**Playstation Home Assignment**

**Question 1**

My understanding is you want to obtain number of players belongs to different rank or level.

The mistakes I see in the query are:

1. The Rank is in selected columns but not in the group by statement which will give an error. Here it’s only grouped by Level\_Name which will give number of players in each level.
2. We need to group it by both Rank and Level\_ID (or Level\_Name) to see the number of players of each rank playing in each level. I would prefer Level\_ID over Level\_Name as the name can be duplicate or can have inconsistent case.
3. The LevelName is missing ‘\_’ . It should be Level\_Name (This can be a typo but thought worth mentioning)

The query I would write to get the expected result:

SELECT

Pl.Rank,

Pl.Level\_ID,

Le.Level\_Name,

Count(Pl.Player\_ID) as num\_players

FROM Players Pl

INNER JOIN Levels Le ON PL.Level\_ID = Le.Level\_ID --this is optional, If you want to see the Level\_Name

Order by Pl.Rank, Pl.Level\_ID

GROUP BY Pl.Rank,

Pl.Level\_ID,

Le.Level\_Name;

**Question 2**

In a nutshell the bash script is printing 2nd column of first 50 even lines of a file named inputfile. The 2nd column of the file is consisting of urls of files that needs to be downloaded. The last command is using these urls to download files.

**`yes | nl | head -50 | cut -f 1`** - This line of code here is complex statement to generate 50 numbers (from 1 to 50) so that the for loop iterates 50 times. This can be simplified by using **seq** command as below:

For a in $(seq 50); do <something>; done

**head -$(($a\*2)) inputfile | tail -1 | \** - This code here is taking first a\*2 number of lines ( where a is current iteration) and then it takes only the last line from those lines. For example, if the current iteration value is a=5 then it will take first 10 lines (5\*2=10) and then the **tail -1** command will take only the last line which is line 10th. Basically its taking only the even number of lines (2,4,6,8…100) from file ‘inputfile’.

Finally the last line will print the 2nd column value using **awk** command. Here it’s assuming the file is tab separated.

And the 2nd column has the urls of files that needs to be downloaded. These urls are then passed to the **wget** command which will download the files. Further, it’s discarding any error message generated.

Below is improvised bash script:

number=50

for seq\_num in $(seq “$number”); do

data=$(sed –n “$((seq\_num\*2))p” “inputfile” | awk -F’\t’ ‘{print $2}’)

If [-n “$data”]; then

wget –c “$data” 2> /dev/null

fi

done

**Question 3**

Last.fm repo has 2 data files. One containing details of users and the other file has details of songs that each user played. My approach is to first load the file and analyse the data. The below analysis was done:

1) Check the count of rows and columns to get insights in the size of the data.

2) Check if there is some data missing. For examples Track\_ID field has lots of nulls

3) Type cast any field if required. Here the timestamp field was type casted to correct datetime format.

After the analysis, we need to create spark session object so that we can perform spark related operation. We then load the music file using .read function into spark dataframe. Below steps were taken to get possible solution for Part A, B and C task.

**Part A:** This can be achieved by grouping the data by userid using .groupBy function and counting distinct tracks played within groups. The field musicbrainz-track-id has lot of nulls so we will also use track-name and artist-id as well to get the distinct track.

Here we got total of 992 userids with number of dictinct songs they played. It appears there are 1000 users and we are missing 8 users in the result. This could be because these 8 users haven’t played any song. To get these user in our output we will need to join the music data with the users data so that we get list of all users.

**Part B:** To get this we need to group the data by track-id or track-name and count the instances in each group. We can then order the groups by the number of counts in descending order and limit the data to only 100 records.

Part C: This can be achieved by grouping the data by userid and sorting on timestamp. We need to get the time difference between each song to see if both the songs belong to same session or a new one. We can do this by creating a new field which will have the previous song timestamp. We then subtract the current timestamp with the previous song timestamp to get the difference in time. If the difference in time is more than 20 minutes then we will mark that as a start of new session.

We then assign session id to each session. This can be achieved using sum function in spark. It will keep on adding the ones when it encounters new session. If the session is not new then it will add 0 so that it can assign same number to all the songs within that session.

Once we have session id assigned our task becomes easy and we can just group the data by userid and session id and get the desired output. We can get last timestamp using max function and first timestamp using min functions. We can get the list of all the songs played in the session by using collect\_list function. We can also get the duration of the session by subtraction last timestamp with the first timestamp of the songs.

We then order the data by session duration and limit it to 100 records to get top 100 sessions.

**Question 4**

The test was very interesting and clever way to analyse candidates approach to a problem. This will also check the candidates ability to paying attention to details. The first question was relatively easy compared to the others. The data and question was very clear which helped me to answer it quickly.

The second question was bit complex as there are lot of operation going on in those 3 lines of code. I spent more time to understand the code and once I understood I was able to quickly modify the script. This code was the example of overthinking I must say ☺

The last question was very interesting. I have done lot of similar operations but using mostly pandas. It took lot of time to set up pyspark on my machine. I was having lot of difficulties with the Hadoop and java compatibility with pyspark. Finally I was able to set pyspark and begin with the solution. The first 2 parts were quite easy and would need me to spend lot of time. The 3rd part was bit challenging as I needed to build sessions. I found out a way to do that using sum function which I think was very challenging. I faced difficulties in writing the output results to the files because of some java errors. Hence I manually saved the data on files for your reference.

Overall I enjoyed the challenge and increased my knowledge from it. It took me couples of hours for 2 to 3 days overall to solve all the problems. I hope you like my approach and I look forward to showcase my skills in more details.