MIE 1628 Assignment 1: HDFS, YARN and HIVE SQL

Q1.

1) What are two main functions and the components of HDFS?

The two main functions of Hadoop Distributed File System (HDFS) are data storage and management. HDFS is Hadoop's primary storage system that provides redundant storage for massive amount of data in order to rescue the system from possible data loss in case of failure. It also makes applications available to parallel processing. HDFS contains two components, NameNode and DataNodes. NameNode is the master node, it manages the file system namespace; controls the access of clients to files; and executes file system namespace operations like opening, closing and renaming files and directories. DataNodes are slave nodes that are responsible for the effective storage of data in HDFS. They perform read and write operations from client request. They also perform block creation, deletion and replication upon instruction from the NameNode.

2) What are main functions and the components of YARN?

The main functions of YARN are resource management and job scheduling. The components of YARN include 1). Resource Manager: Runs on a master daemon and manages the resource allocation in the cluster; 2). Node Manager: Run on the slave daemons and are responsible for the execution of a task on every single DataNode; 3). Application Master: Manages the user job lifecycle and resource need of individual applications; 4). Containers: Set of resources including RAM, CPU, Network, HDD etc on a single node.

Q2. Data is replicated at least thrice on HDFS. Does it imply that any alterations or calculations done on one copy of the data will be reflected in the other two copies also?

The alterations or calculations would be done on the original data and do not reflect on the copies of data. Master Node would identify the location of the original data and performs the calculations. Only if the slave node is not responding or data is corrupted, Master Node would move to the second replica.

Q3. Write the commands for the following HDFS tasks:

1) Copy a file from HDFS to local file system:

hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

2) Move files within HDFS:

hdfs dfs -mv URI [URI…] <dest>

Example: hdfs dfs -mv/user/hadoop/file1/user/hadoop/file2

3). Print contents:

hdfs dfs -cat URI [URI…]

Example: hdfs dfs -cat file:///file3 /user/hadoop/file4

4). Move file from local to HDFS:

hdfs dfs -moveFromLocal <localsrc> <dst>

5). Create new directory in personal directory and title it Lab1_2020_Fall:

Create directory: hdfs dfs -mkdir /user/mie_yhuang/Lab1_2020_Fall

Check if created: hdfs dfs -ls /user/mie_yhuang

```
[mie_yhuang@hdp006 ~]$ hdfs dfs -mkdir /user/mie_yhuang/Lab1_2020_Fall
[mie_yhuang@hdp006 ~]$ hdfs dfs -ls /user/mie_yhuang
Found 4 items
drwx------   - mie_yhuang uoft_mie          0 2020-10-23 01:04 /user/mie_yhuang/.Trash
drwxr-xr-x   - mie_yhuang uoft_mie          0 2020-10-18 15:51 /user/mie_yhuang/.hiveJars
drwxr-xr-x   - mie_yhuang uoft_mie          0 2020-10-23 01:09 /user/mie_yhuang/Lab1_2020_Fall
drwx------   - mie_yhuang uoft_mie          0 2020-10-18 17:43 /user/mie_yhuang/hive
```

6). Move file /tmp/flights.csv to folder Lab1_2020_Fall:

hdfs dfs -mv /tmp/flights.csv /user/mie_yhuang/Lab1_2020_Fall

hdfs dfs -ls /user/mie_yhuang/Lab1_2020_Fall

```
[mie_yhuang@hdp006 ~]$ hdfs dfs -mv /tmp/flights.csv /user/mie_yhuang/Lab1_2020_Fall
[mie_yhuang@hdp006 ~]$ hdfs dfs -ls /user/mie_yhuang/Lab1_2020_Fall
Found 1 items
-rw-r--r--   2 mie_nliu hdfs          0 2020-10-22 19:36 /user/mie_yhuang/Lab1_2020_Fall/flights.csv
```

For the following SQL questions, the csv files have been uploaded to database mie_yhuang_a1.

Q4. Practice CASE Statement:

The HQL code:

```
-- select required columns and write case statement from game
-- then join team_info to find home team name and away team name respectively
select a.season as season,
    b.teamname as home_team_name,
    c.teamname as away_team_name,
    (a.home_goals + a.away_goals) as total_goals,
    case
    when (a.home_goals + a.away_goals) >= 7 then 'High scoring game'
    when (a.home_goals + a.away_goals) < 7 and (a.home_goals + a.away_goals) > 4 then 'Mid scoring game'
    else 'Low scoring game'
    end as category
from game as a
join team_info as b
on (a.home_team_id = b.team_id)
join team_info as c
on (a.away_team_id = c.team_id)
where a.outcome = 'home win OT'
```

The screenshot of the resulting table:

season	home_team_name	away_team_name	total_goals	category
20112012	Flyers	Devils	7	High scoring game
20112012	Devils	Flyers	7	High scoring game
20122013	Bruins	Rangers	5	Mid scoring game
20122013	Rangers	Bruins	7	High scoring game
20122013	Bruins	Penguins	3	Low scoring game
20102011	Sharks	Red Wings	3	Low scoring game

Q5. Practice OVER and GROUPBY Function:

HQL code using GROUPBY:

```
select player_id,
    sum(goals) as sum_goals,
    max(goals) as max_goals,
    avg(goals) as avg_goals,
    sum(assists) as sum_assists,
    max(assists) as max_assists,
    avg(assists) as avg_assist
from game_skater_stats
group by player_id
order by (sum_goals+sum_assists) desc
```

The resulting table using GROUPBY Function:

player_id	sum_goals	max_goals	avg_goals	sum_assists	max_assists	avg_assist
8471675	299	3	0.443620178041543	515	5	0.7640949554896143
8474141	311	3	0.4157754010695187	478	4	0.6390374331550802
8471214	434	4	0.550761421319797	334	3	0.42385786802030456
8473512	223	3	0.30381471389645776	504	5	0.6866485013623979
8471215	282	3	0.4385692068429238	434	4	0.6749611197511665
8473548	285	3	0.36726804123711343	419	4	0.5399484536082474

HQL code using OVER Function:

```
-- require distinct to remove duplicates
select distinct(player_id),
    sum(goals) over (partition by player_id) as sum_goals,
    max(goals) over (partition by player_id) as max_goals,
    avg(goals) over (partition by player_id) as avg_goals,
    sum(assists) over (partition by player_id) as sum_assists,
    max(assists) over (partition by player_id) as max_assists,
    avg(assists) over (partition by player_id) as avg_assist
from game_skater_stats
order by (sum_goals+sum_assists) desc
```

The resulting table using OVER Function:

player_id	sum_goals	max_goals	avg_goals	sum_assists	max_assists	avg_assist
8471675	299	3	0.443620178041543	515	5	0.7640949554896143
8474141	311	3	0.4157754010695187	478	4	0.6390374331550802
8471214	434	4	0.550761421319797	334	3	0.42385786802030456
8473512	223	3	0.30381471389645776	504	5	0.6866485013623979
8471215	282	3	0.4385692068429238	434	4	0.6749611197511665
8473548	285	3	0.36726804123711343	419	4	0.5399484536082474

Discussion: Using OVER function could achieve the same results as using GROUPBY function but there are many repetitive records. The duplicates could be removed through select distinct group id.

Q6. Practice JOINS:

The HQL code: Table Q6 created in database mie_yhuang_a1

```
create table Q6 as
select distinct p.player_id,
    p.lastname as lastname,
    p.firstname as firstname,
    p.nationality as nationality
from player_info as p
join game_skater_stats as g
on p.player_id = g.player_id
where p.primaryposition = 'D' and g.team_id = 2
```

The resulting table:

p.player_id	lastname	firstname	nationality
8458541	Staios	Steve	CAN
8466309	Mottau	Mike	USA
8467580	Eaton	Mark	USA
8467931	Carkner	Matt	CAN
8468101	Martinek	Radek	CZE

Q7. Practice Subquery:

HQL code:

```
select a.game_id as game_id,
    sum(a.shots) as sum_shots,
    max(a.shots) as max_shots,
    avg(a.shots) as avg_shots
from game_skater_stats a
where a.game_id in
    (select b.game_id
     from game b
     where b.home_goals > 3)
group by a.game_id
```

The resulting table:

game_id	sum_shots	max_shots	avg_shots
2010020004	79	7	2.1944444444444446
2010020005	70	5	1.9444444444444444
2010020009	64	6	1.7777777777777777
2010020011	62	8	1.7222222222222223
2010020014	56	5	1.5555555555555556
2010020015	69	6	1.9166666666666667

Q8. Practice Subquery:

HQL code:

```
select a.player_id as player_id,
    avg(a.hits) as avg_hits
from game_skater_stats a
where a.team_id in (
  select b.team_id
  from game_teams_stats b
  where b.team_id %2 != 0 and b.hoa = 'home')
group by player_id
```

The resulting table:

player_id	avg_hits
8446485	1.0727272727272728
8448208	0.2638888888888889
8449645	0.23809523809523808
8456283	1.1702127659574468
8457063	0.5952380952380952
8458520	1.5