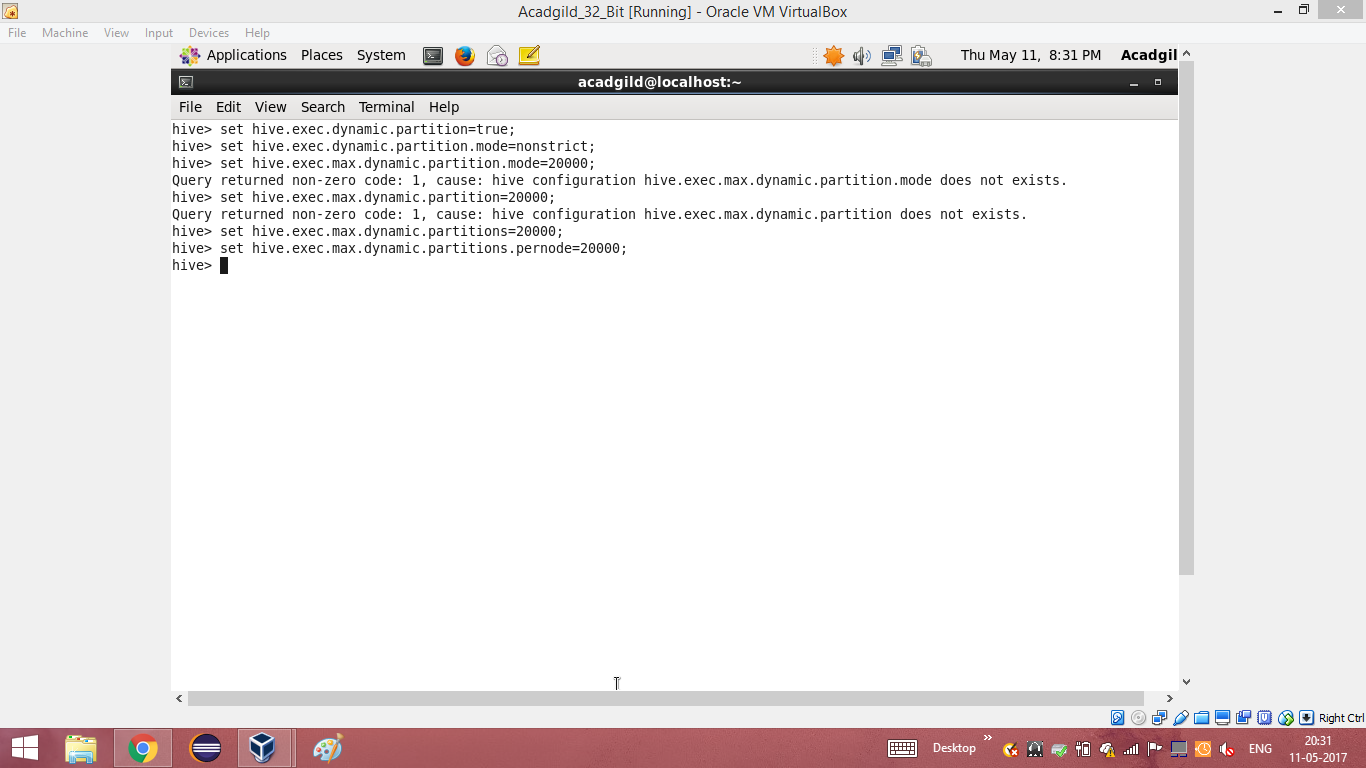
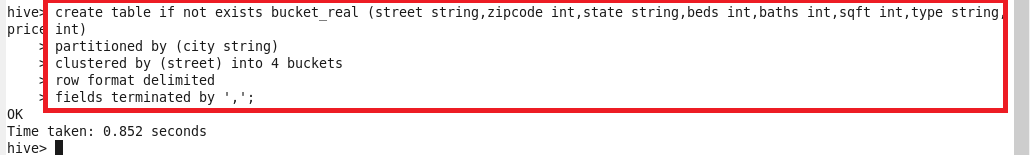
**Bucketing operation:**

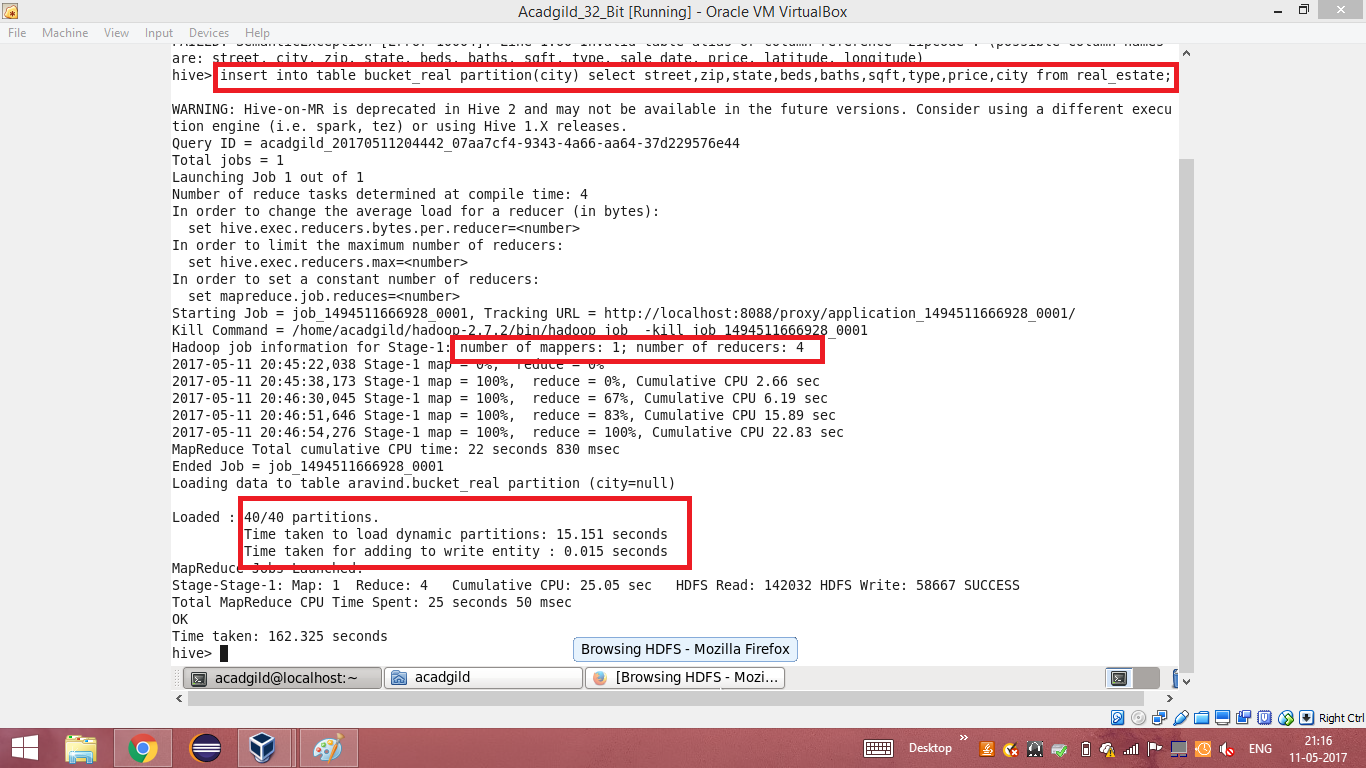
**Step 1: setting configurations in hive shell**



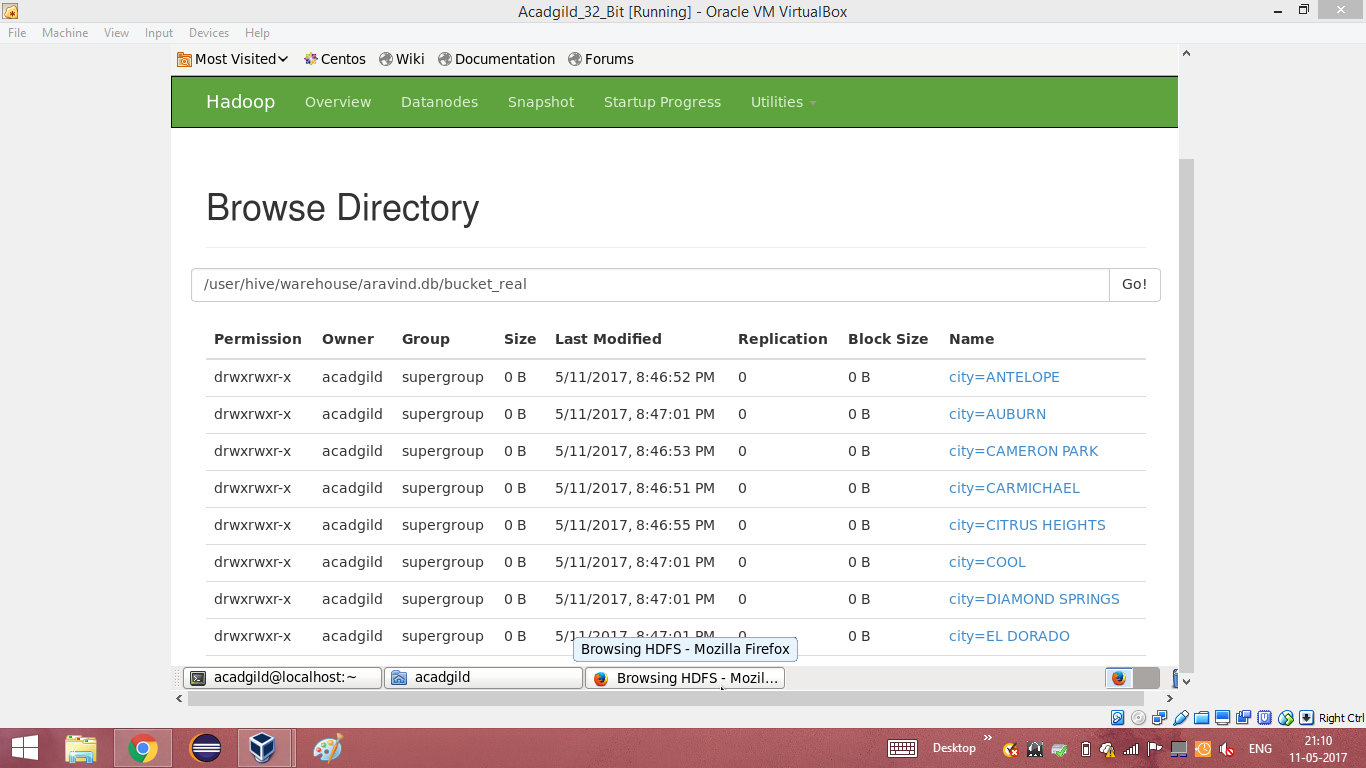
**Step 2: creating a bucketed table**



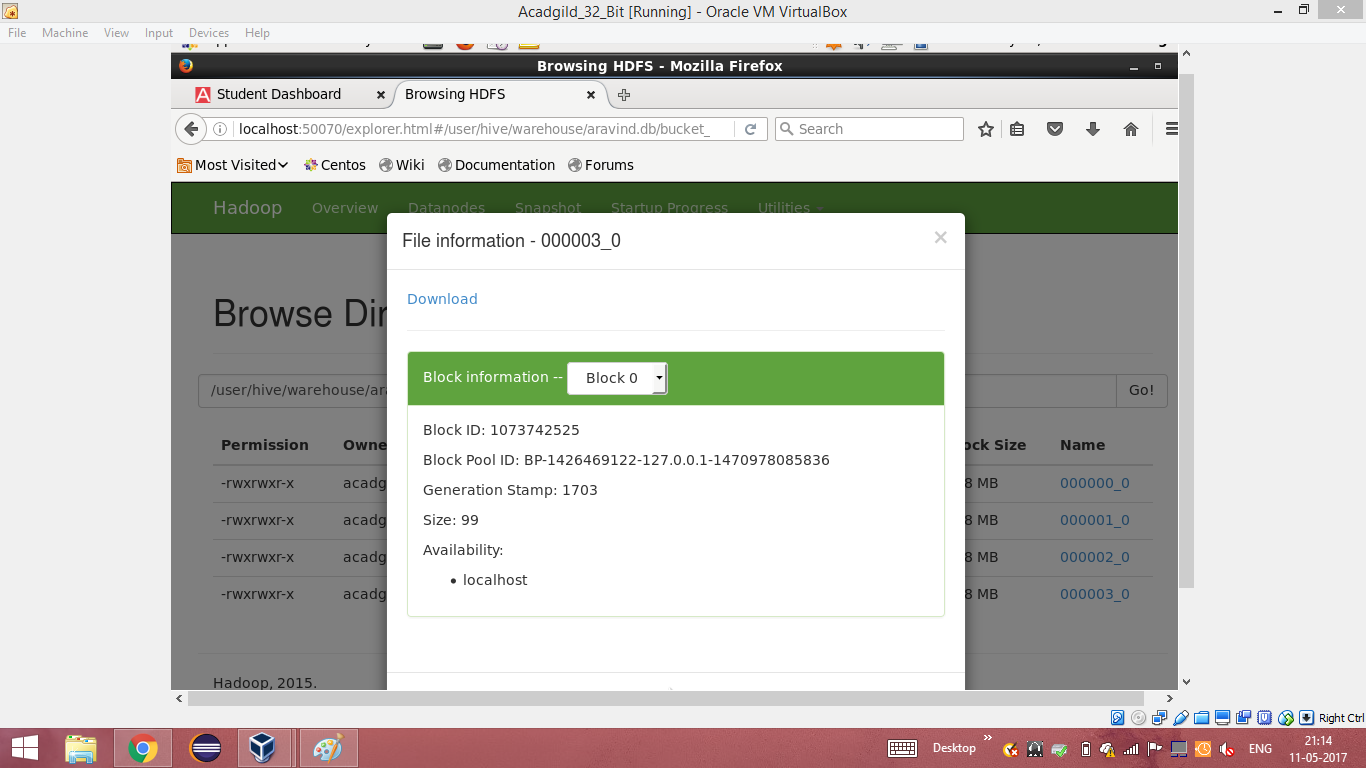
**Step 3: inserting into the bucketed table**



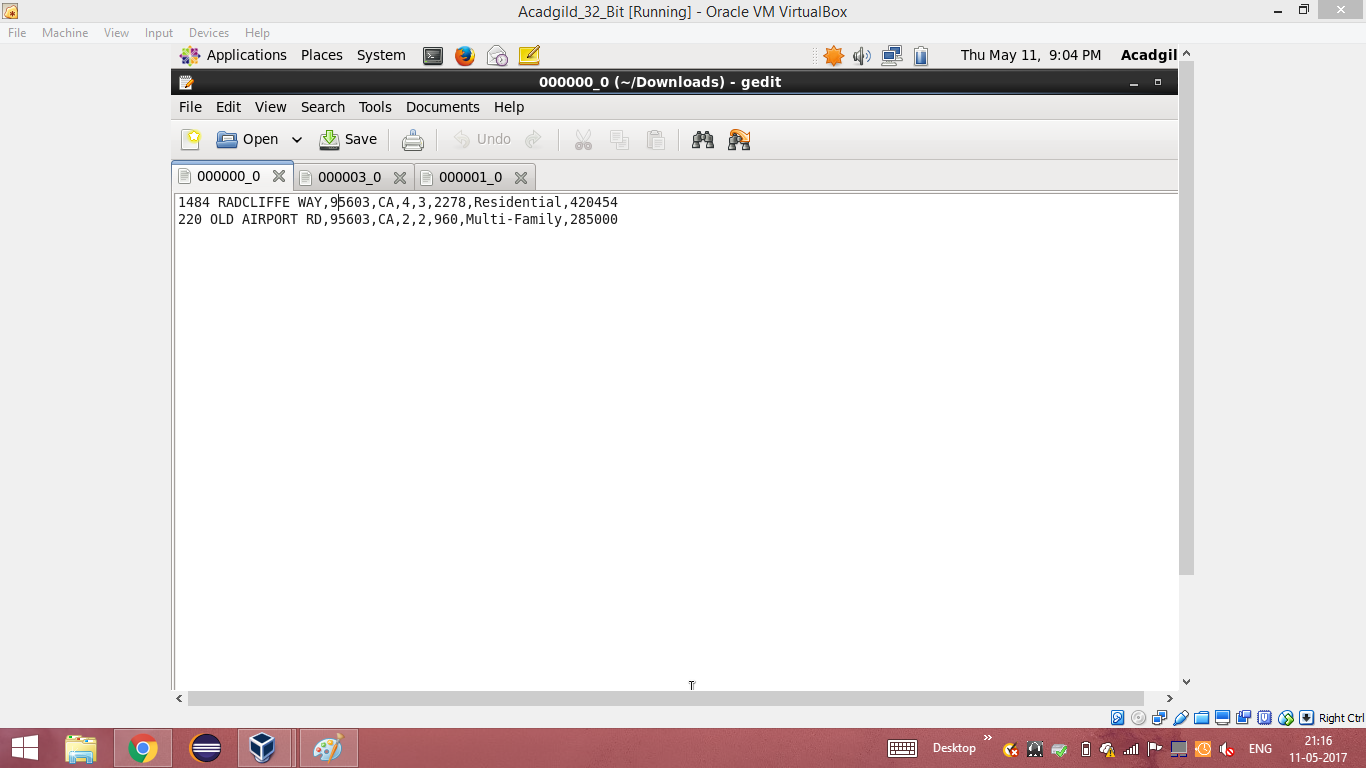
**Step 4: browsing the directory**



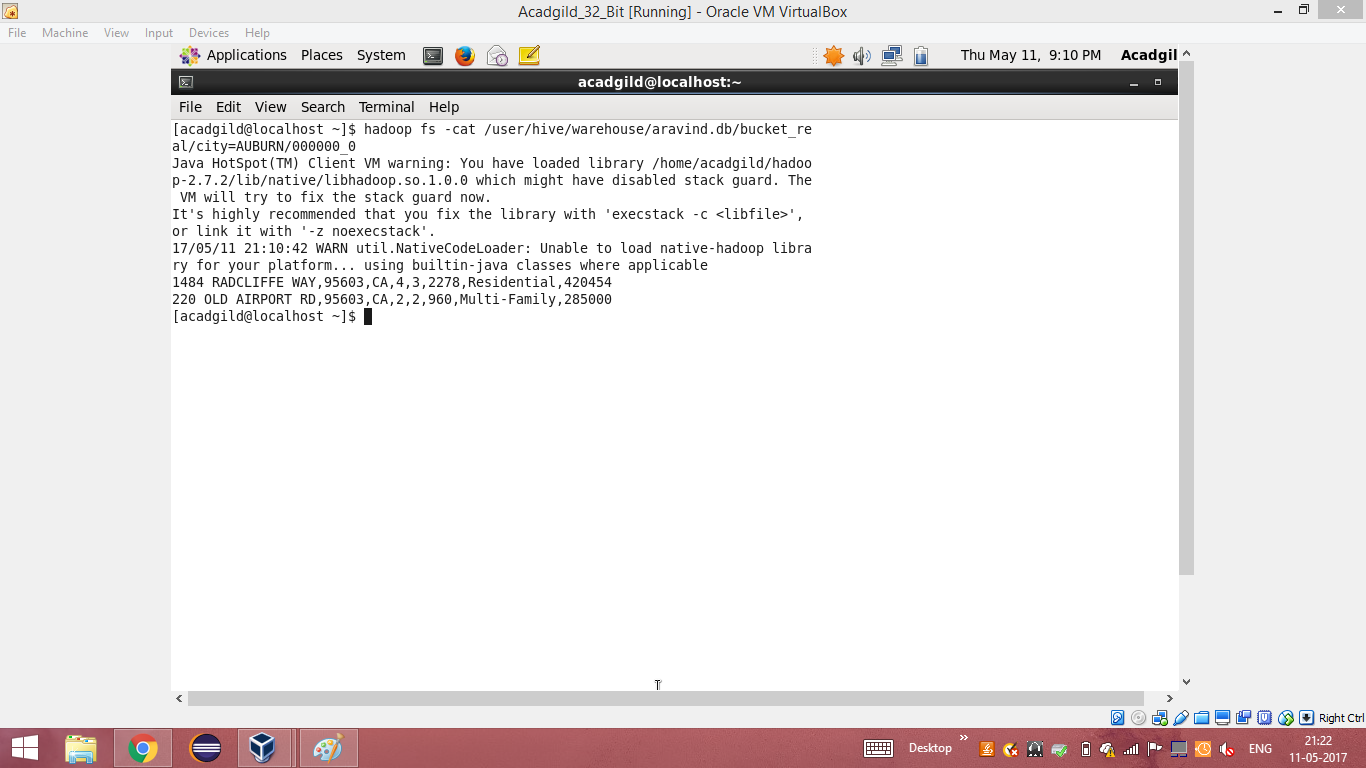
**Step 5: downloading the partitioned data**



**Step 6: viewing the partitioned data**



**Step 7: viewing the data using cat command.**



**FEATURES:**

* Bucketing concept is based on (hashing function on the bucketed column) mod (by total number of buckets).
* The hash\_function depends on the type of the bucketing column.
* Records with the same bucketed column will always be stored in the same bucket.
* We use CLUSTERED BY clause to divide the table into buckets.
* Physically, each bucket is just a file in the table directory, and Bucket numbering is 1-based.
* Bucketing can be done along with Partitioning on Hive tables and even without partitioning.
* Bucketed tables will create almost equally distributed data file parts.

**ADVANTAGES:**

* Bucketed tables offer efficient sampling than by non-bucketed tables. With sampling, we can try out queries on a fraction of data for testing and debugging purpose when the original data sets are very huge.
* As the data files are equal sized parts, map-side joins will be faster on bucketed tables than non-bucketed tables. In Map-side join, a mapper processing a bucket of the left table knows that the matching rows in the right table will be in its corresponding bucket, so it only retrieves that bucket (which is a small fraction of all the data stored in the right table).
* Similar to partitioning, bucketed tables provide faster query responses than non-bucketed tables.
* Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. This makes map-side joins even more efficient, since the join of each bucket becomes an efficient merge-sort.