**Task 05**

**Abdul Razaque**

**Data Engineering**

* **What is the Historical Load:**

Historical load is the one-time initial load of data that the Source already had before the creation of the Pipeline.

I mean it does beg the question, what exactly is a historical data load anyways? When does a data load move over some invisible line to become a historical load, as opposed to a “normal” load? This is a strange question, but generally, I think you know a historical data load when you see one. Probably because it fits into some box that is “outside” normal. Maybe that’s a bad way to approach historical loads, should one process handle it all? What about the ever-infamous “idempotent” data loads, shouldn’t your ETL loads be a Kraken raised from the depths, able to gobble all in its path?

Historical load refers to the process of loading historical data into a data warehouse or a database. Historical data is data that has been collected over a period of time and may include data from months, years, or even decades ago.

Historical load is an important part of data warehousing and analytics, as it allows organizations to analyze trends and patterns over time, and gain insights into long-term business performance. By analyzing historical data, organizations can identify patterns and trends that may not be visible in more recent data, and make informed decisions based on this information.

The process of historical load involves collecting, transforming, and loading large amounts of historical data into a data warehouse or database. This process can be complex and time-consuming, as historical data is often stored in a variety of formats and sources, and may require extensive data cleansing an transformation.

**What makes historical loads suck so much?**

I think there are many and varying reasons that historical loads are such a pain.

* Historical data may or may not be in the correct/different format.
* Historical data has a high likely hood of having data quality issues (that cause loads to break).
* Historical data loads choke the normal “production” pipeline.
* It’s not always clear “how” the historical load should be done.

**3 Practical Approaches to Historical Data Loads.**

* Historical load integrated directly into normal load/transform scripts.
* Historical load completely separate from normal load/transform scripts.
* Quasi-integration of historical load and normal load/transform scripts.
* **What is Full Load in detail:**

Full-load (FTL) means when the transport company delivers the truck to the shipper, which completely fills the truck in one direction.

In simple words, the full load of a machine is the rated power output as per the machine nameplate whereas no load means zero power output.

Full load refers to the process of loading all the data from a source system into a target system, such as a data warehouse or a database, during the initial data integration process. A full load is typically performed when the target system is newly created or when there is a need to refresh the entire dataset in the target system.

In a Full Data Load, the complete dataset is emptied or loaded and then entirely overwritten (i.e. deleted and replaced) with the newly updated dataset in the next data loading run. While comparing the Incremental Data Load vs Full Load, you also don’t need to maintain extra information such as timestamps to carry out a Full Data Load.

**Key Benefits of ETL Full Load::**

A Full Data Load is a traditional Data Loading method that offers the following benefits:

* **Easy-to-Implement**: When comparing the Initial Load vs Full Load ETL, executing a Full Data Load is a straightforward process that simply deletes the whole old table and replaces it with an entire updated dataset.
* **Simple Design**: Based on a particularly easy-to-set uploading process, a Full Data load doesn’t require you to worry about database design and keeping it clean. While comparing Initial Load vs Full Load ETL, you will notice that If an error occurs in a Full Load, you can simply re-run the loading process without having to do much else in the way of data cleanup/preparation.
* **Low Maintenance**: This technique doesn’t require you to manage the keys and whether some data is up to date or not as every time you reload the table, all data will be updated no matter what. For instance when comparing the Initial Load vs Full Load ETL, dtime\_updated, and dtime\_inserted are the most commonly used keys in delta load.

**Challenges of Full Data Load::**

While applying the Full Data Load approach, you may encounter the following hurdles:

* **Unsustainable**: It can be an inconvenient data loading method when you only need to update just a handful of records but have to insert millions of records due to its architecture.
* **Slow Performance**: As you start dealing with massive volumes of data, performing a full data load with a larger dataset is time-consuming and takes up a lot of server resources.
* **Unable to Preserve History**: With Full Data Load, you can’t keep the historical data as it drops the old data and the new dataset completely replaces it. This old data is often important as in some cases you may want to track the changes in the database
* **What is Incremental Load:**

Incremental load is a selective method of moving data from one system to another.

Incremental load refers to the process of loading only the new or changed data from a source system into a target system, such as a data warehouse or a database, since the last load. Incremental load is a popular technique in data integration and is used to reduce the time and resources required to load large datasets into a target system.

Incremental loading is available for many of the Treasure Data integrations. In some cases, it is a simple checkbox choice and in others, after you select incremental loading you are provided with other fields that must be specified.

**Limitations, Supported, Suggestions:**

* For some integrations, if you choose incremental loading, you might need to make sure that there is an index on the columns to avoid a full table scan.
* Only Timestamp, Datetime, and numerical columns are supported as incremental\_columns.
* For the raw query, the incremental\_columns is required because it won't be able to detect the Primary keys for a complex query.

There are 2 types of incremental loads, depending on the volume of data you’re loading;

1. Streaming incremental load
2. Batch incremental load.

**About Incremental Loading for Integrations::**

Treasure Data Incremental loading has 4 patterns (3 types of data connector + 1 workflow td\_load operator.), then the 3 data connector loading examples are as follows:

* Cloud storage service (e.g. AWS S3, GCS and etc.)
* Lexicographic order of file name
* Query (e.g. MySQL, BigQuery and etc.)
* Date time
* Variable period (Google Analytics, etc)
* Use start\_date for loading

Incremental load is commonly used in various industries, such as finance, healthcare, retail, and logistics, among others, where there is a need for up-to-date and accurate data to make informed decisions. It is an essential component of data integration and is often used in conjunction with other data integration techniques to ensure that the target system contains the most up-to-date and accurate data possible.