# Snowflake SQL Commands Exercise

**Step 1: Environment Setup**

1. **Sign in** to Snowsight and open **Projects → Worksheets**.
2. In the worksheet header, set:
   * **Role** → SYSADMIN (object creation)
   * **Warehouse** → COMPUTE\_WH (or create one if missing; see below)
3. (Optional) If you don’t have a warehouse yet, run:

-- As SYSADMIN

CREATE WAREHOUSE IF NOT EXISTS COMPUTE\_WH

WITH WAREHOUSE\_SIZE = 'XSMALL' AUTO\_SUSPEND = 60 AUTO\_RESUME = TRUE INITIALLY\_SUSPENDED = TRUE;

GRANT USAGE ON WAREHOUSE COMPUTE\_WH TO ROLE SYSADMIN;

**Step2 : Databases & Schemas**

**Goal:** Create an isolated playground.

USE ROLE SYSADMIN;

USE WAREHOUSE COMPUTE\_WH;

CREATE DATABASE IF NOT EXISTS LAB\_DB;

CREATE SCHEMA IF NOT EXISTS LAB\_DB.MY\_SCHEMA;

**Checkpoint:**

SHOW DATABASES LIKE 'LAB\_DB';

SHOW SCHEMAS IN DATABASE LAB\_DB;

**Step 3) Tables: Permanent, Transient, Temporary**

**Goal:** Understand lifecycle & recovery features.

-- Permanent table (default; Time Travel + Fail‑safe)

CREATE OR REPLACE TABLE customer\_data (

id INT,

name STRING,

city STRING,

created\_at TIMESTAMP\_NTZ

);

-- Transient table (no Fail‑safe; lower storage cost for short‑lived data)

CREATE OR REPLACE TRANSIENT TABLE session\_logs (

session\_id STRING,

user\_id INT,

login\_time TIMESTAMP\_NTZ

);

-- Temporary table (session‑scoped; drops at end of session)

CREATE TEMPORARY TABLE temp\_sales (

order\_id INT,

amount NUMBER(10,2),

order\_date DATE

);

**Load some data**

INSERT INTO customer\_data VALUES

(1,'Asha','Mumbai',CURRENT\_TIMESTAMP()),

(2,'Rohit','Delhi',CURRENT\_TIMESTAMP()),

(3,'Neha','Pune',CURRENT\_TIMESTAMP());

INSERT INTO session\_logs VALUES

('s1', 101, CURRENT\_TIMESTAMP()),

('s2', 102, CURRENT\_TIMESTAMP()),

('s3', 101, DATEADD('minute',-5,CURRENT\_TIMESTAMP()));

INSERT INTO temp\_sales VALUES

(1001, 899.00, '2025-08-01'),

(1002, 1299.50, '2025-08-02');

**Checkpoint:**

SELECT \* FROM customer\_data;

SELECT \* FROM session\_logs;

SELECT \* FROM temp\_sales;

**Step 4) Views: Standard & Materialized**

**Standard view**

CREATE OR REPLACE VIEW customer\_city\_view AS

SELECT name, city FROM customer\_data;

SELECT \* FROM customer\_city\_view;

**Materialized view (if supported)**

CREATE OR REPLACE MATERIALIZED VIEW active\_sessions AS

SELECT user\_id, COUNT(\*) AS session\_count

FROM session\_logs

GROUP BY user\_id;

SELECT \* FROM active\_sessions;

**Test refresh behavior**

INSERT INTO session\_logs VALUES ('s4', 101, CURRENT\_TIMESTAMP());

-- Compare results

SELECT \* FROM customer\_city\_view; -- standard view (reads base table live)

SELECT \* FROM active\_sessions; -- MV (may take a moment to refresh)

**Fallback (if MV not available):**

-- Emulate MV with a summary table and manual refresh

CREATE OR REPLACE TABLE active\_sessions\_tbl AS

SELECT user\_id, COUNT(\*) AS session\_count

FROM session\_logs

GROUP BY user\_id;

-- “Refresh” it manually when data changes

CREATE OR REPLACE TABLE active\_sessions\_tbl AS

SELECT user\_id, COUNT(\*) AS session\_count

FROM session\_logs

GROUP BY user\_id;

**Step 5) Stored Procedures (SQL & JavaScript)**

**Note:** If **SQL stored procedures** are not enabled on your account, use the JavaScript version.

**4A. Stored Procedure (SQL language)**

CREATE OR REPLACE PROCEDURE add\_customer(id INT, name STRING, city STRING)

RETURNS STRING

LANGUAGE SQL

AS

$$

INSERT INTO customer\_data (id, name, city, created\_at)

VALUES (:id, :name, :city, CURRENT\_TIMESTAMP());

RETURN 'Customer Added';

$$;

-- Call it

CALL add\_customer(10,'Karan','Bengaluru');

SELECT \* FROM customer\_data WHERE id = 10;

**4B. Stored Procedure (JavaScript language)**

CREATE OR REPLACE PROCEDURE get\_customer\_count()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

var stmt = snowflake.createStatement({sqlText: `SELECT COUNT(\*) AS total FROM customer\_data`});

var rs = stmt.execute();

rs.next();

return 'Total Customers: ' + rs.getColumnValue('TOTAL');

$$;

-- Call it

CALL get\_customer\_count();

**Optional:** Create a JS stored procedure insert\_session(p\_user\_id INT) that inserts a new row into session\_logs with a generated session id.

**Step 5) DDL, DML & Query Constructs**

**DDL: evolve schema**

ALTER TABLE customer\_data ADD COLUMN email STRING;

**DML: insert / update / delete**

INSERT INTO customer\_data (id, name, city, created\_at, email)

VALUES (11,'Riya','Hyderabad',CURRENT\_TIMESTAMP(),'riya@example.com');

UPDATE customer\_data SET city = 'Gurugram' WHERE id = 11;

DELETE FROM customer\_data WHERE id = 11;

**Queries: grouping, ordering, limits**

SELECT city, COUNT(\*) AS num\_customers

FROM customer\_data

GROUP BY city

ORDER BY num\_customers DESC

LIMIT 2;

**Optional:** Write a query to list customers created in the last 10 minutes.

**Step 6) Roles & Access Control (Security Model)**

**Best practice roles:**

* Use SECURITYADMIN to create/manage users & roles.
* Use SYSADMIN to create/manage databases/schemas/tables.

**6A. Create a role and grant object privileges**

USE ROLE SECURITYADMIN;

CREATE ROLE IF NOT EXISTS TRAINEE\_ROLE;

-- Allow the role to see & use our objects

GRANT USAGE ON DATABASE LAB\_DB TO ROLE TRAINEE\_ROLE;

GRANT USAGE ON SCHEMA LAB\_DB.MY\_SCHEMA TO ROLE TRAINEE\_ROLE;

GRANT SELECT ON ALL TABLES IN SCHEMA LAB\_DB.MY\_SCHEMA TO ROLE TRAINEE\_ROLE;

GRANT SELECT ON ALL VIEWS IN SCHEMA LAB\_DB.MY\_SCHEMA TO ROLE TRAINEE\_ROLE;

-- Future grants so new objects are auto‑granted

ALTER SCHEMA LAB\_DB.MY\_SCHEMA

GRANT SELECT ON FUTURE TABLES TO ROLE TRAINEE\_ROLE;

ALTER SCHEMA LAB\_DB.MY\_SCHEMA

GRANT SELECT ON FUTURE VIEWS TO ROLE TRAINEE\_ROLE;

-- Warehouse access so queries can run

GRANT USAGE ON WAREHOUSE COMPUTE\_WH TO ROLE TRAINEE\_ROLE;

**6B. Assign the role to yourself (your user)**

-- Replace YOUR\_USERNAME with your actual Snowflake login name

GRANT ROLE TRAINEE\_ROLE TO USER YOUR\_USERNAME;

**6C. Test with the new role**

USE ROLE TRAINEE\_ROLE;

USE WAREHOUSE COMPUTE\_WH;

USE SCHEMA LAB\_DB.MY\_SCHEMA;

-- Should succeed if grants are correct

SELECT \* FROM customer\_data;

SELECT \* FROM customer\_city\_view;

**Step 7) Sales Analytics**

**Goal:** Build a tiny analytics pipeline end‑to‑end.

USE ROLE SYSADMIN;

USE WAREHOUSE COMPUTE\_WH;

USE SCHEMA LAB\_DB.MY\_SCHEMA;

-- 1) Core tables

CREATE OR REPLACE TABLE products (

product\_id INT,

name STRING,

category STRING,

price NUMBER(10,2)

);

CREATE OR REPLACE TABLE sales\_orders (

order\_id INT,

product\_id INT,

qty INT,

order\_date DATE

);

-- 2) Insert data

INSERT INTO products VALUES

(1,'Headphones','Audio',1999),

(2,'Keyboard','Peripherals',2499),

(3,'Webcam','Video',3299);

INSERT INTO sales\_orders VALUES

(10001,1,2,'2025-08-01'),

(10002,2,1,'2025-08-02'),

(10003,3,3,'2025-08-03');

-- 3) Monthly summary

CREATE OR REPLACE VIEW v\_orders\_enriched AS

SELECT so.order\_id, so.product\_id, so.qty, so.order\_date, p.name, p.category, p.price,

(so.qty \* p.price) AS revenue

FROM sales\_orders so

JOIN products p USING (product\_id);

-- Try MV first

CREATE OR REPLACE MATERIALIZED VIEW monthly\_sales\_summary AS

SELECT TO\_CHAR(DATE\_TRUNC('month', order\_date), 'YYYY-MM') AS month,

SUM(qty \* price) AS revenue

FROM v\_orders\_enriched

GROUP BY 1;

-- If MV fails, use a table and refresh manually

-- CREATE OR REPLACE TABLE monthly\_sales\_summary AS

-- SELECT TO\_CHAR(DATE\_TRUNC('month', order\_date), 'YYYY-MM') AS month,

-- SUM(qty \* price) AS revenue

-- FROM v\_orders\_enriched

-- GROUP BY 1;

-- 4) Procedure to insert a sale (JS) and show new totals

CREATE OR REPLACE PROCEDURE insert\_sale(p\_order\_id INT, p\_product\_id INT, p\_qty INT, p\_date DATE)

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute({sqlText: `INSERT INTO sales\_orders VALUES (:1,:2,:3,:4)`, binds:[P\_ORDER\_ID,P\_PRODUCT\_ID,P\_QTY,P\_DATE]});

try {

// MV will refresh automatically; if you used a table, “refresh” it manually:

// snowflake.execute({sqlText: `CREATE OR REPLACE TABLE monthly\_sales\_summary AS

// SELECT TO\_CHAR(DATE\_TRUNC('month', order\_date), 'YYYY-MM') AS month,

// SUM(qty \* price) AS revenue

// FROM v\_orders\_enriched GROUP BY 1`});

} catch (e) {}

return 'Sale inserted';

$$;

-- 5) Test

CALL insert\_sale(10004, 1, 5, '2025-08-10');

SELECT \* FROM monthly\_sales\_summary ORDER BY month;