# 📈 Lab Manual: SYSTEM$CLUSTERING\_INFORMATION Deep Dive

## 🌟 Objective

Understand how Snowflake internally monitors and evaluates clustering quality using SYSTEM$CLUSTERING\_INFORMATION. This lab helps professionals analyze clustering depth, overlaps, and determine when re-clustering is beneficial.

**Covers:**

* Metadata insights
* Clustering health analysis
* Understanding SYSTEM$CLUSTERING\_INFORMATION

## 🪧 Use Case: Clustering for Query Optimization

**Industry Case (e.g., Amazon/Stripe):** In e-commerce or fintech systems, millions of daily transactions are joined or filtered by customer\_id and transaction\_date. Snowflake tables ingest data in time order, but sometimes out-of-order (due to late-arriving data). This leads to **clustering drift**.

Clustering helps optimize such queries and Snowflake provides SYSTEM$CLUSTERING\_INFORMATION to measure its health.

## 📊 Step 1: Create the Database and Clustered Table

-- Step 1.1: Create database and schema  
CREATE OR REPLACE DATABASE ANALYTICS\_LAB;  
USE DATABASE ANALYTICS\_LAB;  
USE SCHEMA PUBLIC;  
  
-- Step 1.2: Create the base TRANSACTIONS table  
CREATE OR REPLACE TABLE TRANSACTIONS (  
 TRANSACTION\_ID STRING,  
 CUSTOMER\_ID STRING,  
 TRANSACTION\_DATE DATE,  
 AMOUNT NUMBER(10, 2),  
 CHANNEL STRING  
);  
  
-- Step 1.3: Create clustered version  
CREATE OR REPLACE TABLE TRANSACTIONS\_CLUSTERED CLUSTER BY (CUSTOMER\_ID, TRANSACTION\_DATE)  
AS  
SELECT \* FROM TRANSACTIONS;

## 💰 Step 2: Insert Out-of-Order Records

-- Step 2.1: Insert older data first  
INSERT INTO TRANSACTIONS\_CLUSTERED  
SELECT  
 'TXN\_' || SEQ4(),  
 'CUST\_' || UNIFORM(1, 100, RANDOM()),  
 DATEADD(DAY, -UNIFORM(10, 20, RANDOM()), CURRENT\_DATE),  
 UNIFORM(10, 5000, RANDOM()),  
 'WEB'  
FROM TABLE(GENERATOR(ROWCOUNT => 5000));  
  
-- Step 2.2: Insert recent data next  
INSERT INTO TRANSACTIONS\_CLUSTERED  
SELECT  
 'TXN\_' || SEQ4(),  
 'CUST\_' || UNIFORM(1, 100, RANDOM()),  
 DATEADD(DAY, -UNIFORM(0, 5, RANDOM()), CURRENT\_DATE),  
 UNIFORM(10, 5000, RANDOM()),  
 'MOBILE'  
FROM TABLE(GENERATOR(ROWCOUNT => 5000));  
  
-- Step 2.3: Insert random date data (simulate out-of-order arrival)  
INSERT INTO TRANSACTIONS\_CLUSTERED  
SELECT  
 'TXN\_' || SEQ4(),  
 'CUST\_' || UNIFORM(1, 100, RANDOM()),  
 DATEADD(DAY, -UNIFORM(1, 25, RANDOM()), CURRENT\_DATE),  
 UNIFORM(10, 5000, RANDOM()),  
 'POS'  
FROM TABLE(GENERATOR(ROWCOUNT => 5000));

## 📒 Step 3: Inspect Clustering Metadata

-- Step 3.1: Get clustering stats  
SELECT SYSTEM$CLUSTERING\_INFORMATION('TRANSACTIONS\_CLUSTERED');

### 🔎 Key Fields:

* **average\_depth**: Avg # of overlapping partitions per value (lower is better)
* **total\_partition\_count**: Total partitions
* **total\_overlaps**: How many values exist in multiple partitions (higher = worse)
* **partition\_depth\_histogram**: Distribution of depth values

## 🔢 Step 4 (Optional): Recluster & Re-evaluate

If average\_depth > 4 or total\_overlaps is high, consider re-clustering:

-- Manually trigger recluster (or let Snowflake auto manage)  
ALTER TABLE TRANSACTIONS\_CLUSTERED RECLUSTER;  
  
-- Re-check clustering info  
SELECT SYSTEM$CLUSTERING\_INFORMATION('TRANSACTIONS\_CLUSTERED');

## 🔄 Step 5: Test Query Performance Impact

Try running queries before and after clustering:

-- Scan by customer and date range  
SELECT \*  
FROM TRANSACTIONS\_CLUSTERED  
WHERE CUSTOMER\_ID = 'CUST\_12'  
AND TRANSACTION\_DATE BETWEEN CURRENT\_DATE - 10 AND CURRENT\_DATE;

Go to **Snowsight > Query History > Profile View**

* Check **Pruning %** and **micro-partitions scanned**

## 📄 Summary Table

| Metric | Indicates |
| --- | --- |
| average\_depth | Row overlap across partitions |
| total\_partition\_count | Size of table by partition |
| total\_overlaps | Clustering degradation |
| partition\_depth\_histogram | Range/spread of clustering efficiency |

## 🎯 Learning Outcomes

* ✅ Interpret clustering metadata using system functions
* ✅ Understand how clustering quality degrades over time
* ✅ Make decisions about manual re-clustering
* ✅ Optimize pruning and performance via clustering