**Tables**

**TRX\_TAXI\_TRIPS**

Hosts filtered data from the bigquery-public data.new\_york\_taxi\_trips.tlc\_green\_trips\_2014 dataset . Apart from the original columns, additional columns are generated to provide useful or essential information. The table is partitioned daily for optimization purposes. When querying the table, it is recommended to utilize the DT\_TRX\_DATE column for date-related operations.

Lookup tables have been established for columns containing IDs, and you can access them from the following tables.

**LKP\_PAYMENT\_TYPE**

A numeric code signifying how the passenger paid for the trip

**LKP\_RATE**

The final rate code in effect at the end of the trip

**LKP\_TAXI\_VENDOR**

A code indicating the LPEP provider that provided the record.

**LKP\_TRIP\_TYPE**

A code indicating whether the trip was a street-hail or a dispatch that is automatically assigned based on the metered rate in use but can be altered by the driver.

**LKP\_ZONES**

Unique ID number of each taxi zone. Corresponds with the pickup\_location\_id and dropoff\_location\_id in each of the trips tables

Moreover, the following summary tables are available for use in reports or real-time needs.

**F\_TRX\_TAXI\_TRIPS\_PICKUP\_LOCATION\_CNT\_D**

Summarizes daily pickup points commonly used.

**F\_TRX\_TAXI\_TRIPS\_DROPOFF\_LOCATION\_CNT\_D**

Summarizes daily dropoff points commonly used.

**F\_TRX\_TAXI\_TRIPS\_ROUTE\_CNT\_D**

Daily summary of the frequency of taxis using end-to-end routes.

**Querys**

**Sqlquerry.sql**

It encompasses nearly all the codes utilized throughout this study, including scripts for table creation, data control, and more.

**COPY DATA TRX\_TAXI\_TRIPS.sql**

TRX\_TAXI\_TRIPS includes the codes necessary for daily (delta) data transfer of the table. It is the first activity in the ETL job. It must be completed successfully for other jobs to work.

**F\_TRX\_TAXI\_TRIPS\_PICKUP\_LOCATION\_CNT\_D\_LOAD.sql**

For the "TRX\_TAXI\_TRIPS\_PICKUP\_LOCATION\_CNT\_D" table, it identifies the changing days in the main transaction table, calculate summary metrics for these days, and loads them into the table.

**F\_TRX\_TAXI\_TRIPS\_DROPOFF\_LOCATION\_CNT\_D\_LOAD.sql**

For the "TRX\_TAXI\_TRIPS\_DROPOFF\_LOCATION\_CNT\_D" table, it identifies the changing days in the main transaction table, calculate summary metrics for these days, and loads them into the table.

**F\_TRX\_TAXI\_TRIPS\_ROUTE\_CNT\_D\_LOAD.sql**

For the "TRX\_TAXI\_TRIPS\_ROUTE\_CNT\_D" table, it identifies the changing days in the main transaction table, calculate summary metrics for these days, and loads them into the table.

**RESOURCES**

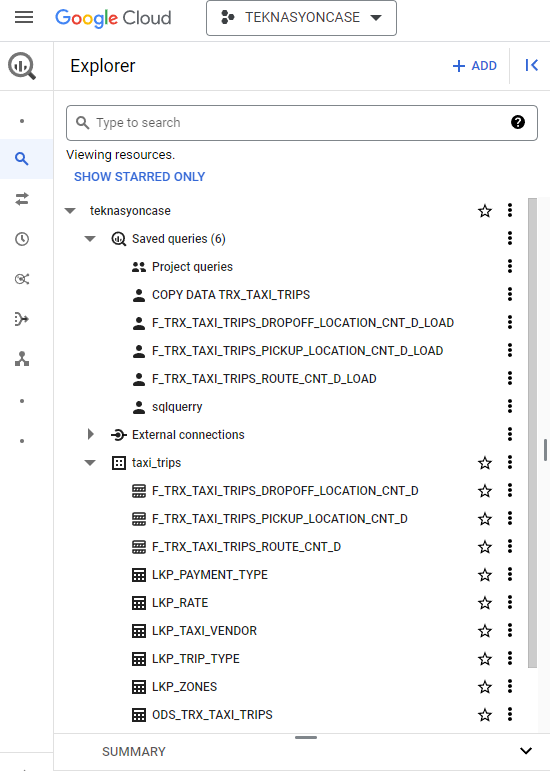
cloud.google.com/bigquery/docs

[google.com](http://www.google.com)

[youtube.com](http://www.youtube.com)

stackoverflow.com

medium.com



Define a data pipeline architecture that would work on the data received and does the above calculations when the data received.

Our data source can be a file system, a database, or any other data repository. The key considerations here revolve around delta data transfer, ensuring the seamless execution of operations without data loss during transfers, and without causing disruptions to the operation of both the source and target systems.

If there is a column, such as 'UPDATE date,' in our dataset, it significantly simplifies the process of data transfer. Specifying the end date with a gap whenever data is being transferred is a preventive measure against data loss. This approach enhances the efficiency of data transfer operations and helps maintain data integrity.

What would be the bottlenecks of such a system and how would your design overcome these situations?

In such a system, there are crucial details that can simplify our workflow. It would be advantageous to have a transaction number to ensure data uniqueness. If this is not available, we can explore generating this number using multiple columns, as we did in our latest table.

Opting for the delta data transfer method introduces specific considerations. It is essential to avoid historical data flow in this scenario. If historical data needs to be maintained, a field like the 'Update' field becomes imperative.

In cases where there is neither a unique column nor an update date, it becomes necessary to go directly to the source and perform bulk transfers for a defined period. However, this approach may lead to increased performance losses and system costs.