**Assignment 27.3**

**Problem Statement:**

Explain the key concepts of Bucketing and perform bucketing operations using our attached Blog. Share and explain the commands used with the final result.

**Solution:**

* **Bucketing in Hive:**

Hive partition divides table into number of partitions and these partitions can be further subdivided into more manageable parts known as Buckets or Clusters. The Bucketing concept is based on Hash function, which depends on the type of the bucketing column. Records which are bucketed by the same column will always be saved in the same bucket.

Here, CLUSTERED BY clause is used to divide the table into buckets.

In Hive Partition, each partition will be created as directory. But in Hive Buckets, each bucket will be created as file.

Bucketing can also be done even without partitioning on Hive tables.

**Advantages of Bucketing:**

Bucketed tables allows much more efficient sampling than the non-bucketed tables. With sampling, we can try out queries on a section of data for testing and debugging purpose when the original data sets are very huge. Here, the user can fix the size of buckets according to the need.

Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. Since the data files are equal sized parts, map-side joins will be faster on the bucketed tables.

* **Bucketing operations:**

Let us see how to create and populate bucketed table in the following example, where we try to subdivide a partition table into multiple manageable parts based on a field by using bucketing technique.

To perform the bucketing operation on a dataset, we require an input dataset. You can refer to the below link for our input dataset real\_state:

<https://drive.google.com/open?id=0B_P3pWagdIrrQzk2Q3dRUFp3YVU>

Dataset Description:

Column 1: Street

Column 2: City

Column 3: Zip

Column 4: State

Column 5: Beds

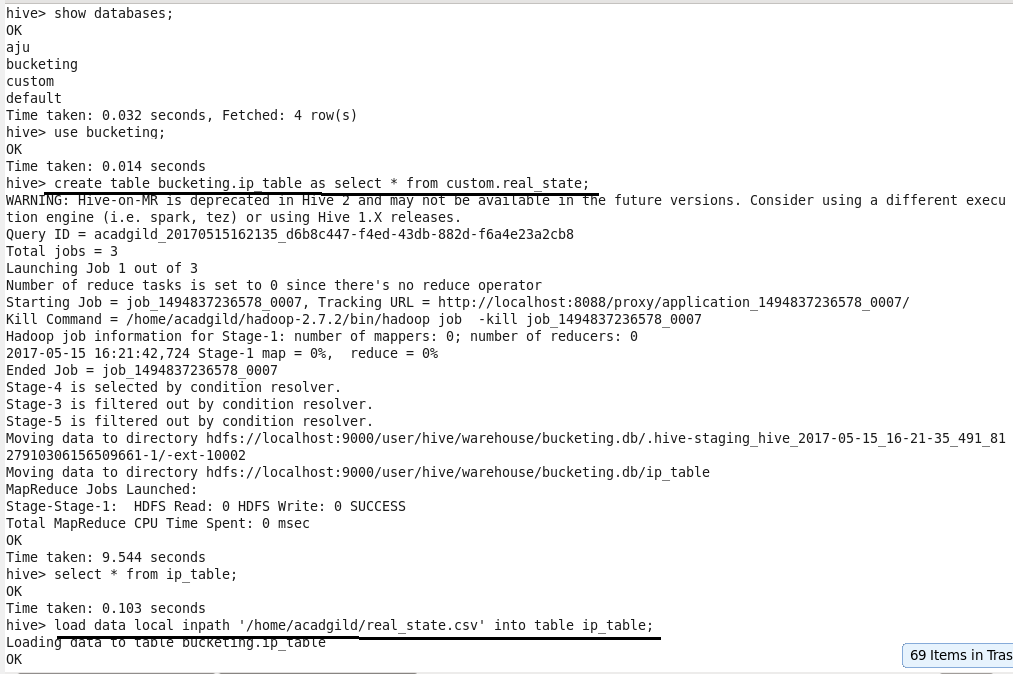
Column 6: Baths

Column 7: Sq\_feet

Column 8: flat\_type

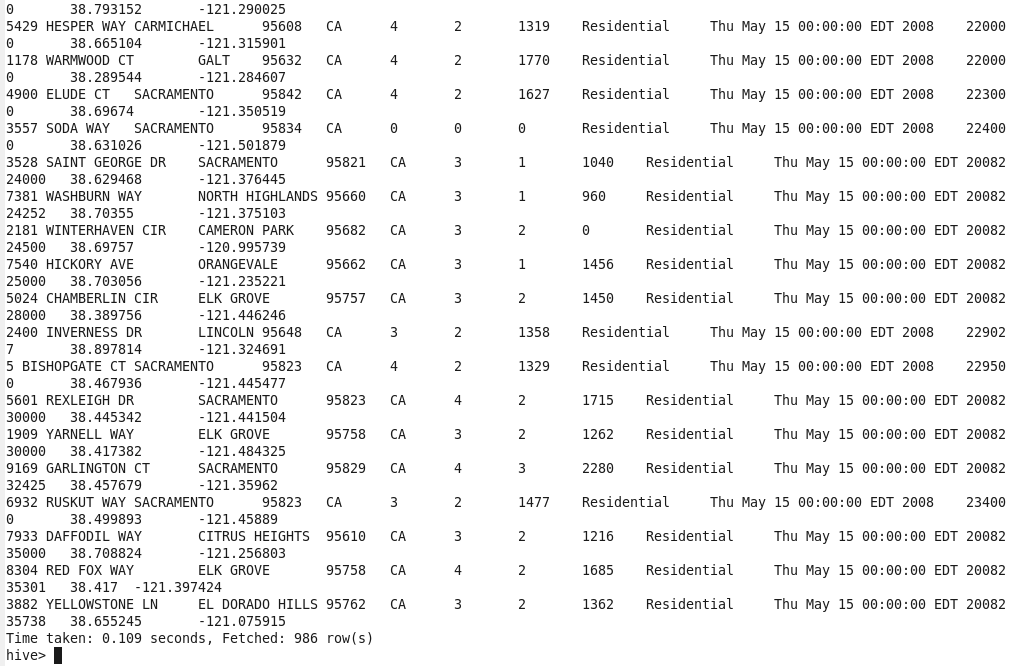
Column 9: Price

* **Selecting the Database**
* **Creating a New Input Table**
* **Load the Input Dataset**

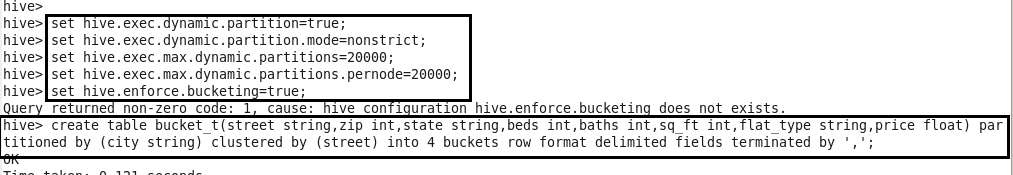


* **Display the Contents of Table input\_table to Ensure Whether the Input File has been Loaded Successfully or Not**

C:\Users\612777\Music\2.png



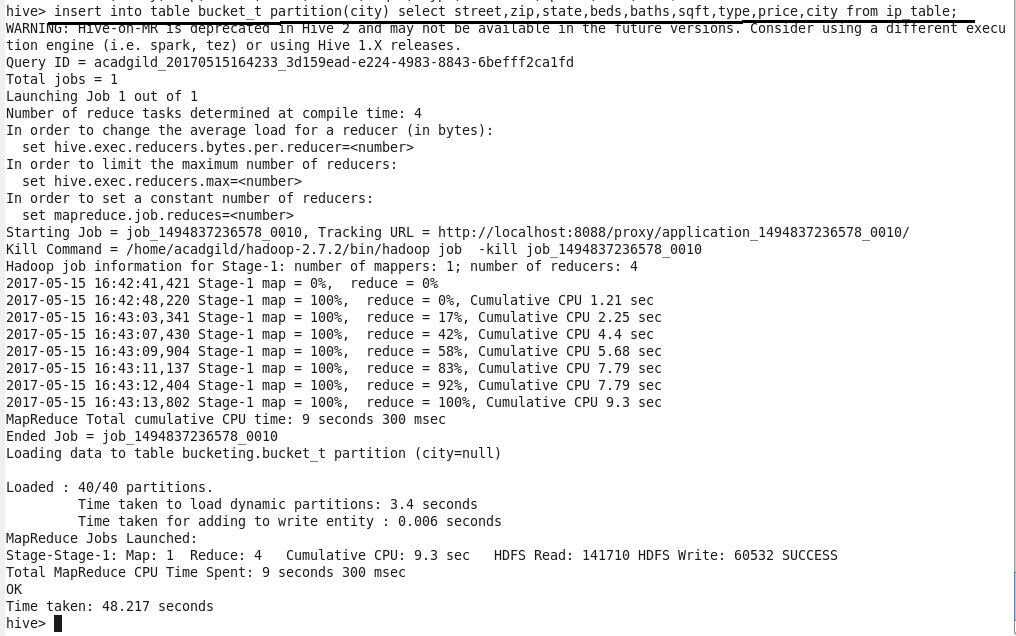
* **Set the Below Properties in Hive Command Line Before Proceeding Further for Bucketing Scripts**
* **Creating Bucket Table**



To populate the bucketed table, we have to set **hive.enforce.bucketing** property to ‘true’, so that the Hive knows to create the number of buckets declared in the table definition.

The property **hive.enforce.bucketing = true**is similar to **hive.exec.dynamic.partition = true**, in Hive partitioning. By setting this property, we will enable dynamic bucketing while loading data into the Hive table.

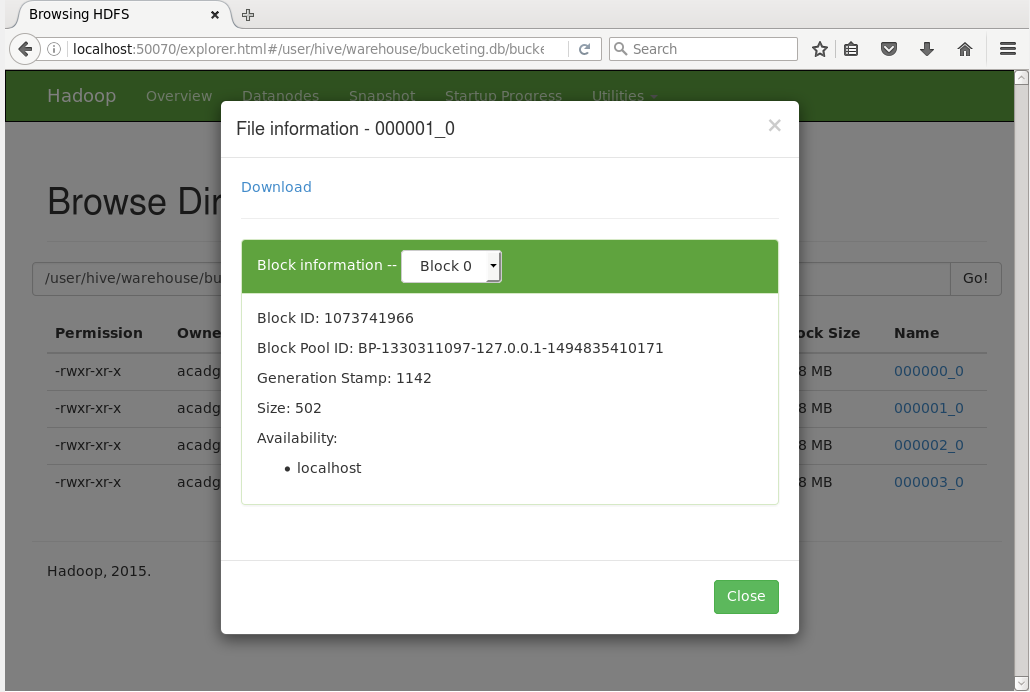
* **Query to Retrieve Data from Bucketed Table**

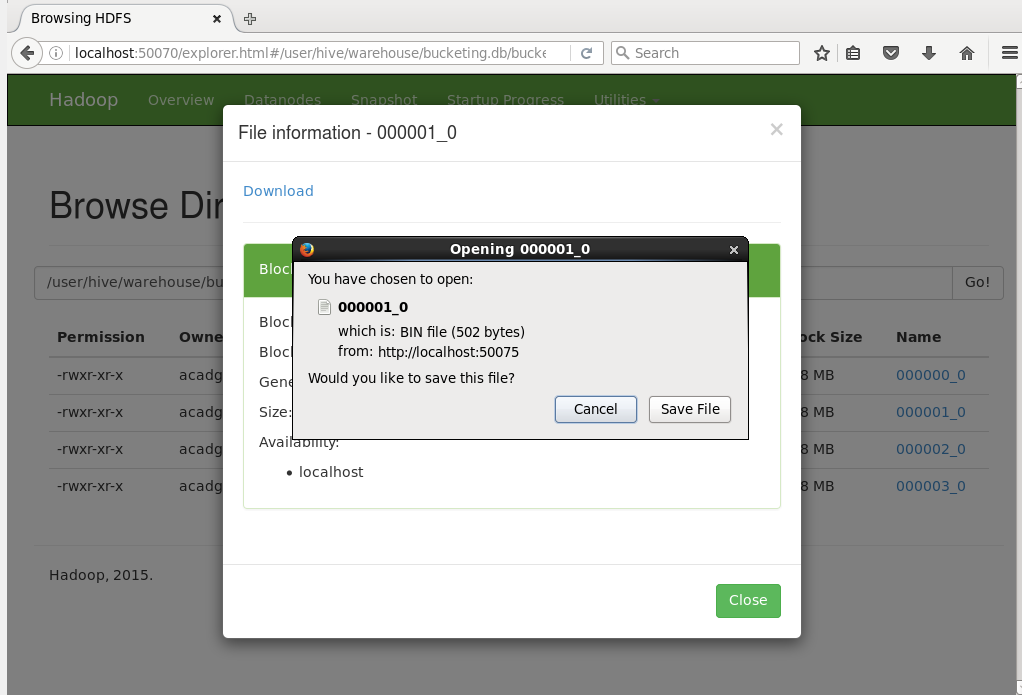


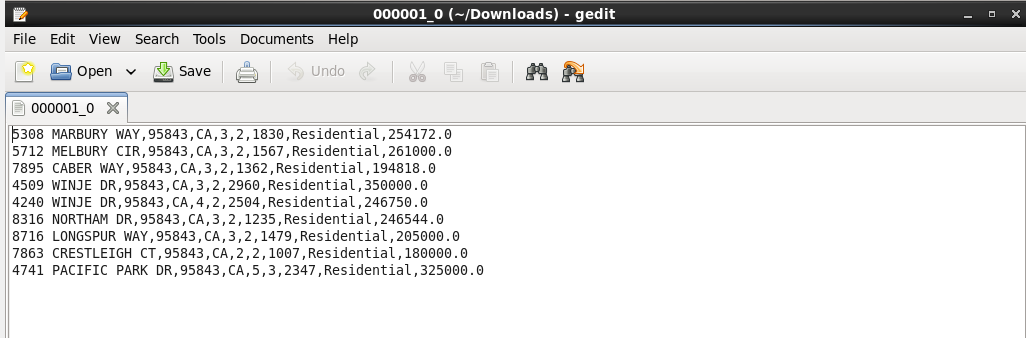
We can observe from the above image that we are selecting the columns from input\_table and inserting it into our bucketed table ‘bucket\_table’, which is partitioned by city.

After successfully inserting the contents of input\_table to bucket\_table, we can see the bucketed output result in the browser and can also download the required city partitioned bucketed files in our local file system.

* **Browse Directory**







We can observe from the above image, the part file is successfully downloaded in the local file system downloads directory.

It is clear that Bucketing is most suitable for sampling purpose and adds some optimization to the query performance.

We can also execute the above output in command line by copy pasting the same or different part file path, according to the user’s requirement.

