nextleap

SQL 4 Data Analyst Fellowship

Saksham Arora

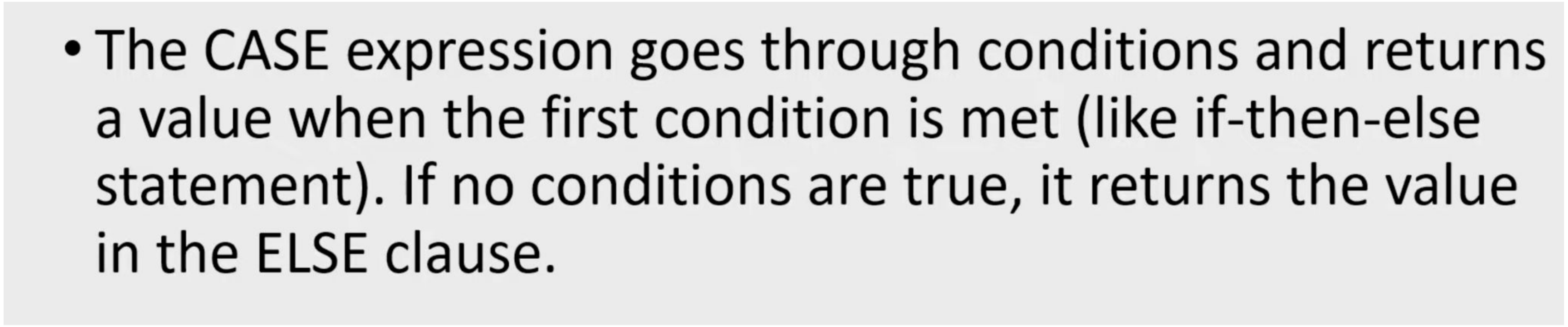
Software Engineer Microsoft

[Saksham Arora - Microsoft | LinkedIn]()

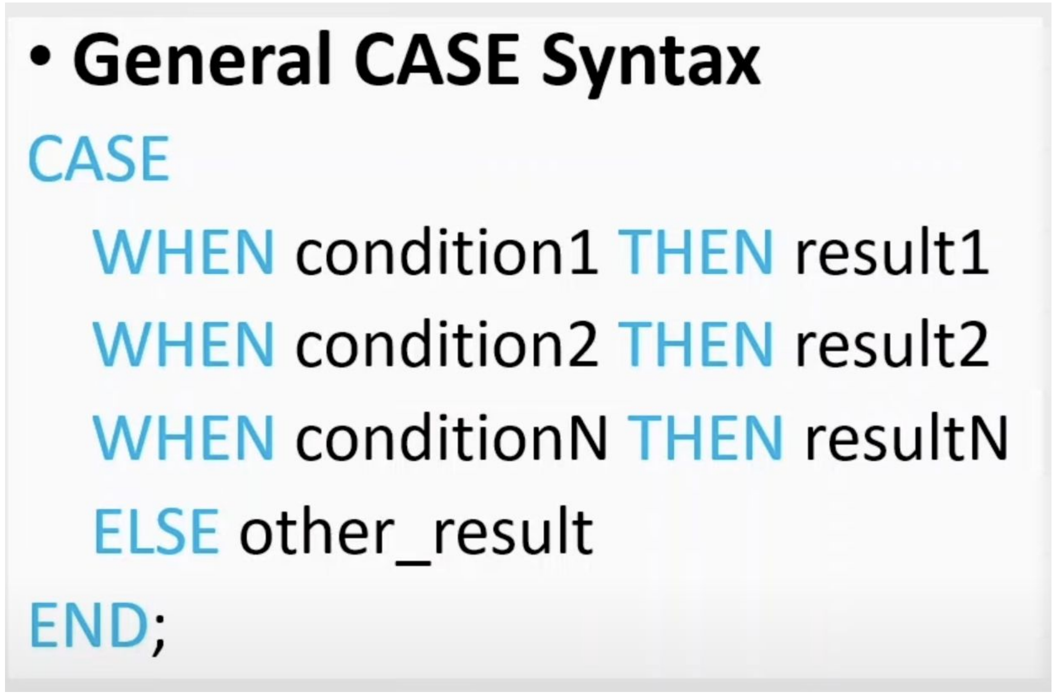
Session 4: CASE WHEN & Subqueries

1. **CASE WHEN**
    - Explanation
        - Define CASE WHEN syntax
        - Use cases of CASE WHEN:
            - Basic Syntax:
            - Multiple Conditions
            - Using CASE in SELECT vs. WHERE Clauses
    - Case 1: Customer Segmentation Analysis
        - Segmenting customers into categories based on their spending frequency.
        - Tasks:
            - Segment Customers by Spending:
                - Write a query using CASE WHEN in the SELECT clause to classify customers as "High Spenders," "Medium Spenders," or "Low Spenders" based on their total purchase amount.
2. **Subqueries**
    - Explanation
        - subqueries and their types (
        - How subqueries are executed(SELECT, FROM, WHERE clauses).
    - Case 2: Identifying Sales Trends with Subqueries
        - Scenario Overview:
            - The sales team needs a report to compare current sales against the previous quarter to identify top-performing products.
        - Tasks:
            - Subqueries in SELECT Clause:
                - Write a subquery to calculate the total sales per product for the current and previous quarters.
                - Use the subquery in the SELECT clause to compute the difference in sales between quarters.
            - Subqueries in WHERE Clause:
                - Use a subquery in the WHERE clause to filter for products with sales growth above 20%.
            - Subqueries with Joins:
                - Combine subqueries with INNER JOIN to merge results with the main sales table for a comprehensive report.

Case WHEN



- The CASE expression goes through conditions and returns a value when the first condition is met (like if-then-else statement). If no conditions are true, it returns the value in the ELSE clause.

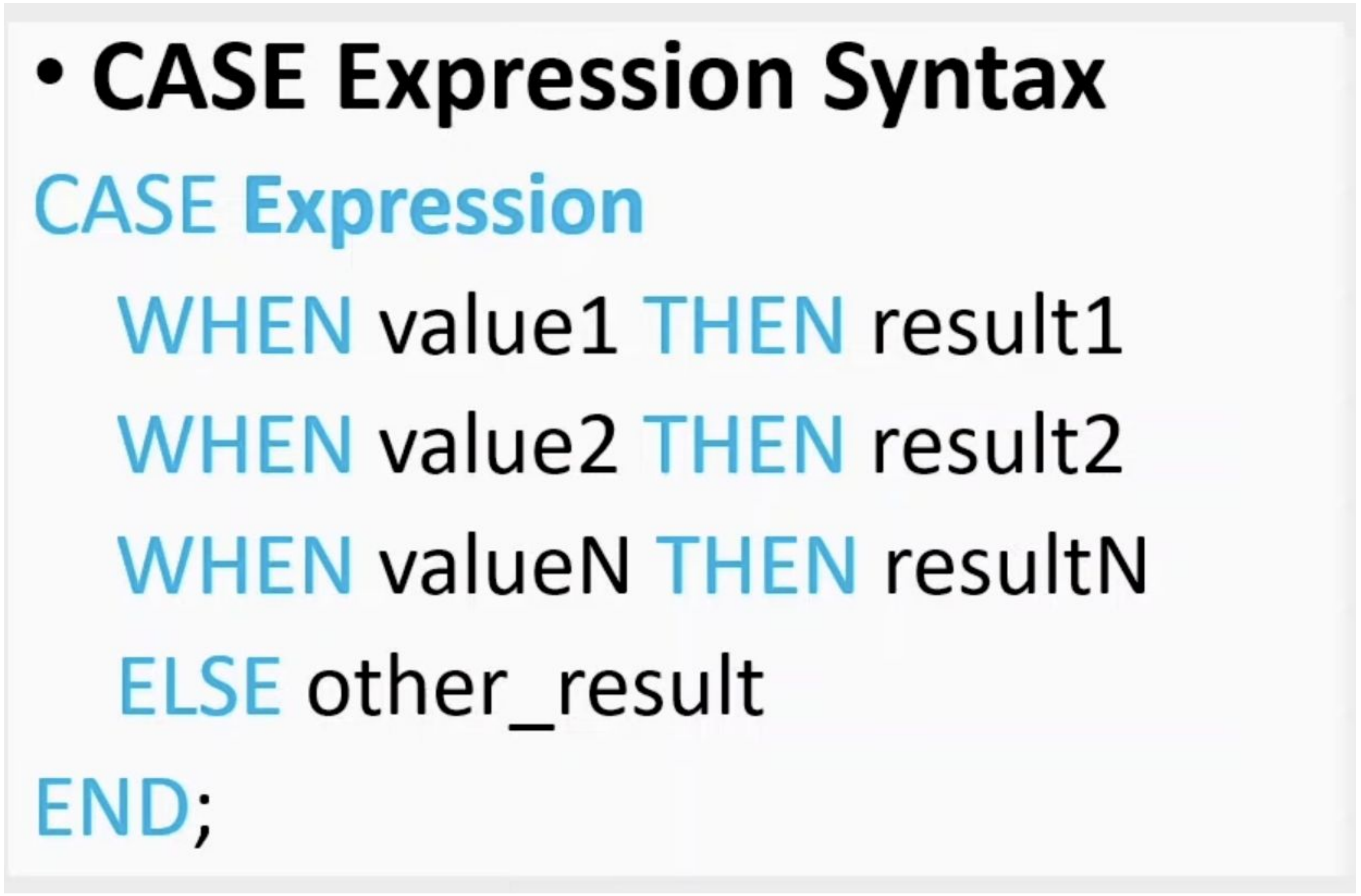


- **General CASE Syntax**

```
CASE
   WHEN condition1 THEN result1
   WHEN condition2 THEN result2
   WHEN conditionN THEN resultN
   ELSE other_result
END;
```

ELSE part here is optional.

If you don’t write ELSE part and no conditions are true, it returns NULL.



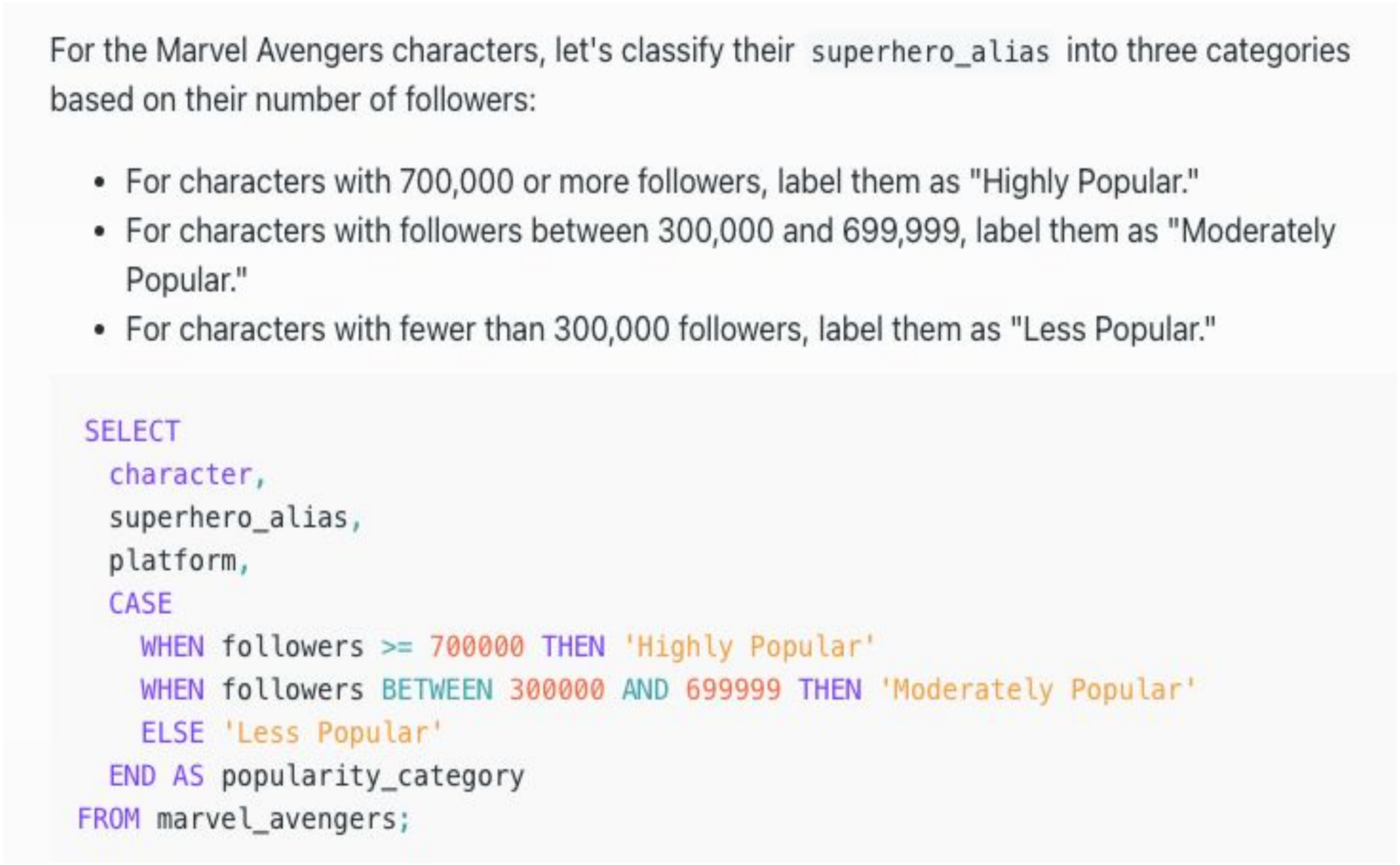
- **CASE Expression Syntax**

```
CASE Expression
   WHEN value1 THEN result1
   WHEN value2 THEN result2
   WHEN valueN THEN resultN
   ELSE other_result
END;
```

CASE WHEN Table

actor		superhero_alias	platform	followers	posts	engagement_rate	avg_likes	avg_comments
Robert Downey Jr.	Tony Stark	Iron Man	Instagram	500000	200	8.20	12000	800
Chris Evans	Steve Rogers	Captain America	Twitter	300000	150	6.50	8000	500
Scarlett Johansson	Natasha Romanoff	Black Widow	Instagram	700000	300	7.80	15000	1000
Chris Hemsworth	Thor	Thor	YouTube	400000	100	9.10	20000	1200
Mark Ruffalo	Bruce Banner	Hulk	Twitter	200000	80	5.30	6000	400

Example 1 for CASE WHEN



For the Marvel Avengers characters, let's classify their `superhero_alias` into three categories based on their number of followers:

- For characters with 700,000 or more followers, label them as "Highly Popular."
- For characters with followers between 300,000 and 699,999, label them as "Moderately Popular."
- For characters with fewer than 300,000 followers, label them as "Less Popular."

```sql
 SELECT
  character,
  superhero_alias,
  platform,
  CASE
    WHEN followers >= 700000 THEN 'Highly Popular'
    WHEN followers BETWEEN 300000 AND 699999 THEN 'Moderately Popular'
    ELSE 'Less Popular'
  END AS popularity_category
FROM marvel_avengers;
```

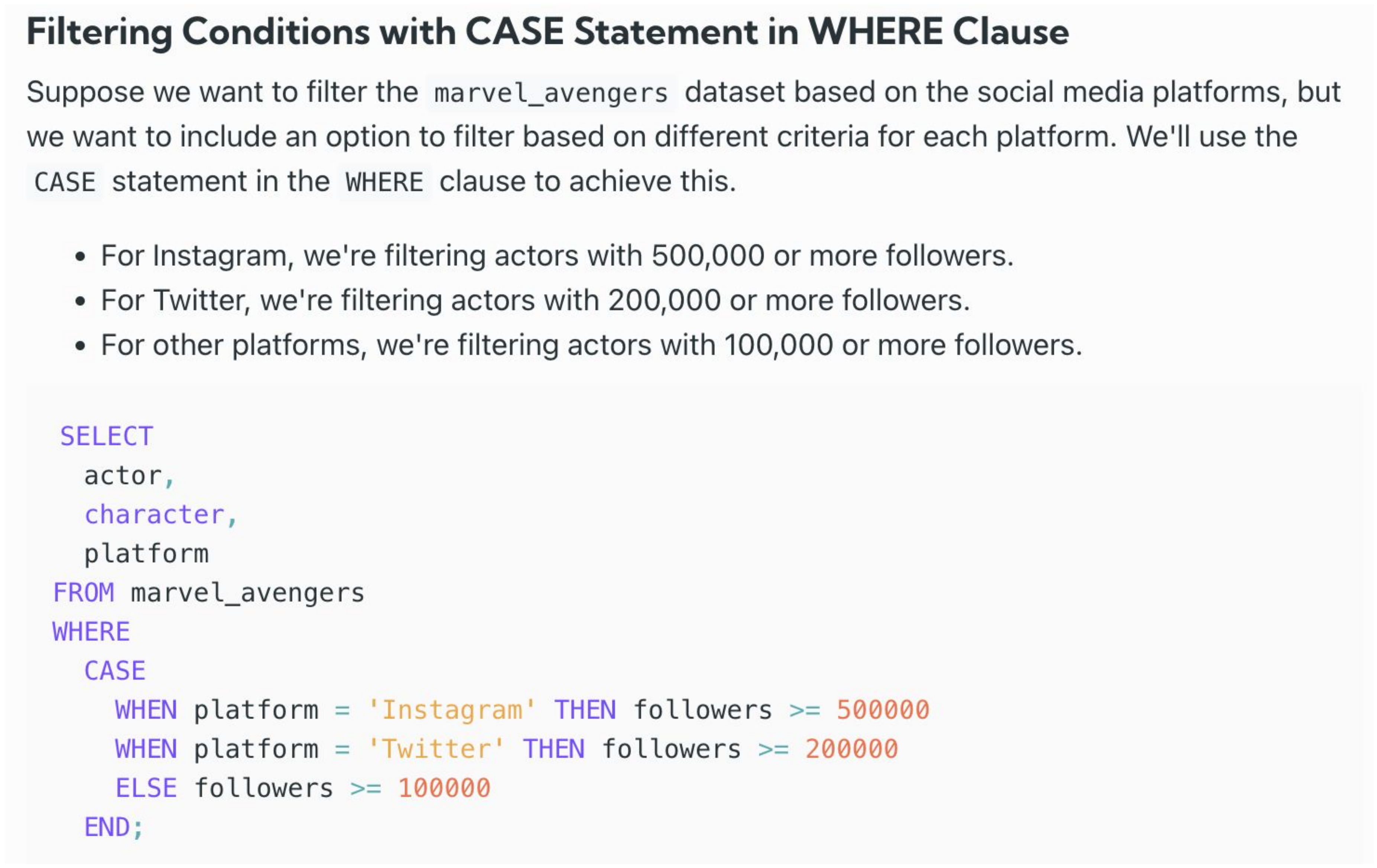
Using CASE Statement in WHERE Clause

The `CASE` statement in the `WHERE` clause is used to **filter rows based on specified conditions** within the dataset.

```
 SELECT
  column_1,
  column_2
FROM table_1
WHERE CASE
    WHEN condition_1 THEN result_1
    WHEN condition_2 THEN result_2
    WHEN ... THEN ...
    ELSE result_3 -- If condition_1 and condition_2 are not met, return result_3 in
  END;
```

The `CASE` statement evaluates conditions for each row, determining whether it meets the filtering criteria. Rows satisfying the conditions specified in the `CASE` statement are included in the result.

Example 2 for CASE WHEN



Filtering Conditions with CASE Statement in WHERE Clause

Suppose we want to filter the `marvel_avengers` dataset based on the social media platforms, but we want to include an option to filter based on different criteria for each platform. We'll use the `CASE` statement in the `WHERE` clause to achieve this.

- For Instagram, we're filtering actors with 500,000 or more followers.
- For Twitter, we're filtering actors with 200,000 or more followers.
- For other platforms, we're filtering actors with 100,000 or more followers.

```sql
 SELECT
   actor,
   character,
   platform
 FROM marvel_avengers
 WHERE
   CASE
     WHEN platform = 'Instagram' THEN followers >= 500000
     WHEN platform = 'Twitter' THEN followers >= 200000
     ELSE followers >= 100000
   END;
```

What is a Subquery?

Subqueries, also known as inner queries, are powerful tools to embed one query within another. By nesting queries within parentheses, you can generate temporary tables to perform calculations and filter data within the main query. Subqueries enable granular control over your data, enhancing the precision of your analysis.

- Concept of outer and inner query
- First inner query will execute then outer query will use result of inner query and then execute

Subquery Table Example

artist_id	artist_name	genre	concert_revenue	year_of_formation	country	number_of_members	album_released	label
103	Taylor Swift	Pop	700000	2004	United States	1	9	Republic Records
104	BTS	K-Pop	800000	2013	South Korea	7	7	Big Hit Music
105	Adele	Pop	600000	2006	United Kingdom	1	3	Columbia Records
109	Blackpink	K-Pop	450000	2016	South Korea	4	5	YG Entertainment
110	Maroon 5	Pop	550000	1994	United States	5	7	Interscope Records

Example 1 for Subqueries

Single-Value Comparison in WHERE Clauses: When you need to compare a single value to a result from another query, utilize the subquery in the WHERE clause to enable dynamic data filtering. This enhances query flexibility and precision by allowing on-the-fly condition adjustments based on subquery results.

```
    SELECT artist_name
    FROM concerts
    WHERE concert_revenue > (
  SELECT AVG(concert_revenue) FROM concerts);
```

Example for Correlated Subquery

Let’s suppose we need to find the artist with 3rd highest revenue.

```
SELECT artist_id, artist_name
FROM concerts as E1
WHERE 2 = (
  SELECT COUNT(artist_id) FROM concerts as E2
  WHERE E2.concert_revenue > E1.concert_revenue
)
```

Problem Statements

1. Calculate Special Bonus - LeetCode
2. https://nextleap.app/problem/compressed-mode
3. https://nextleap.app/problem/retail-industry-leader
4. https://nextleap.app/interview-preparation/sql/questions/topic/subquery

nextleap

Q&A

What's on your mind?

SKELLYSCOO
I have several questions.

Feed us back!