# CSE 4308

## LAB 4 ASSIGNMENT

Task statement:

Our main goal for this task was to showcase various SQL commands by writing complex queries to show different information from given information.

Solution approach:

For the solution, we had to use aggregate functions such as avg,min,max and also use some commands such as GROUP BY and HAVING to a set of tuples. We had to use nested queries in order to match primary keys between separate tables and maintain data integrity. IN and LIKE operators were also used but the biggest part of the task was using nested sub queries.

A snippet of the solution is provided below:

*1*

*SELECT MOV\_YEAR FROM MOVIE WHERE MOV\_YEAR>=1990 AND MOV\_YEAR<=2000;*

*2*

*SELECT ACT\_FIRSTNAME FROM ACTOR WHERE ACT\_FIRSTNAME LIKE '%k' OR ACT\_FIRSTNAME LIKE 'a%' OR ACT\_FIRSTNAME LIKE '%a%' OR ACT\_FIRSTNAME LIKE '%a';*

*3*

*SELECT ACT\_FIRSTNAME FROM ACTOR NATURAL JOIN DIRECTOR WHERE ACTOR.ACT\_FIRSTNAME=DIRECTOR.DIR\_FIRSTNAME;*

*4*

*SELECT A.ACT\_FIRSTNAME FROM ACTOR A, ACTOR B WHERE A.ACT\_FIRSTNAME=B.ACT\_FIRSTNAME AND A.ACT\_ID!=B.ACT\_ID;*

*5*

*SELECT MOV\_TITLE FROM MOVIE LEFT JOIN RATING WHERE RATING.REV\_STARS=NULL;*

*6*

*SELECT avg(REV\_STARS) FROM RATING;*

*7*

*SELECT count(MOV\_COUNTRY), MOV\_COUNTRY FROM MOVIE GROUP BY MOV\_COUNTRY;*

*8*

*SELECT min(REV\_STARS),MOV\_ID FROM RATING GROUP BY MOV\_ID ORDER BY min(REV\_STARS) DESC;*

*9*

*SELECT M.MOV\_TITLE FROM MOVIE M WHERE*

*( SELECT avg(REV\_STARS) FROM RATING WHERE REV\_STARS IS NOT NULL) <*

*( SELECT avg(REV\_STARS) FROM RATING R WHERE R.MOV\_ID=M.MOV\_ID AND R.REV\_STARS IS NOT NULL)*

*AND M.MOV\_ID IN (SELECT MOV\_ID FROM RATING);*

*//SELECT max(A\_REV) FROM (SELECT avg(REV\_STARS) AS A\_REV FROM RATING GROUP BY MOV\_ID) A;*

*10*

*SELECT A.ACT\_FIRSTNAME AS ACT\_NAME,*

*(SELECT sum(R.REV\_STARS) FROM RATING R WHERE R.MOV\_ID IN*

*(SELECT C.MOV\_ID FROM CASTS C WHERE C.ACT\_ID=A.ACT\_ID))*

*AS SUM\_RATING FROM ACTOR A WHERE A.ACT\_ID IN*

*(SELECT C2.ACT\_ID FROM CASTS C2 WHERE C2.MOV\_ID IN*

*(SELECT MOV\_ID FROM RATING))*

*ORDER BY SUM\_RATING DESC;*

*11*

*SELECT DIR\_NAME, AVG\_RATING FROM*

*( SELECT D.DIR\_FIRSTNAME AS DIR\_NAME,*

*(SELECT avg(R.REV\_STARS) FROM RATING R WHERE R.MOV\_ID IN*

*( SELECT DN.MOV\_ID FROM DIRECTION DN WHERE DN.DIR\_ID=D.DIR\_ID)*

*AND R.REV\_STARS IS NOT NULL)*

*AS AVG\_RATING*

*FROM DIRECTOR D WHERE DIR\_ID IN*

*( SELECT DIR\_ID FROM DIRECTION DN2 WHERE DN2.MOV\_ID IN*

*( SELECT MOV\_ID FROM RATING))*

*ORDER BY AVG\_RATING DESC*

*)*

*WHERE ROWNUM = 1;*

*12*

*SELECT \* FROM MOVIE WHERE MOV\_ID IN*

*(SELECT DN.MOV\_ID FROM DIRECTION DN WHERE DN.DIR\_ID IN*

*(SELECT D.DIR\_ID FROM DIRECTOR D WHERE D.DIR\_FIRSTNAME IN*

*(SELECT D1.DIR\_FIRSTNAME FROM DIRECTOR D1*

*INTERSECT*

*SELECT A.ACT\_FIRSTNAME FROM ACTOR A)));*

*13*

*SELECT M.MOV\_TITLE, ( SELECT avg(REV\_STARS) FROM RATING R WHERE R.MOV\_ID=M.MOV\_ID AND REV\_STARS IS NOT NULL) AS AVG\_RATE*

*FROM MOVIE M WHERE M.MOV\_ID IN*

*(SELECT RR.MOV\_ID FROM RATING RR WHERE RR.REV\_STARS IS NOT NULL*

*GROUP BY RR.MOV\_ID HAVING avg(RR.REV\_STARS) > 7 )*

*ORDER BY AVG\_RATE DESC;*

*14*

*SELECT R.REV\_NAME FROM REVIEWER R WHERE R.REV\_ID IN*

*(SELECT RT.REV\_ID FROM RATING RT WHERE RT.REV\_STARS =( SELECT min (REV\_STARS) FROM RATING));*

The commands that were used are:

AVG (Average):

The AVG function is used to calculate the average value of a numeric column in a SQL query. It takes the values of the specified column, sums them up, and divides by the number of rows to compute the average.

MIN (Minimum):

The MIN function is used to find the minimum (smallest) value in a numeric or text column in a SQL query.

MAX (Maximum):

The MAX function is used to find the maximum (largest) value in a numeric or text column in a SQL query.

GROUP BY:

The GROUP BY clause is used to group rows in a result set based on the values in one or more columns. It's often used in combination with aggregate functions like SUM, AVG, MIN, and MAX to perform calculations on groups of rows.

HAVING:

The HAVING clause is used in conjunction with the GROUP BY clause to filter the result set based on aggregate function results. It allows us to specify conditions for the groups created by GROUP BY.

IN:

The IN operator is used to filter rows based on a list of specific values. It allows us to specify a set of values that a column's value should match in order for the row to be included in the result set.

Problems I faced:

This weeks task was particularly hard for me and I couldn’t finish the 15th task, as I had a lot of trouble in joining the tables in a meaningful way and finding all the useful data. I faced a lot of hurdles while writing the nested queries and kept running into errors. Aside from that I often accidentally misused the aggregate functions by trying to nest them as another group and calling another column alongside them that wasn’t using the aggregate function.

SUBMITTED BY

MEHREEN HOSSAIN CHOWDHURY

210041219