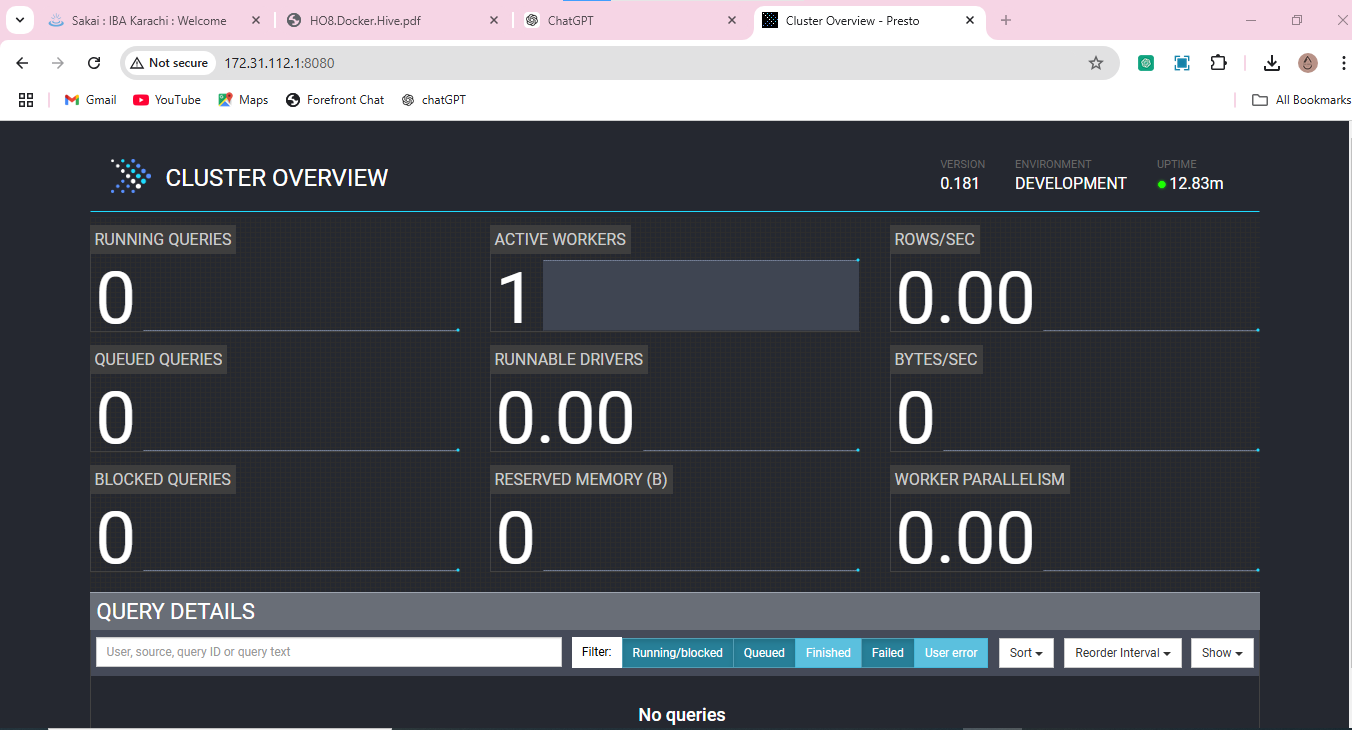
**Big Data Docker lab 8 HIVE**

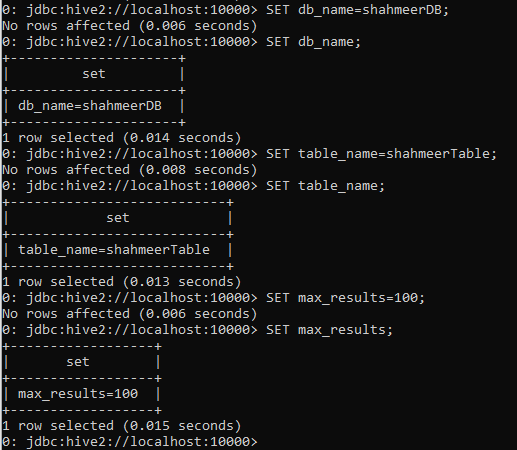
**Shahmeer Khan 25156**

**Presto output port 8080**

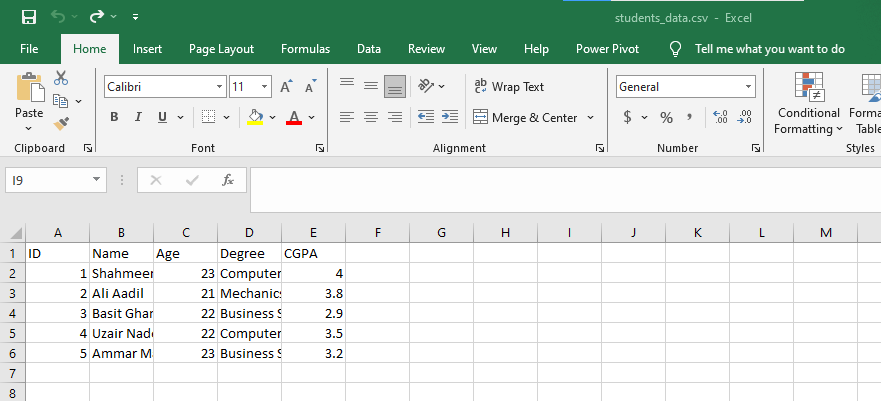


**Changing variable names**

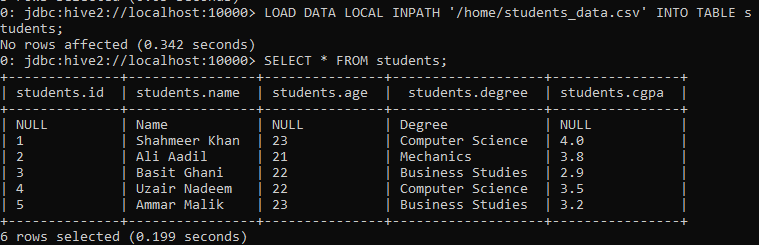
Set! Didn’t work so I used Set instead



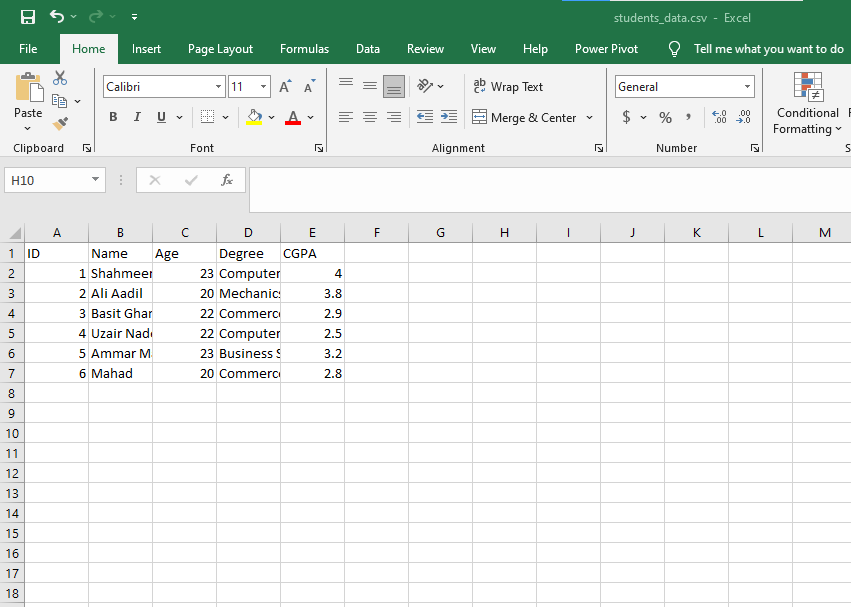
**CSV file**



**Loaded into table**

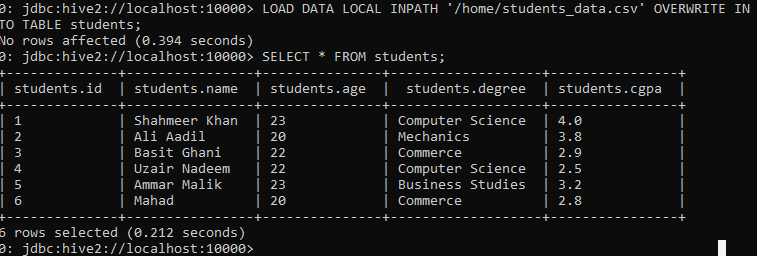


**Updated csv**

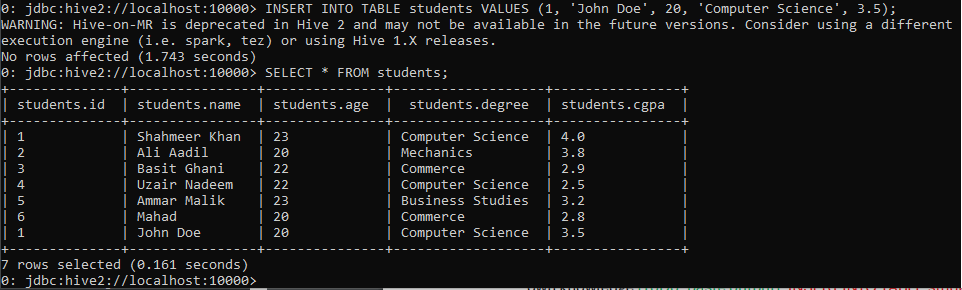


**Overwrite command**

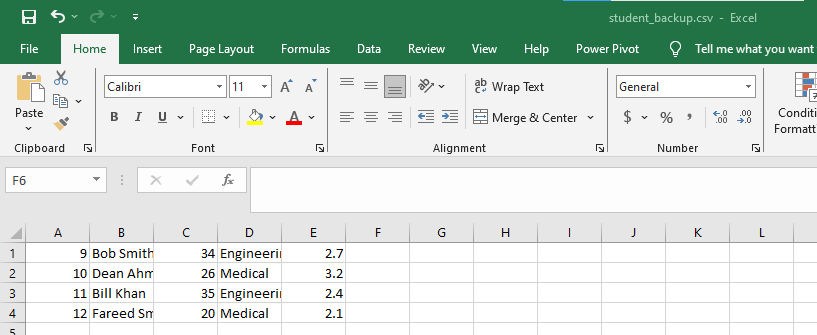
To overwrite after changes made, I had to first delete the csv file from the hive server container and then copy it from local again, after that the overwrite command showed needed changes



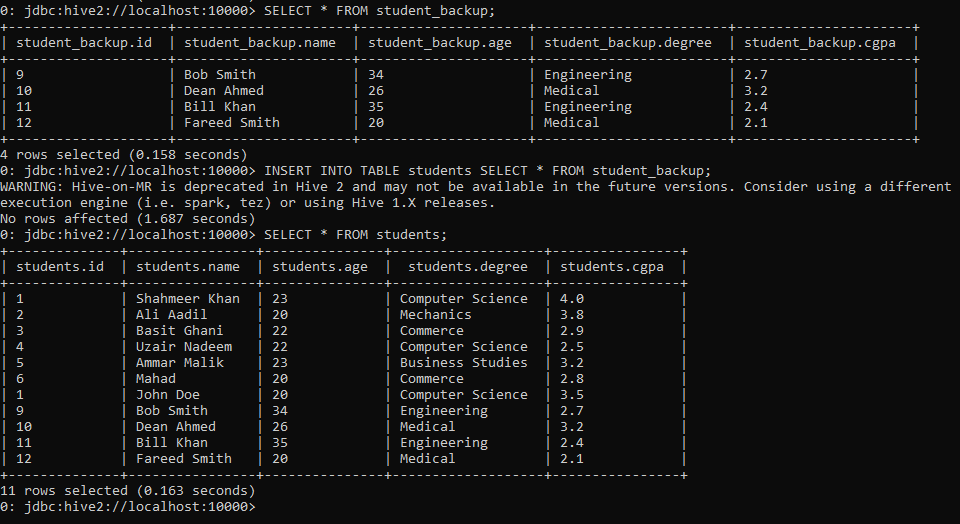
**Insert command**



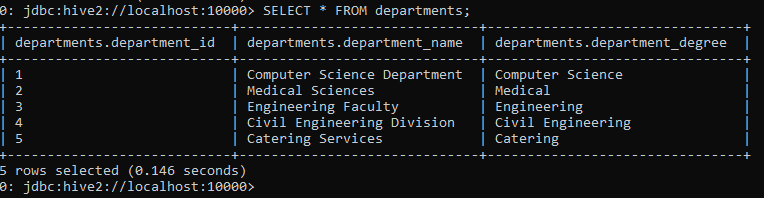
**New table**



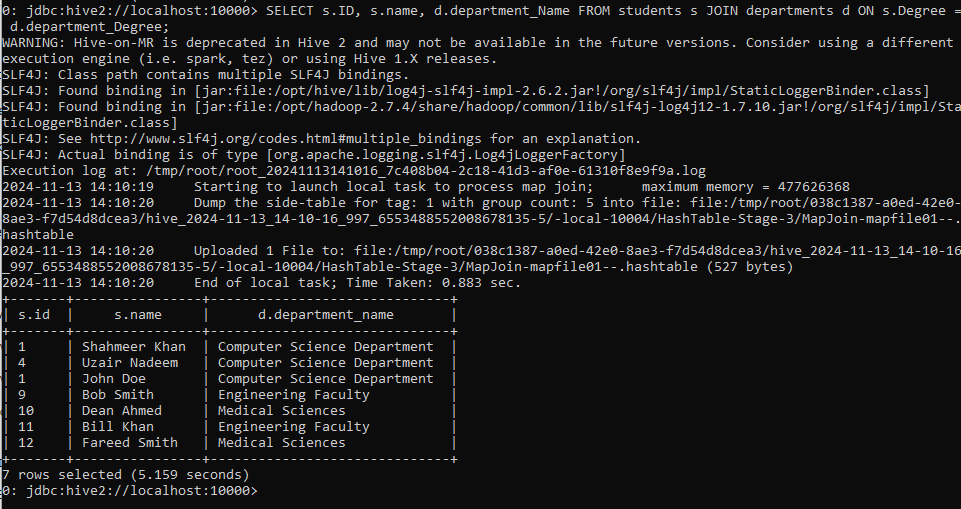
**Insert data from another table**



**Dummy department table**

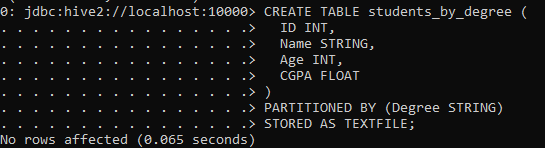


**Join with departments query**

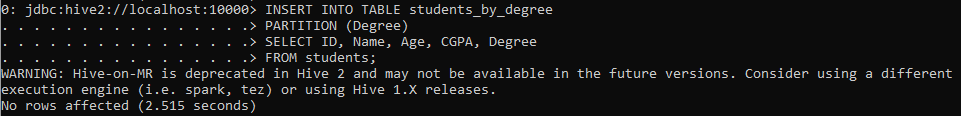


**Partitioning**

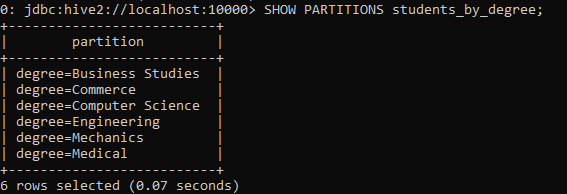
Create partition table



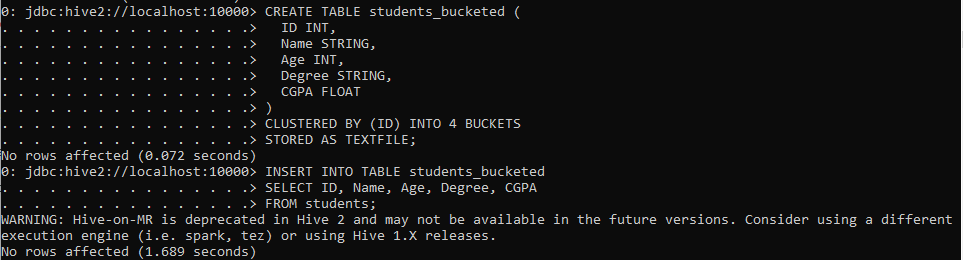
Copy data from main table



Show partitions

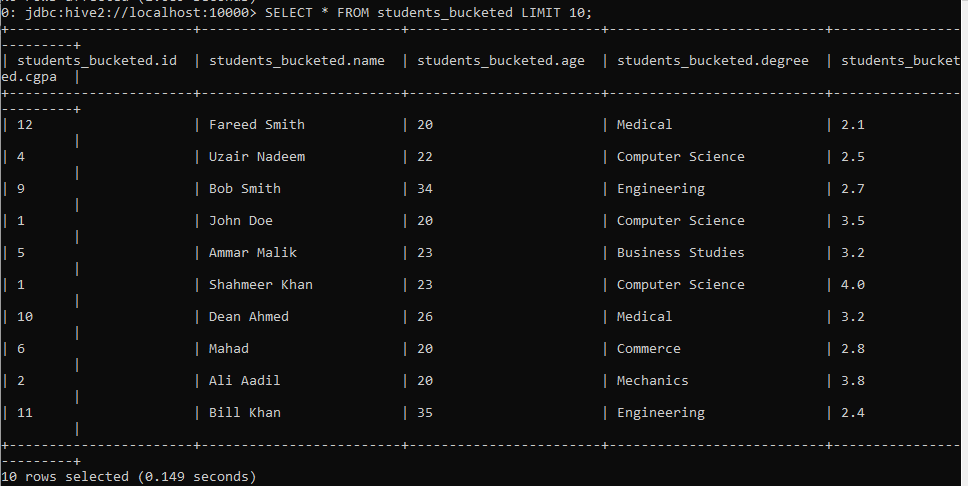


**Bucketing**

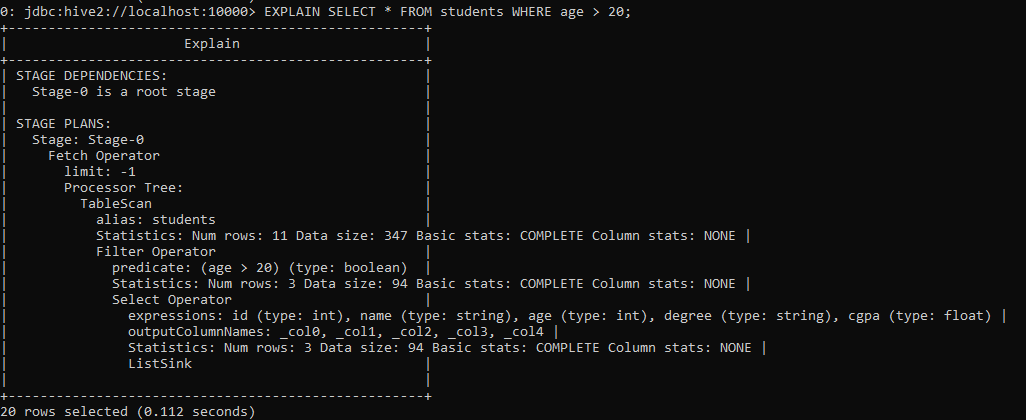


**Checking bucketed data**

We can see here the order of rows has changed based on the way it was bucketed



**Explain query plan**

****

Stage 0 is where the execution starts from

In stage zero the fetch operator fetches data from the table, limit = -1 means to fetch all matching rows

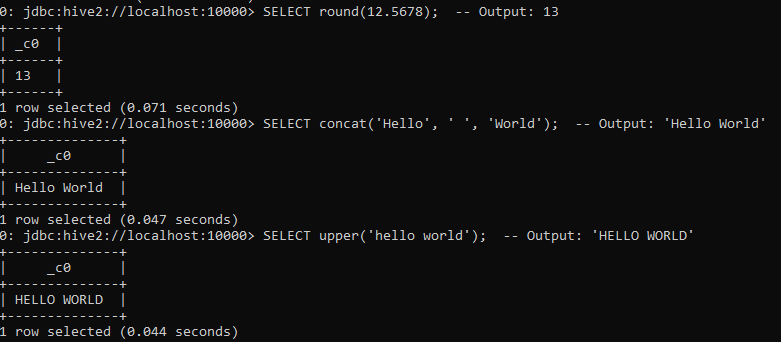
Table scan scans the table for info

It gives stats on no of rows, size of data and if scan is complete

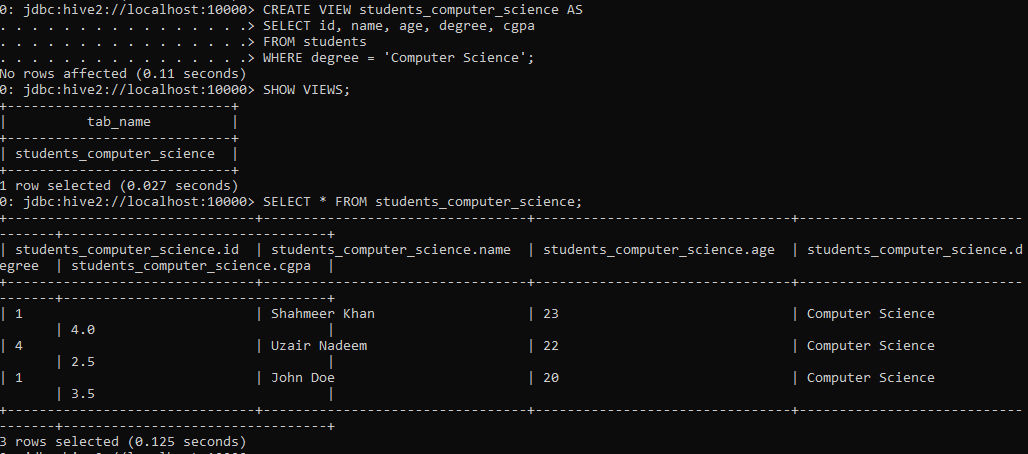
Filter operator displays the condition we set for the query

It displays the number of rows that match the query and the size of that data

**Built In functions**

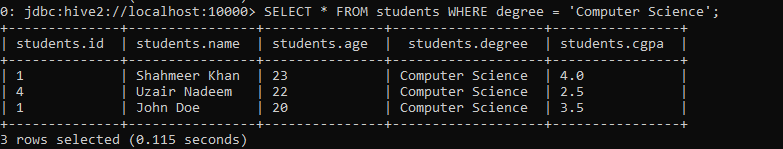


**View**

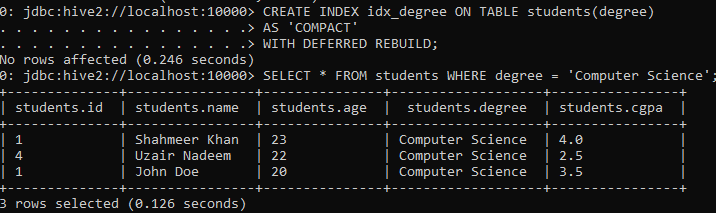


**Indexing**

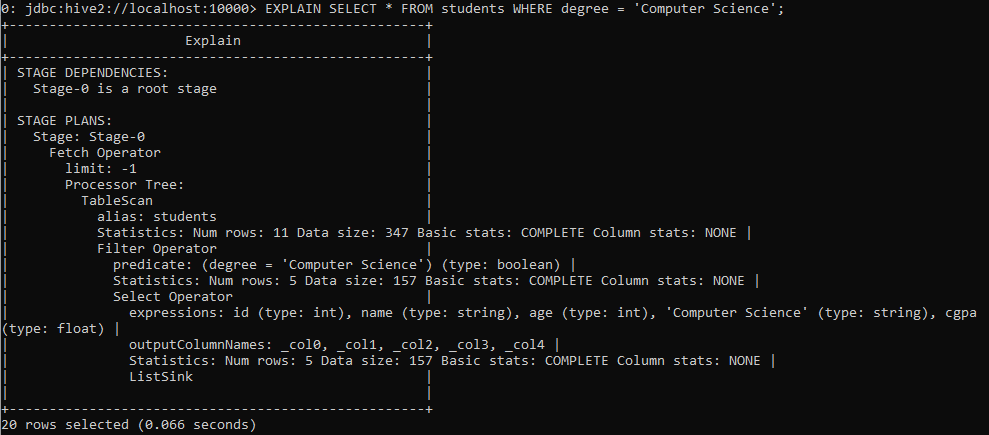
Without indexing



**After indexing**



Indexing should improve speed but it appears indexing wasn’t availed when checking explain command

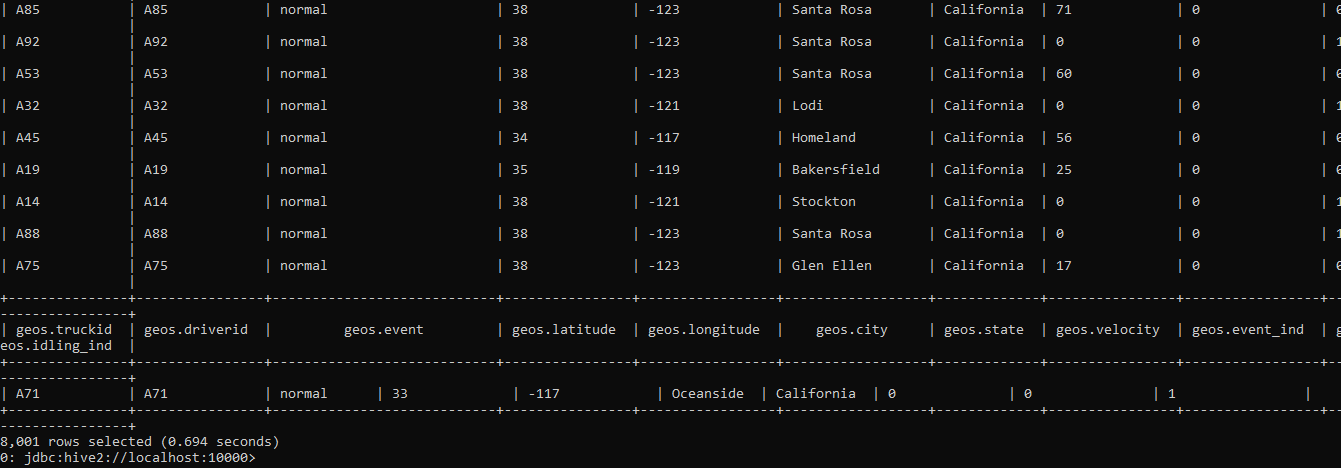


Could be due to query execution taking such little time that indexing wasn’t needed.

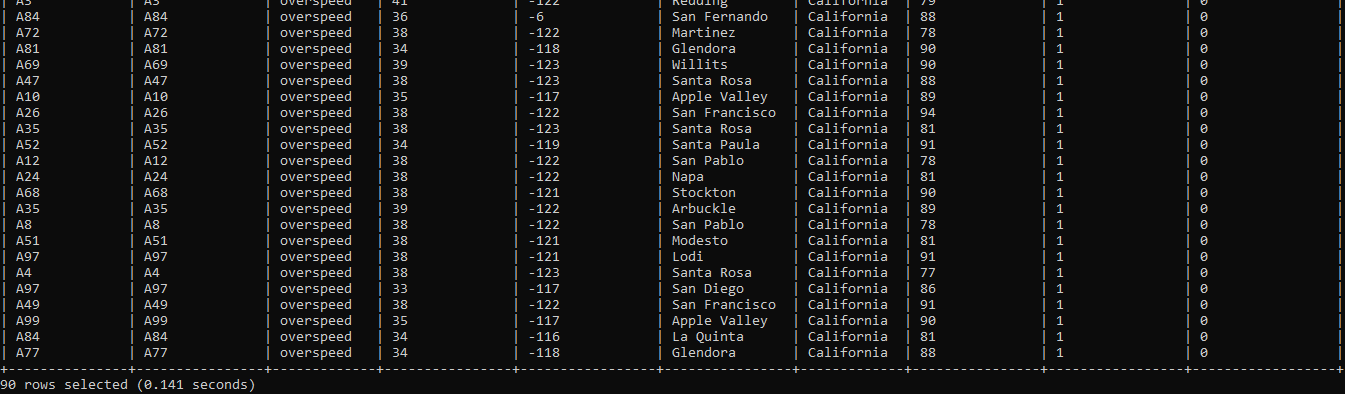
**Geo Location Table**

Note: Too many records so had to crop out till the last few

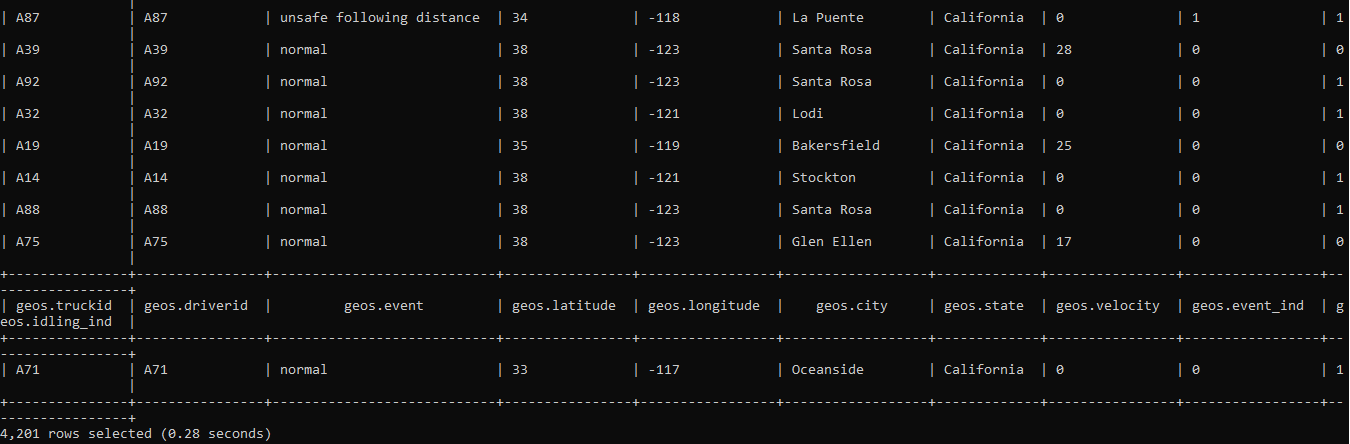
**select \* from geos;**



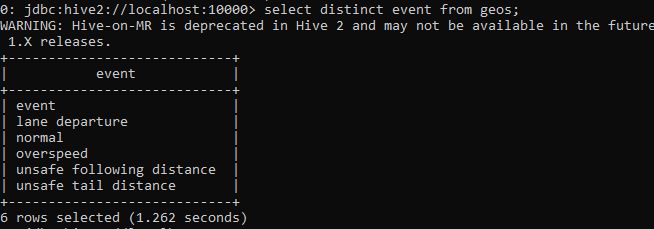
**select \* from geos where event = "overspeed";**



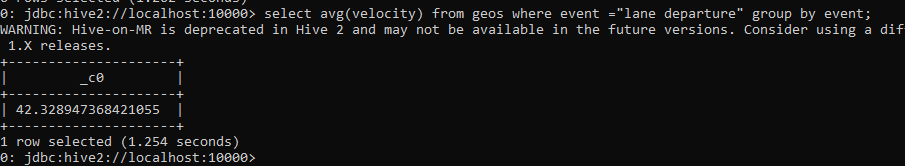
**select \* from geos where velocity < 40;**



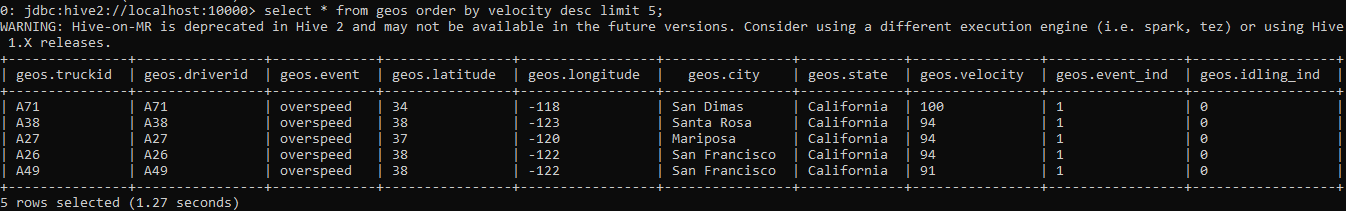
**select distinct event from geos;**



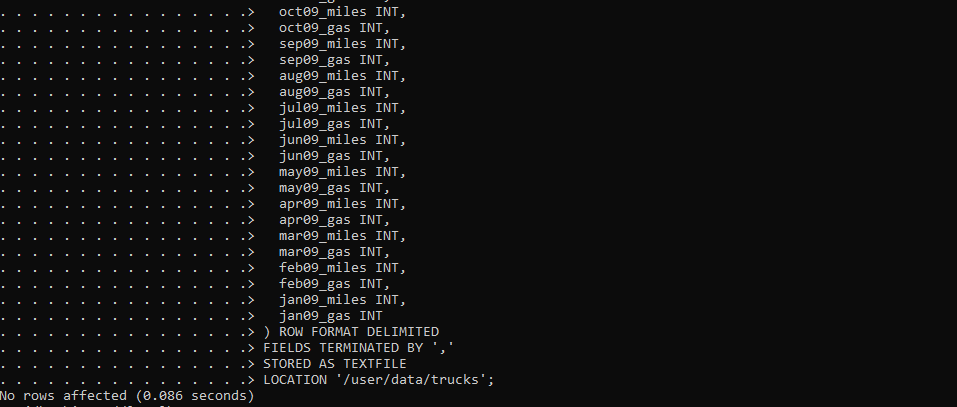
**select avg(velocity) from geos where event ="lane departure" group by event;**



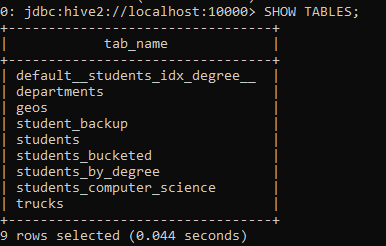
**select \* from geos order by velocity desc limit 5;**



**Trucks table**

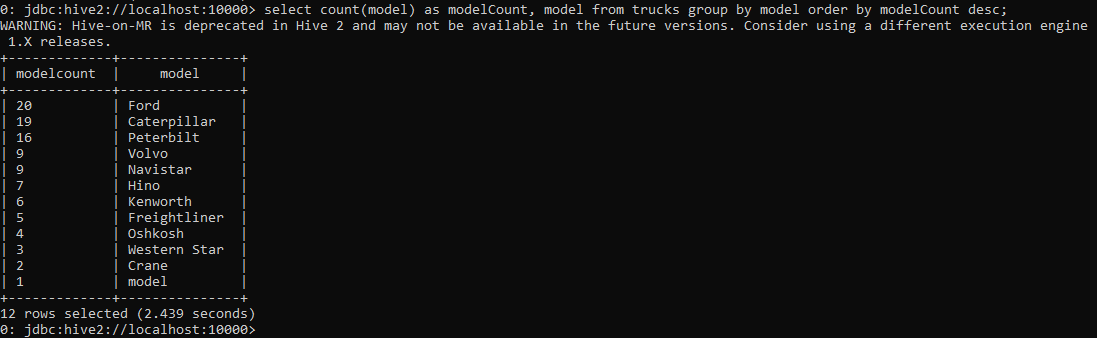


**Table created**



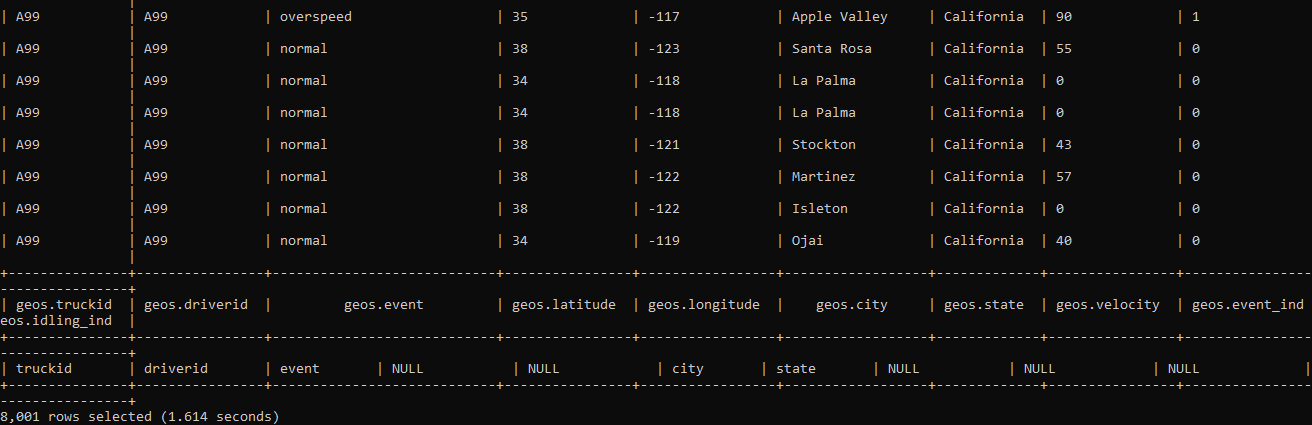
This allows us to have a table that points directly to externally stored data, this allows for benefits like persistence i.e. data won’t be lost if external table is dropped, easier data sharing and even support for partitioning.

**select count(model) as modelCount, model from trucks group by model order by modelCount desc;**



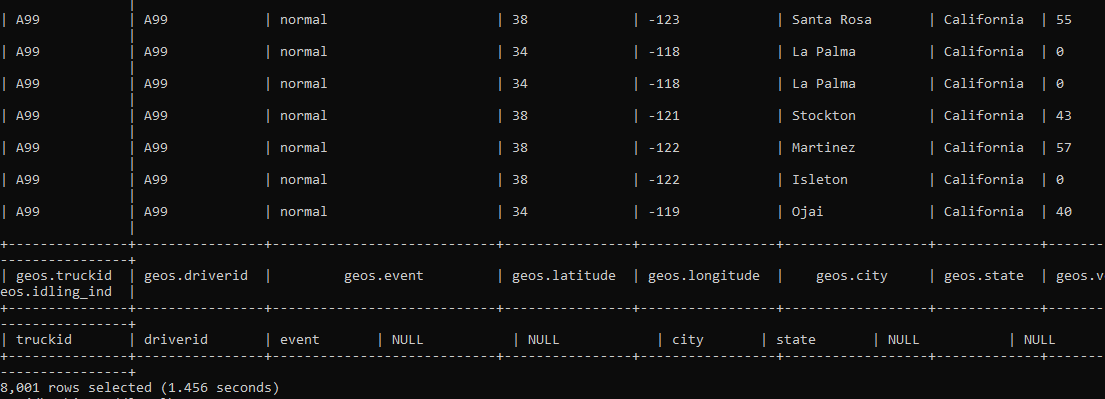
Counts the number of trucks for each model and displays it in descending order of count.

**select \* from geos ORDER BY truckid;**



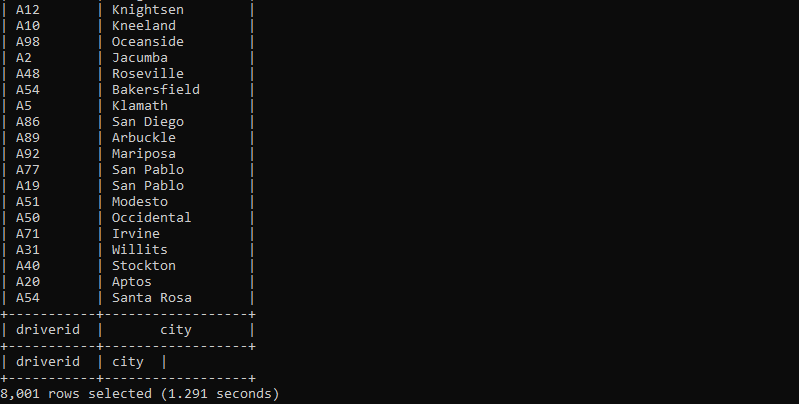
Displays all records in the geos table ordered by truck id in ascending order by default.

**select \* from geos SORT BY driverid ASC;**



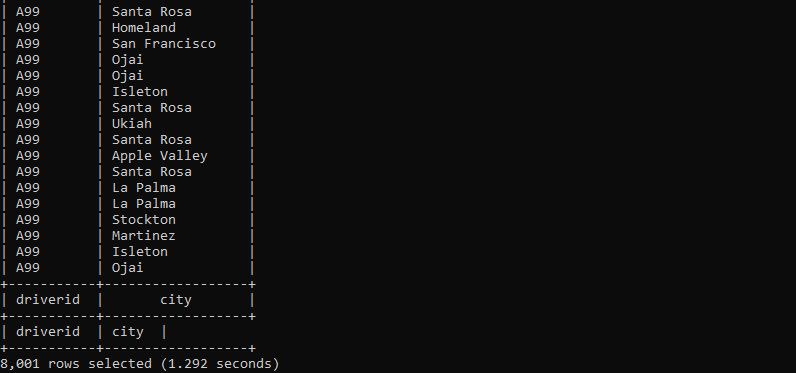
Displays all records in the geos table ordered by driver id but in ascending order.

**select driverid, city from geos DISTRIBUTE BY driverid;**



Distribute by assigns data to reducers that group similar data, this allows for parallel processing across multiple reduces

**select driverid, city from geos CLUSTER BY driverid;**



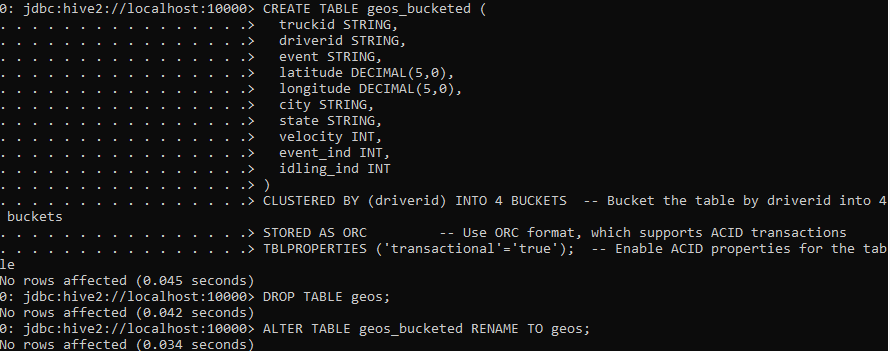
Clustering ensures that rows with the same value for the specified column (driver id) are sent to the same reducer, this enables parallel sorting

**UPDATE geolocation SET driverid = null WHERE event = “normal”;**

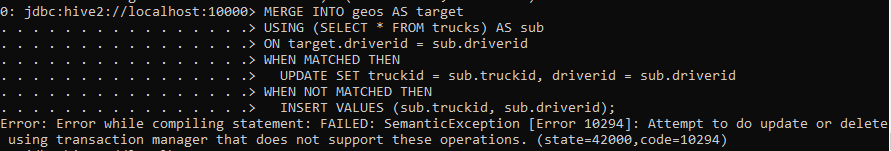


Error due to ACID properties not being enabled for the table

**Using bucketing to enable ACID**

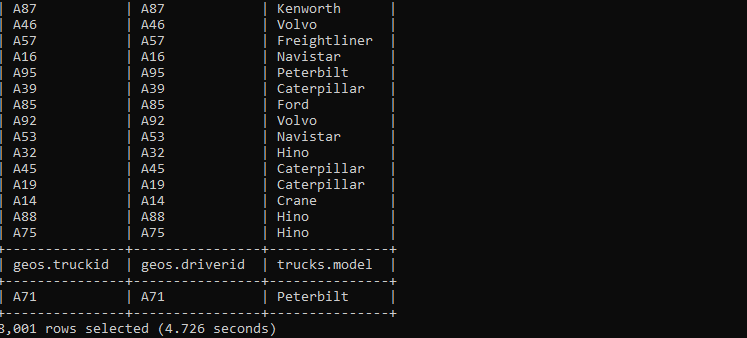


Still getting errors for merge and update



**Display truckid, driverid and model for every abnormal event**

select geos.truckid,geos.driverid,trucks.model from geos join trucks where geos.driverid=trucks.driverid;



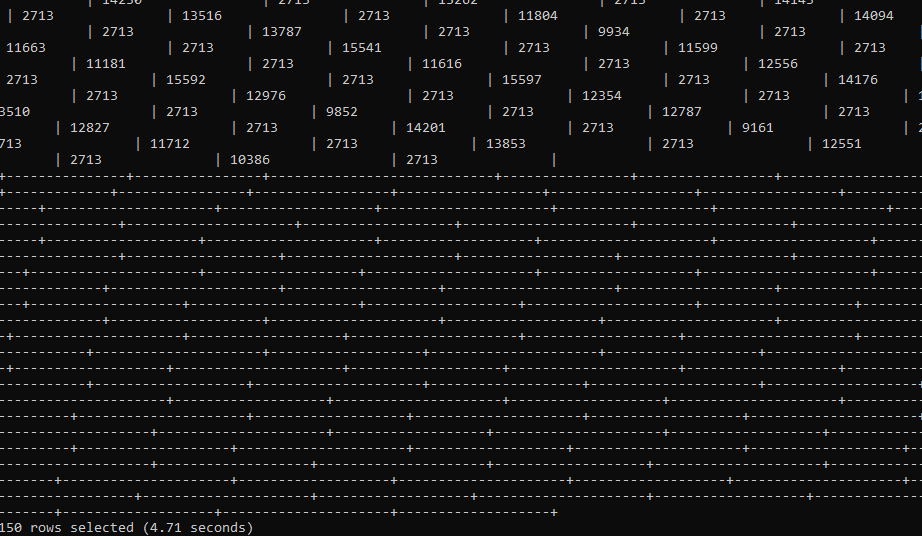
**Display truckid, driverid, model and city for velocity > 25**

SELECT geos.truckid, geos.driverid, trucks.model, geos.city FROM geos JOIN trucks ON geos.driverid = trucks.driverid WHERE geos.velocity > 25;



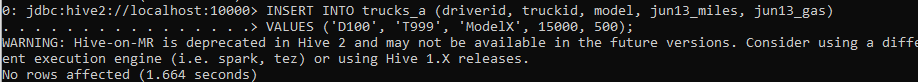
**Display complete record of the trucks for events where they were on unsafe following distance**

SELECT \*FROM geos JOIN trucks ON geos.driverid = trucks.driverid WHERE geos.event = 'unsafe following distance';

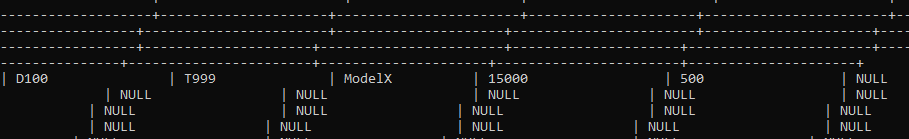


**ACID commands on trucks table**

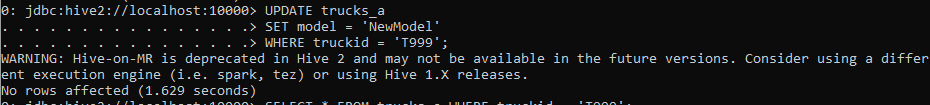
Insert



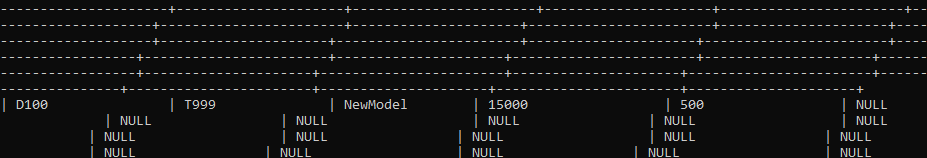
Viewing insert



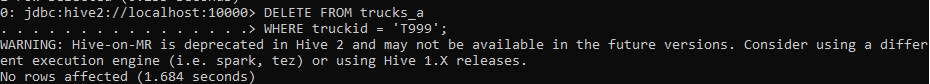
Update



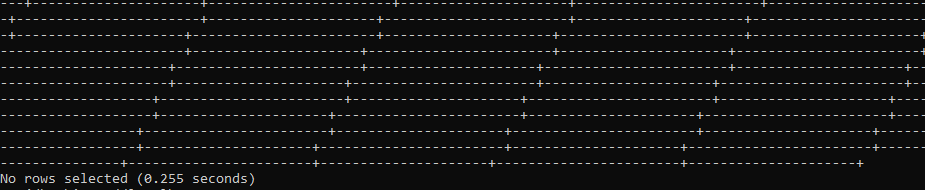
Viewing update



Delete



Verifying delete



No rows found after select query

Performance difference: Acid tables took longer to execute queries but offered ACID properties