Search Fundamentals → Query Re-formulation Metric

Table search: session_id, user_id, query, timestamp

1. **Objective:**
    a. **Query Reformulation Rate (Percentage of sessions with reformulations)**
    b. **Average Number of Query Reformulations per Session**
2. **Learning:**
    a. **Greatest**
        i. Greatest vs. Coalesce
    b. Count(*) **FILTER** (WHERE X>1) → to create a formula
        i. Order of execution in this case

Query Reformulation Rate (percentage of sessions with reformulations)

```
With session_query_count AS(
Select
Session_id,
count(DISTINCT query) as num_queries
From search_logs
Group by session_id
)
Select
count(*) FILTER (where num_queries >1)*100.0/count(*) as query_reformulation_rate
From session_query_count;
```

Explanation:

1. Count distinct queries per session
2. **Sessions with num_queries > 1 are considered reformulated**
3. Divide by total sessions to get the percentage

Average Number of Query Reformulations per Session

```
With session_query_count AS (
Select
Session_id,
count(distinct query) AS num_queries
From search_logs
Group by session_id
)
```

```
Select
AVG(GREATEST(num_queries - 1,0)) as avg_query_reformulations_per_sessions
From search_query_count
```

Explanation:

1. For each session, subtract 1 (the initial query) to count **only reformulations**.
2. Use **GREATEST(...,0)** to avoid negative values for sessions with only one query
3. Average across all sessions

Reformulations per user

```
With user_session_queries AS (
Select
User_id,
Session_id,
count(DISTINCT query) as num_queries
From search_logs
Group by user_id, session_id
)
Select
user_id,
AVG(GREATEST(num_queries - 1,0) as avg_reformulations_per_session
From user_session_queries
Group by user_id
```

Key Insight: → Shows **which users tend to reformulate more**, useful for personalization analysis

Summary:

1. Num_queries > 1 → session involved reformulation
    a. **Interpretation/Key Note: num_queries - 1 or num_queries > 1 → Counts only reformulations (ignores the first query)**
2. Percentage of such sessions = Query Reformulation Rate
3. Average num_queries -1 per session = Average Query Reformulations per Session

---

Filter Clause (PostgreSQL/modern SQL)

1. Purpose: Allows to **apply an aggregate function to a subset of rows** without using a separate WHERE clause
2. Syntax:
    a. AGG_FUNCTION(column) FILTER (WHERE condition)

Greatest → Returns the largest value
Filter → Applies aggregate on a subset of rows