**Task 1**

**Data:** Dataset1

**Instructions:**

* Create database targetdb at HDFS location /data/target/targetdb
* Create tables: movies, ratings, tags in targetdb database
* Load the relevant data files to those tables
* Prepare a SQL to load movie\_stats table.

Following is the column detail for movie\_stats table.

movieid, movie\_name,avg\_rating,hash\_tag,hash\_tag\_cnt

**Note**: Choose appropriate column data type as per your knowledge.

CREATE DATABASE targetdb LOCATION ‘/data/target/targetdb’

● Create tables: movies, ratings, tags in targetdb database

create table movies (movieId int, title string, genres string) row format delimited fields terminated by ',';

create table ratings (userId int,movieId int,rating double,t\_stamp decimal(12,0)) row format delimited fields terminated by ',';

create table tags (userId int,movieId int,tag string,t\_stamp decimal(12,0)) row format delimited fields terminated by ',';

● Load the relevant data files to those tables

hadoop fs -put Desktop/data/Dataset1/movies.csv /data/target/targetdb/movies/

hadoop fs -put Desktop/data/Dataset1/ratings.csv /data/target/targetdb/ratings/

hadoop fs -put Desktop/data/Dataset1/tags.csv /data/target/targetdb/tags/

OR

hadoop fs -chmod -R 777  /data

hadoop fs -mkdir '/data/input/impala/'

hadoop fs -put Desktop/data/Dataset1/movies.csv /data/input/impala/

load data inpath '/data/input/impala/' into table movies;

● Prepare a SQL to load movie\_stats table.

create TABLE movie\_stats (movieid int, movie\_name string, avg\_rating decimal(9,2), hash\_tag string, hash\_tag\_cnt int) row format delimited fields terminated by ','

INSERT INTO movie\_stats

Select t1.\*,  t2.tag, t2.tag\_cnt from

(select m.movieid, m.title, CAST(avg(r.rating) AS DECIMAL(9,2)) from movies m join ratings r on m.movieid = r.movieid  group by m.movieid, m.title) t1

JOIN

(select  movieid, tag, CAST(count(tag) AS INT) as tag\_cnt from tags group by movieid, tag) t2

ON

T1.movieid = t2.movieid

(t2 can be replaced just by simple inner join)

#### Method2:

**Reference only: Its not going to give you correct result**

SELECT m.movieid, m.title, avg(r.rating) over(partition by m.movieid), tag, count(tag) over(partition by t.movieid, t.tag)

FROM

    Movies m

JOIN

    Ratings r

ON

    M.movieid = r.movieid

JOIN

    Tags t

ON

    M.movieid = t.movieid

\*\*\*\* sql windows function will give many more duplicates ( t1 \* t2 \* t3)

=======================================================================

**Task 2**

Let’s apply some integrity constraints to our table

* Modify movies table to make column movieId UPI and NOT NULL

UPI: unique primary index (primary key). Impossible, because we don’t have constrains in Hive or Impala

**EXTRA:**

create table t1 (tx\_id int, name string) partitioned by (tx\_date string) row format delimited fields terminated by ',';

insert into t1 partition(tx\_date='2019-01-01') values(1,'Cookies');

**Task 3**

**Data:** Dataset2

**Instructions:**

* Create new database stagedb at location /data/staging/stagedb
* Create tables movies, ratings, tags under database stagedb with same details as targetdb tables

create table movies like targetdb.movies;

*Do not create this way, it doesn’t copy the exact format*

create table ratings as select \* from targetdb.ratings limit 0

create table ratings like targetdb.ratings;

create table tags like targetdb.tags;

● Load the relevant data files to those tables

hadoop fs -put Desktop/data/Dataset2/movies.csv /data/stage/stagedb/movies/

hadoop fs -put Desktop/data/Dataset2/ratings.csv /data/stage/stagedb/ratings/

hadoop fs -put Desktop/data/Dataset2/tags.csv /data/stage/stagedb/tags/

* Prepare three different scripts, one for each to load stage tables with given data files considering all possible integrity checks as
  + Movie ID is unique and not null field. Hence have to check for any unique key as well as not null violation

select count(\*) from movies where movieid is null;

select movieid, count(\*) from movies group by movieid having count(\*)>1;

select count(\*) from movies m1 join targetdb.movies m2 on m1.movieid = m2.movieid;

How to run

impala-shell -f movies.hql -d stagedb

* + Rating should be for a valid movie. Hence movieId in rating data file should be present in Movie table
  + If there is any exact duplicate record then we must discard one.

Insert overwrite movies select distinct from movies;

=======================================================================

**Task 4**

**Instructions:**

* Prepare full refresh load script to refresh movie\_stats table

=======================================================================

**Task 5**

**Data:** Dataset 3

**Instructions:**

* Movies table is Slowly Changing Dimension (SCD) type 2 table.
* Generally, we receive Insert/Update/Delete flag with source file to load Slowly Changing Dimensions (SCD)
* Modify the movies table load script to implement IUD logic

=======================================================================

**Task 6**

**Instructions:**

We have some clients using MySQL database to generate reports. Hence we need to push our movie\_stats table to that MySQL database.