**Snowflake Advanced Hands-On Labs**

**Lab 1: Clustering Keys for Performance Optimization**

**Objective:** Optimize query performance using clustering keys.

**Steps:**

1. Create a large sample 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 sample data (can use Snowflake sample DB or synthetic inserts).
10. INSERT INTO SALES\_DATA
11. SELECT SEQ8(), UNIFORM(1,10000,RANDOM()), REGION, CURRENT\_DATE - UNIFORM(1,365,RANDOM()), UNIFORM(10,1000,RANDOM())
12. FROM TABLE(GENERATOR(ROWCOUNT=>100000));
13. Run a filter query on REGION and check performance.
14. Create a clustered table:
15. CREATE OR REPLACE TABLE SALES\_CLUSTERED CLUSTER BY (REGION) AS
16. SELECT \* FROM SALES\_DATA;
17. Compare query performance on clustered vs non-clustered tables.
18. SELECT \* FROM SALES\_CLUSTERED WHERE REGION='East';

**Validation:** Clustered table scans fewer partitions, improving query speed.

**Lab 2: Materialized Views**

**Objective:** Create and query a materialized view for faster results.

**Steps:**

1. Create a sales summary materialized view:
2. CREATE OR REPLACE MATERIALIZED VIEW SALES\_SUMMARY 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\_SUMMARY;
8. Insert new data into the base table and check if view updates automatically.
9. INSERT INTO SALES\_DATA VALUES (100001, 555, 'North', CURRENT\_DATE, 999.99);
10. SELECT \* FROM SALES\_SUMMARY WHERE REGION='North';

**Validation:** The materialized view reflects updated totals.

**Lab 3: Secure Data Sharing Between Accounts**

**Objective:** Share data with another Snowflake account securely.

**Steps:**

1. Create a share:
2. CREATE SHARE SALES\_SHARE;
3. Grant usage on database and schema:
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 TO SHARE SALES\_SHARE;
7. Add a consumer account:
8. ALTER SHARE SALES\_SHARE ADD ACCOUNTS=<consumer\_account\_locator>;

**Validation:** Consumer can see shared database when they run SHOW DATABASES;.

**Lab 4: Role-Based Access Control (RBAC)**

**Objective:** Manage access using roles.

**Steps:**

1. Create a role:
2. CREATE ROLE ANALYST\_ROLE;
3. Assign role to a user:
4. GRANT ROLE ANALYST\_ROLE TO USER <username>;
5. Grant permissions:
6. GRANT USAGE ON WAREHOUSE LAB\_WH TO ROLE ANALYST\_ROLE;
7. GRANT SELECT ON TABLE PROJECT\_DB.RAW\_DATA.SALES TO ROLE ANALYST\_ROLE;
8. Switch role and test access:
9. USE ROLE ANALYST\_ROLE;
10. SELECT \* FROM PROJECT\_DB.RAW\_DATA.SALES;

**Validation:** User can only perform granted actions.

**Lab 5: Snowflake Streams & Tasks**

**Objective:** Automate incremental ETL with Streams and Tasks.

**Steps:**

1. Create a stream on the sales table:
2. CREATE OR REPLACE STREAM SALES\_STREAM ON TABLE SALES\_DATA;
3. Insert new records into SALES\_DATA.
4. Check stream contents (new rows tracked):
5. SELECT \* FROM SALES\_STREAM;
6. Create a task to process new data:
7. CREATE OR REPLACE TASK SALES\_ETL\_TASK
8. WAREHOUSE = LAB\_WH
9. SCHEDULE = '5 MINUTE'
10. AS
11. INSERT INTO SALES\_AUDIT SELECT \* FROM SALES\_STREAM;
12. Enable the task:
13. ALTER TASK SALES\_ETL\_TASK RESUME;

**Validation:** Task runs automatically, inserting new rows into SALES\_AUDIT.

**Lab 6: Zero-Copy Cloning**

**Objective:** Create dev/test environments instantly.

**Steps:**

1. Clone a table:
2. CREATE OR REPLACE TABLE SALES\_DEV CLONE SALES\_DATA;
3. Verify cloned data:
4. SELECT COUNT(\*) FROM SALES\_DEV;
5. Update dev table without affecting original:
6. DELETE FROM SALES\_DEV WHERE REGION='West';
7. SELECT COUNT(\*) FROM SALES\_DATA WHERE REGION='West'; -- unaffected

**Validation:** Original table remains unchanged.

**Lab 7: Snowflake Pricing Model Optimization**

**Objective:** Optimize cost by monitoring warehouses.

**Steps:**

1. Create a **resource monitor**:
2. CREATE OR REPLACE RESOURCE MONITOR MONTHLY\_MONITOR
3. WITH CREDIT\_QUOTA = 100
4. TRIGGERS ON 75 PERCENT DO NOTIFY
5. ON 90 PERCENT DO SUSPEND
6. ON 100 PERCENT DO SUSPEND\_IMMEDIATE;
7. Assign it to a warehouse:
8. ALTER WAREHOUSE LAB\_WH SET RESOURCE\_MONITOR = MONTHLY\_MONITOR;
9. Run queries to simulate credit usage.
10. Check usage:
11. SHOW RESOURCE MONITORS;

**Validation:** Warehouse suspends automatically when credit quota exceeded.