**Prepare a report detailing the HQL queries for creating Hive tables and the results obtained for the HQL tasks**

**Note: We will also leverage the Hive External Tables (non_event_data and event_data) here that were created for generating the S3 datasets as documented in DataPreparation.pdf**

## Hive connection using beeline:

```
[hadoop@ip-172-31-82-238 ~]$ beeline -u jdbc:hive2://localhost:10000/default -n hadoop

Connecting to jdbc:hive2://localhost:10000/default
Connected to: Apache Hive (version 2.3.4-amzn-2)
Driver: Hive JDBC (version 2.3.4-amzn-2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.4-amzn-2 by Apache Hive
```

## Hive database creation:

```
0: jdbc:hive2://localhost:10000/default> create database if not exists telco_db;
No rows affected (1.378 seconds)


0: jdbc:hive2://localhost:10000/default> show databases;

+----------------+
| database_name  |
+----------------+
| default        |
| telco_db       |
+----------------+
2 rows selected (0.318 seconds)


0: jdbc:hive2://localhost:10000/default> describe database telco_db;

+-----------+----------+----------------------------------------------------+-------------+-------------+----
---------+
|  db_name  | comment  |                      location                      | owner_name  | owner_type  |
parameters  |
+-----------+----------+----------------------------------------------------+-------------+-------------+----
---------+
| telco_db  |          | hdfs://ip-172-31-82-238.ec2.internal:8020/user/hive/warehouse/telco_db.db | hadoop
| USER        |             |
+-----------+----------+----------------------------------------------------+-------------+-------------+----
---------+
1 row selected (0.058 seconds)


0: jdbc:hive2://localhost:10000/default> use telco_db;
No rows affected (0.042 seconds)
```

## Hive configuration:

```
0: jdbc:hive2://localhost:10000/default> set hive.cli.print.header=true;
No rows affected (0.011 seconds)


0: jdbc:hive2://localhost:10000/default> set hive.resultset.use.unique.column.names=false;
No rows affected (0.015 seconds)
```

```
0: jdbc:hive2://localhost:10000/default> set hive.exec.dynamic.partition=true;
No rows affected (0.011 seconds)


0: jdbc:hive2://localhost:10000/default> set hive.exec.dynamic.partition.mode=nonstrict;
No rows affected (0.006 seconds)


0: jdbc:hive2://localhost:10000/default> set hive.strict.checks.cartesian.product=false;
No rows affected (0.008 seconds)


0: jdbc:hive2://localhost:10000/default> set hive.mapred.mode=nonstrict;
No rows affected (0.011 seconds)
```

## Hive external table creation and data loading for 'train' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists train (
. . . . . . . . . . . . . . . . . . . .> device_id string,
. . . . . . . . . . . . . . . . . . . .> gender string,
. . . . . . . . . . . . . . . . . . . .> age bigint,
. . . . . . . . . . . . . . . . . . . .> group_train string
. . . . . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . . . . .> row format delimited
. . . . . . . . . . . . . . . . . . . .> fields terminated by ','
. . . . . . . . . . . . . . . . . . . .> lines terminated by '\n'
. . . . . . . . . . . . . . . . . . . .> location '/user/hadoop/telco/train/';
No rows affected (0.457 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from train;

+--------+
|  _c0   |
+--------+
| 74645  |
+--------+
1 row selected (20.985 seconds)


0: jdbc:hive2://localhost:10000/default> select * from train limit 5;

+----------------------+---------+------+--------------+
|       device_id      | gender  | age  | group_train  |
+----------------------+---------+------+--------------+
| -7548291590301750000 | M       | 33   | M32+         |
| 6943568600617760000  | M       | 37   | M32+         |
| 5441349705980020000  | M       | 40   | M32+         |
| -5393876656119450000 | M       | 33   | M32+         |
| 4543988487649880000  | M       | 53   | M32+         |
+----------------------+---------+------+--------------+
5 rows selected (2.622 seconds)
```

## Hive external table creation and data loading for 'brand_device' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists brand_device (
. . . . . . . . . . . . . . . . . . . .> device_id string,
. . . . . . . . . . . . . . . . . . . .> phone_brand string,
. . . . . . . . . . . . . . . . . . . .> device_model string
. . . . . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . . . . .> row format delimited
. . . . . . . . . . . . . . . . . . . .> fields terminated by ','
. . . . . . . . . . . . . . . . . . . .> lines terminated by '\n'
```

```
. . . . . .> location '/user/hadoop/telco/brand_device/';
No rows affected (0.091 seconds)

0: jdbc:hive2://localhost:10000/default> select count(*) from brand_device;

+---------+
|   _c0   |
+---------+
| 187245  |
+---------+
1 row selected (7.478 seconds)


0: jdbc:hive2://localhost:10000/default> select * from brand_device limit 5;

+----------------------+--------------+---------------+
|       device_id      | phone_brand  | device_model  |
+----------------------+--------------+---------------+
| 1845358998536310000  | meitu        | 2             |
| 3126957642374570000  | meitu        | 2             |
| -3051457881268070000 | meitu        | 2             |
| 4608241502940040000  | meitu        | 2             |
| 6005031767544890000  | meitu        | 2             |
+----------------------+--------------+---------------+
5 rows selected (0.269 seconds)
```

## Hive external table creation and data loading for 'events' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists events (
. . . . . .> event_id bigint,
. . . . . .> device_id string,
. . . . . .> event_time string,
. . . . . .> longitude string,
. . . . . .> latitude string
. . . . . .> )
. . . . . .> row format delimited
. . . . . .> fields terminated by ','
. . . . . .> lines terminated by '\n'
. . . . . .> location '/user/hadoop/telco/events/';
No rows affected (0.09 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from events;

+----------+
|   _c0    |
+----------+
| 3252950  |
+----------+
1 row selected (19.082 seconds)


0: jdbc:hive2://localhost:10000/default> select * from events limit 5;

+-----------+----------------------+-----------------------+------------+-----------+
| event_id  |       device_id      |       event_time      | longitude  | latitude  |
+-----------+----------------------+-----------------------+------------+-----------+
| 1         | 29182687948017100    | 2016-05-01 00:55:25.0 | 121.38     | 31.24     |
| 2         | -6401643145415150000 | 2016-05-01 00:54:12.0 | 103.65     | 30.97     |
| 3         | -4833982096941400000 | 2016-05-01 00:08:05.0 | 106.6      | 29.7      |
| 4         | -6815121365017310000 | 2016-05-01 00:06:40.0 | 104.27     | 23.28     |
| 5         | -5373797595892510000 | 2016-05-01 00:07:18.0 | 115.88     | 28.66     |
+-----------+----------------------+-----------------------+------------+-----------+
5 rows selected (0.196 seconds)
```

## Hive external table creation and data loading for 'app_events' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists app_events (
. . . . . .> event_id string,
. . . . . .> app_id string,
. . . . . .> is_installed bigint,
. . . . . .> is_active bigint
. . . . . .> )
. . . . . .> row format delimited
. . . . . .> fields terminated by ','
. . . . . .> lines terminated by '\n'
. . . . . .> location '/user/hadoop/telco/app_events/';
No rows affected (0.059 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from app_events;

+-----------+
|    _c0    |
+-----------+
| 32473067  |
+-----------+
1 row selected (13.979 seconds)


0: jdbc:hive2://localhost:10000/default> select * from app_events limit 5;

+-----------+----------------------+---------------+------------+
| event_id  |        app_id        | is_installed  | is_active  |
+-----------+----------------------+---------------+------------+
| 2         | 5927333115845830913  | 1             | 1          |
| 2         | -5720078949152207372 | 1             | 0          |
| 2         | -1633887856876571208 | 1             | 0          |
| 2         | -653184325010919369  | 1             | 1          |
| 2         | 8693964245073640147  | 1             | 1          |
+-----------+----------------------+---------------+------------+
5 rows selected (0.18 seconds)
```

## Hive external table creation and data loading for 'app_labels' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists app_labels (
. . . . . .> app_id string,
. . . . . .> label_id bigint
. . . . . .> )
. . . . . .> row format delimited
. . . . . .> fields terminated by ','
. . . . . .> lines terminated by '\n'
. . . . . .> stored as textfile
. . . . . .> location '/user/hadoop/telco/app_labels/'
. . . . . .> tblproperties ('skip.header.line.count' = '1');
No rows affected (0.065 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from app_labels;

+---------+
|   _c0   |
+---------+
| 459943  |
+---------+
1 row selected (7.131 seconds)


0: jdbc:hive2://localhost:10000/default> select * from app_labels limit 5;
```

```
+----------------------+-----------+
|        app_id        | label_id  |
+----------------------+-----------+
| 7324884708820027918  | 251       |
| -4494216993218550286 | 251       |
| 6058196446775239644  | 406       |
| 6058196446775239644  | 407       |
| 8694625920731541625  | 406       |
+----------------------+-----------+
5 rows selected (0.238 seconds)
```

## Hive external table creation and data loading for 'label_categories' table:

```
0: jdbc:hive2://localhost:10000/default> create external table if not exists label_categories (
. . . . . . . . . . . . . . . . . . . .> label_id bigint,
. . . . . . . . . . . . . . . . . . . .> category string
. . . . . . . . . . . . . . . . . . . .> )
. . . . . . . . . . . . . . . . . . . .> row format delimited
. . . . . . . . . . . . . . . . . . . .> fields terminated by ','
. . . . . . . . . . . . . . . . . . . .> lines terminated by '\n'
. . . . . . . . . . . . . . . . . . . .> stored as textfile
. . . . . . . . . . . . . . . . . . . .> location '/user/hadoop/telco/label_categories/'
. . . . . . . . . . . . . . . . . . . .> tblproperties ('skip.header.line.count' = '1');
No rows affected (0.131 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from label_categories;

+------+
| _c0  |
+------+
| 930  |
+------+
1 row selected (6.642 seconds)


0: jdbc:hive2://localhost:10000/default> select * from label_categories limit 5;

+-----------+--------------------+
| label_id  |      category      |
+-----------+--------------------+
| 1         |                    |
| 2         | game-game type     |
| 3         | game-Game themes   |
| 4         | game-Art Style     |
| 5         | game-Leisure time  |
+-----------+--------------------+
5 rows selected (0.207 seconds)
```

---

## Hive queries and analysis:

### 1. The 10 most popular brands and the percentage of the respective Male and Female owners of these brands [Handle the device id duplicates from brand_device table.]

We will first de-duplicate the brand_device table with inline OVERWRITE by grouping over the device_id column and selecting the first row in each group to eliminate the duplicates using this Hive Query –

```
0: jdbc:hive2://localhost:10000/default> insert OVERWRITE table brand_device
. . . . . . . . . . . . . . . . . . . .> select t1.device_id, t1.phone_brand, t1.device_model from (
. . . . . . . . . . . . . . . . . . . .> select t2.device_id, t2.phone_brand, t2.device_model,
```

```
. . . . . .> Row_number() OVER (partition by t2.device_id
. . . . . .> order by t2.device_id, t2.phone_brand) AS rno
. . . . . .> from brand_device t2) as t1 where t1.rno = 1;
No rows affected (12.285 seconds)


0: jdbc:hive2://localhost:10000/default> select count(*) from brand_device;

+---------+
|   _c0   |
+---------+
| 186713  |
+---------+
1 row selected (0.331 seconds)
```

We can see that the brand_device row count **reduced from 187245 to 186713** after eliminating the duplicate rows. This matches the **Total_Device_IDs_Count** and **Unique_Device_IDs_Count** numbers as seen in **SQLTasks.pdf**

We will leverage the **non_event_data** table to check the 10 most popular brands overall irrespective of gender -

```
0: jdbc:hive2://localhost:10000/default> select phone_brand, count(phone_brand) as bcount
. . . . . .> from non_event_data group by phone_brand
. . . . . .> order by bcount desc limit 10;

+--------------+---------+
| phone_brand  | bcount  |
+--------------+---------+
| Xiaomi       | 17299   |
| samsung      | 13669   |
| Huawei       | 12960   |
| OPPO         | 5783    |
| vivo         | 5637    |
| Meizu        | 4699    |
| Coolpad      | 3339    |
| lenovo       | 2691    |
| Gionee       | 1123    |
| HTC          | 1013    |
+--------------+---------+
10 rows selected (7.553 seconds)
```

Finally we will extract the 10 most popular brands from the **non_event_data** table with the respective gender split counts and percentages using above query as a Common Table Expression (CTE) -

```
0: jdbc:hive2://localhost:10000/default> with t1 as (
. . . . . .> select phone_brand, count(phone_brand) as bcount,
. . . . . .> sum(case when gender='F' then 1 else 0 end) as fcount,
. . . . . .> sum(case when gender='M' then 1 else 0 end) as mcount
. . . . . .> from non_event_data group by phone_brand
. . . . . .> order by bcount desc limit 10)
. . . . . .> select t1.phone_brand as brand, t1.bcount as brand_count,
. . . . . .> t1.fcount as female_count, t1.mcount as male_count,
. . . . . .> round(t1.fcount * 100 / (t1.fcount + t1.mcount), 2)
. . . . . .> as female_percentage,
. . . . . .> round(t1.mcount * 100 / (t1.fcount + t1.mcount), 2)
. . . . . .> as male_percentage from t1;

+----------+--------------+---------------+-------------+--------------------+------------------+
|  brand   | brand_count  | female_count  | male_count  | female_percentage  | male_percentage  |
+----------+--------------+---------------+-------------+--------------------+------------------+
| Xiaomi   | 17299        | 5918          | 11381       | 34.21              | 65.79            |
| samsung  | 13669        | 5431          | 8238        | 39.73              | 60.27            |
| Huawei   | 12960        | 4244          | 8716        | 32.75              | 67.25            |
| OPPO     | 5783         | 2571          | 3212        | 44.46              | 55.54            |
| vivo     | 5637         | 2651          | 2986        | 47.03              | 52.97            |
```

```
| Meizu    | 4699         | 1302          | 3397        | 27.71              | 72.29            |
| Coolpad  | 3339         | 1079          | 2260        | 32.32              | 67.68            |
| lenovo   | 2691         | 893           | 1798        | 33.18              | 66.82            |
| Gionee   | 1123         | 402           | 721         | 35.8               | 64.2             |
| HTC      | 1013         | 320           | 693         | 31.59              | 68.41            |
+----------+--------------+---------------+-------------+--------------------+------------------+
10 rows selected (17.21 seconds)
```

---

### 2. The 10 most popular brands for Male and Female?

We will leverage the **non_event_data** table to extract and concatenate (UNION operation) the 10 most popular brands for Female and Male genders -

```
0: jdbc:hive2://localhost:10000/default> (select 'F' as gender, phone_brand as brand,
. . . . . .> count(phone_brand) as brand_count from
. . . . . .> non_event_data where gender='F' group by phone_brand
. . . . . .> order by brand_count desc limit 10)
. . . . . .> UNION
. . . . . .> (select 'M' as gender, phone_brand as brand,
. . . . . .> count(phone_brand) as brand_count from
. . . . . .> non_event_data where gender='M' group by phone_brand
. . . . . .> order by brand_count desc limit 10)
. . . . . .> order by 1,3 desc;

+---------+----------+--------------+
| gender  |  brand   | brand_count  |
+---------+----------+--------------+
| F       | Xiaomi   | 5918         |
| F       | samsung  | 5431         |
| F       | Huawei   | 4244         |
| F       | vivo     | 2651         |
| F       | OPPO     | 2571         |
| F       | Meizu    | 1302         |
| F       | Coolpad  | 1079         |
| F       | lenovo   | 893          |
| F       | Gionee   | 402          |
| F       | HTC      | 320          |
    ------------------------
| M       | Xiaomi   | 11381        |
| M       | Huawei   | 8716         |
| M       | samsung  | 8238         |
| M       | Meizu    | 3397         |
| M       | OPPO     | 3212         |
| M       | vivo     | 2986         |
| M       | Coolpad  | 2260         |
| M       | lenovo   | 1798         |
| M       | Gionee   | 721          |
| M       | HTC      | 693          |
+---------+----------+--------------+
20 rows selected (13.784 seconds)
```

We can see that the 10 most popular brands are overall the same for Female and Male genders but with some interim ranking changes. For example, Samsung and Huawei are seen to swap ranks 2 and 3 across the two genders.

---

### 3. The count and percentage analysis of the Gender in the train data set

We will extract the respective gender counts and percentages using group by gender on the 'train' table along with a CTE to self-join with the 'train' table -

```
0: jdbc:hive2://localhost:10000/default> with t1 as (select count(*) as total_count from train)
. . . . . .> select t2.gender, count(t2.gender) as gender_count,
```

```
. . . . . .> round(count(t2.gender) * 100 / t1.total_count, 2)
. . . . . .> as gender_percentage from train t2
. . . . . .> join t1 group by t1.total_count, t2.gender;

+---------+---------------+--------------------+
| gender  | gender_count  | gender_percentage  |
+---------+---------------+--------------------+
| F       | 26741         | 35.82              |
| M       | 47904         | 64.18              |
+---------+---------------+--------------------+
2 rows selected (23.08 seconds)
```

---

### 4. The top mobile phone brands offering the highest number of models [Provide details about the top three brands.]

We will extract the top phone brands using group by phone_brand on the 'brand_device' table and then count the distinct device_model's within each brand in descending order to yield the top 3 brands -

```
0: jdbc:hive2://localhost:10000/default> select phone_brand as brand,
. . . . . .> count(distinct device_model) as models_count
. . . . . .> from brand_device group by phone_brand
. . . . . .> order by models_count desc limit 3;

+----------+---------------+
|  brand   | models_count  |
+----------+---------------+
| lenovo   | 194           |
| samsung  | 163           |
| Huawei   | 145           |
+----------+---------------+
3 rows selected (16.679 seconds)
```

---

### 5. The average number of events per device id [Applicable to the device_id column from the train table, which has at least one associated event in the event table]

We will leverage the **event_data** table to check the device_events_count for a few device_id rows that have one or more associated events -

```
0: jdbc:hive2://localhost:10000/default> select device_id, count(event_id) as device_events_count
. . . . . .> from event_data where event_id IS NOT NULL
. . . . . .> group by device_id limit 5;

+----------------------+----------------------+
|       device_id      | device_events_count  |
+----------------------+----------------------+
| -1001337759327040000  | 109                  |
| -1002595372059170000  | 7                    |
| -1002733576670970000  | 55                   |
| -1002969456091700000  | 7                    |
| -1005411102947240000  | 44                   |
+----------------------+----------------------+
5 rows selected (14.474 seconds)
```

Finally we will use above query as a CTE to aggregate and compute the overall avg_events_per_device number -

```
0: jdbc:hive2://localhost:10000/default> with t1 as (
. . . . . .> select device_id, count(event_id) as device_events_count
. . . . . .> from event_data where event_id IS NOT NULL group by device_id)
. . . . . .> select round(avg(t1.device_events_count),2)
. . . . . .> as avg_events_per_device from t1;
```

```
+------------------------+
| avg_events_per_device  |
+------------------------+
| 52.15                  |
+------------------------+
1 row selected (14.767 seconds)
```

---

### 6. Whether the count and percentage of the device_id column in the train table have corresponding events data available

We will first check the count of device_id's from the 'train' table that have corresponding associated events from the 'events' table -

```
0: jdbc:hive2://localhost:10000/default> select count(device_id) from train where device_id in
. . . . . . . . . . . . . . . . . . . .> (select device_id from events);

+--------+
|  _c0   |
+--------+
| 23310  |
+--------+
1 row selected (25.037 seconds)
```

Finally we will extract the percentage value using a CTE to self-join with the 'train' table -

```
0: jdbc:hive2://localhost:10000/default> with t1 as (select count(*) as total_count from train)
. . . . . . . . . . . . . . . . . . . .> select t1.total_count as total_devices_count,
. . . . . . . . . . . . . . . . . . . .> count(t2.device_id) as have_events_count,
. . . . . . . . . . . . . . . . . . . .> round(count(t2.device_id) * 100 / t1.total_count, 2)
. . . . . . . . . . . . . . . . . . . .> as have_events_percentage
. . . . . . . . . . . . . . . . . . . .> from train t2 join t1 where t2.device_id in
. . . . . . . . . . . . . . . . . . . .> (select device_id from events)
. . . . . . . . . . . . . . . . . . . .> group by t1.total_count;

+----------------------+--------------------+-------------------------+
| total_devices_count  | have_events_count  | have_events_percentage  |
+----------------------+--------------------+-------------------------+
| 74645                | 23310              | 31.23                   |
+----------------------+--------------------+-------------------------+
1 row selected (27.561 seconds)
```