Administering Snowflake exercises

List of all assignments for Administering Snowflake together at one place. Please find all DDL and all other needed queries for the exercises in following file on GitHub: `/exercises.sql`

01 Authentication

First exercise related to authentication is using key pair instead of user and password. We have to generate private & public key and then assign it to our user.

1. Let's check if our user does not have assigned some key by performing `desc user <my_username>`

We will get a list of user's parameters where one is called RSA_PUBLIC_KEY.

2. We can generate the encrypted private key with following command:

```
openssl genrsa 2048 | openssl pkcs8 -topk8 -v2 des3 -inform PEM -out
rsa_key.p8
```

We need to provide our passphrase.

3. Once we have a private key, we can generate the public key:

```
openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
```

4. Now we have to assign the public key to our user - remove the public key delimiters

```
 Alter user <username> set rsa_public_key = 'key'
```

Can you guess how would you perform a key rotation?

Second exercise is about finding users who don't use MFA. We have to options. First is checking a Snowflake metadata. You can check `ACCOUNT_USAGE. LOGIN_HISTORY` view where is a column `SECOND_AUTHENTICATION_FACTOR` saying if any second authentication method is used. All users with NULL in this column don't use MFA.

Second option is then using a new feature called Trust Center. You can find it under monitoring. Trust center performs series of industry standard checks to validate your account security. One of the performed check is also verification if users use MFA.

MFA enforcement
Let's have a look on a new feature for MFA enforcement. You can create a authentication policy to enforce users for local users or SSO as well.

You can create an authentication policy for whole account or per user

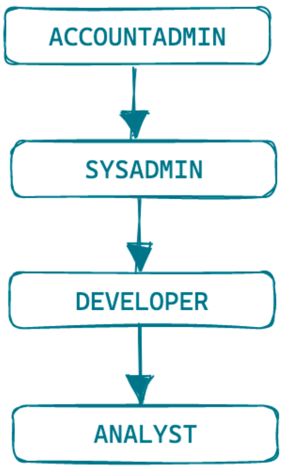
02 Authorization

This exercise is dedicated to authorization tasks, but first need to have some DB, schema and tables to work with. Let's use Snowflake sample data (Tasty Bites) for it. You should find a worksheet to create DB, schema, table, stage and load some data in Getting Started Tutorials folder and Load sample data with SQL from S3 file. I have included those queries also in GitHub repo. Please create all those objects now.

Now We can proceed and create a bunch of custom roles to practice the RBAC model. We will try to follow the best practices, meaning that:

- SECURITYADMIN will be owner of those roles

- SYSADMIN should have access to those roles



Create a role called `ANALYST` with following privileges:
- Read access to table `MENU` (SELECT privileges)
- Ability to use warehouse `COMPUTE_WH`

Create a role called `DEVELOPER` with following privileges:
- Access to `ANALYST` Role
- Full access to DB `TASTY_BITES_SAMPLE_DATA`

Use role `ANALYST` and try to create a new table in RAW_POS schema

Use role `DEVELOPER` and try to create a new table in RAW_POS schema

Note: If you need to grant access to tables you have to grant USAGE privilege DB level.

03 Database roles

This exercise is about database roles and how to use them for data sharing. Using database roles for data sharing has benefit in possibility to grant different database roles to different account roles in consumer account which means you have granular access to shared data. The other approach is about all or nothing.

As a prerequisite for this exercise you need two Snowflake accounts in same Snowflake region. Otherwise you won't be able to test the data sharing. We will be working in both accounts here.

Let's start in the account which will be data provider:

We will create a new database called SHARING_TEST where we are going to put some DB objects.

Let's create a table called `landing_table` with following structure:

```
id number,
value varchar,
created timestamp
```

Then let's create a `landing_view`

```
create or replace secure view landing_view as
select
  id,
  value
from landing_table;
```

And finally a `customer` table with these columns:

```
 id number,
 name varchar,
 surname varchar,
 phone varchar
```

Now we can proceed and create two database roles where each one will have access to different objects. Let's create DB roles `share_provider1` and `share_provider2`.

Now we need to grant access to particular objects, so `share_provider1`:

- Usage on DB and schema
- Select on customer table

And `share_provider2`:

- Usage on DB and schema
- Select on landing_table and landing_view

As a next step we need to create a share called `s_sample_share` and grant our database roles to this share.

And last step for data provider is adding the consumer account into the share. You can do it with `alter share` command.

Data consumer part

Now we can switch to data consumer account where we need to create a DB from the share. We can do it directly in Snowsight: Data Products -> Private Sharing and there you can find direct Shares. Let's create a db from share we just created as data providers.

Then let's proceed and create developer and analyst account roles. Later we will grant the database roles from share to these account roles to get granular access for shared data.

Grant those roles to your user so you can test it.

And then you can grant our database roles to these account roles:

```
grant database role sample_share.share_provider1 to role developer;
```

```
grant database role sample_share.share_provider2 to role analyst;
```

Where `sample_share` is a DB name I created from the share.

Then we can validate whole setup.

Let's activate `developer` role - you should see only customer table. Then try `analyst` role and you should see `landing_table` and `landing_view`.

04 Privileges

This exercise is about understanding of usage SHOW GRANTS command as there are different use case where it could be used.

You can try following:

1. List all users and roles who has granted roles ANALYST and DEVELOPER
2. List all privileges granted no role DEVELOPER
3. List all privileges granted on schema RAW_POS
4. List all privileges granted no table MENU
5. List all privileges granted to role DEVELOPER
6. List all privileges granted to your user
7. Check what are future grants in in schema RAW_POS
8. Check all privileges on schema RAW_POS and filter the result only for ANALYST role
9. Grant INSERT on MENU table to role ANALYST and then REVOKE it

Let's do it in SQL but of course there is also way how to manage grants in UI.

05 Auditing user activity

During this exercise we will try to use Snowflake metadata to find out who and how is using data in our Snowflake account. We will be mainly working with ACCESS_HISTORY view in ACCOUNT_USAGE schema. Let's try to get an complete overview about data access starting from importing data from stage as we have our external stage for loading data in called TASTY_BYTES_SAMPLE_DATA.PUBLIC.BLOB_STAGE.

First task is about finding all data accessed related with stages. This can be retrieved from direct object access column.

Then we can add info about tables which are being used as targets for this load. Such info could be retrieved from objects modified column.

And in the end we can add info about particular columns which have been loaded. This can be again retrieved from objects_modified where we will find a key called **columns.**

We need to flatten semi structured data in this exercise quite heavily as all the needed columns are stored as JSONs. You will need to use lateral flatten () many times.

Once we know how to retrieved these information for our ingestion pipeline we can try to check same data for a view.

We can try to retrieve many other useful informations. Please try to write also queries covering following use cases:

- Find all tables accessed by your user
- Find all users who have modified table 'TASTY_BYTES_SAMPLE_DATA.RAW_POS.MENU'
- Find the most accessed columns in table 'TASTY_BYTES_SAMPLE_DATA.RAW_POS.MENU'

06 Resource Monitors

This exercise is dedicated to practice creation and administration of resource monitors. Let's ty to create a resource monitor with following params:

- Credit quota = 50
- Monthly frequency
- It will start immediately
- It will notify at 75%
- It will suspend at 98%
- It will suspend immediately at 100%

Then assign it to our COMPUTE_WH.

As a next exercise let's create another resource monitor which will be serving as resource monitor for whole account. It will have 1000 credits as quota and suspend on 100%. Assign the monitor to whole account.

We will also try to enable notifications to non admin users. For that case let's create a user called JOE. If you want to finish the enablement you also need to assign him an email address and verify it.

Grant a developer role to JOE and then you can try to enable resource notifications for him.

07 WH configuration

This exercise is focused on effective configuration of virtual warehouses. What parameters should be set and how. We are going to create a single virtual warehouse with following specs:

Create your first virtual warehouse called `COMPUTE_WH`, with following parameters:

- Extra small size
- Enabled auto resume
- Enabled auto suspend
- Suspend after 1 minute
- It will be suspended after creation

Create a multi cluster warehouse called `MULTI_COMPUTE_WH` with following parameters:

- Small size
- Enabled auto resume
- Enable auto suspend
- Suspend after 1 minutes
- It will be suspended after creation
- Min number of clusters: 1
- Max number of clusters: 3
- Economy scaling policy

Next we are going to modify the `STATEMENT_TIMEOUT_IN_SECONDS` parameter. The default value (2 days) is too high and can lead to unintentional cost spikes. First we need to find out what is currently configured:

- On warehouse level
- On session level
- On account level

We will modify the value only on the warehouse level. This is the lowest level and Snowflake will always use this value also in case it would be configured on session and account level. Modify the `STATEMENT_TIMEOUT_IN_SECONDS` on `COMPUTE_WH` to 1 hour (3600 seconds).

As a last step in effective warehouse configuration we are going to create a resource monitor which can suspend our warehouse automatically in case of reaching the defined limit. Once we have the resource monitor in place, we will assign it to our `COMPUTE_WH`.

Create resource monitor `compute_quota` with following parameters:

- Credit quota = 50 credits
- Monthly frequency
- Starts immediately
- It will notify when reaching 75% of the quota
- It will suspend the warehouse when reaching 98% of the quota
- It will suspend the warehouse immediately when reaching 100% of the quota

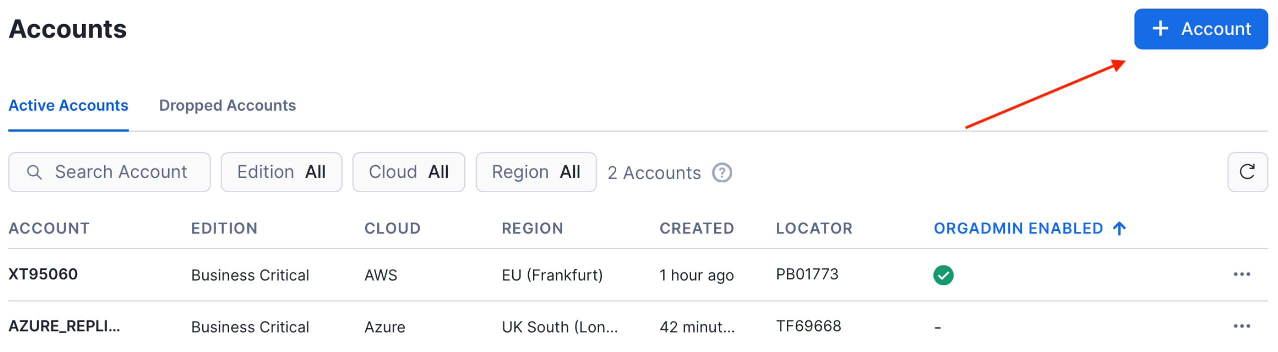
Assign this resource monitor to our `COMPUTE_WH`.

08 Database replication & failover

This exercise will be dedicated to practicing database replication. It has some prerequisites:

- we need two accounts tight to organization
- We need some data to replicate

If you have your trial account, the orgadmin role should be enabled for you. You can go into Admin -> Accounts where you can add second account where you will replicate the data.



ACCOUNT	EDITION	CLOUD	REGION	CREATED	LOCATOR	ORGADMIN ENABLED
XT95060	Business Critical	AWS	EU (Frankfurt)	1 hour ago	PB01773	✓
AZURE_REPLI...	Business Critical	Azure	UK South (Lon...	42 minut...	TF69668	-

Next we need some data to replicate. Let's reuse Snowflake sample data for Tasty Bites which we already have available.

We will create also one role called TESTER to verify that account level objects like roles are also replicated. TESTER role should be granted to SYSADMIN role.

Then we can proceed to create failover group. We want to replicate following objects:

- Roles
- Warehouses
- Databases

As ALLOWED_DATABASE we will use our TASTY_BYTES_SAMPLE_DATA
As ALLOWED_ACCOUNTS then our secondary account
As REPLICATION_SCHEDULE then try 10 mins

Once we do this on primary account, we have to create replica of this failover group also on the secondary account.

Replica creation should start the replication and we need to monitor it. It's possible to do it directly in Snowsight. Go to Admin -> Accounts -> Replication

You probably can't see the replication tab under Accounts. We have to grant monitor privileges for our replication group to accountadmin role or any other role you have used for creation of failover group.

```
grant monitor on failover group my_failover_group to role accountadmin;
```

Then you can check how the replication is running. It will be probably failing because we try to replicate also account objects which are already created in target account (system roles). In order to make it work we have to assign to those objects created outside of the replication the unique global identifiers. You can use for that following command:

```
SELECT SYSTEM$LINK_ACCOUNT_OBJECTS_BY_NAME('my_failover_group');
```

Then you can try to manually refresh the failover group in target account.

Please find all the queries used in this exercise on **GitHub** in `/exersice.sql` file.