Powdr 101 – Getting Started with Snowflake

Hello/Bonjour/Hola,

We've put together a simple walk through on how to get up and running with Snowflake in a few easy-to-follow steps. During this demonstration we will be creating a Snowflake environment within the Microsoft Azure environment. We will not only be creating the account and a couple of roles, and maybe a dashboard to monitor our usage.

For us to get started you will need to have a Snowflake account and access to the ACCOUNTADMIN, SECURITYADMIN, and SYSADMIN roles. If you need to create an account, you can use the link here: https://trial.snowflake.com/?owner=SPN-PID-169286

Alright so the demo we're doing today will start with the creation of the environment. When creating a database environment, think of it as building a house. We don't rush in trying to install the shingles without laying the foundation. We will start with the foundation of the Snowflake environment focusing initially on Databases, Schemas, Security, and Monitoring. At the end of this article, we will have a basic environment setup completed and a better understanding of a few advanced features.

What we will be creating:

A Snowflake environment that has the following databases:

- DEV (Development)
- TST (Test)
- PRD (Production)
- SNOWMONITOR (Organizational Usage Meta Data)

The following schemas will be created within the databases:

- DEV/TST/PRD Databases
    - STG (Stage)
    - INT (Integration)
    - DIM (Dimension)
- SNOWMONITOR
    - USAGE (Org Usage Meta Data)

We will create the following roles:

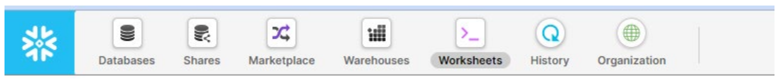
- Development access
    - DEVOPS
        - Create, Select, Update, Delete, Drop, Alter
    - DEVREAD
        - Select
- TST (Test)
    - TSTOPS

        - Create, Select, Update, Delete, Drop, Alter
    - TSTREAD
        - Select
- PRD (Production)
    - PRDOPS
        - Create, Select, Update, Delete, Drop, Alter
    - PRDREAD
        - Select
- REPORTING
    - Select access on all objects in all databases through inherited roles
- MASKINGADMIN
    - Role to create and apply masking policies
        - Not covered in this how-to, here's Snowflakes Support section for dynamic masking policies: https://docs.snowflake.com/en/sql-reference/sql/create-masking-policy.html
- PRIVATE
    - Role to allow for access to the masked columns via a masking policy

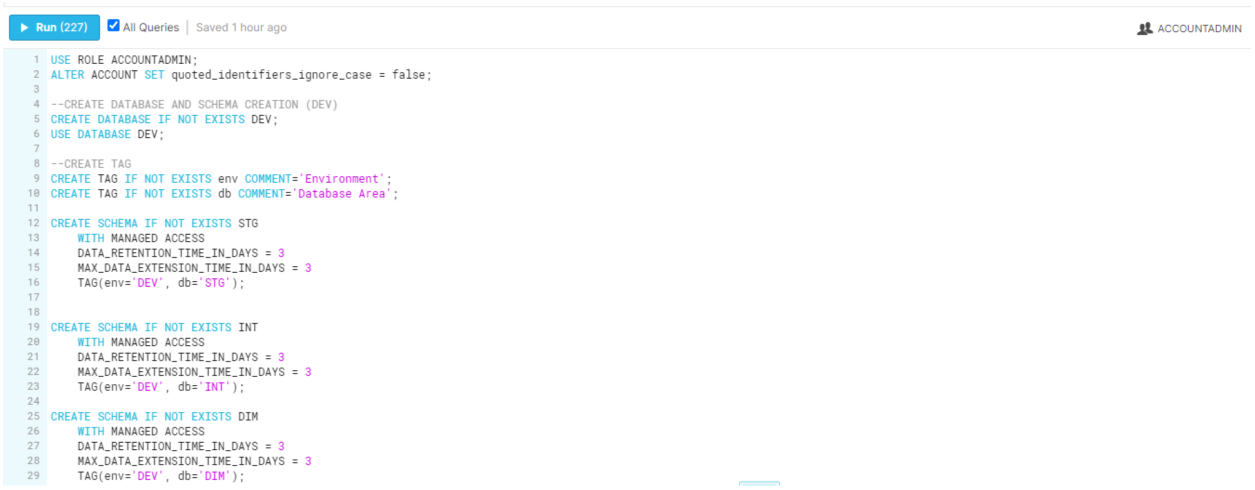
Let's get started:

First, we will log into our Snowflake account, the account URL will look something like this for an Azure based instance: <https://abcdef-1234567.snowflakecomputing.com/>

Once logged in, we navigate to the Worksheets tab on the user interface:



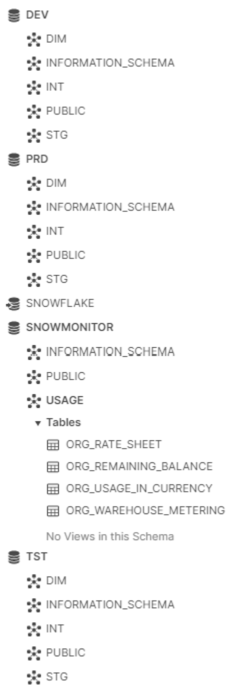
Next we will copy the contents from the Powdr101.sql file the contains the SQL scripts to create the databases, schema, roles, and SNOWMONITOR objects. Paste all line of the script in the worksheet, check the box All Queries, and click Run(236):



```
USE ROLE ACCOUNTADMIN;
ALTER ACCOUNT SET quoted_identifiers_ignore_case = false;

--CREATE DATABASE AND SCHEMA CREATION (DEV)
CREATE DATABASE IF NOT EXISTS DEV;
USE DATABASE DEV;

--CREATE TAG
CREATE TAG IF NOT EXISTS env COMMENT='Environment';
CREATE TAG IF NOT EXISTS db COMMENT='Database Area';

CREATE SCHEMA IF NOT EXISTS STG
    WITH MANAGED ACCESS
    DATA_RETENTION_TIME_IN_DAYS = 3
    MAX_DATA_EXTENSION_TIME_IN_DAYS = 3
    TAG(env='DEV', db='STG');


CREATE SCHEMA IF NOT EXISTS INT
    WITH MANAGED ACCESS
    DATA_RETENTION_TIME_IN_DAYS = 3
    MAX_DATA_EXTENSION_TIME_IN_DAYS = 3
    TAG(env='DEV', db='INT');

CREATE SCHEMA IF NOT EXISTS DIM
    WITH MANAGED ACCESS
    DATA_RETENTION_TIME_IN_DAYS = 3
    MAX_DATA_EXTENSION_TIME_IN_DAYS = 3
    TAG(env='DEV', db='DIM');
```

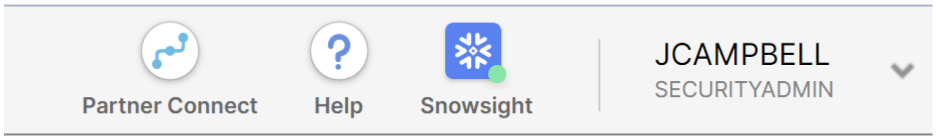
Please note that if your user profile doesn't have access to the ACCOUNTADMIN, SECURITYADMIN, and SYSADMIN roles this script will fail to run. If you're lacking these roles, request that they be granted or ask your ACCOUNTADMIN to complete the steps of this article.

After the statements are successfully executed you will now see four new databases with schemas have been created.

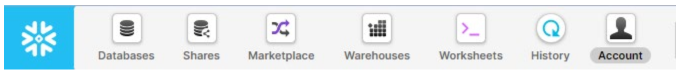


```
DEV
  DIM
  INFORMATION_SCHEMA
  INT
  PUBLIC
  STG
PRD
  DIM
  INFORMATION_SCHEMA
  INT
  PUBLIC
  STG
SNOWFLAKE
SNOWMONITOR
  INFORMATION_SCHEMA
  PUBLIC
  USAGE
    Tables
      ORG_RATE_SHEET
      ORG_REMAINING_BALANCE
      ORG_USAGE_IN_CURRENCY
      ORG_WAREHOUSE_METERING
    No Views in this Schema
TST
  DIM
  INFORMATION_SCHEMA
  INT
  PUBLIC
  STG
```

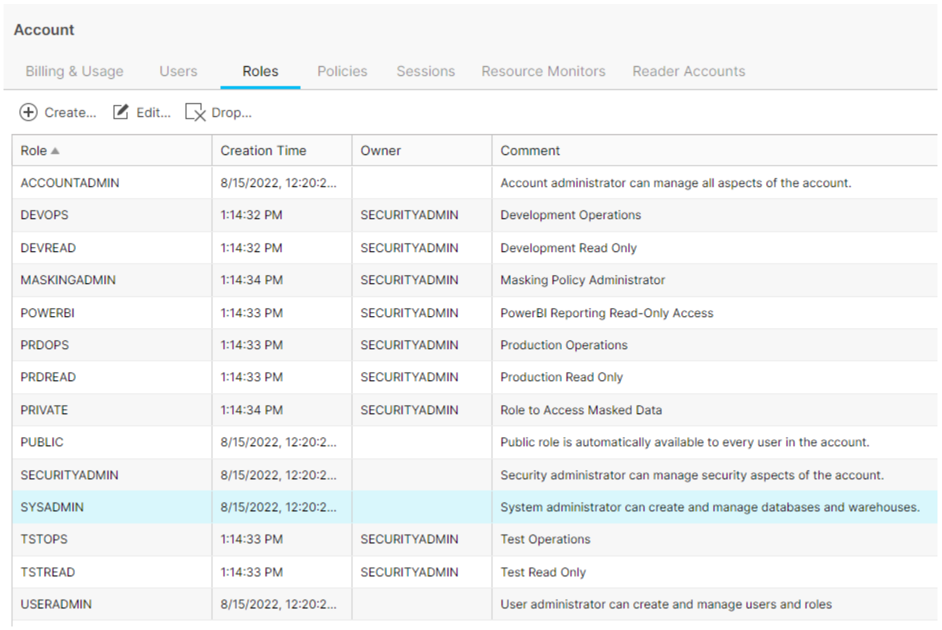
Now let's view the roles that we have created, we will do this by switching over to the SECURITYADMIN role on the top right-hand corner of the UI:



Once we have set our role as SECURITYADMIN we will now be able to see the Account tab on the UI:



We will select the Account tab, and then select the Roles option on the Account page. This will list the roles that currently exist for our Snowflake account.



Role ▲	Creation Time	Owner	Comment
ACCOUNTADMIN	8/15/2022, 12:20:2...		Account administrator can manage all aspects of the account.
DEVOPS	1:14:32 PM	SECURITYADMIN	Development Operations
DEVREAD	1:14:32 PM	SECURITYADMIN	Development Read Only
MASKINGADMIN	1:14:34 PM	SECURITYADMIN	Masking Policy Administrator
POWERBI	1:14:33 PM	SECURITYADMIN	PowerBI Reporting Read-Only Access
PRDOPS	1:14:33 PM	SECURITYADMIN	Production Operations
PRDREAD	1:14:33 PM	SECURITYADMIN	Production Read Only
PRIVATE	1:14:34 PM	SECURITYADMIN	Role to Access Masked Data
PUBLIC	8/15/2022, 12:20:2...		Public role is automatically available to every user in the account.
SECURITYADMIN	8/15/2022, 12:20:2...		Security administrator can manage security aspects of the account.
SYSADMIN	8/15/2022, 12:20:2...		System administrator can create and manage databases and warehouses.
TSTOPS	1:14:33 PM	SECURITYADMIN	Test Operations
TSTREAD	1:14:33 PM	SECURITYADMIN	Test Read Only
USERADMIN	8/15/2022, 12:20:2...		User administrator can create and manage users and roles

Access within Snowflake is enforced through roles that are assigned to a given user or another role. Roles can inherit the rights granted to other roles allow for team-based access control. In the current security model that we just created, we have a distinct separation from DEV, TST, and PRD. This will allow for us to grant users access to a specific database (environment) or create team roles that would be able to inherit access through assignment of the roles created during this exercise.

The security aspects of Snowflake should be considered before creating objects, addressing your security strategy later on can lead to headaches down the road. More information regarding Snowflakes access policies can be located here: https://docs.snowflake.com/en/user-guide/security-access-control-overview.html

Monitoring Snowflake Usage

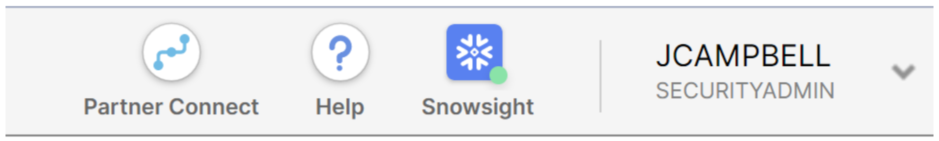
One challenge that early adopters of Snowflake face is how am I sure I'm not over or under utilizing this platform. To address this concern up front, we have created SNOWMONITOR. SNOWMONITOR is a database that is used to store the historical usage information for a rolling 12 months. This can be adjusted as needed if you have the use case to preserve the historical summary usage information. During the build step, we executed the script to build the SNOWMONITOR database and the objects needed to do basic reporting. These objects (tables) can be found in the SNOWMONITOR database under the USAGE schema.

These objects are:

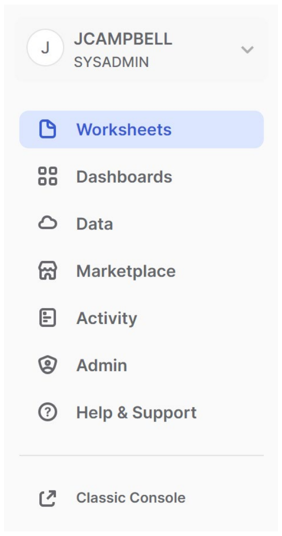
- ORG_RATE_SHEET
  - Daily snapshot of the rates associated with Snowflake services for our organization.
- ORG_REMAINING_BALANCE
  - Remaining capacity agreement balance, assuming you have a capacity agreement.
- ORG_USAGE_IN_CURRENCY
  - The daily usage of services in the currency assigned to the account/contract
- ORG_WAREHOUSE_METERING
  - Breakdown of credit usage by warehouse

These objects are refreshed daily and will contain usage for the prior day. If the account is new, we may not have any usage information within these tables. Once the Organization starts to use Snowflake these summary objects will contain daily information. With these four objects we can see the usage at a high level by querying the tables, but let's create some visuals to provide quick value to those responsible for managing the Snowflake cost.

Snowflake has introduced Dashboards through Snowsight, this can be accessed by clicking the Snowsight icon on the Snowflake UI:



When we click on the icon, we will be prompted to log into application using the same credentials we use to access the Snowflake UI (Classic). Once we log into Snowsight we will the new user interface with tabs now running down the left-hand side of the screen:



Here we will click on the Dashboards option to create our Snowflake Usage Reporting using Snowsight. Snowsight works by using tiles, within these tiles we use a SQL statement to build the visualizations that cover the following metrics:

- Contract Remaining Balance
  - Purchased + Free
- Current Month Usage
- Prior Month Usage
- Daily Cost by Service
- Forecasted Credit Exhaustion
  - 15 Day Average Usage
  - 30 Day Average Usage
  - 60 Day Average Usage
  - 90 Day Average Usage

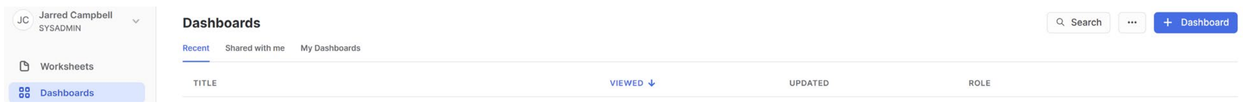
The following queries will be used within the tiles to create the dashboard:

(Please note that without any activity within the account these queries will be empty)

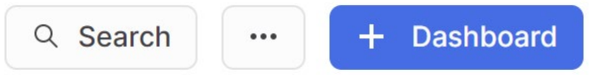
```
-- Current Month Cost

Select SUM(USAGE_IN_CURRENCY) Monthly_Cost

from SNOWMONITOR.USAGE.ORG_USAGE_IN_CURRENCY

WHERE MONTH(CURRENT_DATE) = MONTH(USAGE_DATE)

AND YEAR(CURRENT_DATE) = YEAR(USAGE_DATE);
```

```
-- Prior Month Cost
Select SUM(USAGE_IN_CURRENCY) Monthly_Cost
from SNOWMONITOR.USAGE.ORG_USAGE_IN_CURRENCY
WHERE MONTH(ADD_MONTHS(CURRENT_DATE,-1)) = MONTH(USAGE_DATE)
AND YEAR(ADD_MONTHS(CURRENT_DATE,-1)) = YEAR(USAGE_DATE);
```

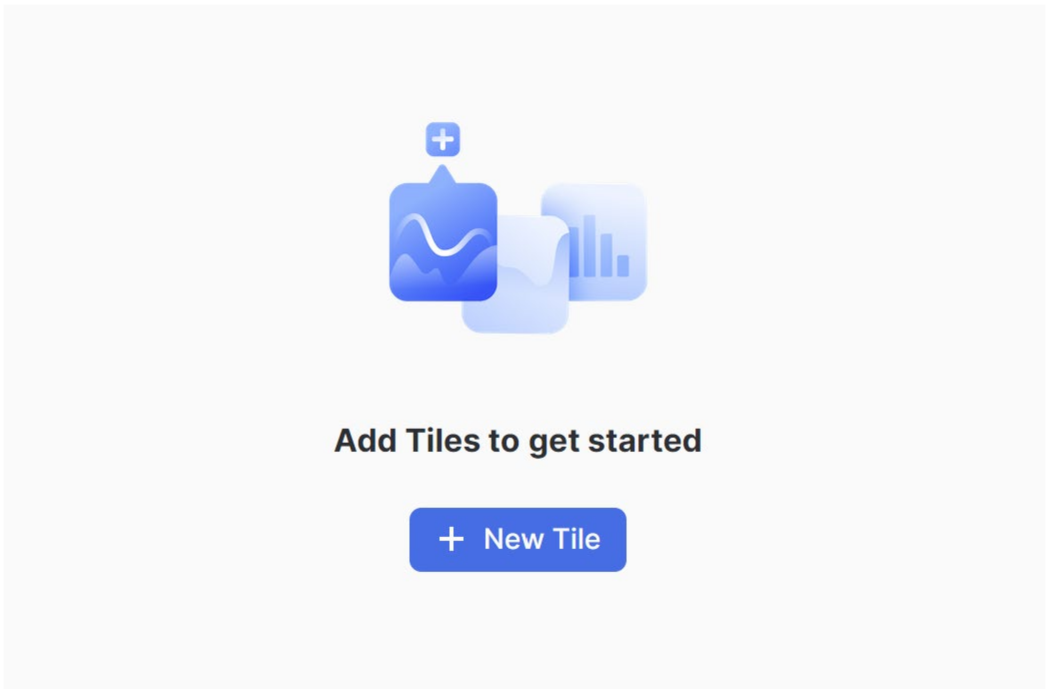
These queries will return a single value that can be placed on the dashboard as a KPI. Now we will create our first tile with the queries above. When we click the Dashboard tab, we will be navigated to the Dashboard page that will show our created dashboards along with dashboards that are shared with us:



We will click on the + Dashboard button in the top right corner of the page:

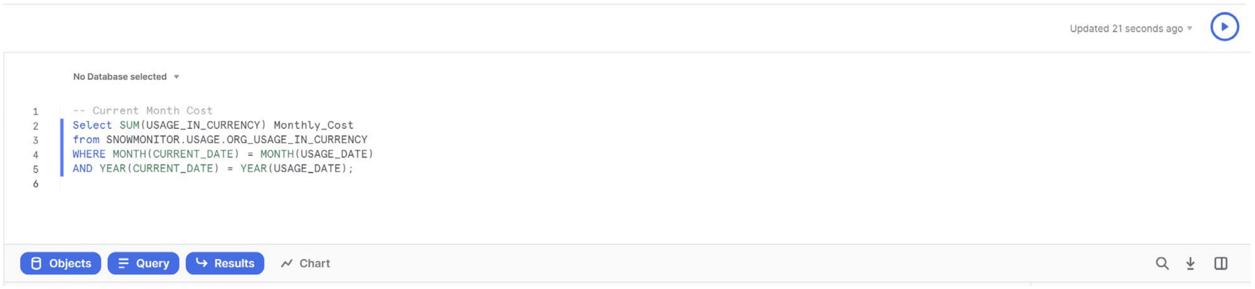


We will name the dashboard, Snowflake Usage, to align with the content that will be displayed within the dashboard. Once created we will be given the option to add tiles to the newly created dashboard:



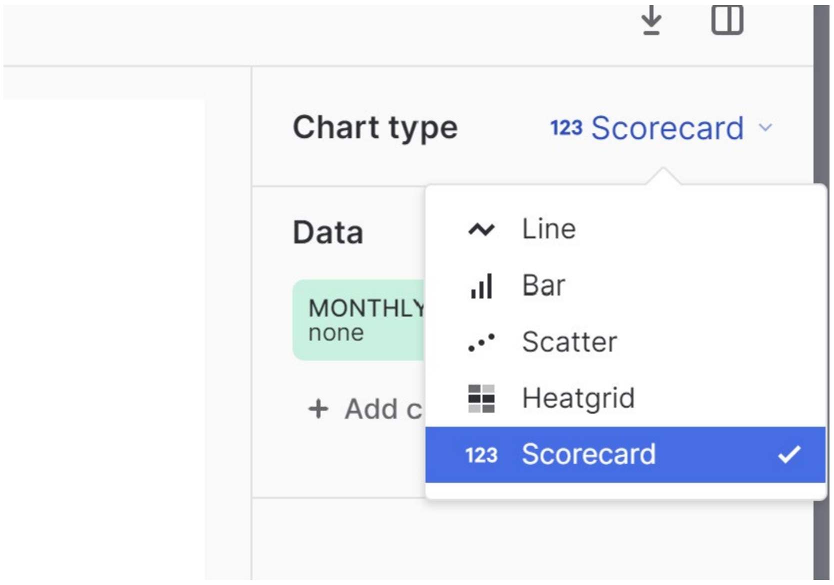
We will click on + New Tile to add our first visualization to the dashboard. If you are currently under a capacity agreement with Snowflake, we can use the Remaining Balance query to show our remaining capacity balance. For all users, we can use the Current Month Cost and Prior Month Cost to highlight the cost of our Snowflake environment for the prior and current month.

Copy the query for –Current Month Cost, and past in the worksheet area that opened when we clicked + New Tile. Paste this query in the screen, and click the Run Arrow on the right side of the screen:

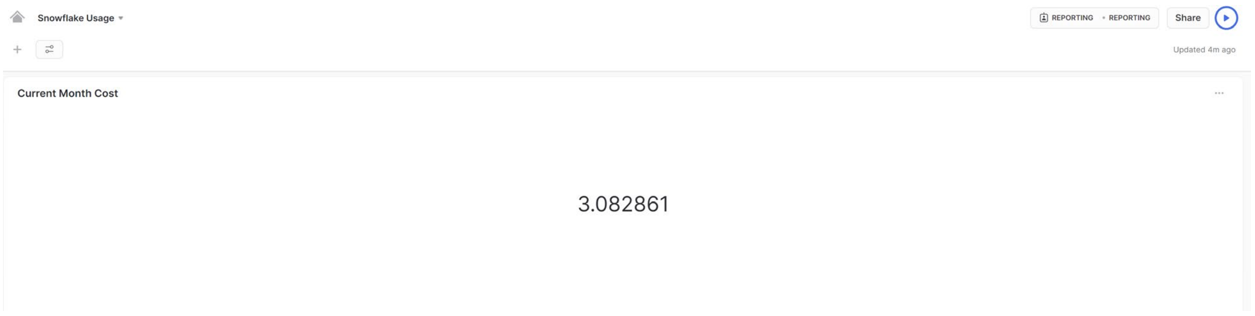


```
-- Current Month Cost
Select SUM(USAGE_IN_CURRENCY) Monthly_Cost
from SNOWMONITOR.USAGE.ORG_USAGE_IN_CURRENCY
WHERE MONTH(CURRENT_DATE) = MONTH(USAGE_DATE)
AND YEAR(CURRENT_DATE) = YEAR(USAGE_DATE);
```

With the results we will click on the chart option below our query text to switch the result to a visualization. From here we will change the visual to present the value as a scorecard:



After we have set our measure as a Scorecard we can return to our Snowflake Usage dashboard. We will now see our Scorecard visual in the center of the dashboard:



Current Month Cost

3.082861

We will now complete the same steps for the Prior Month Cost, to add a new visual click on the + on the left menu and select add new from worksheet at the bottom.

- Copy the query
- Add a new Tile
- Paste the query
- Adjust the visual to Scorecard
- Return to Snowflake Usage

Now we will have to measures that show the cost of our Snowflake account; Current Month Usage ($) and Prior Month Usage ($). We've now created a simple Snowsight dashboard that contains two KPI's.

Now we will look at if our account has a capacity agreement, meaning that we have an annual financial commitment of usage with Snowflake. These two additional queries will provide us with additional cost related KPI's along with predictive exhaustion of our capacity agreement.

The following queries will be used to help us manage our capacity agreement:

```
--Remaining Balance

SELECT FREE_USAGE_BALANCE + CAPACITY_BALANCE AS REMAINING_BALANCE

FROM "SNOWMONITOR"."USAGE"."ORG_REMAINING_BALANCE"

Where DATE = (SELECT MAX(DATE) FROM "SNOWMONITOR"."USAGE"."ORG_REMAINING_BALANCE");
```

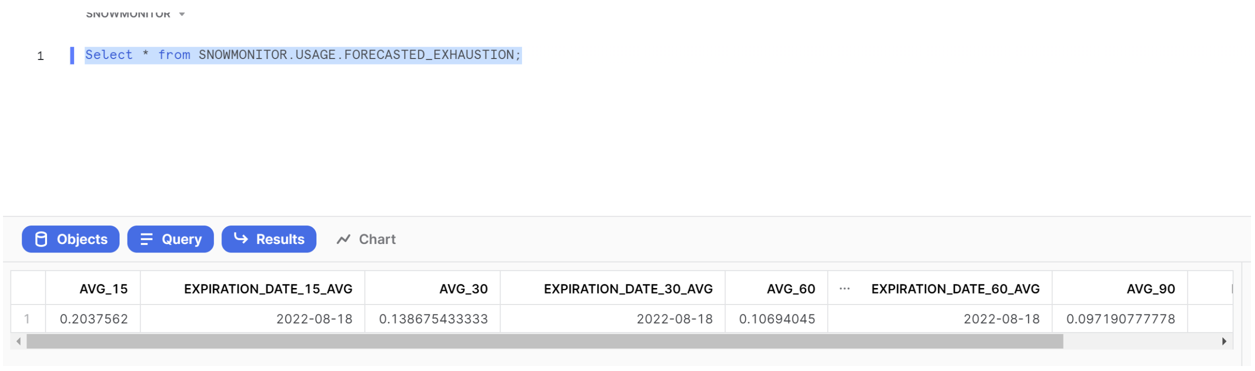
```
--Credit Forecast

SELECT * FROM SNOWMONITOR.USAGE.FORECASTED_EXHAUSTION;
```

(FORECASTED_EXHAUSTION is a view that was created during the initial build step)

We will add the Remaining Balance Scorecard just like the last two:

- Copy the query
- Add a new Tile
- Paste the query
- Adjust the visual to Scorecard
- Return to Snowflake Usage

For Credit Forecast, we will complete the same steps up to Adjust visual to Scorecard. For this query we will keep the output as a tabular set:

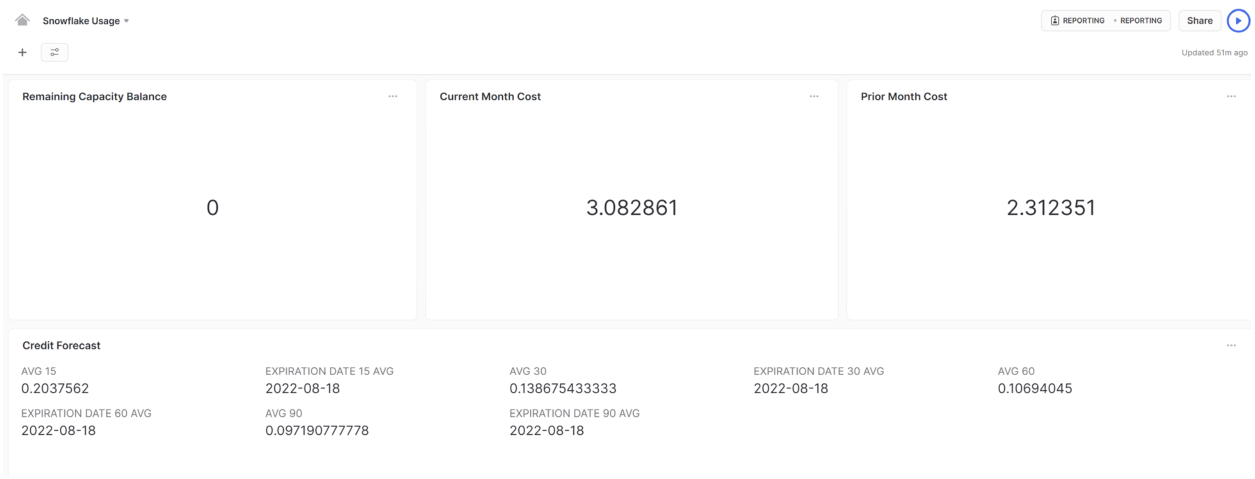


```
Select * from SNOWMONITOR.USAGE.FORECASTED_EXHAUSTION;
```

	AVG_15	EXPIRATION_DATE_15_AVG	AVG_30	EXPIRATION_DATE_30_AVG	AVG_60	EXPIRATION_DATE_60_AVG	AVG_90
1	0.2037562	2022-08-18	0.138675433333	2022-08-18	0.10694045	2022-08-18	0.097190777778

We will keep the results option selected below the SQL text, doing so will keep it in the result format.

With two additional measures added we are now able to leverage this information to track the cost of our Snowflake environment.



Snowflake Usage

Updated 51m ago

Remaining Capacity Balance	Current Month Cost	Prior Month Cost
0	3.082861	2.312351

Credit Forecast

AVG 15	EXPIRATION DATE 15 AVG	AVG 30	EXPIRATION DATE 30 AVG	AVG 60
0.2037562	2022-08-18	0.138675433333	2022-08-18	0.10694045

EXPIRATION DATE 60 AVG	AVG 90	EXPIRATION DATE 90 AVG
2022-08-18	0.097190777778	2022-08-18