Islamic University of Technology

Lab 04

CSE 4308 - DBMS Lab

Submitted To :

Zannatun Naim Sristy
Lecturer, CSE Department
Islamic University of Technology

Submitted By :

Rhidwan Rashid
ID : 200042149
Prog. : SWE
Dept. : CSE

Queries:

(1) Analysis: The problem requires finding the name of the actor/actress who are also directors with and without "Intersect" clause.

Working code without Intersect clause:

```
SELECT ACT_FIRSTNAME, ACT_LASTNAME
FROM ACTOR, DIRECTOR
WHERE ACT_FIRSTNAME = DIR_FIRSTNAME AND ACT_LASTNAME =
DIR_LASTNAME;
```

Working code with Intersect clause:

```
SELECT ACT_FIRSTNAME, ACT_LASTNAME
FROM ACTOR
INTERSECT
SELECT DIR_FIRSTNAME, DIR_LASTNAME
FROM DIRECTOR;
```

Explanation: Without intersect clause we have to select first name and last name of actor/actress from actors and director table and check whether director names and actors/actresses names match or not.
With intersect clause we have to select first name and last name from actor table and using intersect select first name and last name from director table. Intersect clause filters out common entries in both table.

(2) Analysis: This problem requires finding the list of all first names stored in the database.

Working code:

```
SELECT ACT_FIRSTNAME
FROM ACTOR
UNION
SELECT DIR_FIRSTNAME
FROM DIRECTOR;
```

Explanation: Only actor and director table contains first name. We have to select first name from actor table and then with the help of union keyword select first name from director table also. Union combines entries of two tables.

(3) Analysis: This problem requires finding the movie titles that didn't receive any ratings with and without minus clause.

Working code without minus clause:

```
SELECT DISTINCT MOV_TITLE
FROM MOVIE
WHERE MOVIE.MOV_ID NOT IN(SELECT MOV_ID FROM RATING);
```

Working code with minus clause:

```
SELECT MOV_TITLE
FROM MOVIE
MINUS
SELECT MOV_TITLE
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID;
```

Explanation: Without minus operation we have to select movie title from movie table and in where clause we put a nested query which selects all movie id from rating table, then with not in clause we compare all movie id from movie table with movie id from rating table and selects that are not in them. Distinct should be used to avoid duplication.

With help of minus clause we select all movie titles from movie table and also those titles which got ratings, then minus the rated titles from all titles.

(4) Analysis: This problem requires finding the average rating of all movies.

Working code:

```
SELECT MOV_TITLE, AVG(NVL(REV_STARS,0))
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID
GROUP BY MOVIE.MOV_TITLE;
```

Explanation: We have to select movie title and average rating using avg function and then group them by movie title.

(5) Analysis: This problem requires finding minimum rating for each movie and display them in descending order.

Working code:

```
SELECT MOV_TITLE, MIN(NVL(REV_STARS,0)) AS MIN_RATING
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID
GROUP BY MOVIE.MOV_TITLE
ORDER BY MIN_RATING DESC;
```

Explanation: We have to select movie titles from movie and rating table and use min function to calculate minimum, then group them using titles and order them using order by clause.

(6) Analysis: This problem requires finding the last name of actors/actresses and number of rating received by movies they played role.

Working code:

```
SELECT ACT_LASTNAME, COUNT(RATING.MOV_ID)
FROM ACTOR, CASTS, RATING
WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID =
RATING.MOV_ID
GROUP BY ACTOR.ACT_LASTNAME;
```

(7) Analysis: This problem requires finding the last name and average runtime of movies acted by different actors/actresses and excluding actors/actresses who worked with James Cameron with and without having clause.

Working code without having clause:

```
SELECT ACT_LASTNAME, AVG(MOVIE.MOV_TIME)
FROM ACTOR, CASTS, MOVIE
WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID =
MOVIE.MOV_ID
GROUP BY ACTOR.ACT_LASTNAME
MINUS
SELECT ACT_LASTNAME, AVG(MOVIE.MOV_TIME)
FROM ACTOR, CASTS, MOVIE, DIRECTION, DIRECTOR
WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID =
MOVIE.MOV_ID AND MOVIE.MOV_ID = DIRECTION.MOV_ID AND
DIRECTION.DIR_ID = DIRECTOR.DIR_ID AND
DIRECTOR.DIR_FIRSTNAME = 'James' AND DIRECTOR.DIR_LASTNAME =
'Cameron'
GROUP BY ACTOR.ACT_LASTNAME;
```

Working code with having clause:

```
SELECT ACT_LASTNAME, AVG(MOVIE.MOV_TIME)
FROM ACTOR, CASTS, MOVIE, DIRECTION, DIRECTOR
HAVING ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID =
MOVIE.MOV_ID AND MOVIE.MOV_ID = DIRECTION.MOV_ID AND
DIRECTION.DIR_ID = DIRECTOR.DIR_ID AND
DIRECTOR.DIR_FIRSTNAME = 'James' AND DIRECTOR.DIR_LASTNAME =
'Cameron';
```

(8) Analysis: This problem requires finding the first name and last name of the director of the movie having the highest average rating with and without all clause.

Working code without all clause:

```
SELECT MAX(DIR_FIRSTNAME) FIRSTNAME, MAX(DIR_LASTNAME)
LASTNAME, MAX(A.REVIEW_AVERAGE)
FROM DIRECTOR D1, DIRECTION D2, RATING R, (SELECT M.MOV_ID,
avg(REV_STARS) REVIEW_AVERAGE
FROM RATING R, MOVIE M
WHERE M.MOV_ID = R.MOV_ID
GROUP BY M.MOV_ID) A
WHERE (D1.DIR_ID = D2.DIR_ID) AND (D2.MOV_ID = R.MOV_ID);
```

Working code with all clause:

```
SELECT MAX(ALL DIR_FIRSTNAME) FIRSTNAME, MAX(ALL
DIR_LASTNAME) LASTNAME, MAX(A.REVIEW_AVERAGE)
FROM DIRECTOR D1, DIRECTION D2, RATING R, (SELECT ALL
M.MOV_ID, avg(REV_STARS) REVIEW_AVERAGE
FROM RATING R, MOVIE M
WHERE M.MOV_ID = R.MOV_ID
GROUP BY M.MOV_ID) A
WHERE (D1.DIR_ID = D2.DIR_ID) AND (D2.MOV_ID = R.MOV_ID);
```

(9) Analysis: This problem requires finding all movie related information of movies directed and acted by same person.

Working code:

```
SELECT MOVIE.MOV_ID, MOVIE.MOV_TITLE, MOVIE.MOV_YEAR,
MOVIE.MOV_TIME, MOVIE.MOV_LANGUAGE, MOVIE.MOV_RELEASEDATE,
MOVIE.MOV_COUNTRY
FROM MOVIE, ACTOR, CASTS, DIRECTION, DIRECTOR
WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID =
MOVIE.MOV_ID AND MOVIE.MOV_ID = DIRECTION.MOV_ID AND
DIRECTION.DIR_ID = DIRECTOR.DIR_ID AND
DIRECTOR.DIR_FIRSTNAME = ACTOR.ACT_FIRSTNAME AND
DIRECTOR.DIR_LASTNAME = ACTOR.ACT_LASTNAME;
```

(10) Analysis: This problem requires finding the title and average rating of the movies that have average rating more than 7 with and without using having clause.

Working code without having clause:

```
SELECT MOV_TITLE, AVG_RATING
FROM(SELECT MOV_TITLE, AVG(NVL(REV_STARS,0)) AS AVG_RATING
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID
GROUP BY MOVIE.MOV_TITLE)
WHERE AVG_RATING > 7;
```

Working code with having clause:

```
SELECT MOV_TITLE, AVG(NVL(REV_STARS,0))
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID
GROUP BY MOVIE.MOV_TITLE
HAVING AVG(NVL(REV_STARS,0)) > 7;
```

(11) Analysis: This problem requires finding the title of movies having average rating higher than averaging rating of all movies.

Working code:

```
SELECT MOV_TITLE, AVG(NVL(REV_STARS,0))
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID
GROUP BY MOVIE.MOV_TITLE
HAVING AVG(NVL(REV_STARS,0)) > (SELECT AVG(NVL(REV_STARS,0))
FROM RATING);
```

(12) Analysis: This problem requires finding the title and average rating of the movies without using group by.

Working code:

```
SELECT DISTINCT MOV_TITLE,(SELECT AVG(NVL(REV_STARS,0))
FROM RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID)
FROM MOVIE, RATING
WHERE MOVIE.MOV_ID = RATING.MOV_ID;
```

(13) Analysis: This problem requires finding the actresses with same first name.

Working code:

```
SELECT AC1.ACT_FIRSTNAME
FROM ACTOR AC1
INNER JOIN ACTOR AC2 ON AC1.ACT_FIRSTNAME =
AC2.ACT_FIRSTNAME
WHERE AC1.ACT_GENDER = 'F';
```