# Use of Single Scan

* + Hive does complete table scan, so it's recommend to use single scan to perform multiple operations.

INSERT INTO page\_views\_20140101 SELECT \* FROM page\_views WHERE date='20140101';

And

INSERT INTO page\_views\_20140102 SELECT \* FROM page\_views WHERE date='20140102';

Here the processor does complete table scan for two times. To avoid this, we can execute following query

FROM page\_views

INSERT INTO page\_views\_20140101 SELECT \* WHERE date='20140101'

INSERT INTO page\_views\_20140102 SELECT \* WHERE date='20140102'

# Partitioning Tables

* + Low cardinal attribute should be selected as Partition key.(column which have low set of values eg Male Female)
  + Partitioning store data in sub directory under table location.
  + Make SQL faster which queried on Partitioned keys.

# De-normalize Data

* + Normalise data will have multiple relational table and it will increase the number of joins.
  + De Normalise data to avoid multiple joins which in turn will increase the performance.

# Compress Map Reduce Output

* + Compression techniques significantly reduce the intermediate data volume.
  + Compression can applied on mapper and reducer output.
  + Gzip is not splittable so it should be applied to small files.
  + Compressed file size should not be larger than few hundred Megabytes.

1. Map Join
   * Map join is efficient if other table is small enough to fit into memory.
   * Map join is efficient if other table is small enough to fit into memory.
   * One way of Map join is use of hint **SELECT /\*+ MAPJOIN(c) \*/ \* FROM orders o JOIN cities c ON (o.city\_id = c.id);** Table ‘c’ would be loaded into memory.
   * Another way to turn on map joins is to let Hive do it automatically by setting hive.auto.convert.join to true, and Hive will automatically use map joins for any tables smaller than hive.mapjoin.smalltable.filesize
2. Bucketing
   * It improves the performance if bucket key and join keys are common.
   * Use Bucketing when it is difficult to create partition on a column as it would be having huge variety of data in that column on which we want to run queries.
   * Set bucketing flag to true every time before writing the data (SET hive.enforce.bucketing=true).
   * Set optimize flag true to use bucket level join (SET hive.optimize.bucketmapjoin=ture).
3. Input Format Selection
   * Prefer RCFile, ORC input format over text,json input format.
4. Parallel Execution
   * It is recommended to set parallel flag true SET hive.exec.parallel=true
5. Vectorization
   * It is recommended to set vectorization flag true SET hive.vectorized.execution.enabled=true
6. Unit Testing
   * Hive query should be running in local mode for unit testing.
7. Sampling
   * Use TABLESAMPLE clause to get subset of data for analysis.
8. Security –
   * Data analysts accessing only Hiveserver2, with limited access to HDFS files
   * Data analysts accessing both Hiveserver2, and HDFS files through Pig/MR jobs
   * Data analysts accessing Hive CLI
9. Correlation Optimizer
10. MultiTable Inserts
11. Cost Based Optimizer
12. Predicate Push Down
    * Process expressions as early in the plan as possible.
    * The default plan generation adds filters where they are seen but in some instances some of the filter expressions can be pushed nearer to the operator that sees this particular data for the first time.
    * select a.\*, b.\* from a join b on (a.col1 = b.col1) where a.col1 > 20 and b.col2 > 40 for this query, the predicates (a.col1 > 20) and (b.col2 > 40), without predicate pushdown, would be evaluated after the join processing has been done.
    * Suppose the two predicates filter out most of the rows from a and b, the join is unnecessarily processing these rows.
    * With predicate pushdown, these two predicates will be processed before the join.
    * Predicate pushdown is enabled by setting hive.optimize.ppd to true. It is disable by default.
13. Preventing run-away queries
    * A strict value for this variable indicates that an error is thrown by the compiler in case no partition predicate is provided on a partitioned table. This is used to protect against a user inadvertently issuing a query against all the partitions of the table.
    * SET hive.partition.pruning=nonstrict
14. Intermediate Compression
    * Determines whether the output of the intermediate map/reduce jobs in a query is compressed or not.
    * SET hive.exec.compress.intermediate=false
15. Indexing