**Assignment 2**

By Harsh Sharma

Part 2

**Part II. SQL Query**

Question 1. **1.** Retrieve the movie number, title, cost, and genre of movies that are priced under 50 and belong to either the “ACTION” or “COMEDY” genres. Order the results in ascending order based on the genre. (**NOTE**: Use the IN operator, even though it can be achieved using the OR conjunction.)

Solution:

select movie\_num, movie\_title, movie\_cost, movie\_genre

From movie

Where movie\_cost < 50 and movie\_genre in ('ACTION', 'COMEDY')

Order by movie\_genre;

Result:

A screenshot of a computer

Description automatically generated

Question 2. Retrieve the movie title, year, and cost of all movies with a title that includes the word “hope” in any case variation, such as “Hope”, “HOPE”, “hOpe” . Ensure your query is case-insensitive to capture all variations.

(**NOTE**: For this query, utilize a single-block query structure.)

Solution:

Select movie\_title, movie\_year, movie\_cost

from movie

Where lower(movie\_title) like '%hope%';

A screenshot of a computer

Description automatically generated

**3.** Provide the last name, house number, street name and city of all members located in Tennessee (TN). The results should display the house number and street name as separate columns, as shown below.

Solution:

Query:

SELECT mem\_lname AS "LAST NAME",

SUBSTR(mem\_street, 1, INSTR(mem\_street, ' ') - 1) AS "HOUSE NO",

SUBSTR(mem\_street, INSTR(mem\_street, ' ') + 1) AS "STREET NAME",

mem\_city AS "CITY"

FROM membership

WHERE mem\_state = 'TN';

Result:

A screenshot of a black screen

Description automatically generated

**4**. Display the genre name along with the total count of movies for each genre. Sort the results with the genre having the highest movie count at the top and the one with the least count at the bottom.

Solution:

Select movie\_genre as "Movie Genre", count(\*) as "Number of Movies"

From movie

group by movie\_genre

order by count(\*) desc;

Results:

A screenshot of a movie menu

Description automatically generated

5. Display the video number and its rental frequency for all videos that have been rented a minimum of 2 times.

Solution:

SELECT VID\_NUM AS "Video Number", COUNT(\*) AS "Rental Frequency"

FROM DETAILRENTAL

GROUP BY VID\_NUM

HAVING COUNT(\*) >= 2;

Result:

A screenshot of a computer

Description automatically generated

**6.** Identify the year in which the highest number of movies were added to the database. Provide both the year and the corresponding movie count.

(**NOTE**: You might need to sort the count of movies per year and select the top entry. Use ROWNUM in Oracle, or LIMIT in other RDBMS.)

Solution:

SELECT MOVIE\_YEAR AS "Year", COUNT(\*) AS "Movie Count"

FROM MOVIE

GROUP BY MOVIE\_YEAR

ORDER BY COUNT(\*) DESC

FETCH FIRST 1 ROW ONLY;

Result:

A screenshot of a computer

Description automatically generated

7. List the membership number, first name, last name, and zip code of all members who have not rented any videos.

(**NOTE**: Use a nested query structure for this task, even though a JOIN could also achieve the same result.)

Solution:

SELECT MEM\_NUM AS "Membership Number",

MEM\_FNAME AS "First Name",

MEM\_LNAME AS "Last Name",

MEM\_ZIP AS "Zip Code"

FROM MEMBERSHIP

WHERE MEM\_NUM NOT IN (

SELECT DISTINCT MEM\_NUM

FROM RENTAL

);

Result:

A screenshot of a computer

Description automatically generated

8.Identify all members who have rented more than 2 videos. Display their membership numbers in the results.

Solution:

SELECT MEM\_NUM AS "Membership Number"

FROM RENTAL

GROUP BY MEM\_NUM

HAVING COUNT(\*) > 2;

Result:

A black background with white text

Description automatically generated

There’s no such data which fits are query.

**9.** Identify movies that have never been rented. Display their movie titles in the result.

(**NOTE**: Consider using a subquery structure combined a join operation)

Solution:

SELECT MOVIE\_TITLE

FROM MOVIE

LEFT JOIN DETAILRENTAL ON MOVIE.MOVIE\_NUM = DETAILRENTAL.VID\_NUM

WHERE DETAILRENTAL.VID\_NUM IS NULL;

Result:

A screenshot of a computer screen

Description automatically generated

**10.** Display the minimum, maximum and average balances of memberships that have made rentals. Round the average balance to two decimal places.

SELECT

MIN(MEM\_BALANCE) AS "Minimum Balance",

MAX(MEM\_BALANCE) AS "Maximum Balance",

ROUND(AVG(MEM\_BALANCE), 2) AS "Average Balance"

FROM

MEMBERSHIP

WHERE

EXISTS (

SELECT 1

FROM RENTAL

WHERE RENTAL.MEM\_NUM = MEMBERSHIP.MEM\_NUM

);

Result:

A black screen with white text

Description automatically generated

**11.** List the movie title, genre, price description, and rental fee of all movies.

(**NOTE**: Utilize the join operator for this query)

Solution:

SELECT

MOVIE.MOVIE\_TITLE AS "Movie Title",

MOVIE.MOVIE\_GENRE AS "Genre",

PRICE.PRICE\_DESCRIPTION AS "Price Description",

PRICE.PRICE\_RENTFEE AS "Rental Fee"

FROM

MOVIE

JOIN

VIDEO ON MOVIE.MOVIE\_NUM = VIDEO.MOVIE\_NUM

JOIN

PRICE ON MOVIE.PRICE\_CODE = PRICE.PRICE\_CODE;

Result:

A screen shot of a movie list

Description automatically generated

A screen shot of a computer

Description automatically generated

**12.** Calculate the total number of videos available for each movie title.

Solution:

SELECT MOVIE.MOVIE\_TITLE AS "Movie Title",

COUNT(\*) AS "Total Videos"

FROM MOVIE

JOIN VIDEO ON MOVIE.MOVIE\_NUM = VIDEO.MOVIE\_NUM

GROUP BY MOVIE.MOVIE\_TITLE;

Result:

A screenshot of a movie

Description automatically generated

**13**. List the full names of members (formatted as first name followed by |last name), the movie titles they've rented, and the due dates associated with each rental. Order the results by the members’ full names.

Solution: SELECT

MEMBERSHIP.MEM\_FNAME || ' | ' || MEMBERSHIP.MEM\_LNAME AS "Name",

MOVIE.MOVIE\_TITLE AS "Movie Title",

DETAILRENTAL.DETAIL\_DUEDATE AS "Due Date"

FROM

MEMBERSHIP

JOIN

RENTAL ON MEMBERSHIP.MEM\_NUM = RENTAL.MEM\_NUM

JOIN

DETAILRENTAL ON RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

JOIN

MOVIE ON DETAILRENTAL.VID\_NUM = MOVIE.MOVIE\_NUM

ORDER BY

"Name";

Result:



**14**. Display the rental number, rental date, video number, movie title, due date, and return date for all videos returned past their due date. Order the query result by rental number followed by movie title.

Solution: SELECT

RENTAL.RENT\_NUM AS "Rental Number",

RENTAL.RENT\_DATE AS "Rental Date",

VIDEO.VID\_NUM AS "Video Number",

MOVIE.MOVIE\_TITLE AS "Movie Title",

DETAILRENTAL.DETAIL\_DUEDATE AS "Due Date",

DETAILRENTAL.DETAIL\_RETURNDATE AS "Return Date"

FROM

RENTAL

JOIN

DETAILRENTAL ON RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

JOIN

VIDEO ON DETAILRENTAL.VID\_NUM = VIDEO.VID\_NUM

JOIN

MOVIE ON VIDEO.MOVIE\_NUM = MOVIE.MOVIE\_NUM

WHERE

DETAILRENTAL.DETAIL\_RETURNDATE > DETAILRENTAL.DETAIL\_DUEDATE

ORDER BY

"Rental Number", "Movie Title";

Result:

A screenshot of a computer

Description automatically generated

**15.** Retrieve the membership number, first name, last name, and the total number of rentals. Include members who haven’t rented any videos, displaying their rental count as 0. Order the results by total rentals in descending order.

Query:

SELECT

MEMBERSHIP.MEM\_NUM AS "Membership Number",

MEMBERSHIP.MEM\_FNAME AS "First Name",

MEMBERSHIP.MEM\_LNAME AS "Last Name",

COUNT(RENTAL.RENT\_NUM) AS "Total Rentals"

FROM

MEMBERSHIP

LEFT JOIN

RENTAL ON MEMBERSHIP.MEM\_NUM = RENTAL.MEM\_NUM

GROUP BY

MEMBERSHIP.MEM\_NUM, MEMBERSHIP.MEM\_FNAME, MEMBERSHIP.MEM\_LNAME

ORDER BY

"Total Rentals" DESC;

Result:

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

Q 16. For each movie, determine if it has been rented out, and if so, provide the name of the last member who rented it and the corresponding rental date.

Query:

SELECT

MOVIE.MOVIE\_TITLE AS "Movie Title",

MEMBERSHIP.MEM\_FNAME AS "First Name",

MEMBERSHIP.MEM\_LNAME AS "Last Name",

MAX(RENTAL.RENT\_DATE) AS "Last Rental Date"

FROM

MOVIE

LEFT JOIN

DETAILRENTAL ON MOVIE.MOVIE\_NUM = DETAILRENTAL.VID\_NUM

LEFT JOIN

RENTAL ON DETAILRENTAL.RENT\_NUM = RENTAL.RENT\_NUM

LEFT JOIN

MEMBERSHIP ON RENTAL.MEM\_NUM = MEMBERSHIP.MEM\_NUM

GROUP BY

MOVIE.MOVIE\_TITLE, MEMBERSHIP.MEM\_FNAME, MEMBERSHIP.MEM\_LNAME

ORDER BY

"Movie Title";

Result: A screen shot of a computer

Description automatically generated

**17.** List the membership number, first name, and last name of members who have rented both movies “Richard Goodhope” and “Beatnik Fever”. These movies may have been rented at different times with separate rental numbers.

Query: SELECT MEMBERSHIP.MEM\_NUM, MEMBERSHIP.MEM\_FNAME, MEMBERSHIP.MEM\_LNAME

FROM RENTAL

JOIN DETAILRENTAL ON RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

JOIN MOVIE ON DETAILRENTAL.VID\_NUM = MOVIE.MOVIE\_NUM

JOIN MEMBERSHIP ON RENTAL.MEM\_NUM = MEMBERSHIP.MEM\_NUM

WHERE MOVIE.MOVIE\_TITLE = 'Richard Goodhope'

INTERSECT

SELECT MEMBERSHIP.MEM\_NUM, MEMBERSHIP.MEM\_FNAME, MEMBERSHIP.MEM\_LNAME

FROM RENTAL

JOIN DETAILRENTAL ON RENTAL.RENT\_NUM = DETAILRENTAL.RENT\_NUM

JOIN MOVIE ON DETAILRENTAL.VID\_NUM = MOVIE.MOVIE\_NUM

JOIN MEMBERSHIP ON RENTAL.MEM\_NUM = MEMBERSHIP.MEM\_NUM

WHERE MOVIE.MOVIE\_TITLE = 'Beatnik Fever';

Result:

A black background with white text

Description automatically generated