (type). When choosing a column to partition on, it is most effective to choose one that would frequently be used in the where clause.

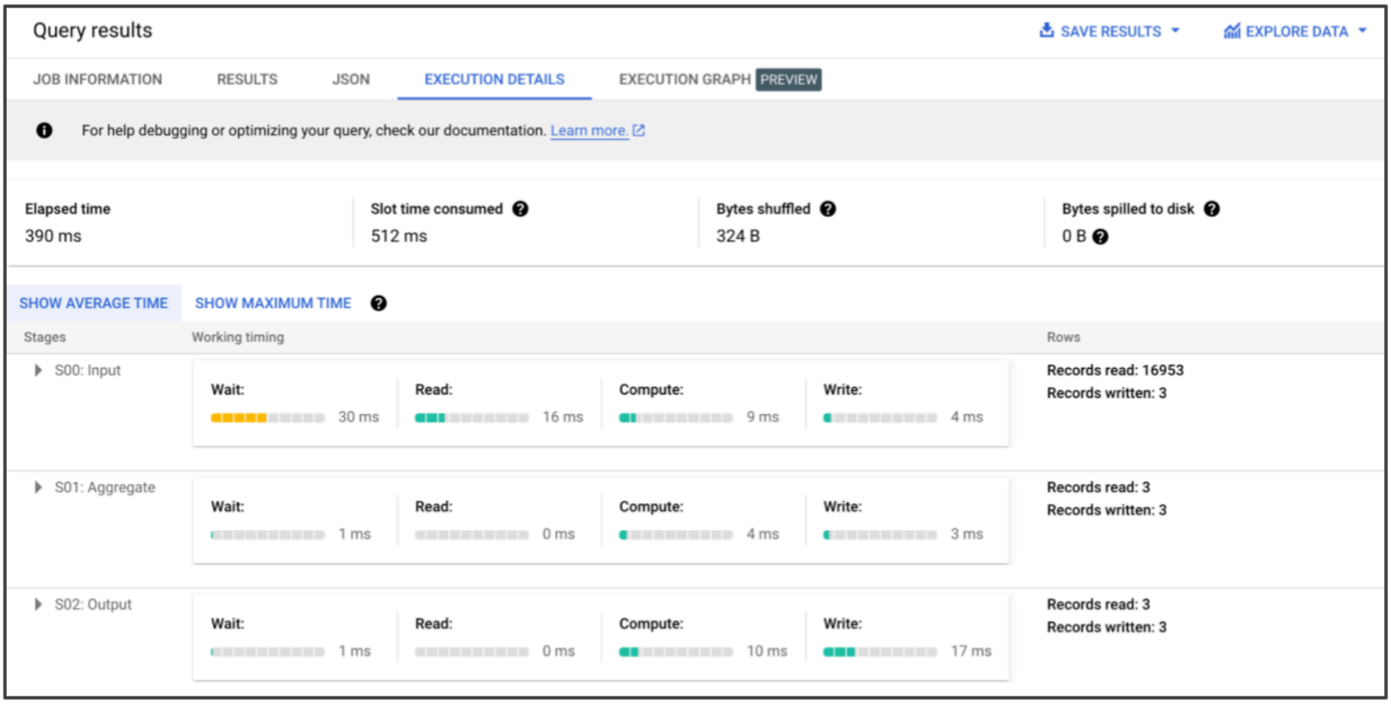


For help debugging or optimizing your query, check our documentation. Learn more.

| Elapsed time | Slot time consumed | Bytes shuffled | Bytes spilled to disk |
|---|---|---|---|
| 347 ms | 237 ms | 324 B | 0 B |

| Stages | Wait | Read | Compute | Write | Rows |
|---|---|---|---|---|---|
| S00: Input | 8 ms | 6 ms | 8 ms | 6 ms | Records read: 16953<br>Records written: 3 |
| S01: Aggregate | 1 ms | 0 ms | 7 ms | 5 ms | Records read: 3<br>Records written: 3 |
| S02: Output | 2 ms | 0 ms | 8 ms | 57 ms | Records read: 3<br>Records written: 3 |

This is the **Execution Details** pane for the query on the table you created that is clustered by the **type** column. Records read: 16,953 and Records written: 3. Typically, a query on the clustered table would process fewer records than the partitioned one. However, the dataset you are using in this activity is too small to properly demonstrate that difference. In other projects, you might find that clustering a table leads to a significant decrease in records read.



For help debugging or optimizing your query, check our documentation. Learn more.

| Elapsed time | Slot time consumed | Bytes shuffled | Bytes spilled to disk |
|---|---|---|---|
| 390 ms | 512 ms | 324 B | 0 B |

| Stages | Working timing | | | | Rows |
|---|---|---|---|---|---|
| | Wait: | Read: | Compute: | Write: | |
| S00: Input | 30 ms | 16 ms | 9 ms | 4 ms | Records read: 16953<br>Records written: 3 |
| S01: Aggregate | 1 ms | 0 ms | 4 ms | 3 ms | Records read: 3<br>Records written: 3 |
| S02: Output | 1 ms | 0 ms | 10 ms | 17 ms | Records read: 3<br>Records written: 3 |

Key takeaways

This activity demonstrates the impact of using partitions and indexes (known as clusters in BigQuery) in database tables. You can use them to optimize query performance and minimize processing costs. In this exercise, applying partitions and clustering means that BigQuery can break all 41,025 records into smaller, more manageable tables to read. The benefits of partitioning will be even more evident with larger datasets. Use this technique to optimize database performance in your future projects.

Go to next item ✓Completed