Parul University

Name: Prerak Doshi
Email: prerak12102004@gmail.com
Roll no: 25UG033422
Phone: 8849921118
Branch: Parul University
Department: CSE10_Batch 1
Batch: 2028
Degree: B.Tech - CSE

Scan to verify results

PIET_Oracle DBMS_Course

PIEt_oracle DBMS_Session 8_CY_Updated

Attempt : 2
Total Mark : 40
Marks Obtained : 40

Section 1 : coding

1. Problem Statement

Sona, a data analyst at a private institute, has been tasked with analyzing course enrollment data by combining information from STUDENTS, COURSES, and ENROLLMENTS tables. She needs to perform JOIN operations to understand enrollment patterns, course popularity, and student engagement for academic planning and strategic decision-making.

Table Details:

Symbol refers to the primary key

Symbol refers to the Foreign key

NN refers to Not NULL

Sample Input Records

Tasks to perform:

Display all course details along with the number of students enrolled in each course to understand the popularity and demand for different courses.Find all students enrolled in courses that have more than 10 students to identify popular courses and their participants for targeted communication and resource planning.List courses and students where course fee is greater than 5000 to analyze the enrollment patterns in high-value courses and understand the premium segment customer base.

Note: The user must write only the view queries.

Answer

oracle.sql

```
SELECT C.*, COUNT(E.STUDENT_ID) AS ENROLLMENT_COUNT
FROM COURSES C
LEFT JOIN ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
GROUP BY C.COURSE_ID, C.COURSE_NAME, C.COURSE_FEE, C.DURATION;

SELECT S.STUDENT_NAME, C.COURSE_NAME
FROM STUDENTS S
JOIN ENROLLMENTS E ON S.STUDENT_ID = E.STUDENT_ID
JOIN COURSES C ON E.COURSE_ID = C.COURSE_ID
WHERE C.COURSE_ID IN (
   SELECT COURSE_ID
   FROM ENROLLMENTS
   GROUP BY COURSE_ID
   HAVING COUNT(*) > 3
);

SELECT C.COURSE_NAME, S.STUDENT_NAME, E.ENROLLMENT_DATE
FROM COURSES C
```

```
JOIN ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
JOIN STUDENTS S ON E.STUDENT_ID = S.STUDENT_ID
WHERE C.COURSE_FEE > 5000;

/*
```

Status : Correct **Marks : 10/10**

2. Problem Statement

George, a data analyst at a sports analytics company, is tasked with analyzing player performance and game outcomes to provide insights into the best-performing teams and players. To accomplish this, George needs to perform joins between the PLAYERS, GAMES and PERFORMANCE tables.

Table Details:

Sample Input Records:

Table Name: PLAYERS

Table Name: GAMES

Table Name: PERFORMANCE

Tasks to perform:

Write a query to fetch the PLAYER_NAME and the GAME_NAME they participated in. Display the results by joining the PLAYERS, GAMES, and PERFORMANCE tables. Write a query to calculate the total score for each player by joining the PLAYERS and PERFORMANCE tables. Display the

PLAYER_NAME and the total score.Write a query to fetch the GAME_NAME and PLAYER_NAME for players belonging to the 'Dragons' team. Display the results by joining the PLAYERS, GAMES, and PERFORMANCE tables. Write a query to fetch the GAME_NAME and the highest SCORE in each game by joining the GAMES and PERFORMANCE tables.Write a query to fetch the PLAYER_NAME of players who have played for more than 40 minutes in at least one game by joining the PLAYERS and PERFORMANCE tables.

Note:

The user must write only the join queries to select the required records.

Answer

oracle.sql

```
SELECT p.PLAYER_NAME, g.GAME_NAME
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
JOIN GAMES g ON pf.GAME_ID = g.GAME_ID;

SELECT p.PLAYER_NAME, SUM(pf.SCORE) AS TOTAL_SCORE
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
GROUP BY p.PLAYER_NAME;

SELECT g.GAME_NAME, p.PLAYER_NAME
FROM GAMES g
JOIN PERFORMANCE pf ON g.GAME_ID = pf.GAME_ID
JOIN PLAYERS p ON pf.PLAYER_ID = p.PLAYER_ID
WHERE p.TEAM_NAME = 'Dragons';

SELECT g.GAME_NAME, MAX(pf.SCORE) AS HIGHEST_SCORE
FROM GAMES g
JOIN PERFORMANCE pf ON g.GAME_ID = pf.GAME_ID
GROUP BY g.GAME_NAME;

SELECT DISTINCT p.PLAYER_NAME
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
WHERE pf.PLAY_TIME > 40;
```

Status : Correct *Marks : 10/10*

3. Problem Statement

Emma, a data analyst at an e-commerce company, has been tasked with analyzing various aspects of the product inventory and orders to provide actionable insights for the management team. To do this, she needs to perform join queries on the PRODUCTS and ORDERS tables.

Table Details:

Sample Input Records:

Table Name: ORDERS

Table Name: PRODUCTS

Tasks to perform:

Write a query to retrieve the ORDER_ID, PRODUCT_NAME, QUANTITY, and TOTAL_PRICE for each order. The query should join the ORDERS table with the PRODUCTS table to get the product details using PRODUCT_ID.Write a query to find the PRODUCT_NAME of products which has order ID 4. The query should use a join between the PRODUCTS and ORDERS tables using PRODUCT_ID.Write a query to calculate the total quantity sold for each product. Retrieve the PRODUCT_NAME and the total quantity sold with an alias name TOTAL_QUANTITY_SOLD. The query should join the PRODUCTS and ORDERS tables using PRODUCT_ID and group the results by PRODUCT_NAME.Write a query to list the PRODUCT_NAME and TOTAL_PRICE for products where the total price in any order exceeds 500. The query should join the PRODUCTS and ORDERS tables using PRODUCT_ID and filter based on the TOTAL_PRICE.Write a query to find products where the STOCK is greater than the QUANTITY ordered in any order. Retrieve the PRODUCT_NAME, STOCK, and QUANTITY of such products. The query should join the PRODUCTS and ORDERS tables using PRODUCT_ID and compare the stock to the ordered quantity.

Note: The user must write only the join queries to select the required

records.

Answer

oracle.sql

```
SELECT o.ORDER_ID, p.PRODUCT_NAME, o.QUANTITY, o.TOTAL_PRICE
FROM ORDERS o
JOIN PRODUCTS p ON o.PRODUCT_ID = p.PRODUCT_ID;

SELECT p.PRODUCT_NAME
FROM PRODUCTS p
LEFT JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE o.ORDER_ID = 4;

SELECT p.PRODUCT_NAME, SUM(o.QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
GROUP BY p.PRODUCT_NAME;

SELECT p.PRODUCT_NAME, o.TOTAL_PRICE
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE o.TOTAL_PRICE > 500;

SELECT p.PRODUCT_NAME, p.STOCK, o.QUANTITY
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE p.STOCK > o.QUANTITY;
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Ravi manages a system tracking airline miles earned by frequent flyers. He wants to calculate total miles per flyer, average miles per flight, and identify flyers with no recorded miles.

Table Details:

Sample Input Records:

Tasks to perform:

Calculate total miles earned per flyer.Calculate average miles per flight across all flyers.Identify flyers with no miles recorded.

Note: The user must write only the join queries to select the required records.

Answer

oracle.sql

```
SELECT F.FLYER_NAME, COALESCE(SUM(M.MILES_EARNED), 0) AS
TOTAL_MILES
FROM FLYERS F
LEFT JOIN MILES M ON F.FLYER_ID = M.FLYER_ID
GROUP BY F.FLYER_NAME
ORDER BY F.FLYER_NAME;

SELECT ROUND(AVG(MILES_EARNED), 2) AS AVERAGE_MILES_PER_FLIGHT
FROM MILES;

SELECT F.FLYER_NAME
FROM FLYERS F
LEFT JOIN MILES M ON F.FLYER_ID = M.FLYER_ID
WHERE M.FLYER_ID IS NULL;
```

Status : Correct ***Marks : 10/10***