**Snowflake Advanced Labs**

**Lab 1: Clustering – Apply Clustering Key and Compare Performance**

**Objective:** Improve query performance by applying clustering keys.

**Steps:**

1. Create a large table without clustering:
2. CREATE OR REPLACE TABLE SALES\_DATA (
3. ORDER\_ID INT,
4. CUSTOMER\_ID INT,
5. REGION STRING,
6. ORDER\_DATE DATE,
7. AMOUNT DECIMAL(10,2)
8. );
9. Load synthetic data:
10. INSERT INTO SALES\_DATA
11. SELECT SEQ8(), UNIFORM(1,10000,RANDOM()),
12. CASE WHEN UNIFORM(1,4,RANDOM())=1 THEN 'East'
13. WHEN UNIFORM(1,4,RANDOM())=2 THEN 'West'
14. WHEN UNIFORM(1,4,RANDOM())=3 THEN 'North'
15. ELSE 'South' END,
16. CURRENT\_DATE - UNIFORM(1,365,RANDOM()),
17. UNIFORM(10,1000,RANDOM())
18. FROM TABLE(GENERATOR(ROWCOUNT=>500000));
19. Run a filter query on REGION and note query time:
20. SELECT SUM(AMOUNT) FROM SALES\_DATA WHERE REGION='East';
21. Create a **clustered table**:
22. CREATE OR REPLACE TABLE SALES\_CLUSTERED
23. CLUSTER BY (REGION) AS
24. SELECT \* FROM SALES\_DATA;
25. Re-run the same query:
26. SELECT SUM(AMOUNT) FROM SALES\_CLUSTERED WHERE REGION='East';

**Expected Outcome:** Query on SALES\_CLUSTERED should be faster due to partition pruning.

**Lab 2: Materialized View – Faster Analytics**

**Objective:** Use materialized views for pre-computed query results.

**Steps:**

1. Create a materialized view on the sales data:
2. CREATE OR REPLACE MATERIALIZED VIEW SALES\_BY\_REGION AS
3. SELECT REGION, SUM(AMOUNT) AS TOTAL\_SALES
4. FROM SALES\_DATA
5. GROUP BY REGION;
6. Query the view:
7. SELECT \* FROM SALES\_BY\_REGION;
8. Insert new data into base table:
9. INSERT INTO SALES\_DATA VALUES
10. (999999, 999, 'East', CURRENT\_DATE, 5000.00);
11. Query again:
12. SELECT \* FROM SALES\_BY\_REGION WHERE REGION='East';

**Expected Outcome:** The materialized view automatically updates and reflects new totals.

**Lab 3: Secure Share – Share Dataset Across Accounts**

**Objective:** Share data securely between Snowflake accounts.

**Steps (Provider account):**

1. Create a share:
2. CREATE SHARE SALES\_SHARE;
3. Grant usage and select permissions:
4. GRANT USAGE ON DATABASE PROJECT\_DB TO SHARE SALES\_SHARE;
5. GRANT USAGE ON SCHEMA PROJECT\_DB.RAW\_DATA TO SHARE SALES\_SHARE;
6. GRANT SELECT ON TABLE PROJECT\_DB.RAW\_DATA.SALES\_DATA TO SHARE SALES\_SHARE;
7. Add consumer account (replace with account locator):
8. ALTER SHARE SALES\_SHARE ADD ACCOUNTS = <consumer\_account\_locator>;

**Steps (Consumer account):**

1. Create a database from the share:
2. CREATE DATABASE SHARED\_DB FROM SHARE <provider\_account\_locator>.SALES\_SHARE;
3. Query shared data:
4. SELECT \* FROM SHARED\_DB.RAW\_DATA.SALES\_DATA;

**Expected Outcome:** Consumer queries provider’s table without copying data.

**Lab 4: Streams & Tasks – Change Capture Automation**

**Objective:** Track changes in a table and process them automatically.

**Steps:**

1. Create a stream:
2. CREATE OR REPLACE STREAM SALES\_STREAM ON TABLE SALES\_DATA;
3. Insert new records into sales table:
4. INSERT INTO SALES\_DATA VALUES (1000001, 101, 'West', CURRENT\_DATE, 1200.00);
5. Check stream contents:
6. SELECT \* FROM SALES\_STREAM;
7. Create an audit table:
8. CREATE OR REPLACE TABLE SALES\_AUDIT LIKE SALES\_DATA;
9. Create a task to capture changes:
10. CREATE OR REPLACE TASK SALES\_AUDIT\_TASK
11. WAREHOUSE = LAB\_WH
12. SCHEDULE = '5 MINUTE'
13. AS
14. INSERT INTO SALES\_AUDIT SELECT \* FROM SALES\_STREAM;
15. Resume the task:
16. ALTER TASK SALES\_AUDIT\_TASK RESUME;

**Expected Outcome:** Every few minutes, new changes flow into SALES\_AUDIT.

**Lab 5: Clone Table – Zero-Copy Development Copy**

**Objective:** Create instant dev/test copies using zero-copy cloning.

**Steps:**

1. Clone the sales table:
2. CREATE OR REPLACE TABLE SALES\_DEV CLONE SALES\_DATA;
3. Verify record count:
4. SELECT COUNT(\*) FROM SALES\_DEV;
5. Make changes in dev table:
6. DELETE FROM SALES\_DEV WHERE REGION='South';
7. Verify original table remains unaffected:
8. SELECT COUNT(\*) FROM SALES\_DATA WHERE REGION='South';

**Expected Outcome:** Dev copy (SALES\_DEV) can be modified independently without affecting original table.