Demo for loading your data in sort key order for time series data

https://docs.aws.amazon.com/redshift/latest/dg/vacuum-load-in-sort-key-order.html

The purpose of the test to prove the followings:

COPY automatically adds new rows to the table's sorted region when all of the following are true:

- The table uses a compound sort key with only one sort column.
- The sort column is NOT NULL.
- The table is 100 percent sorted or empty.
- All the new rows are higher in sort order than the existing rows, including rows marked for deletion. In this instance, Amazon Redshift uses the first eight bytes of the sort key to determine sort order

The test set up:

1. Create a table with the following DDLs in your redshift database:

```
CREATE TABLE demo.time_serie_demo
(
vendorid bigint ,
lpep_pickup_datetime timestamp without TIME zone not null,
lpep_dropoff_datetime timestamp without TIME zone ,
store_and_fwd_flag CHARACTER varying(65535) ,
ratecodeid bigint ,
pulocationid bigint ,
dolocationid bigint ,
passenger_count bigint ,
trip_distance DOUBLE PRECISION,
fare_amount DOUBLE PRECISION ,
extra DOUBLE PRECISION ,
mta_tax DOUBLE PRECISION ,
tip_amount DOUBLE PRECISION ,
tolls_amount DOUBLE PRECISION ,
ehail_fee CHARACTER varying(65535) ,
improvement_surcharge DOUBLE PRECISION ,
total_amount DOUBLE PRECISION ,
payment_type bigint ,
trip_type bigint
) distkey(lpep_pickup_datetime) compound sortkey(lpep_pickup_datetime);
```

2. Create S3 bucket to store data from Jan 2018 to Jun 2018 and the S3 bucket structure as below:

```
s3://demo

        /Jan2018/green_tripdata_2018-01.csvn_tripdata_2018-04.cseven_tripdata_2018-04.csv

        /Feb2018/green_tripdata_2018-02.csv

        /Mar2018/green_tripdata_2018-03.csv

        /Apr2018/green_tripdata_2018-04.csv

        /May2018/green_tripdata_2018-05.csv

        /Jun2018/green_tripdata_2018-06.csv
```

Note: Please see the csv files in the attachment and the data in each csv are not sorted. And for initial loading, the file sequence does not matter, redshift copy command will store the data in the sorted manner according the definition of the sort key.

3. Run the copy command to ingest all csv files into table demo.time_serie_demo table

```
copy demo.time_serie_demo from 's3://demo/' iam_role '<your redshift role>' csv;
```

4. Run the following command to verify if the data in the table 100% sorted or not

```
select * from svv_table_info where unsorted =0 and schema='demo';     ---change to your schema name or tablename
```

Output as below and you will see "unsorted" column is "0.00", which means all data are sorted after the initial load

ssb demo 154019 time_serie_demo1 Y KEY(lpep_pickup_datetime) lpep_pickup_datetime 65535 lzo 1 352 0.0461 0 0.00 0.00 604688 1.00 1.00

5. Upload green_tripdata_2018-09.csv to s3://demo/Sep2018 to simulate the incremental ingestion

> *s3://demo*
>
> */Jan2018/green_tripdata_2018-01.csvn_tripdata_2018-04.cseven_tripdata_2018-04.csv*
>
> */Feb2018/green_tripdata_2018-02.csv*
>
> */Mar2018/green_tripdata_2018-03.csv*
>
> */Apr2018/green_tripdata_2018-04.csv*
>
> */May2018/green_tripdata_2018-05.csv*
>
> */Jun2018/green_tripdata_2018-06.csv*
>
> */Sep2018/green_tripdata_2018-09.csv*

Note: The data in the green_tripdata_2018-09.csv iis not needed to be pre-sorted, but each data in sort key column should be higher than existing rows

6. Run the copy command to ingest green_tripdata_2018-09.csv only into table demo.time_serie_demo table which already having initial data from 6 csv files loaded

> ***copy** demo.time_serie_demo **from** 's3://demo/Sep2018/' iam_role '<your redshift role>' **csv**;*

7. Run the following command to verify if the data in the table is still 100% sorted or not after having one incremental load

> ***select * from** svv_table_info **where** unsorted **=**0 **and schema=**'demo'; ---change to your schema name or tablename*
>
> ***select * from** svv_table_info **where** unsorted **>**0 **and schema=**'demo';*

Note: You should see data in the table still 100% sorted and you don't need to run vacuum command fro such use cases.

Summary: You can use one copy command to load multiple files per table base, the number of files should be multiple of the number of slices you redshift cluster has when conducting initial loadingIf the following conditions are satisfied ,

and

OSDS-523 - Prepare the demo use case for loading time series data in general to initial load and incremental load

TO DO

you don't need to run vacuum command for this particular use cases:

- The table uses a compound sort key with only one sort column.
- The sort column is NOT NULL.
- The table is 100 percent sorted or empty.
- All the new rows are higher in sort order than the existing rows, including rows marked for deletion. In this instance, Amazon Redshift uses the first eight bytes of the sort key to determine sort order