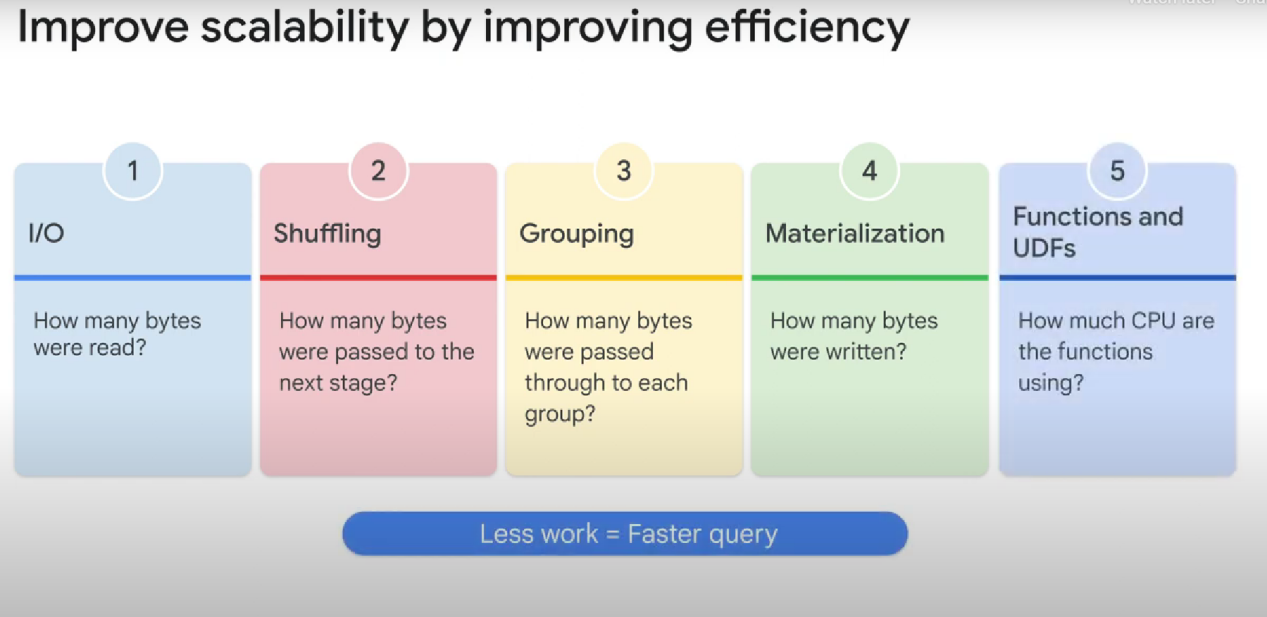
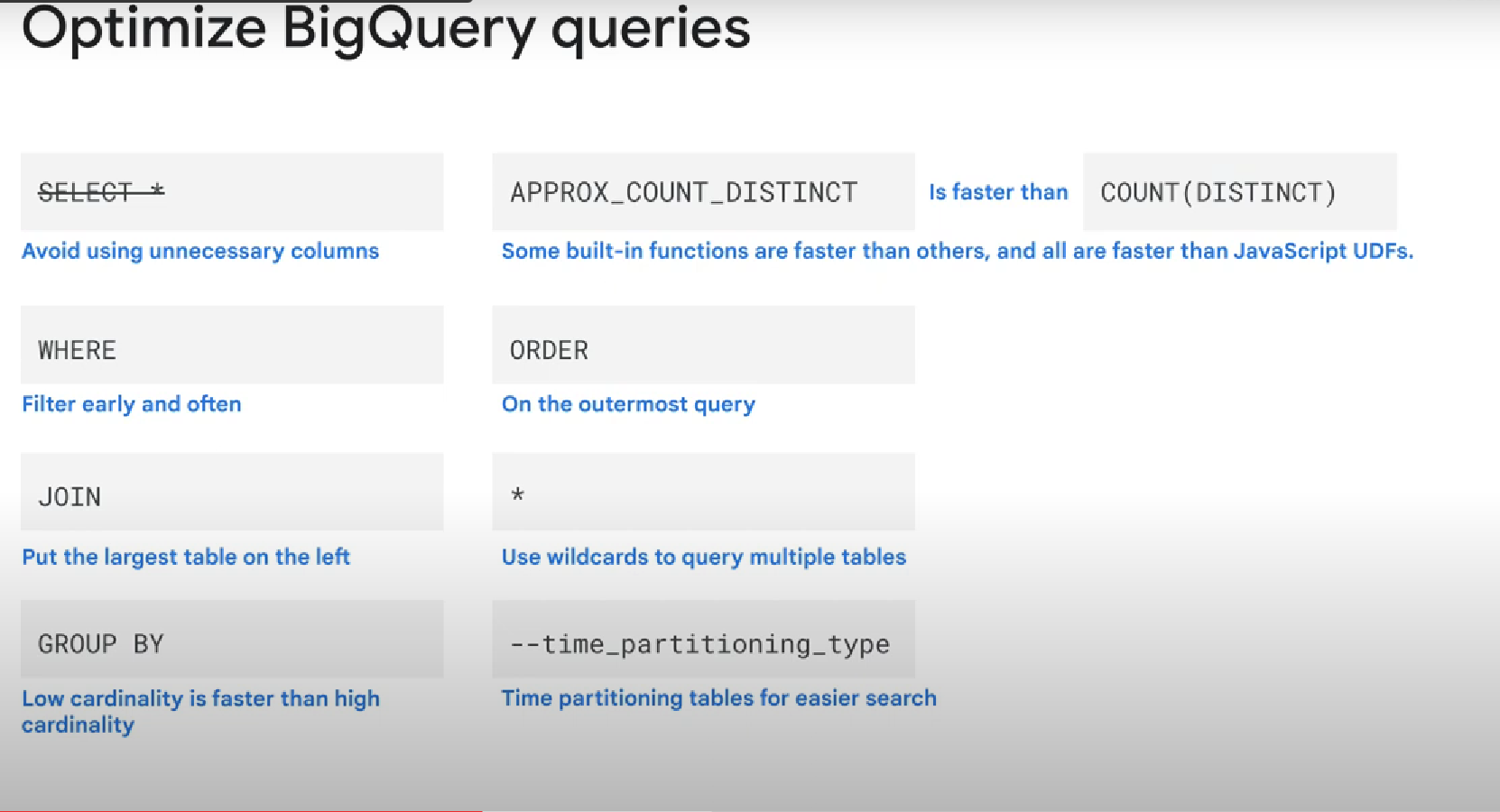
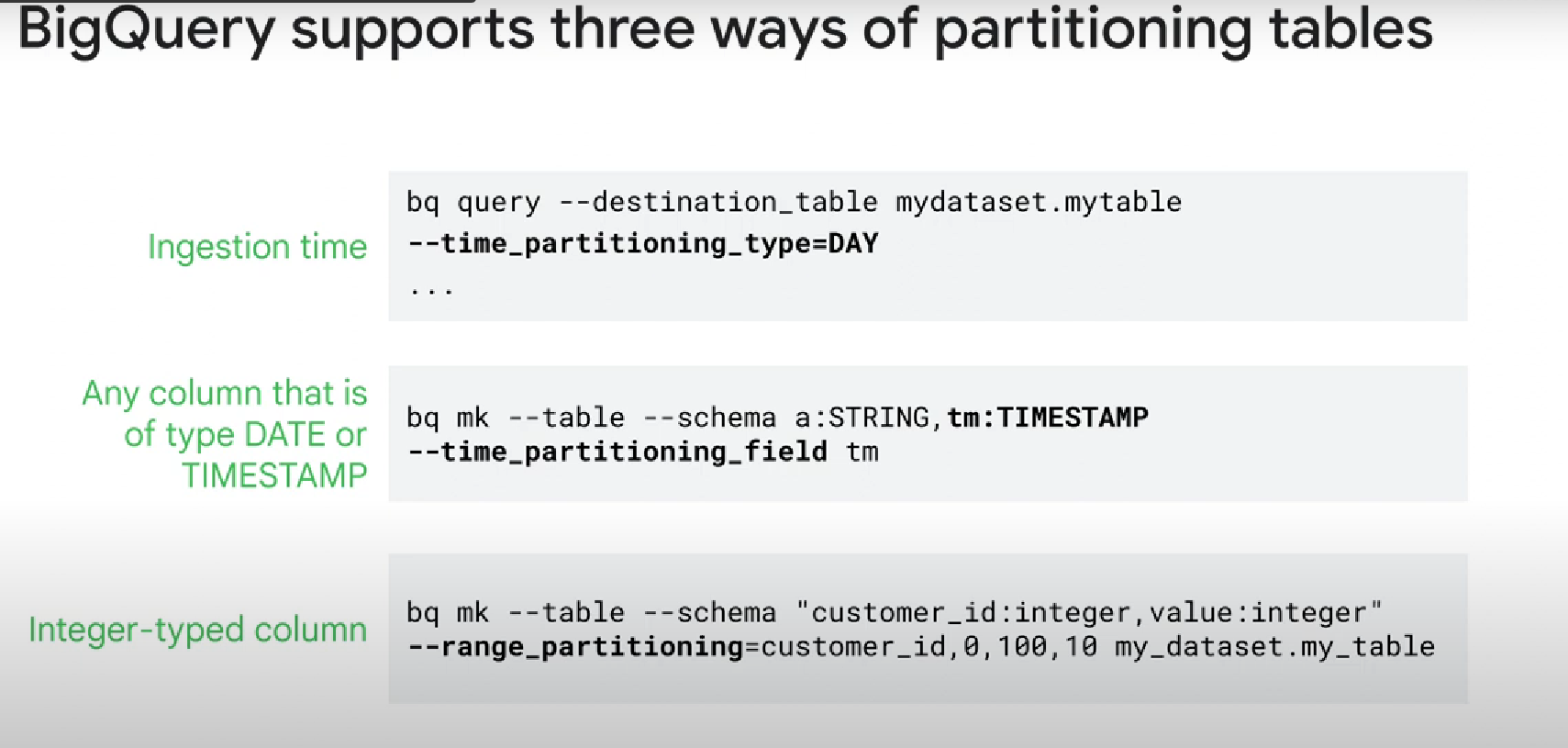
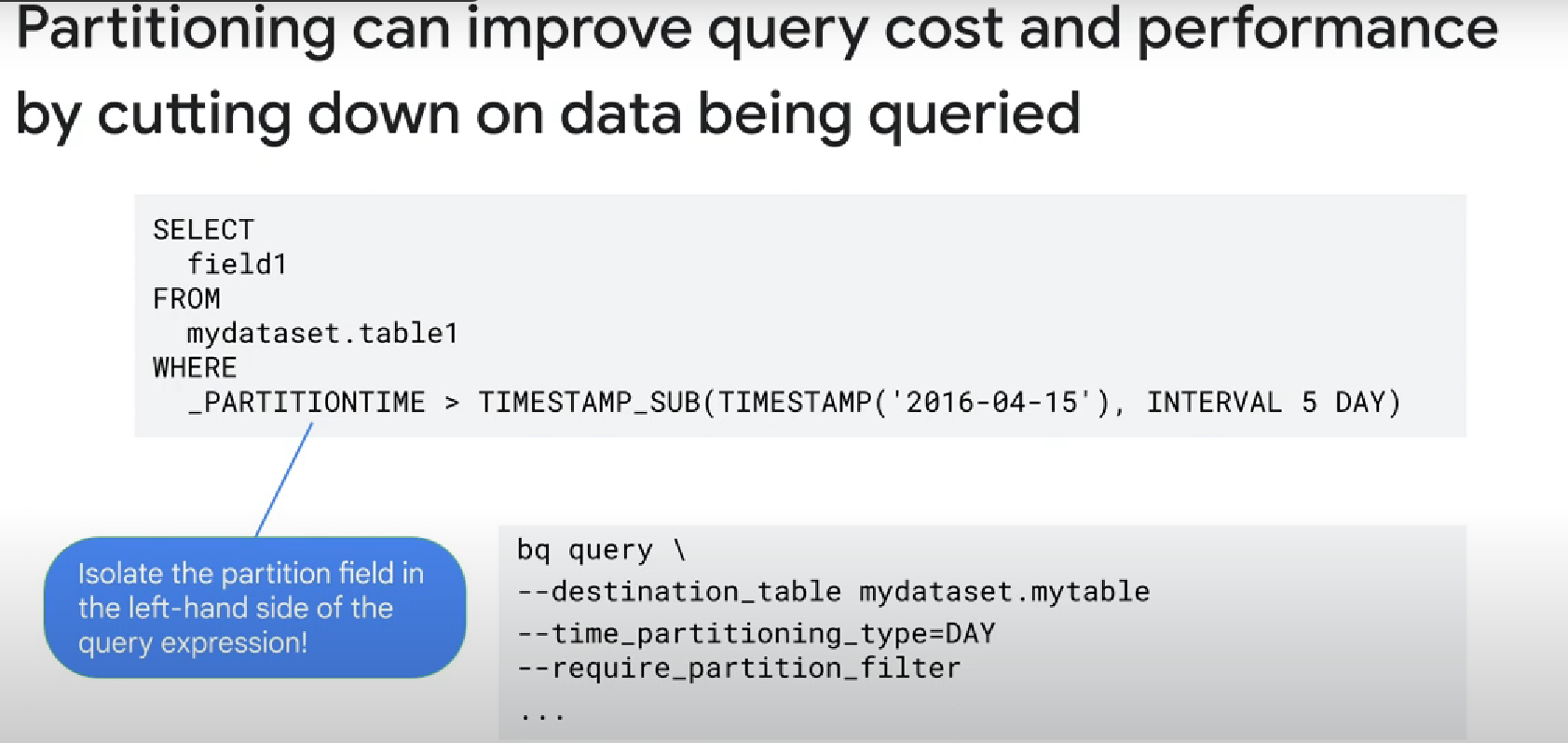


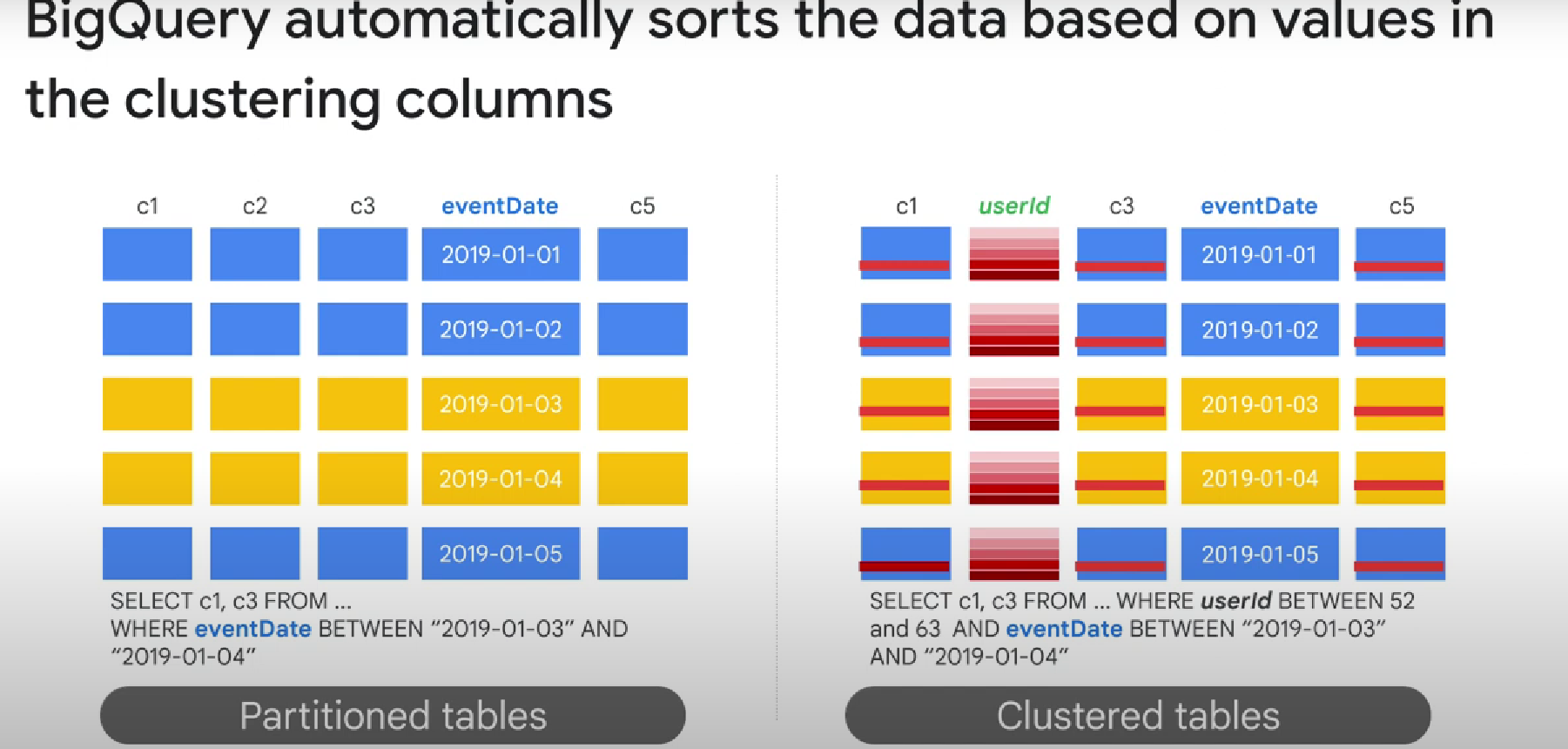
See how data changes over time, see if it is still evenly distributed, see if you need to change the schema, etc...

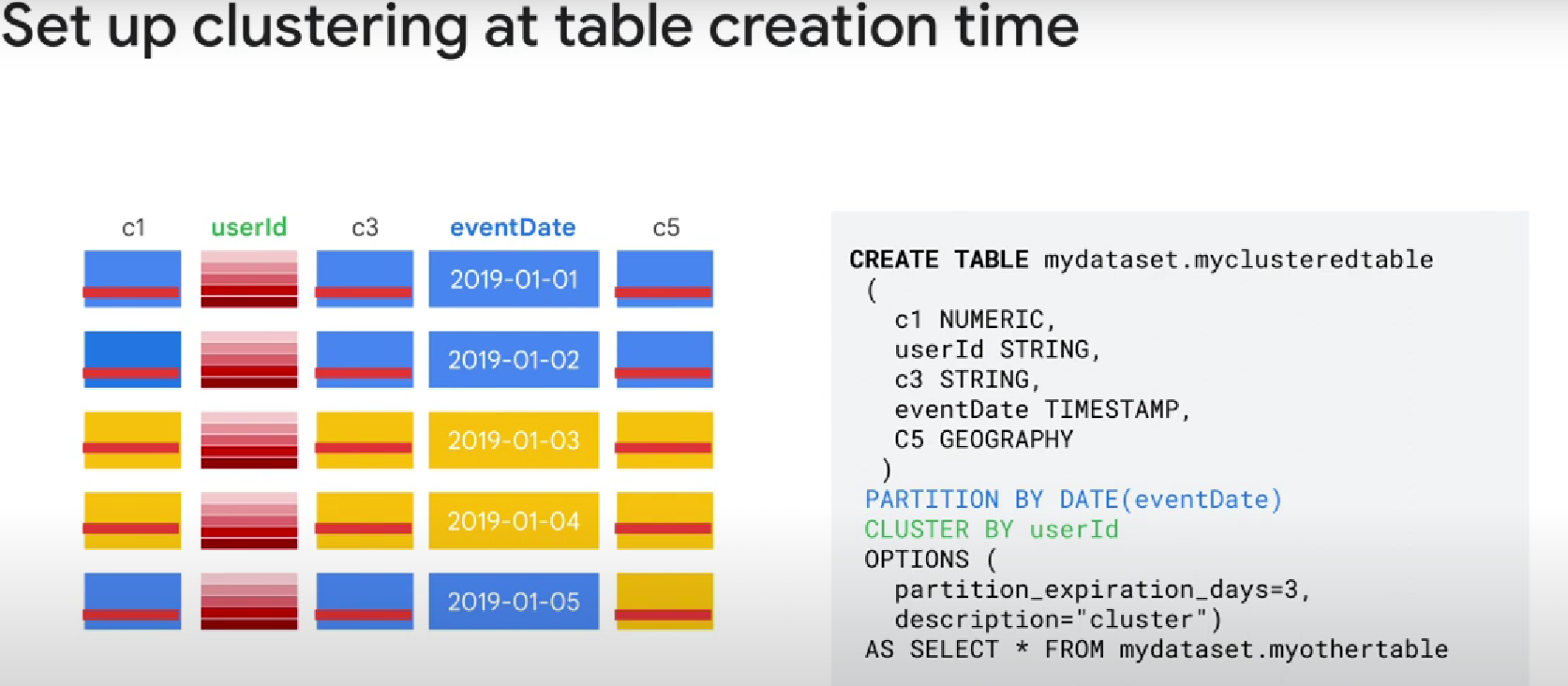




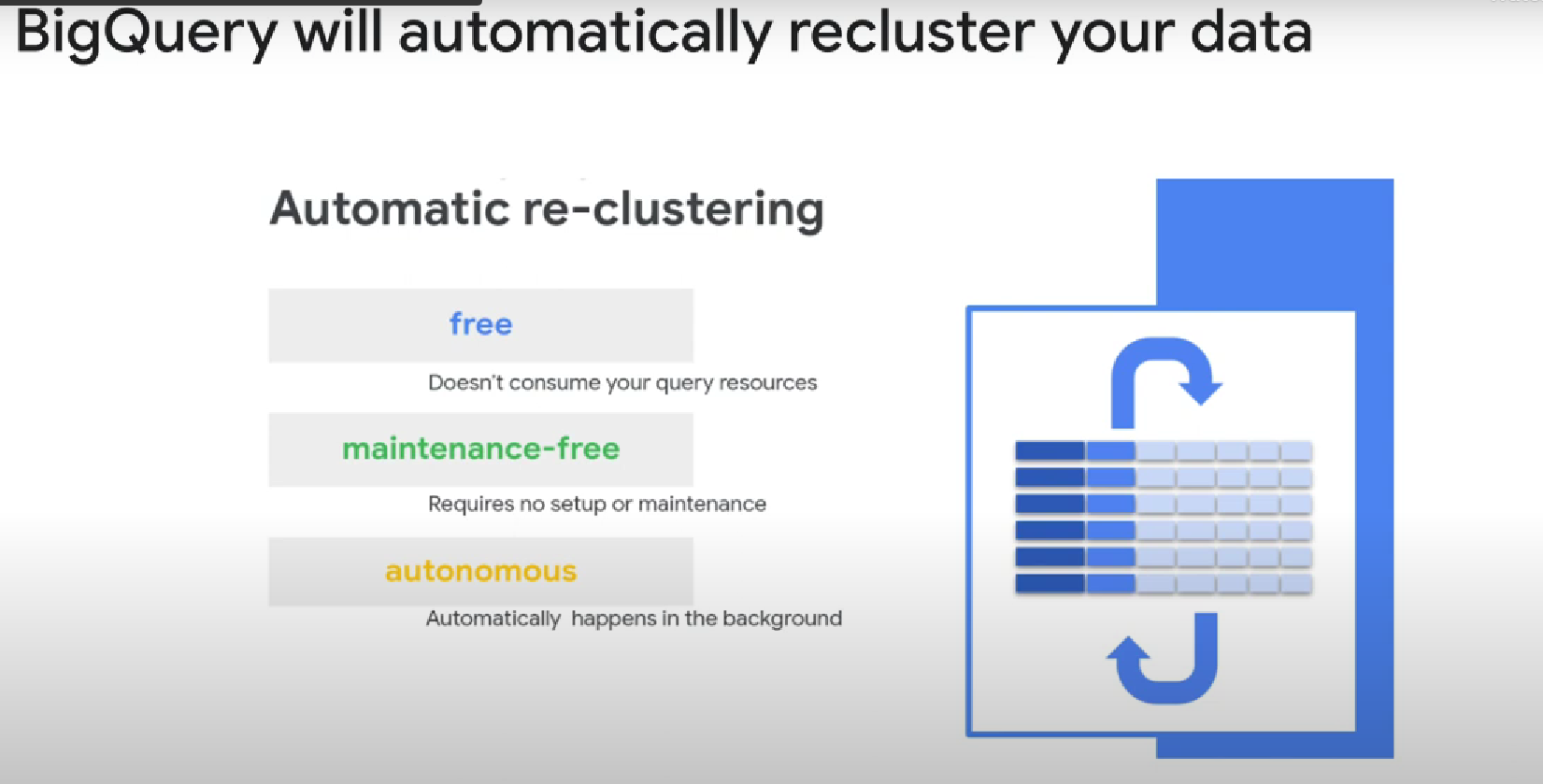




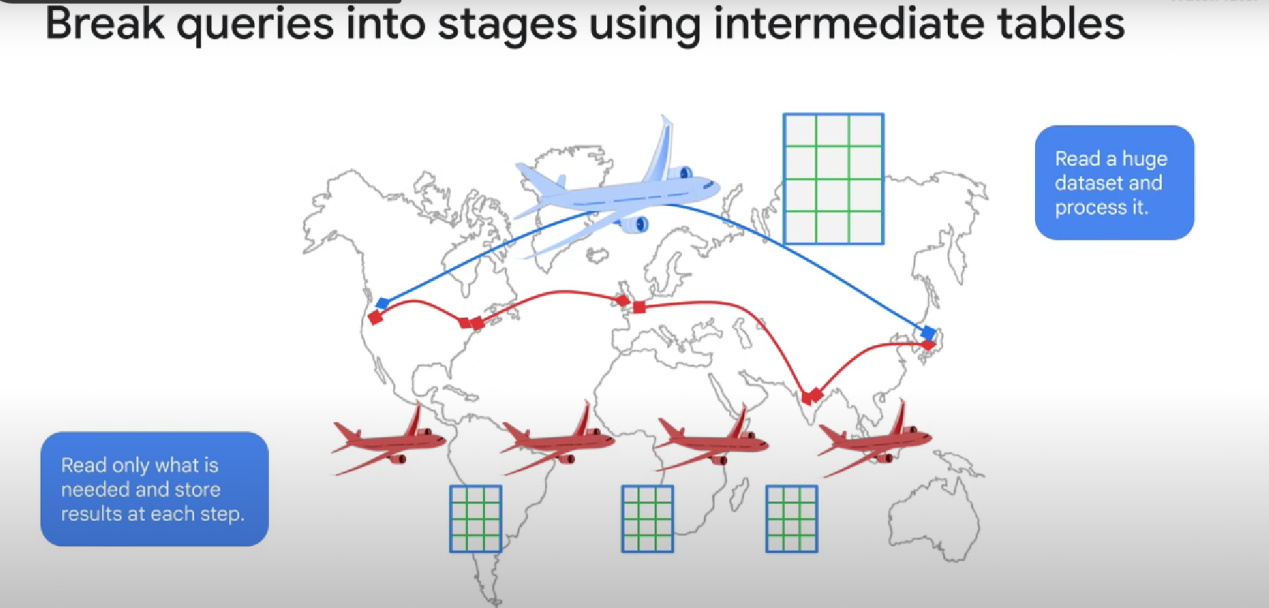




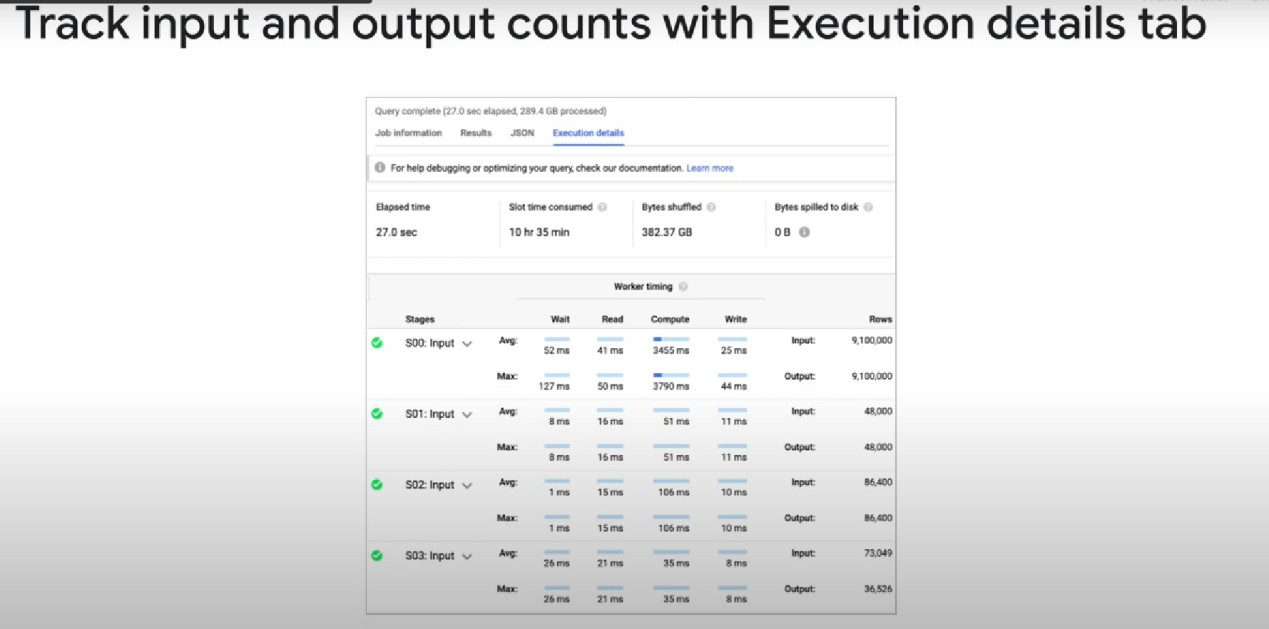
When you cluster data using multiple columns, the order of the specified columns determine the sort order of the data

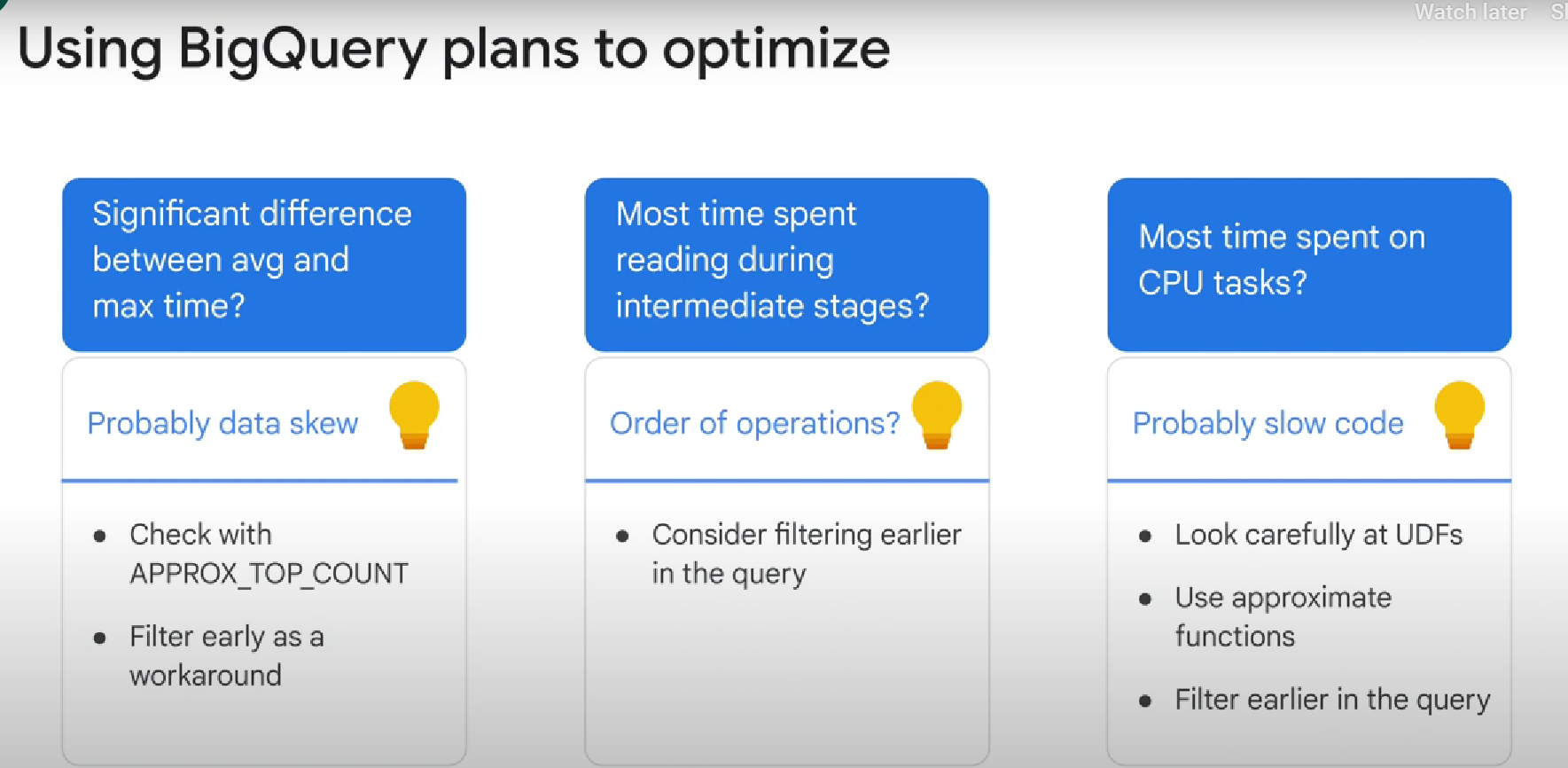


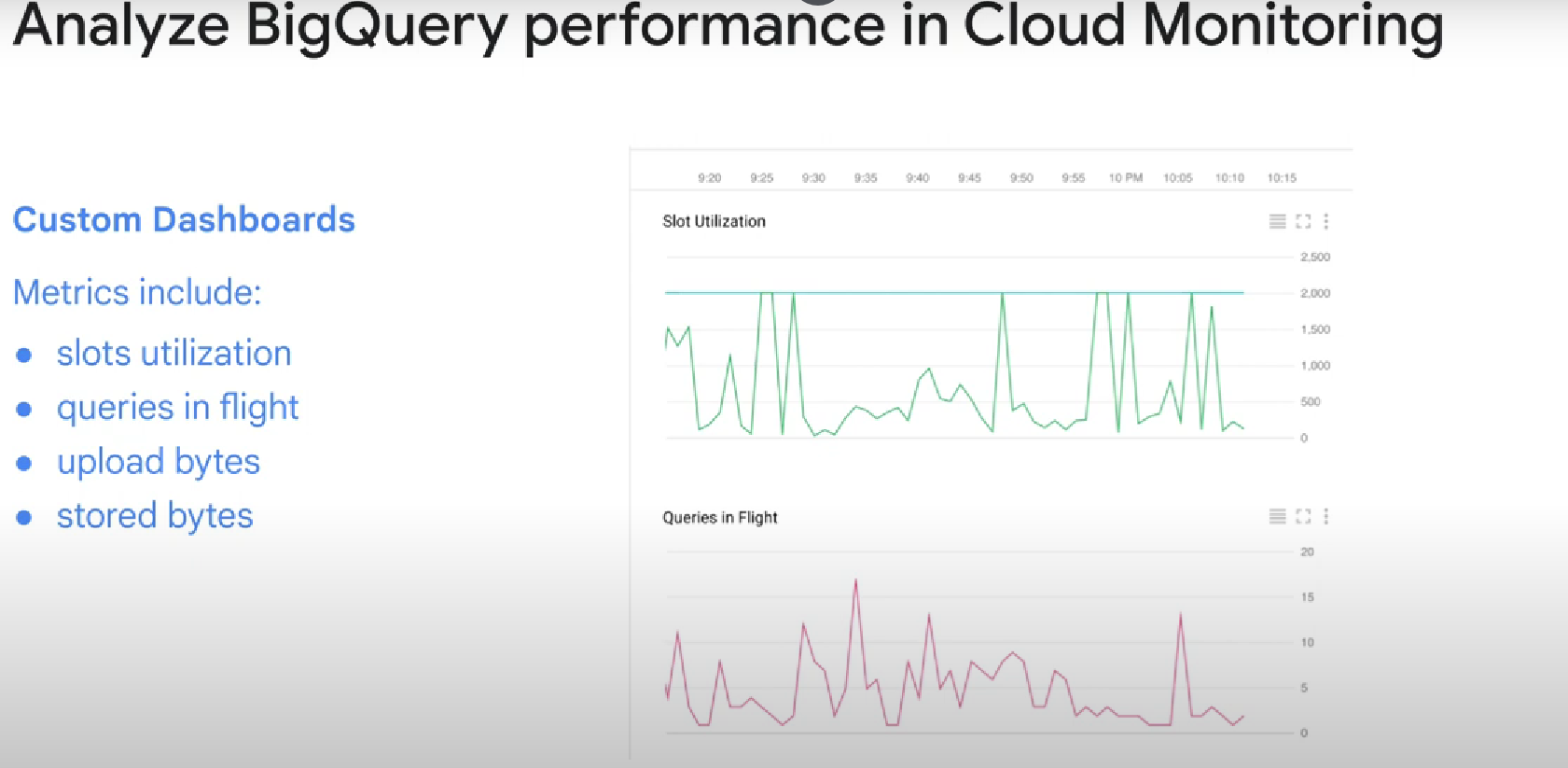
Note: partionning help bigquery gives you better approximations on the cost of a query



Will use more storage but less cpu . And generally, cpu power costs more than storage capacity.



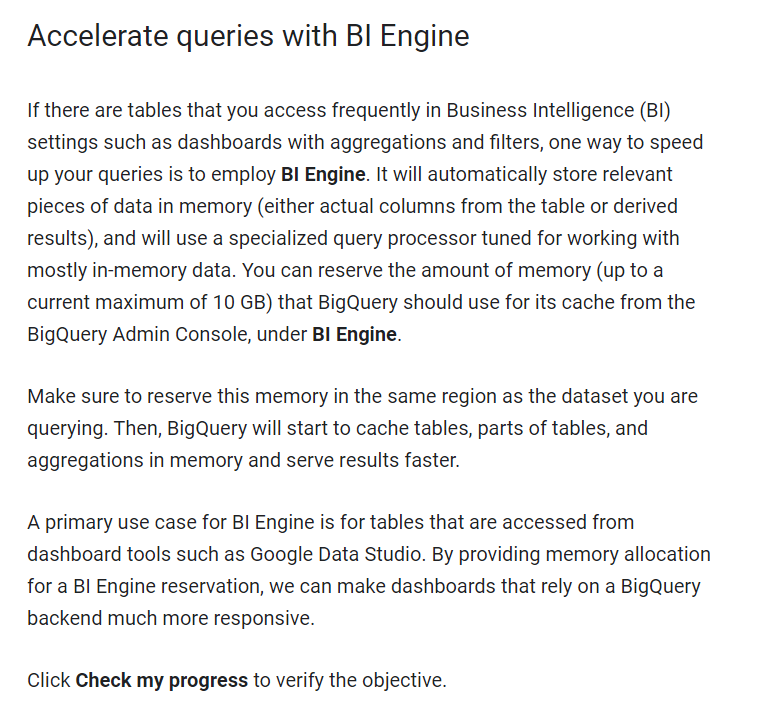




Note: If you find yourself using a WITH clause, view, or a subquery often, one way to potentially improve performance is to store the result into a table (or materialized view).

 The table is not refreshed when new data is added to the original table.

One way to solve this problem of stale data is to **use a materialized view** or to **schedule queries to update the table periodically**. You should measure the cost of such updates to see whether the improvement in query performance makes up for the extra cost of maintaining the table or materialized view up-to-date.



### Avoid self-joins of large tables

Self-joins happen when a table is joined with itself. While BigQuery supports self-joins, they can lead to performance degradation if the table being joined with itself is very large.

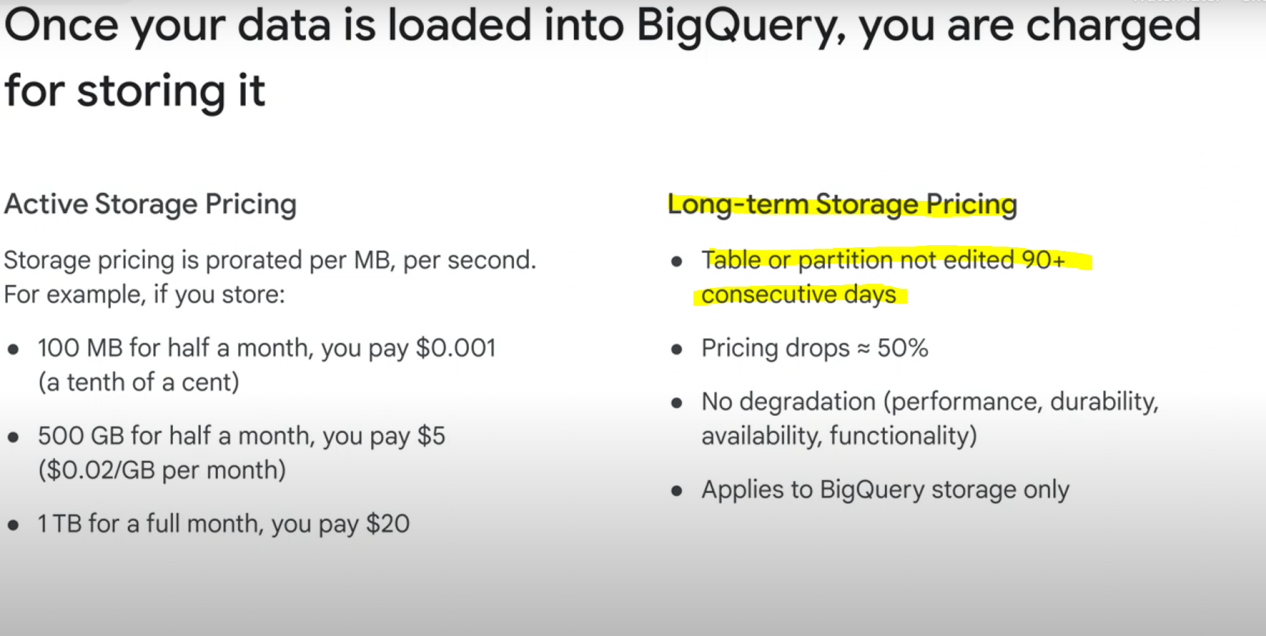
TIMESTAMP\_DIFF ( to guess the next person who rented the same bike without doing join on the table itself)

?????

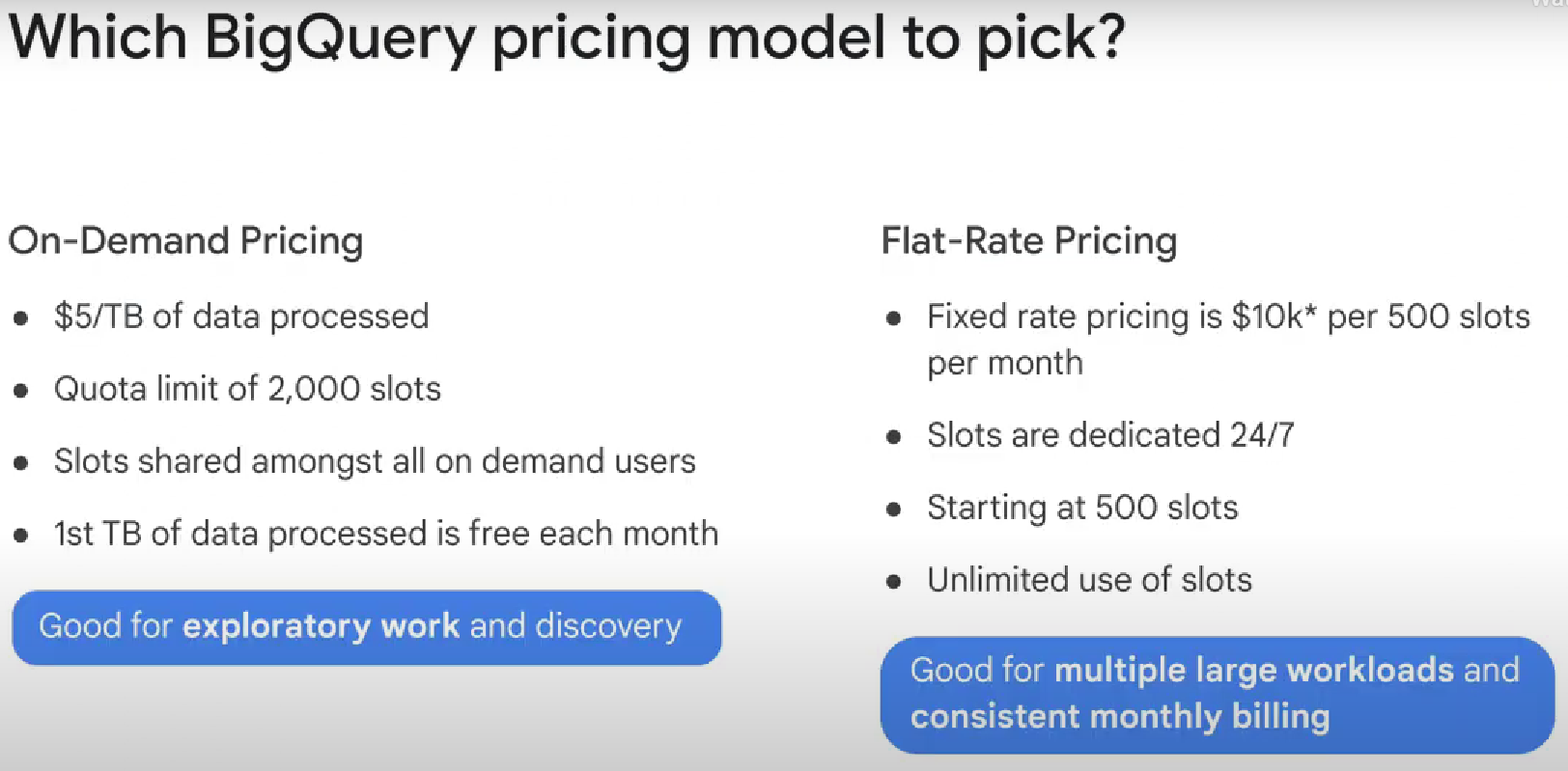


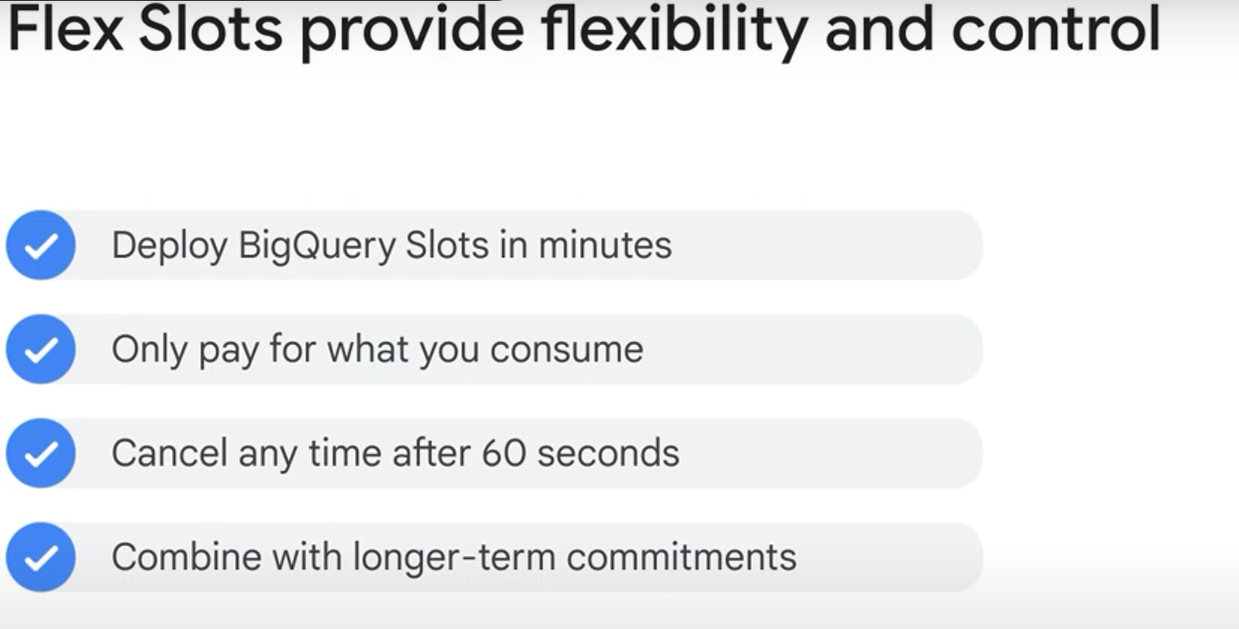
Note: Do processing (cpu heavy operations) on small quantities of data. Separate your data into smaller blocks and do the processing on these blocks. (sort the bike rents of each day instead of sorting all the bike rents you have in your dataset).

If you do require sorting all the data, use more granular keys (i.e. distribute the group’s data over more workers) and then aggregate the results corresponding to the desired key. For example, instead of grouping only by the time zone, it is possible to group by both timezone and repo\_name and then aggregate across repos to get the actual answer for each timezone:

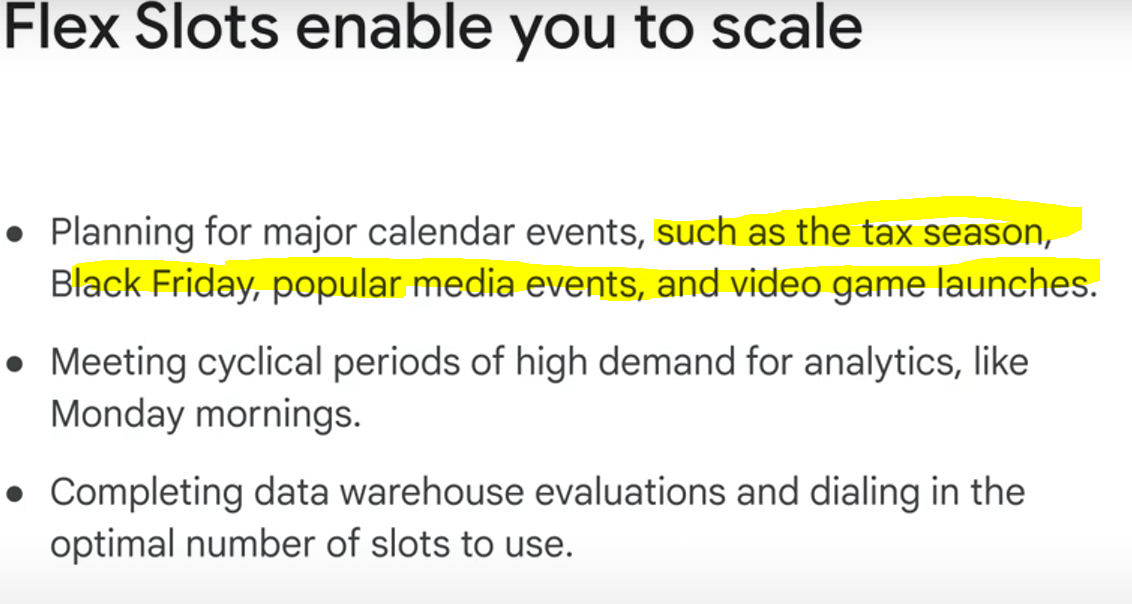


**Querying a table (read) or creating views from a table doesn’t affect the long-term storage policy.** I think only updates of the table will reverse it to the active storage pricing policy.

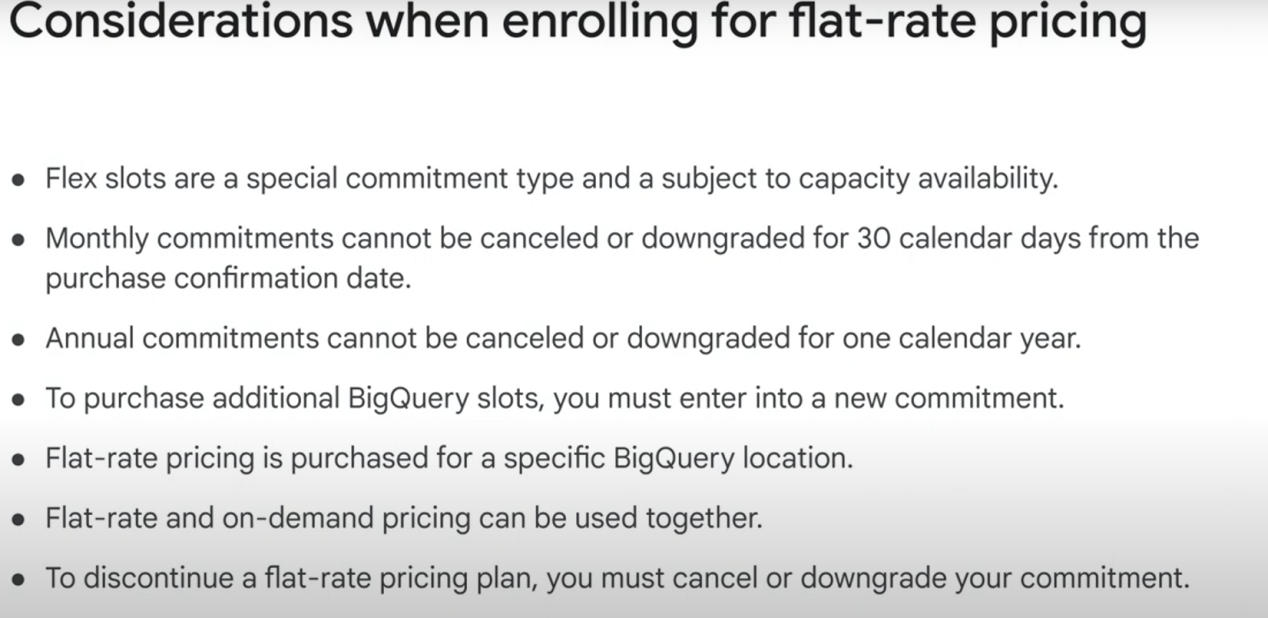




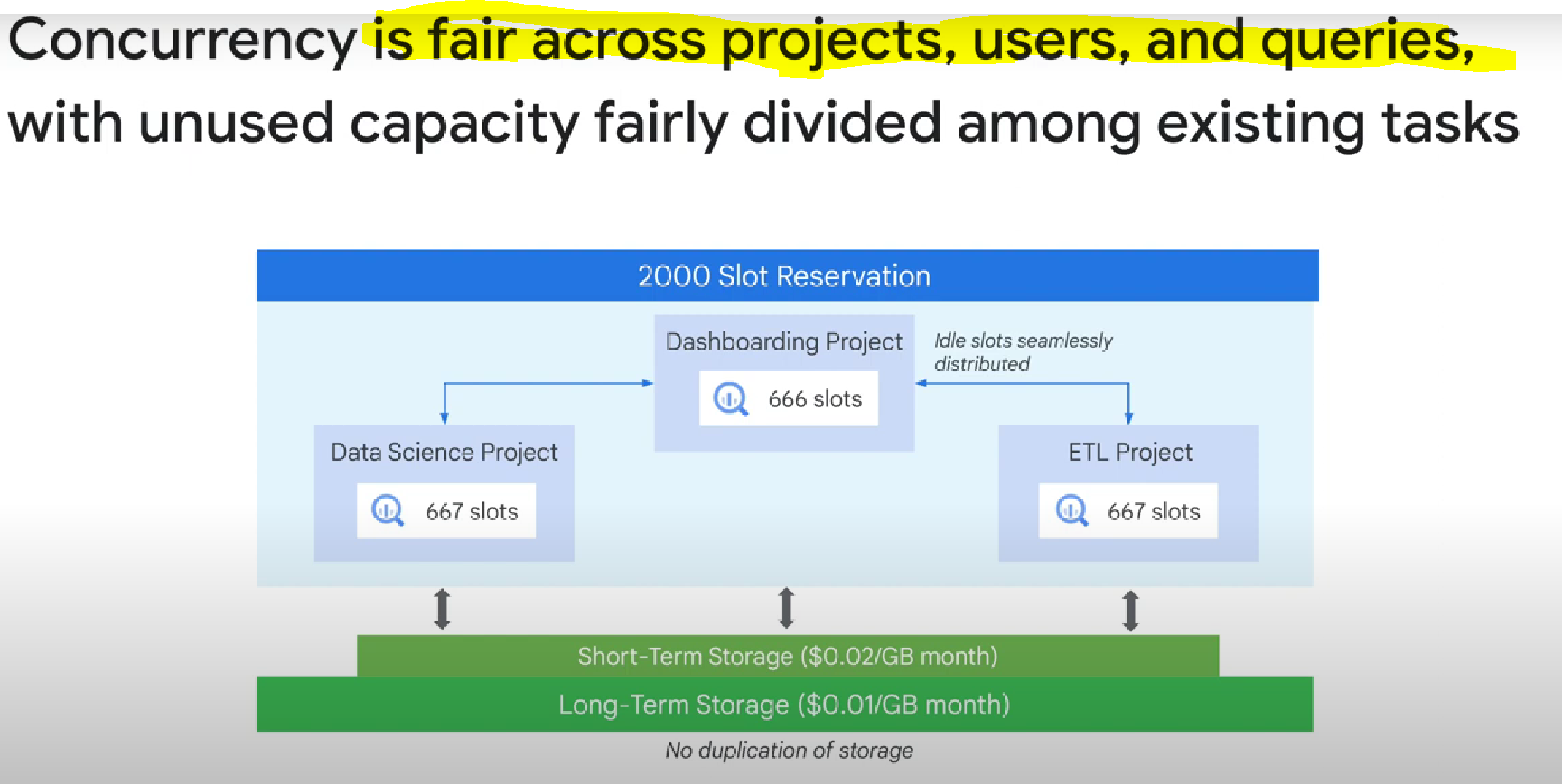
Flex slots = short term



Flex slots are not always available. (If you pay for one and the transaction is accepted, then you are guaranteed a slot you paid for)







(1 very heavy/complex query won’t take all the slots if you’re running multiple queries at the same time)!!

