09 CTEs -Nikhil Sharma X KirkYagami

Common Table Expressions (CTEs)

CTEs are temporary result sets that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement.

```
-- Find programmers who earn more than the average salary using a CTE
WITH AverageSalary AS (
    SELECT AVG(Salary) AS avg_salary FROM programmers
)
SELECT p.Programmer_Name, p.Salary
FROM programmers p, AverageSalary a
WHERE p.Salary > a.avg_salary;

-- Calculate profit for each software and find the most profitable ones
WITH SoftwareProfits AS (
    SELECT Programmer_Name, Software_Name,
           (Software_Cost * Sold) - Development_Cost AS Profit
    FROM software
)
SELECT sp.Programmer_Name, sp.Software_Name, sp.Profit
FROM SoftwareProfits sp
ORDER BY sp.Profit DESC
LIMIT 3;
```

Real-World Scenarios and Complex Queries

Performance Review Dashboard

Create a dashboard that shows each programmer's salary relative to their peers with the same primary language:

```
-- Calculate average salary by primary language
WITH LanguageAvgSalary AS (
    SELECT Primary_Language, AVG(Salary) AS avg_lang_salary
    FROM programmers
    GROUP BY Primary_Language
)
SELECT p.Programmer_Name, p.Primary_Language, p.Salary,
       las.avg_lang_salary,
       ROUND((p.Salary / las.avg_lang_salary) * 100 - 100, 2) AS percent_diff_from_avg
FROM programmers p
JOIN LanguageAvgSalary las ON p.Primary_Language = las.Primary_Language
ORDER BY percent_diff_from_avg DESC;
```

Questions

Q1. Write a query to create a new column called `rating_category` with three categories based on the `ratings`

column:

- **High** → ratings **greater than or equal to 4.30**
- **Medium** → ratings **between 3.50 and 4.30 (exclusive of 4.30)**
- **Low** → ratings **less than or equal to 3.50**

Then, for each category, calculate:

1. The total count of products
2. The total sales
3. The percentage contribution of each category to the overall count and overall sales.

Exclude rows where the `ratings` column is NULL. Do **not** modify the original table.

```
WITH Categories AS (
    SELECT
        *,
        CASE
            WHEN rating >= 4.30 THEN 'high'
            WHEN rating BETWEEN 3.50 AND 4.30 THEN 'medium'
            ELSE 'low'
        END AS rating_category
    FROM
        BigBasket_Products
    WHERE
        rating IS NOT NULL
)

SELECT
    rating_category,
    COUNT(*) AS total_count,
    SUM(sale_price) AS total_sale_price,
    ROUND((COUNT(*) * 100.0) / (SELECT COUNT(*) FROM Categories), 2) AS count_percentage,
    ROUND((SUM(sale_price) * 100.0) / (SELECT SUM(sale_price) FROM Categories), 2) AS
sale_price_percentage
FROM
    Categories
GROUP BY
    rating_category;
```

rating_category	total_count	total_sale_price	count_percentage	sale_price_percentage
medium	625	136358	49.45	41.15
low	206	76837	16.30	23.19
high	433	118159	34.26	35.66

2. Write a query that will display the topmost preferred category for each brand based on the total sales of that category.

```
WITH CategorySales AS (
    SELECT
        brand,
        category,
```

```
        SUM(sale_price) AS total_sales,
        ROW_NUMBER() OVER (PARTITION BY brand ORDER BY SUM(sale_price) DESC) AS
category_rank
    FROM
        BigBasket_Products
    GROUP BY
        brand, category
)

SELECT
    brand,
    category,
    total_sales
FROM
    CategorySales
WHERE
    category_rank = 1;
```