Top 10 Ultimate Snowflake Challenges

by

Cristian Scutaru

Table of Contents

Introduction

Welcome

Welcome to **Top 10 Ultimate Snowflake Challenges**, an exciting collection of hidden hands-on gems for Snowflake, presented in an original manner. The live interactive version of this book has been implemented as an **Udemy video course** as well.

My name is Cristian Scutaru and I am a world-class expert in Snowflake, former *Snowflake* "*Data Superhero*" and *SnowPro SME (Subject Matter Expert)*. I advised, designed and implemented end-to-end systems for dozens of Snowflake customers. And I lately started to share my knowledge, as a successful Udemy instructor. So you are in good hands...

I'll challenge you here with questions in 10 different Snowflake areas that I found being exciting over the years. I will provide hands-on solutions in either SQL or Python.

All the code used in this book is open-sourced in my public **snowflake-challenges** GitHub repository. A new database will be created for each new use case in a *setup.sql* file. You usually copy this code into a new SQL Worksheet, in a *free Snowflake trial* account that you can get within seconds at **signup.snowflake.com**. No credit card or any commitment is required. Select also the Enterprise Edition, in an AWS region, if possible. As most recent Snowflake features are frequently available only for AWS.

Each challenge will start with a **Question**, after which I may give you some additional explanation and hints. When ready, read my own **Answer**, starting with the next page. Write back to me if you find a better solution (I'm always glad for feedback). A final **Conclusion** chapter will close the challenge.

What you will learn more about

- Some obscure but very interesting things in Snowflake.
- Solve tricky issues with Snowflake queries and hierarchical data.
- Fix intermediate to advanced SQL queries.
- Solve funny and challenging puzzles using Snowflake.
- Learn more about the Snowflake ecosystem in a funny engaging way.
- See the questions and try to solve them on your own.
- Watch my hands-on experiments and follow my hints.
- Follow extra-curriculum recommended material.
- Learn intermediate to advanced SQL and Python programming in Snowflake.
- Learn something about Streamlit and how to use it in Snowflake.
- Advanced tips and tricks.

In a challenging but funny and engaging way, you'll learn about time travel in Snowflake, pivot queries, Streamlit apps, Python worksheets, data clean rooms, access policies, and so on...

What Snowflake areas will be considered

- Time Travel
- Auditing and Query History
- Generating Synthetic but Realistic Data
- Data Classification

- Snowpark Stored Procedures from Python Code
- Streamlit Apps
- Data Clean Rooms
- Row Access Policies
- SQL Queries in Snowflake
- Querying Metadata
- User-Managed and Serverless Tasks
- Details on Snowflake's Virtual Warehouses
- Charts and Graphs
- Recursive SQL Queries on Hierarchical Data
- Semi-Structured Data
- Cost Management and Cost Estimates
- Multiple-Table Inserts
- Extended Pivot Queries
- Multi-Tenant Architectures
- SnowSQL Variable Substitution
- Object Dependencies
- Python Worksheets

My main objectives are

- To help you become better in Snowflake and SQL;
- To help you discover new interesting things in Snowflake;
- To have fun through a set of well-structured and organized puzzles;
- To introduce you to some Snowflake features in an engaging and entertaining way.

I created this book for

- Data Engineers and Software Developers willing to learn more about Snowflake and SQL;
- Hands-on Data Architects willing to find out about hidden gems, or just solve interesting funny challenges;
- Any other technical person willing to dive deeper into hidden corners of SQL and the Snowflake Data Cloud.

Quick Review of all Challenges

Without giving up the answers, let us walk through all the challenges I included in this selection.

CHALLENGE #1

Using one single SQL statement, copy all '*apple*' entries from the *Fruits* table into the *Apples* table, and all '*orange*' entries into the *Oranges* table:

```
create table Fruits (name string)
   as select * from values ('apple'), ('orange'), ('apple');
create table Apples (name string);
create table Oranges (name string);
```

In this first challenge I will dare you to *simultaneously* insert new entries into more than one Snowflake table, to do this using one single SQL statement.

CHALLENGE #2

Add the *gender* column - with the *male/female* **values - to the pivot query below.**

The new column headers should be the *country*, "'*married male***'", "'***married female***'", "'***single male***'", and "'***single female***'".**

```
select country, "'married'", "'single'"
from (select country, status, orders from Customers)
pivot (sum(orders) for status in ('married', 'single'))
order by country;
```

In this next challenge I will ask you to basically add a second column that you can pivot, in the same query. The current PIVOT clause, in Snowflake, does not allow you do that.

CHALLENGE #3

Jack and Mary separately save in a table only one has access to the food they like most: Jack "*I like grapes*", and Mary "*I like cookies*".

Mary needs to know if they both like the same food, but without telling each other what they like.

How can you do this in Snowflake?

This third challenge here is a funny one and - I will not hide it from you - that's also a hard one.

Chances are you may be asked to reinvent something that was very hard to invent in the first place. But you may learn something very important for your career here, with proven practicality.

CHALLENGE #4

What is the easiest way to add, delete or modify some table data in Snowsight, without typing any SQL?

Hint: **But you may type ...Python! ;)**

In this challenge I will ask you to basically extend the Snowflake web UI with some small code. To add some small feature, to edit table data in place. Using eventually some (small) Python code, but no SQL.

CHALLENGE #5

In Snowsight, display a graph with the tables used by every view from your Snowflake account.

The next challenge will also ask you to extend Snowflake with metadata rendered in a graph format.

Try to find first how you can get the view-tables dependencies. Then what's one easy way to render them as a graph.

CHALLENGE #6

Create a *Customers* table with just *name* **and** *country* **text columns. Enter data for 10 minutes or so, using INSERT SQL statements.**

Display now the number of entries added every minute.

This one is also interesting, because I will ask you to enter data into a new Customers table with no explicit timestamps or other datetime information possible.

However, later on, I want you to give me the number of entries added every minute.

CHALLENGE #7

From an *Employees* table with *employee* **and** *manager* **columns, show each employee name with an indent, right below the immediate manager.**

Remark: **The president of the company has no manager.**

This is all about hierarchical data: you have employees, and each employee always has one manager and only one manager. Only the president of the company has no manager at all.

How can you show this in a hierarchical manner, with each employee name prefixed by a specific indent, right below their immediate manager? To be able for anyone to easily identify who is reporting to whom, directly or not.

CHALLENGE #8

Generate one million rows of fake but realistic *Customers* data - with *name*, *address*, *city*, *state* **and** *email*. **Then label each column based on the type of info it contains.**

What would be the best way to do this in Snowflake, for 10-100x more data?

This is about generating fake but realistic data. And yeah, the question also relates to scalability.

How can you do this with one million rows? Or later with 10 million or 100 million rows?

CHALLENGE #9

Two tasks will execute quickly for just one second or so. The first one will be scheduled every minute, while the second one every two minutes.

What is the best implementation in Snowflake for each of them?

The next challenge is all about tasks and virtual warehouses. And definitely, when you have tasks waking up too frequently, they may consume a lot of compute credits.

I ask you to evaluate what's happening with the warehouse behind (for user-managed tasks). And what's happening if you use serverless tasks instead.

CHALLENGE #10

You'd need to run the same SQL script multiple times, with the object names modified for each business partner and the environments they create. The names will be prefixed with partner's name, and end with a *dev/test/prod* **environment suffix.**

What is the most efficient way to do this in Snowflake?

Finally, this last challenge is about configuring an SQL script, to be more generic.

There is a common requirement in real life to split-up some Snowflake account between departments, business partners or even customers.

However, it's costly and unrealistic to create and maintain one specialized script for each of them, when they will all get very similar resources.

I hope you'll love all this. Buckle up and let's go!

Challenge #1: Multi-Table Insert

Question

Using one single SQL statement, copy all '*apple*' entries from the *Fruits* table into the *Apples* table, and all '*orange*' entries into the *Oranges* table:

```
create table Fruits (name string)
   as select * from values ('apple'), ('orange'), ('apple');
create table Apples (name string);
create table Oranges (name string);
```

So we create a Fruits table using *CTAS (Create Table As Select)*, and we populate right away this table with the values 'apple', 'orange', and 'apple'.

We'll have to copy these three values into the new tables Apples and Oranges. At the end, Apples will have two entries ('apple' and 'apple') and Oranges will have one entry ('orange').

Don't scroll further, go experiment for a few minutes and come back here when you're ready. I will share with you next my solution. And I'll be glad for you to share with me your own solution, if different.

Answer

In your test Snowflake account, create a new SQL Worksheet, paste the following code (from the *setup.sql* file), and execute it all together:

```
create or replace database challenge_insert;


create table Fruits (name string)
   as select * from values ('apple'), ('orange'), ('apple');


create table Apples (name string);
create table Oranges (name string);
```

You may say now that it's super easy, why we don't do this? To use two simple INSERT statements: one to copy data to Apples, the other one for Oranges. *Do not run this, as the devil is in detail!*

```
insert into Apples
select name from Fruits
where name = 'apple';


insert into Oranges
select name from Fruits
where name = 'orange';
```

I'm pretty sure you saw this: I asked you to do this in *one single SQL statement*. We have two SQL statements here, so not so fast 😉

...Did you know about this? **INSERT for multi-table**, in Snowflake, yeah? From the Snowflake documentation. Go there and you'll see that it is possible, yes, to use one single INSERT to add data into multiple tables at once.

The way we can do it is: with name from Fruits, call an **INSERT FIRST** when name is 'apple' then into Apples, and when name is 'orange' then into Oranges. As you can see here: name (one single column) is propagated into different tables. Just run this.

```
INSERT FIRST
   WHEN name='apple' THEN INTO Apples
   WHEN name='orange' THEN INTO Oranges
select name from Fruits;

select * from Apples;
select * from Oranges;
```

So two rows inserted in Apples and one row inserted in Oranges. Worked fine!

And here you have some other variation, with **INSERT ALL**. Let's just create two other tables first, Fruits5 and the FruitsElse. You'll see why.

```
create table Fruits5 (name string);
create table FruitsElse (name string);
```

We'll then use the OVERWRITE clause to truncate the destination tables first, so everything in these target tables will be deleted.

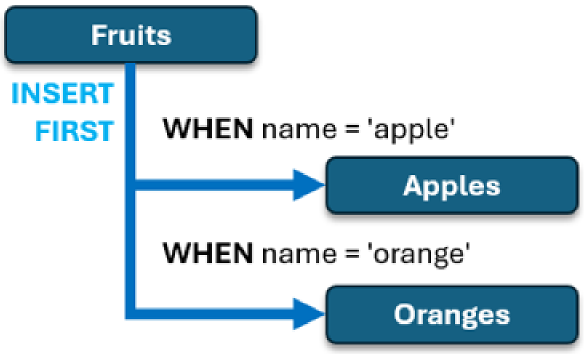
'Apple', as a string value, has a length of 5. So, this will be inserted as well in Fruits5. If no condition here is true, you have an ELSE clause that will insert anything remaining in the Fruits cells.

```
INSERT OVERWRITE ALL
   WHEN name='apple' THEN INTO Apples
   WHEN name='orange' THEN INTO Oranges

   WHEN len(name)=5 THEN INTO Fruits5

   ELSE INTO FruitsElse

select name from Fruits;


select * from Apples;

select * from Oranges;

select * from Fruits5;

select * from FruitsElse;
```

Conclusion

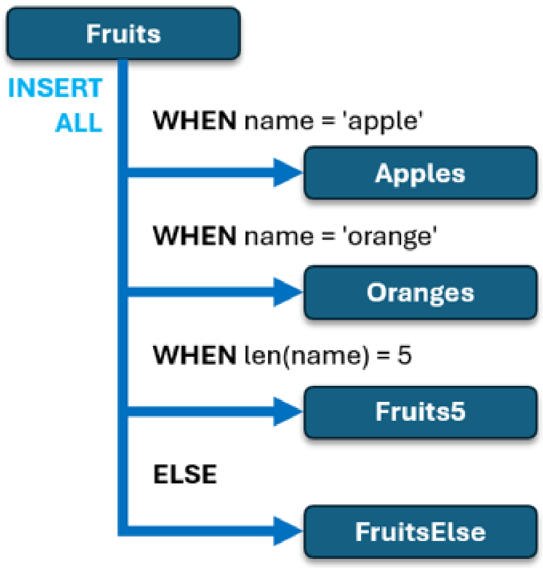
To review my solution here - and let me know if you find something better - I had this restriction: using one single SQL statement. And for this I used the multi-table INSERT implementation of Snowflake.

In the first phase, I just provided the solution to this specific problem. So, with an INSERT FIRST, when the name is 'apple', the entry will be inserted into the ***Apples*** table, and the other conditions will not be checked. When the name is 'orange', the entry will be copied into the ***Oranges*** table.



As we have seen before, I just extended this with an INSERT ALL. With the ALL keyword, *all* these conditions will be evaluated. You can copy the same entry into more than one table at once.

As you can see below, the 'apple' entry will be added in both the ***Apples*** table and the ***Fruits5*** table. At the end, an ELSE clause will collect data when no other condition previously evaluated was found true.


Fruits
INSERT ALL
WHEN name = 'apple'
Apples
WHEN name = 'orange'
Oranges
WHEN len(name) = 5
Fruits5
ELSE
FruitsElse