# 🔍 Snowflake Views vs. Materialized Views

## 🌟 Overview

In Snowflake, both **Views** and **Materialized Views** allow users to define virtual tables based on SQL queries. However, their behavior, performance impact, and use cases differ significantly. This document explains the key differences using real-world scenarios from industry giants like **Meta**, **Amazon**, and **Google**.

## 🏢 Problem: Slow Analytical Queries on Real-Time Data

### Industry Use Case: Meta’s Real-Time Feed Recommendations

Meta runs constant analytics on user activity to tune real-time news feed ranking. The base tables (e.g., clicks, likes, views) are huge and updated constantly.

**Challenge:** Recomputing complex joins or aggregations on-the-fly becomes expensive when the view is queried repeatedly.

### ✅ Snowflake’s Solution

* Use **Views** for ad-hoc exploration, lightweight reuse.
* Use **Materialized Views** for performance-sensitive use cases where precomputed results are beneficial.

## 🔒 What is a View?

### 🔍 Concept

A **View** is a stored SQL query that runs every time it is referenced. It does **not store** any data.

### ⚡ Characteristics

* Real-time reflection of base tables
* Low storage cost
* No performance gain (executes full query each time)

### 💡 Example (Amazon Redshift Migration Team):

CREATE VIEW ORDER\_SUMMARY AS  
SELECT CUSTOMER\_ID, COUNT(\*) AS TOTAL\_ORDERS  
FROM ORDERS  
GROUP BY CUSTOMER\_ID;

Used for: Simple reporting dashboards where data freshness is critical and query cost is acceptable.

## 🔒 What is a Materialized View?

### 🔍 Concept

A **Materialized View** stores the **results** of the SQL query physically. It is automatically refreshed in the background as base tables change.

### ⚡ Characteristics

* Faster query performance
* Consumes storage
* May lag slightly behind real-time (depends on refresh frequency)

### 💡 Example (Google Ads Click Aggregation):

CREATE MATERIALIZED VIEW CLICK\_AGG\_MV AS  
SELECT CAMPAIGN\_ID, COUNT(\*) AS TOTAL\_CLICKS  
FROM AD\_CLICKS  
GROUP BY CAMPAIGN\_ID;

Used for: Performance-intensive queries like dashboards, ML model input tables, and frequently queried metrics.

## 📊 Feature Comparison

| Feature | View | Materialized View |
| --- | --- | --- |
| Data Stored | No | Yes |
| Query Performance | No optimization | Fast (precomputed results) |
| Storage Cost | Minimal | Higher (depends on volume) |
| Reflects Real-Time Data | Yes | No (until refresh) |
| Auto Refresh on Base Change | Not applicable | Yes (automatic or manual) |
| Use Case | Exploration, lightweight BI | Dashboards, ML input tables |

## 📆 Refresh Behavior

Materialized Views are **incrementally and automatically refreshed**:

-- Check refresh history  
SELECT \* FROM TABLE(INFORMATION\_SCHEMA.MATERIALIZED\_VIEW\_REFRESH\_HISTORY)  
WHERE NAME = 'CLICK\_AGG\_MV';

Snowflake optimizes refresh using micro-partition changes and only recalculates impacted portions.

## 📊 Performance Benchmark

### Meta Feed Analytics Benchmark

* **Base Table:** USER\_ACTIONS (~5B rows)
* **Query:** Aggregation of reactions per post

| Method | Time to Query |
| --- | --- |
| View | ~18 seconds |
| Materialized View | ~1.2 seconds |

## 📊 When to Use What?

### ✉ Use **Views** When:

* You need real-time accuracy
* The query is rarely used
* Underlying data changes frequently
* You want to save storage

### 🚀 Use **Materialized Views** When:

* The same complex query runs often
* You need speed (dashboards, ML pipelines)
* Base table change frequency is low-to-moderate

## 💡 Pro Tips from Big Tech

* **Netflix**: Materialized views power their recommendation dashboards for quick filtering by user segments.
* **Apple**: Uses views during prototyping of internal HR analytics to avoid redundant data copies.
* **Amazon**: Runs materialized views on historical order data to speed up supplier performance metrics.

## 🎯 Summary

| Use Case | Recommended Option |
| --- | --- |
| Ad-hoc data exploration | View |
| Dashboard for 10M+ rows | Materialized View |
| Historical aggregation | Materialized View |
| Quick insights during dev | View |

Snowflake gives you the flexibility to balance cost, performance, and freshness by choosing between Views and Materialized Views.

Use **Views** for agility. Use **Materialized Views** for performance.