**Snowflake**

Snowflake is a Data Warehouse that runs entirely on cloud infrastructure, and it cannot be run on a private cloud or on hosted infrastructure. It is primarily available on AWS and Azure cloud. Snowflake is not a relational database. So, it doesn’t enforce primary key and foreign key constraints. But it offers Snowflake SQL commands like DDL, DML and SQL functions.

The Snowflake SQL is rich enough to meet most of the data warehousing requirements. It supports select statements, joins, subqueries, insert, update, delete, merge statements, views, materialized Views, and ACID transactions. It also supports common analytical aggregations such as the cube, rollup, grouping sets, windowing functions, connect by and recursive CTE. Like any typical Data warehouse, it allows connection from most of the popular date integration tools such as IBM Data stage, Informatica, Talend, Pentaho, Power BI, Tableau, Spot fire, Apache Kafka, Apache Spark and data bricks. It also offers native connectors for languages like python, Go lang, and Node.js. You can install Snowflakes CLI on your local machine or connect using DBeaver.

A great thing that Snowflakes do is the real separation of storage and compute recourses. In snowflakes we have a notion of 2 things.

1. Database
2. Virtual Data Warehouse

The database is the storage layer. It allows you to define a database, a bunch of schemas and then add tables to the schema. Once you have tables you can start loading data into those tables. This is the typical structure that we follow from any data warehouse. One thing that is unique in snowflake. The data in tables are stored in Amazon S3, and it consumes storage only cost. There is no compute cost attached to the database unless you are executing DDL or DML queries. It means that you will pay for the compute cost while you are running DDL statements to create a database, a schema, table, or other structural objects. If all these activities take 30 minutes you will pay for the 30 minutes of computing cost. Once your table structure is in place, you would want to load some data into your tables. Rest of the time you will be paying only for the storage cost. That’s how it works at a high level.

The next part of the Snowflake is the virtual Warehouse, which is nothing but a compute cluster. They are named after virtual machines. But instead of machines, Snowflake termed them as warehouse in short we call them VW instead of VM. you can create VM of various sizes depending upon your requirement. It could be single node VW or it could be a multi-node VW such as 2 nodes, 4 nodes, 8 nodes and so on. These nodes are nothing but amazon EC2 instances, but they are internal to VM, and you don't directly interact with them. Creating a VW does not have any cost associated with it. it is just a metadata creation. and you can have more than one VW configuration for example, you might want to create a single node VW for executing the DDLs or performing some other low resource consuming activities. At the same time, you can also define a 4 node VW which you want to use only for the data loading activity and you can create one more a 32 node VW which you wish to apply for a high performance job that triggers every hour and completes in less than 5 minutes.

You create your compute resource definition and start it only when you want to do some computation and shut it down when the computation activity is over. In this way, you pay the compute cost only when your VW is running. You can also use the same VW to execute multiple concurrent queries. In that case as the queries are submitted to the same VW, the warehouse allocates resource to each query and begins running them. If sufficient resources are not available to execute all the queries, additional queries will be queued until the necessary resources become available. So, you have all the flexibility to plan the compute workload and reuse your VWs.

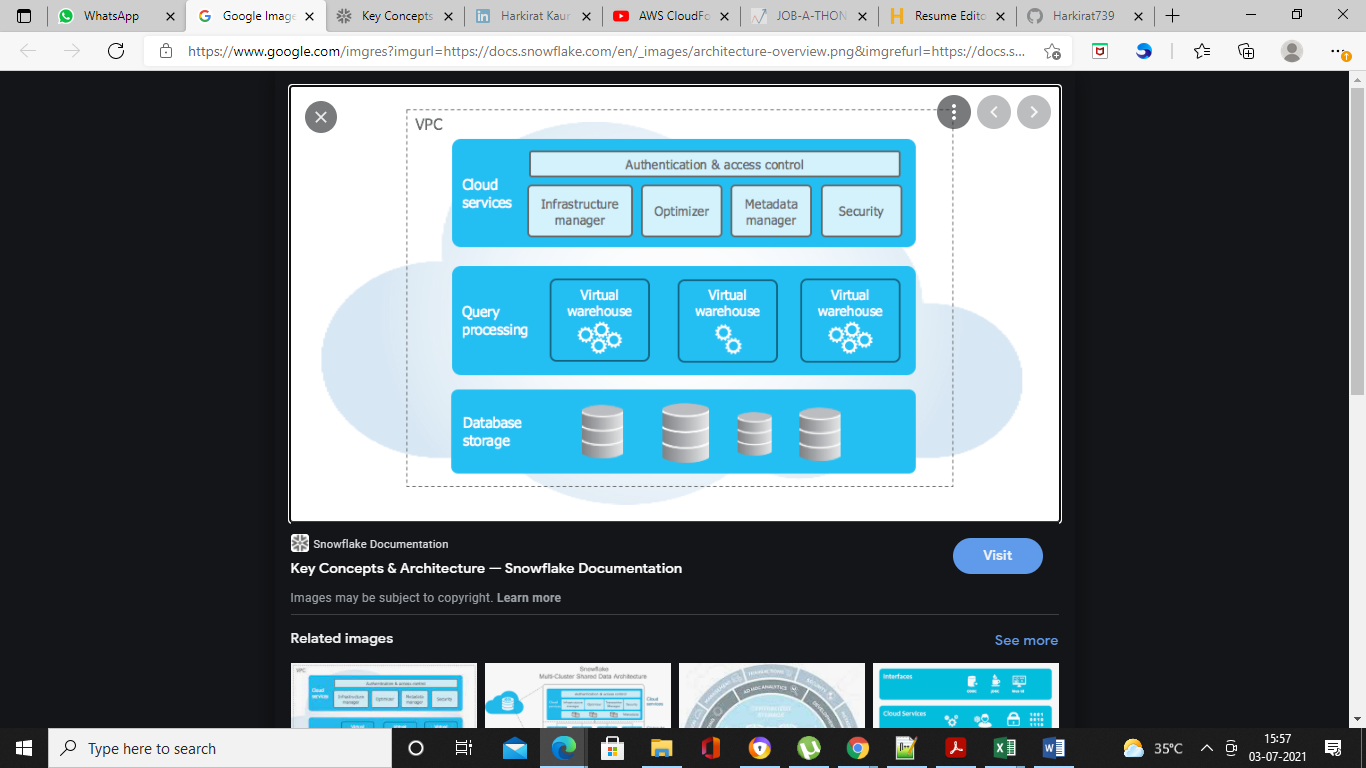
The notion of VW doesn't stop there. Snowflake also offers a unique idea of multi-cluster VW for auto scaling. For example, you created a multi-cluster VW and started a query on the VW. Snowflake would start your query normally is the same VW. However, if you start submitting more and more queries on the same VW, at some point, they will consume all the resources and additional queries will be queued until ones are complete. But you created a multi-cluster warehouse. A multi-cluster VW will detect this scenario and automatically launch a new VW to execute the queued SQLs. Snowflakes allows you to configure automatic scale-up by starting additional VWs and scale-down by shutting down the VWs depending upon the workload. This feature is powerful if you have strict performance SLAS.

**Snowflake Architecture**

In order to understand the arcitecture of snowflake we need to understand two important concepts. The very first is a shared disk architecture and the second is shared nothing architecture.

**Shared disk architecture:** Shared disk architecture is a distributed computing architecture in which all the disks are accessible from all the cluster nodes. We have two major components one is data base and the another is instances. Database is the storage place where we can store the data in the form of tables which underline uses the rows and columns format and it has an indexer.It also carries some logs and the control files. So this data gets shared across the multiple instances which has its own memory and own processes.

**Shared nothing architecture:** Shared nothing architecture is a distributed computing architecture in which all nodes have sole access to distinct disks. We have 3 different disks or the 3 different storages which are nothing but the databases which again hold the different data in the form of rows and columns under the tables with the indexes and it has logs and control files we have the 3 different databases and these different databases has connected to the each instance or each node. And these nodes have its own memory the processes but this all these nodes are connected by coordinator so that means some of the memory components are getting shared or some of the resources are getting shared in this architecture so each node has its own database component and all those are connected either through their internal network or by the shared components.



Snowflake’s unique architecture consists of three key layers:

1. [Database Storage](https://docs.snowflake.com/en/user-guide/intro-key-concepts.html#database-storage) Layer
2. [Query Processing](https://docs.snowflake.com/en/user-guide/intro-key-concepts.html#query-processing) Layer or Compute Layer
3. [Cloud Services](https://docs.snowflake.com/en/user-guide/intro-key-concepts.html#cloud-services) Layer

So this architecture of snowflake internally use the shared disk and shared nothing arhitecture.

**Database Storage Layer:-** In this storage snowflake organizes the data in the compressed columnar format in optimized manner about how this data is stored in a storage system. It could be a file size or metadata or statistics. So all that information is maintained by Snowflake so we don't have to worry about that and it is not exposed to the end user even developers.

**Compute layer:-** Snowflake uses “Virtual Warehouse” for running queries. Snowflake separates the query processing layer from the disk storage. Queries execute in this layer using the data from the storage layer. Each virtual warehouse can work with one storage layer. Generally, a virtual Warehouse has its own independent compute cluster and doesn’t interact with other virtual warehouses.

**Cloud Services :-** All the activities such as authentication, security, metadata management of the loaded data and query optimizer that coordinate across Snowflake happens in this layer.

These three layers scale independently and Snowflake charges for storage and virtual warehouse separately. Services layer is handled within compute nodes provisioned, and hence not charged.

Snowflake decided to use S3 as its storage layer. That was a contrarian decision for database storage because we all know that S3 comes with varying performance and put up several restrictions. For example, S3 has a much higher latency and CPU overhead compared to the local storage. That is why, HDFS on local storage works about better than S3.

The 2nd problem, we can only overwrite files in S3. It does not support updating or even appending the same file. These two problems are so critical that they often put a hold on the decision to use S3 as database storage. Which means you can dump files in there.

However, the Snowflake teams found two main technical reasons to use S3. High availability and Durability.

The S3 doesn’t go down and once saved, you can't lose your files. The second and most critical reason. S3 APIs allowed them to read a part of the data from an S3 file which means, the ability to read in parts without even loading the entire file is a kind of Random access capability. This could be a powerful feature if you have something like an Index that allows you to read exactly what you want to read and avoid reading unnecessary volumes.

**How exactly snowflake leveraged these S3 capabilities?**

The first thing that snowflake does is to break the table into multiple smaller partitions. And they it micro partitions which is not more than 500 MB in size. So, each table partition is between 50 to 500 MB of uncompressed data. The next thing that they do is reorganize the data in each partition to make is columnar. Which means Simple column value in the partition are stored together.

The next step is to compress each Column. And that's unique. You are not compressing the entire partition. You are compressing only the column values individually.

Finally, they had a header to each micro partition. The header contains an offset and length of each column stored within the micro partition. They also store some other metadata information into the header, but the column offsets are the most critical information for us to understand the read pattern. Now, these micro partitions are stored in S3 as immutable files, and a lot of additional metadata and statistics about these micro partitions is maintained in the snowflake metadata layer. And that's why they call the cloud services layer of the Snowflake as the brain of the snowflake.

When you fire a SQL query, Snowflake would know which table and what columns do you want to read. Which micro partition files in S3 belong to the desired table. So they know, which files to read. But instead of reading complete file, they first read the header and then read only those columns which you wanted to read. And this is possible because S3 APIs allow them to read a part of file based on offset and length.

Snowflakes stores metadata and statistics about all the data stored in the micro - partition. So they know the range of the values and the number of distinct values for each of the Columns in the micro – partition. So when we apply a filter in a where clause, they know which micro - partition would have the desired data. This allows the first level of partition pruning and targets only those micro - partition where the desired data is stored. In the next step, they first read the micro-partition header and then only read the desired columns. And this allows the second level of column pruning and minimize the overall I/0 for the SQL. In other words, they automatically apply a generic partitioning strategy and columnar storage and take away this headache from the database designers.

In most of the cases, this generic partitioning works very well due to the tiny size of each micro-partition. If you have a time series date and your queries are filtering on a time frame, this partitioning strategy works like a charm. In certain other uses cases, when you already know your commonly used filter columns, they also give you an option to define table clustering key to increase the degree of partition pruning. The notion of clustering key is similar to what we see in Cassandra. That means, the date is ordered by the clustering key and stored close to each other.

**Difference between Redshift and Big Query and Snowflake**

|  |  |  |
| --- | --- | --- |
| **Redshift** | **Big Query** | **Snowflake** |
| **Setup:**   * Sizing appropriate cluster as storage and compute are not separated. * Designing data workflow to match resource size. * Data optimization required.   **Maintenance:**   * Requires Vacuuming/ Analysing tables periodically.   **Management:**   * Difficult to manage without skilled AWS architect. * Customers may spend hours doing maintenance like updating. | **Setup:**   * No sizing required as storage and compute are separated.   **Maintenance:**   * Low maintenance * Limitations:   No indexes  No columns constraints  No performance tuning capabilities.  **Management:**   * Fully managed services * Backend configuration and tuning is handled by Google. | **Setup:**   * No sizing required as storage and compute are separated. * Selection of cloud provider required.   **Maintenance:**   * Low maintenance. * Automatic and rapid provisioning of greater compute resource.   **Management:**   * Zero management from end users. |

**Features**

* **Faster, Easy to use, more flexible :-** Snowflake is easy to use and it is very flexible**.**
* **Security and Data Protection :-** Snowflake data warehouse offers enhanced authentication. All the communication between the client and server is protected by TLS.
* **Data Sharing :-** You can securely share your data with other Snowflake accounts.
* **Standard and Extended SQL Support :-** Snowflake data warehouse supports most DDL and DML commands of SQL. It also supports advanced DML, transactions, lateral views, stored procedures, etc.

**Mode of connection**

* Browser browsed web Interface
* Command line client (snow SQL)
* JDBC 04 ODBC connection
* 3rd party Partner connection e.g. Informatica

**Cloud platforms**

Amazon web services (AWS)

Google Cloud Platform (GCP)

Microsoft Azure (Azure)

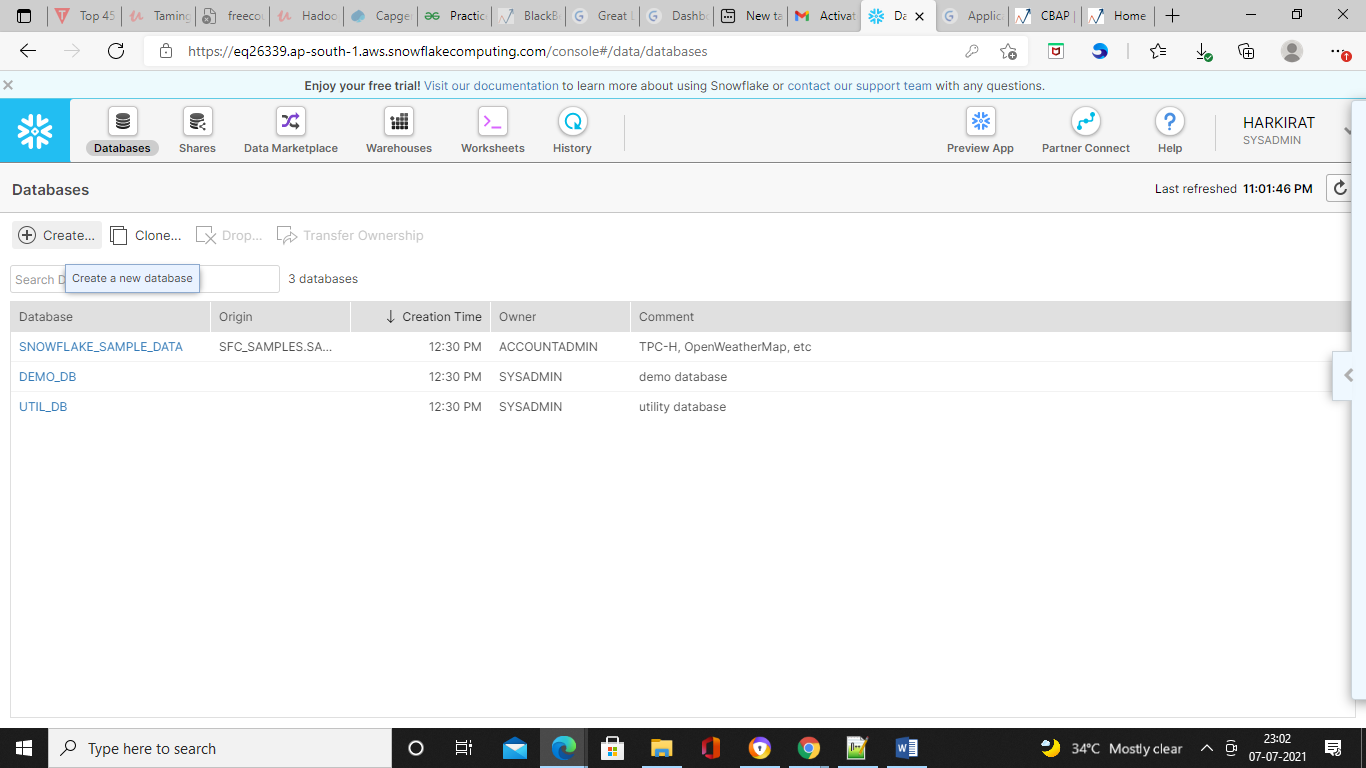
**Snowflake objects**

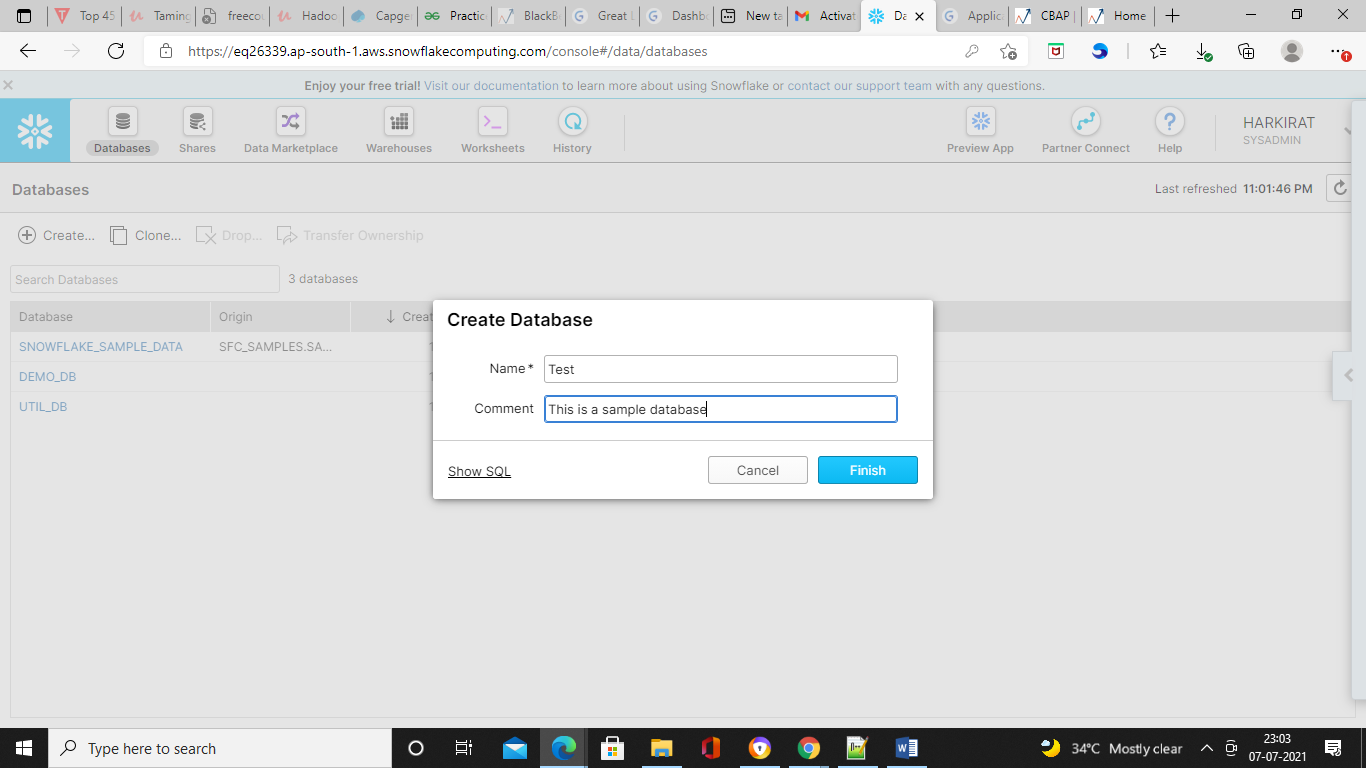
* Databases
* Schema
* Warehouse
* Tables

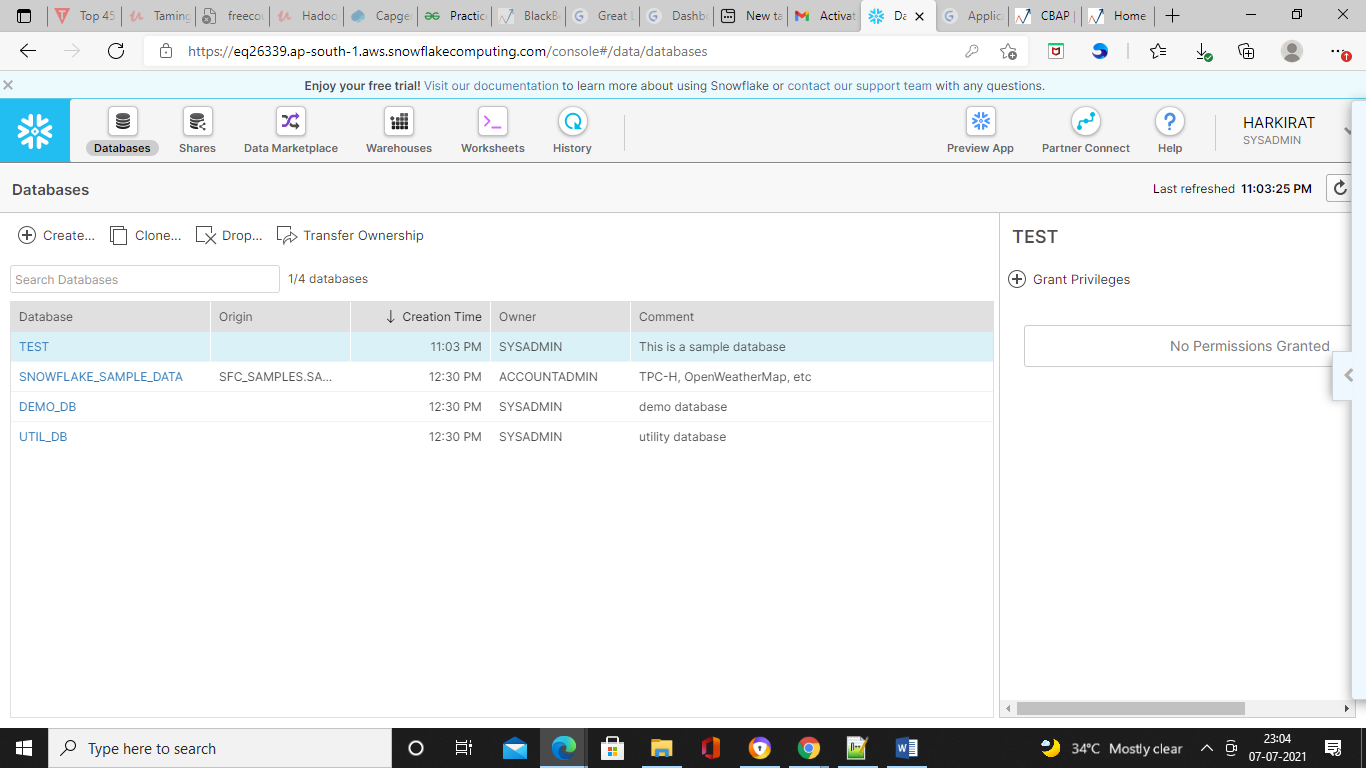
We can create and manage all these snowflake objects using this interface. We can also limit the amount of the data getting loaded into the tables by using this tool or by using this user interface apart from it we can execute DML and DDL operations on adobe basis.

**Database**

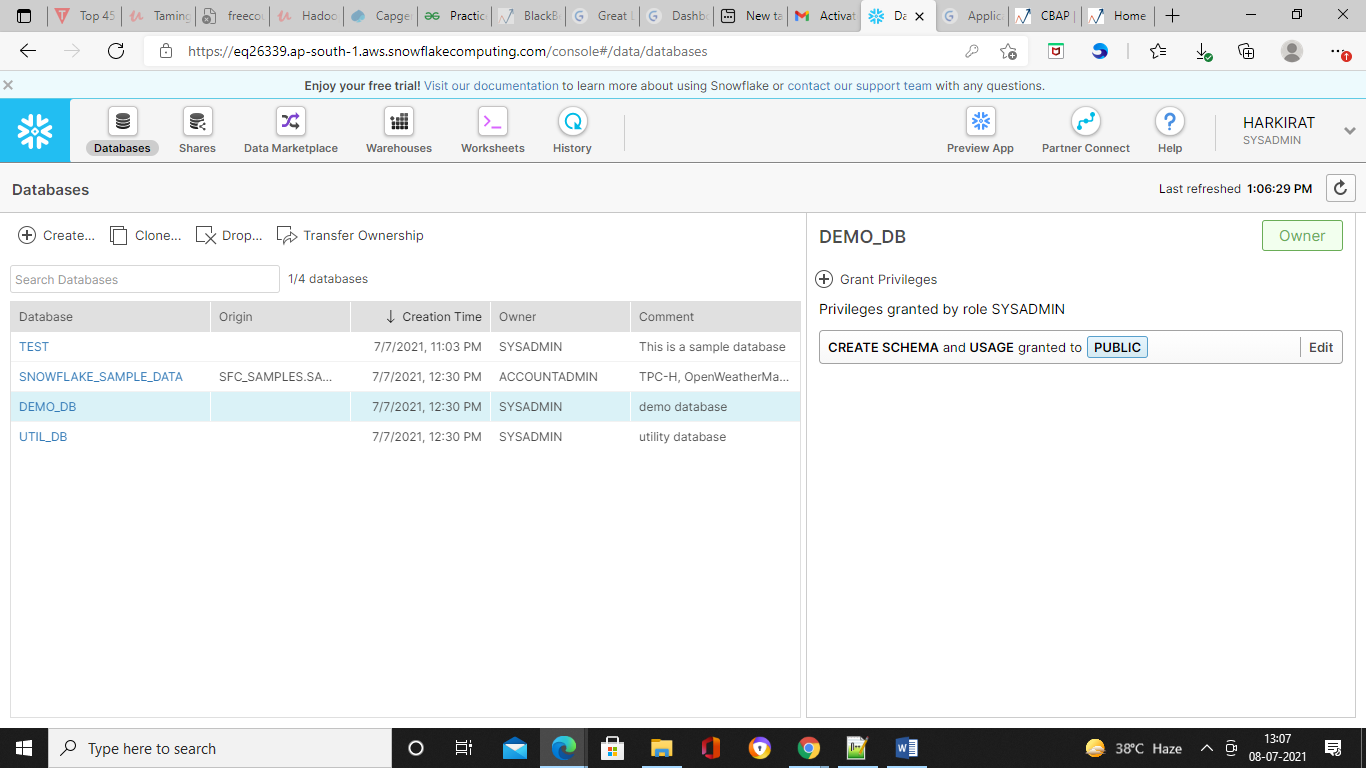
Creating a database

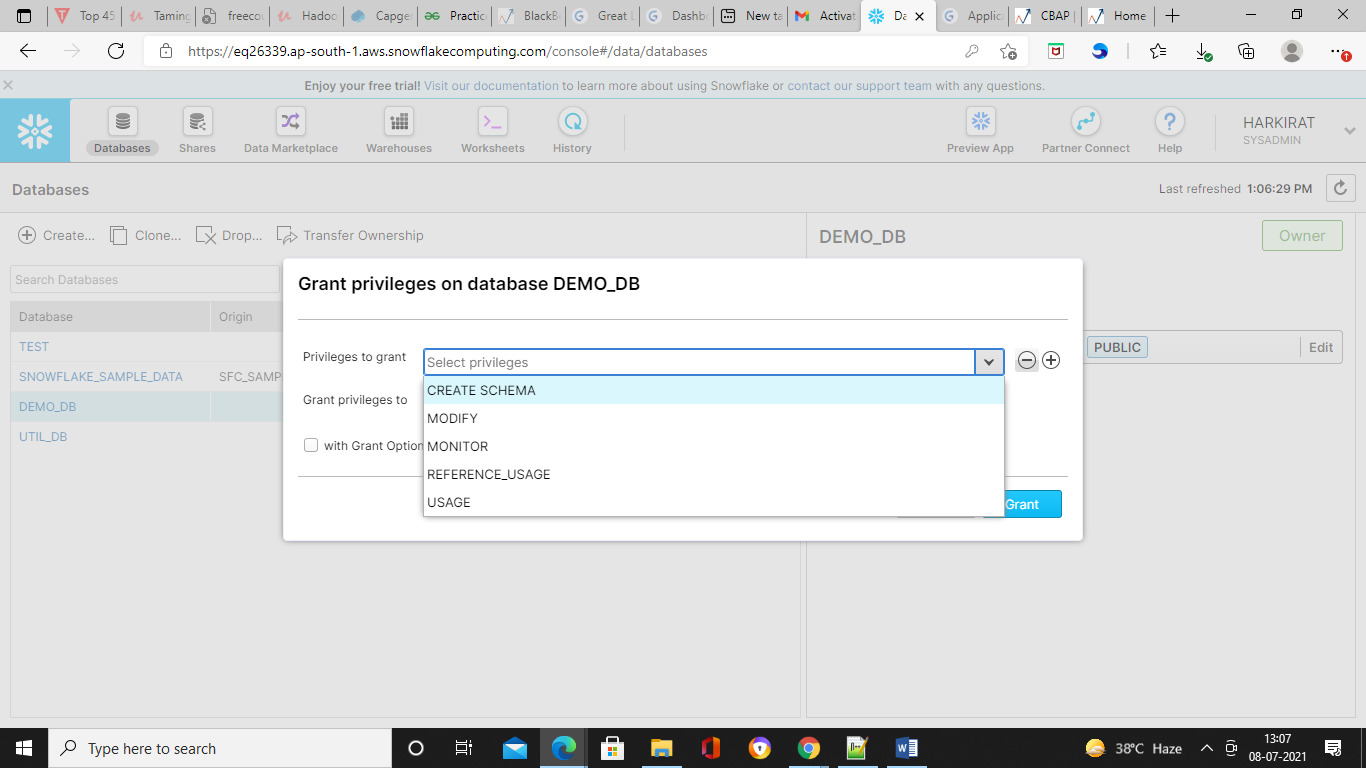






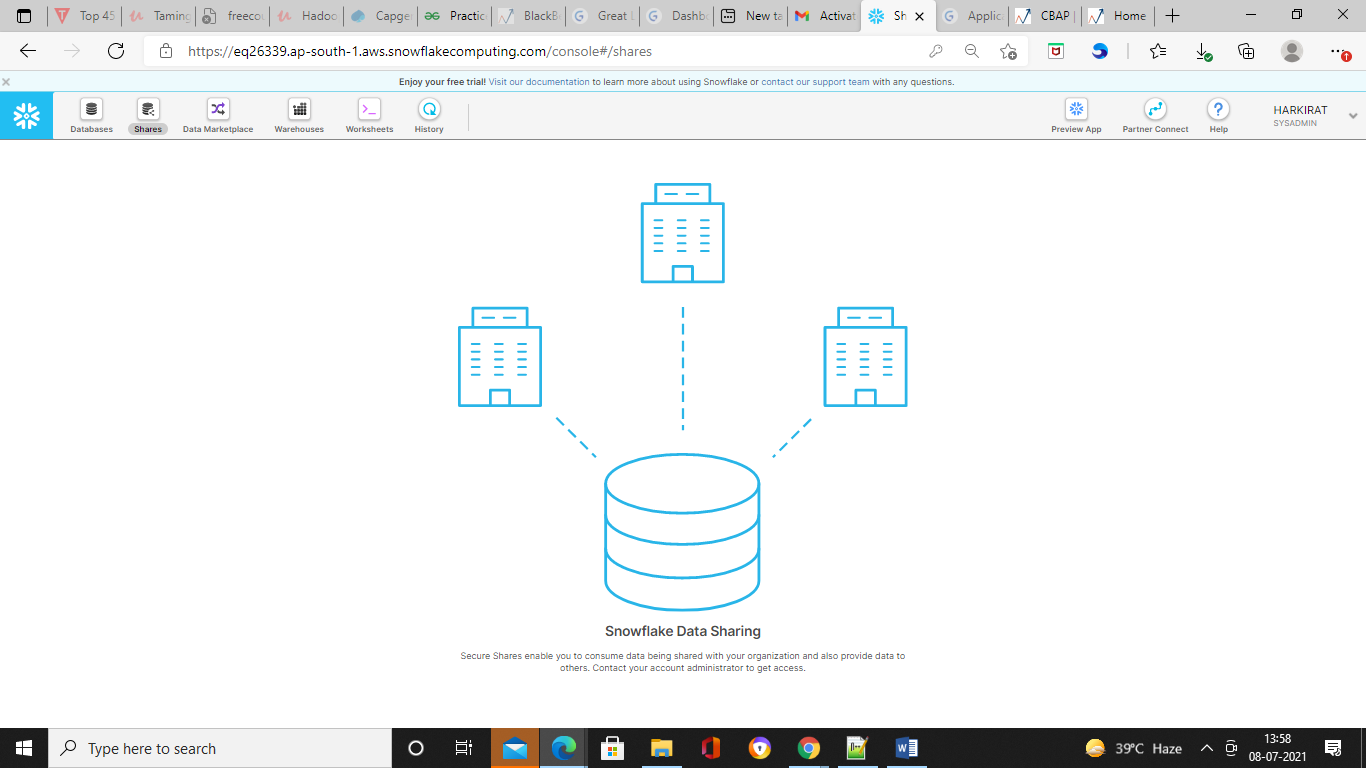
If you want to drop a database, click on drop.





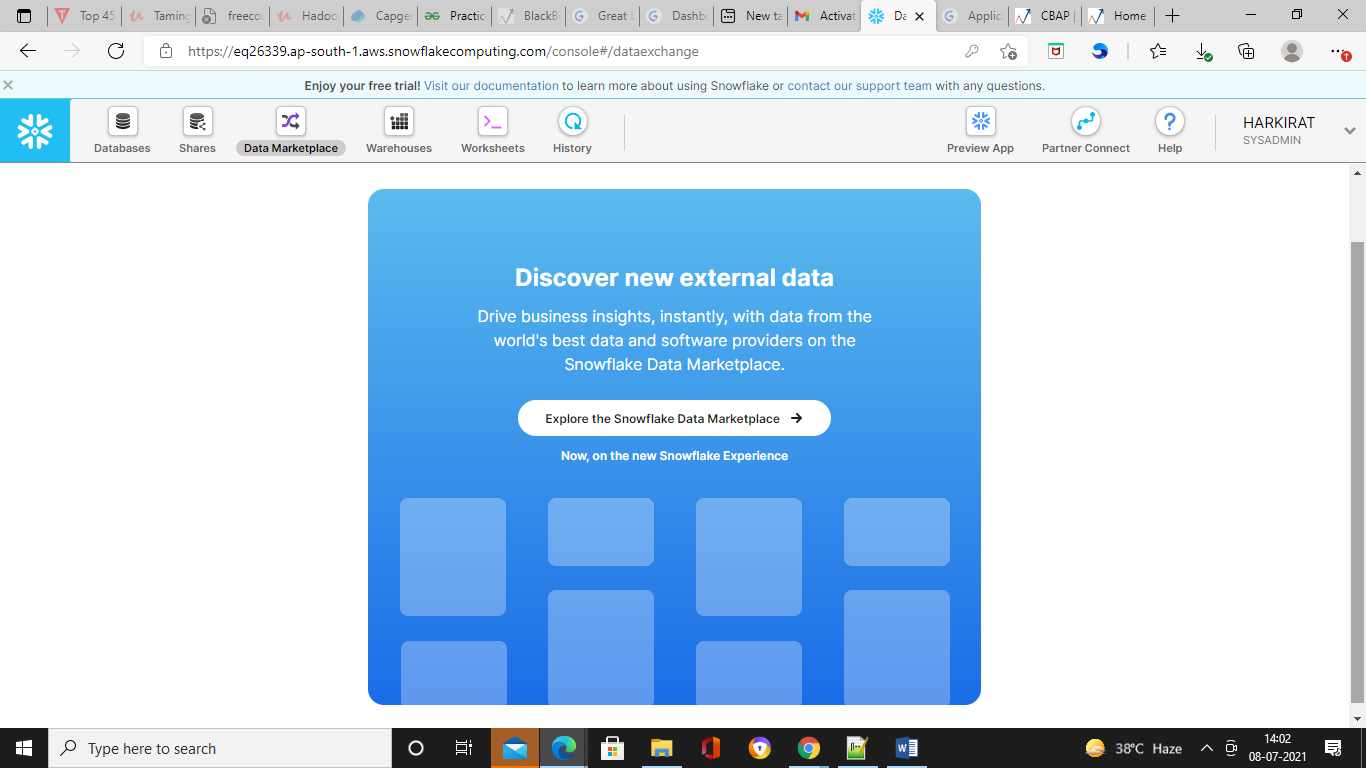
**Shares**

If you like to share the snowflake data, so this is the option will be used to securely share the data.

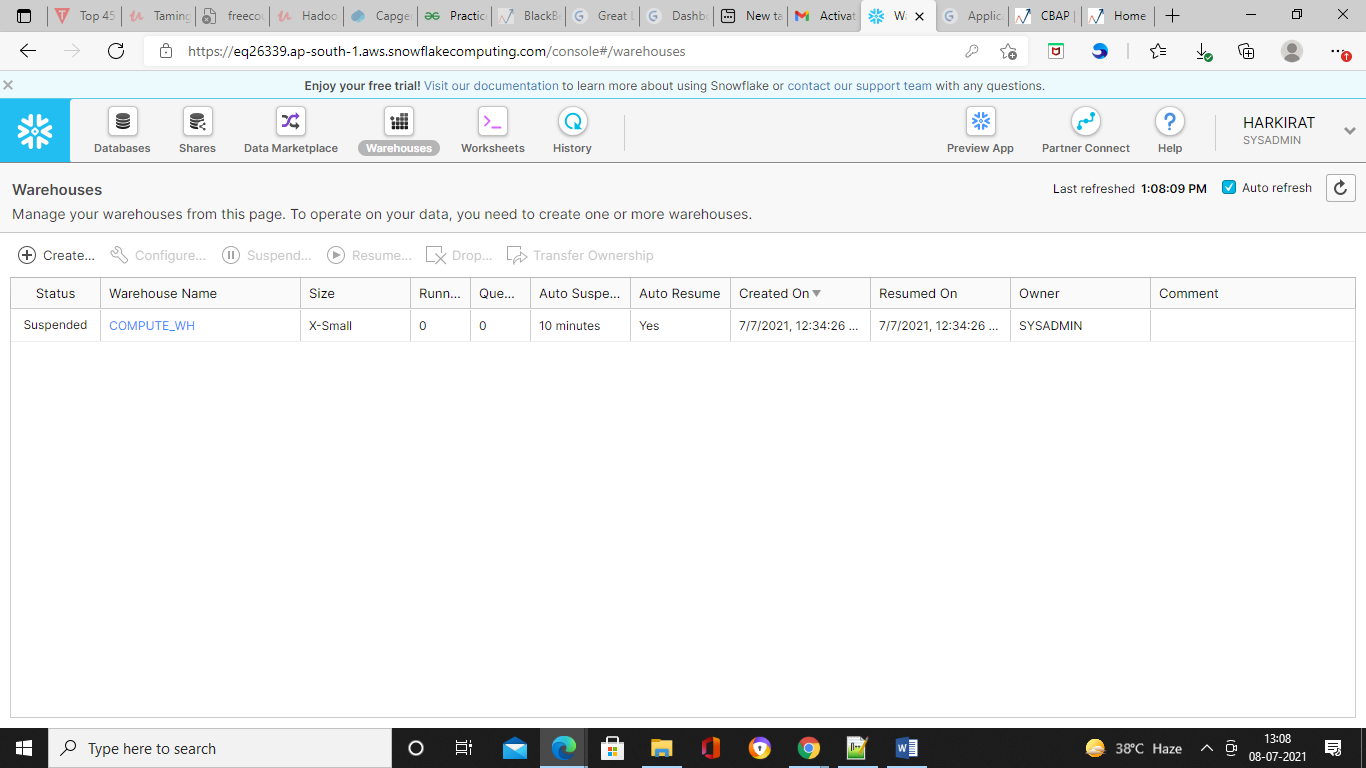


**Data Marketplace**

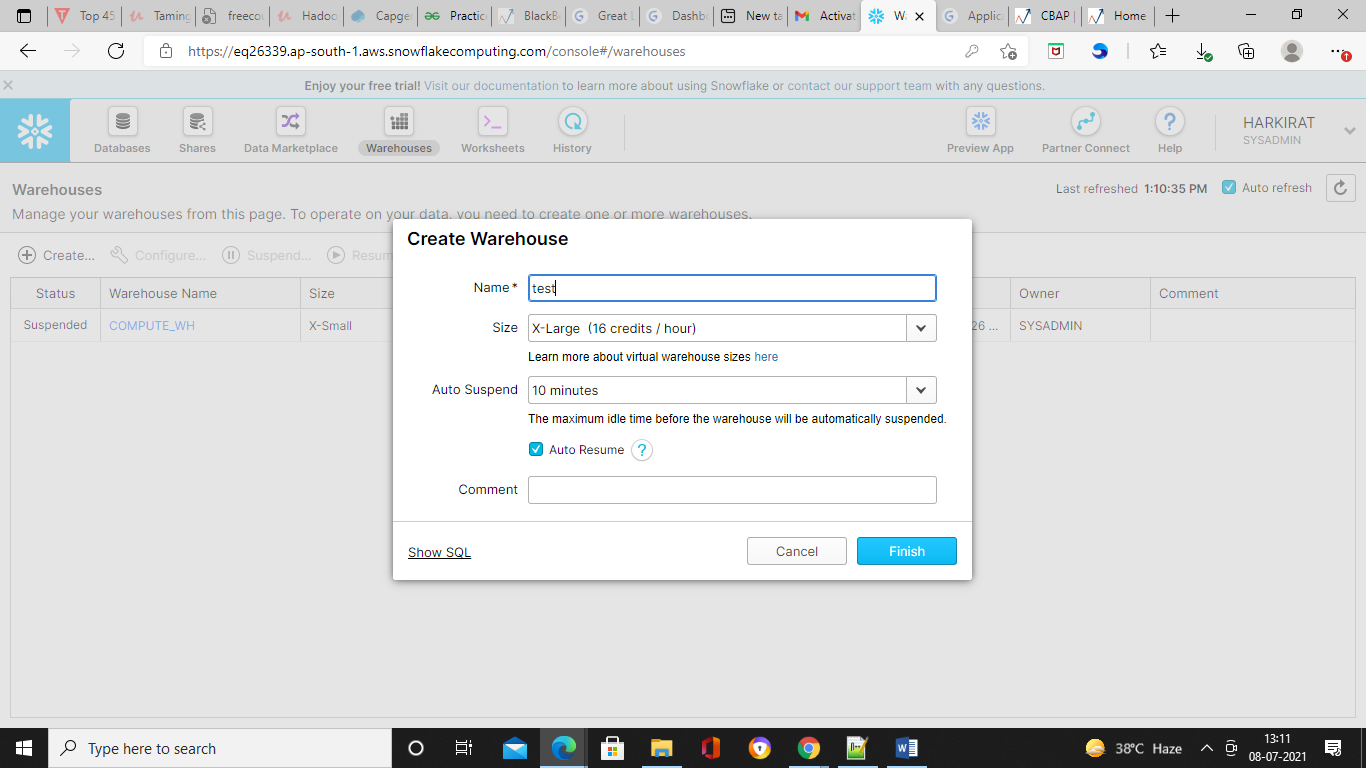
Data Marketplace is used to get the more insights about the business you can explore the more option in the marketplace.



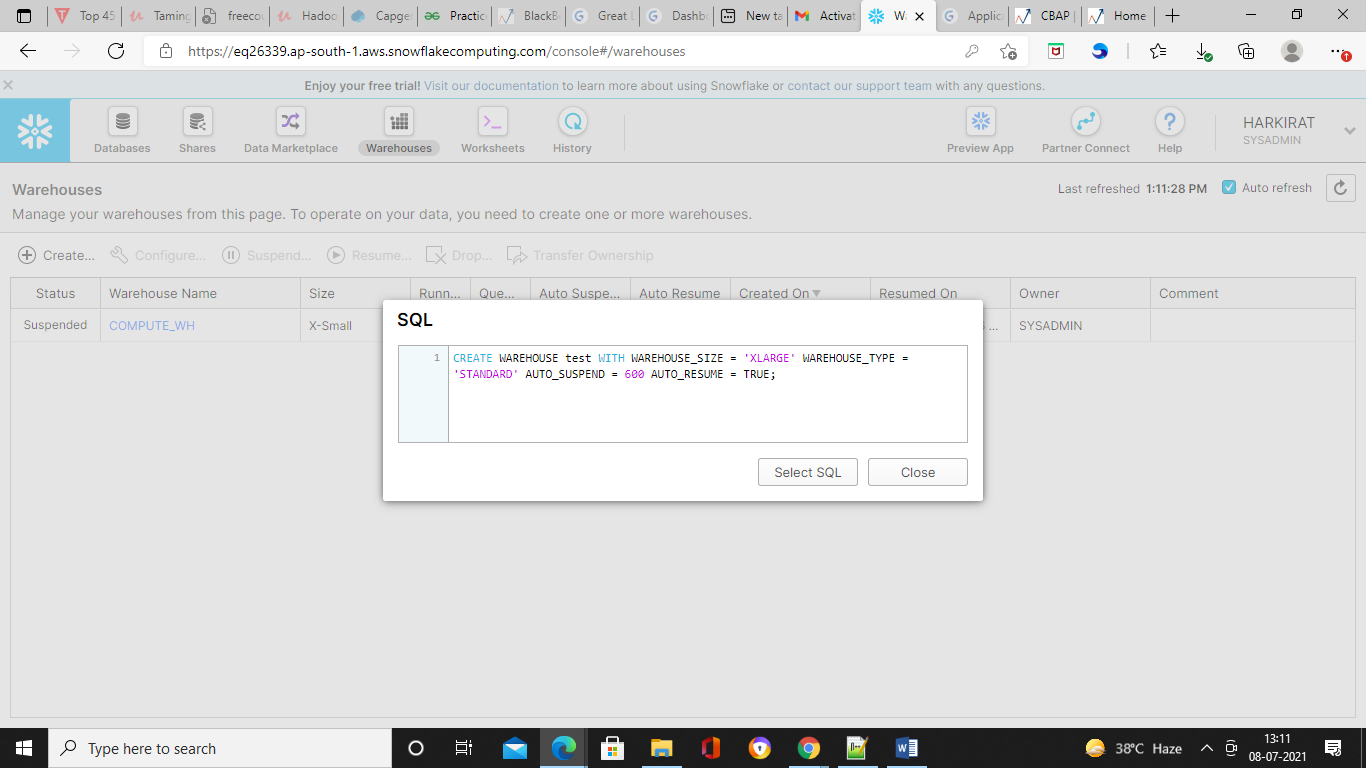
**Warehouse**

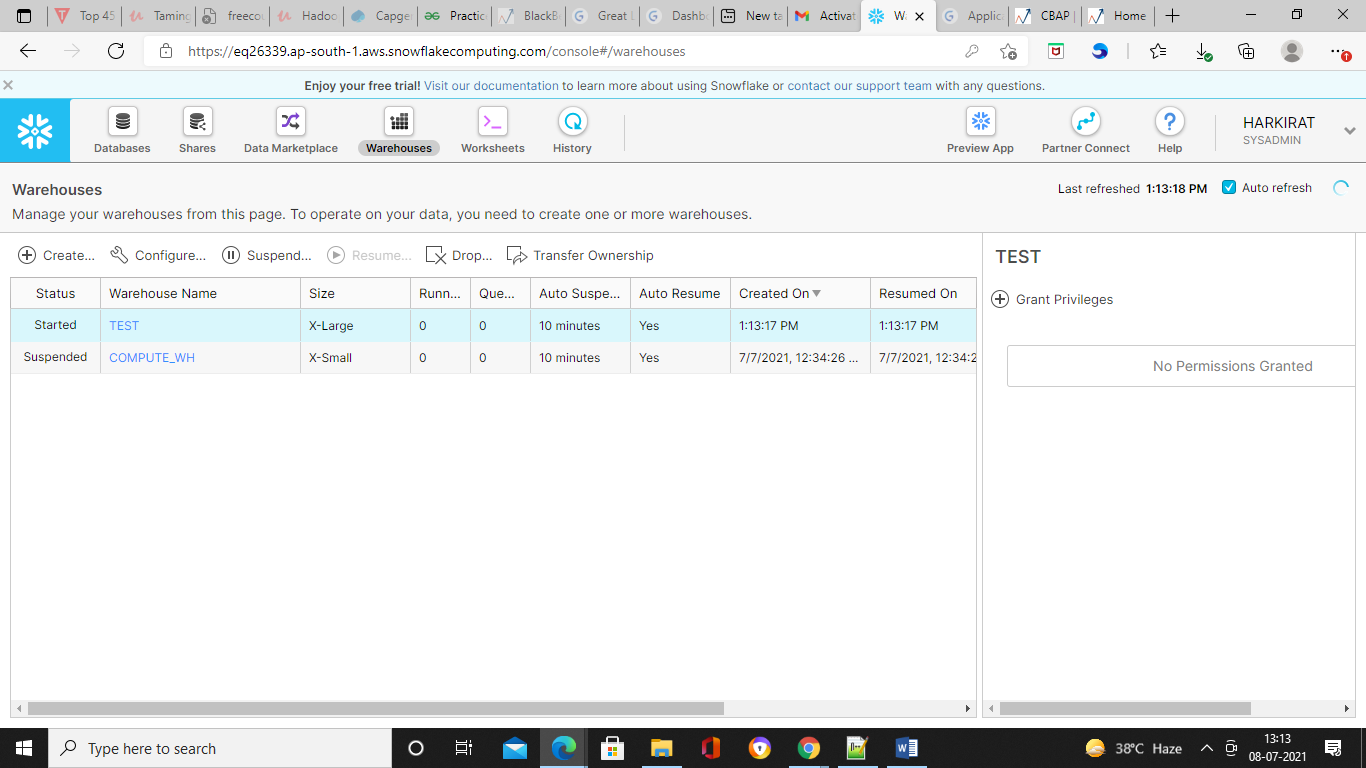


In case you want to create a new warehouse, click on create

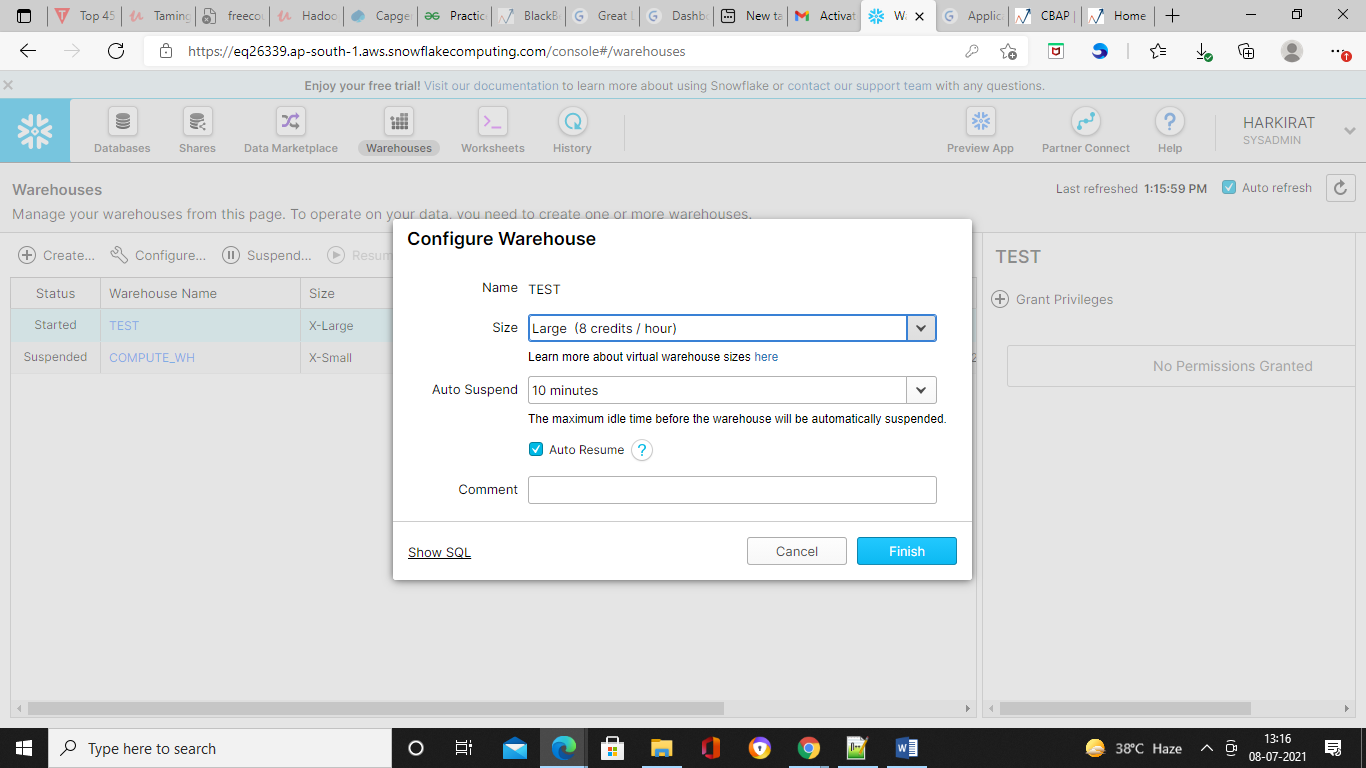


In case you want to see sql, click on show sql





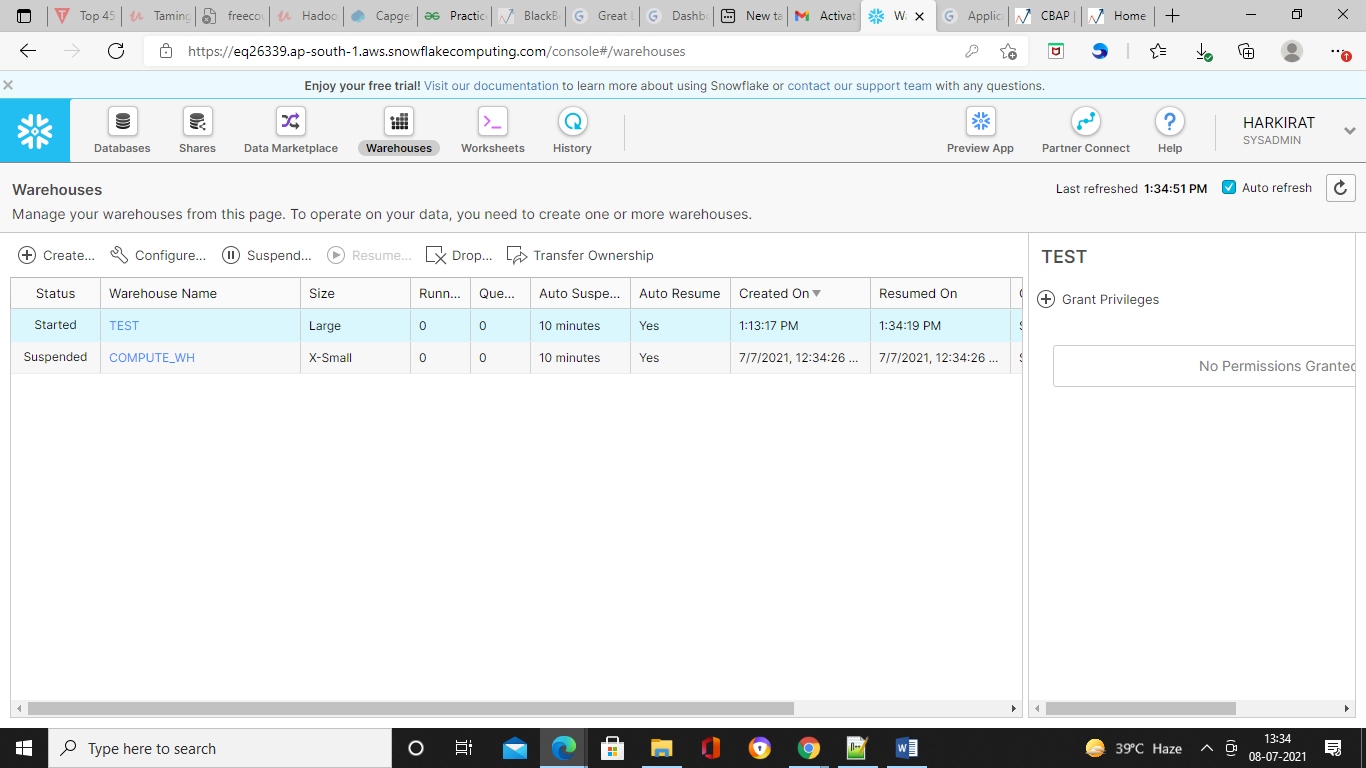
To configure

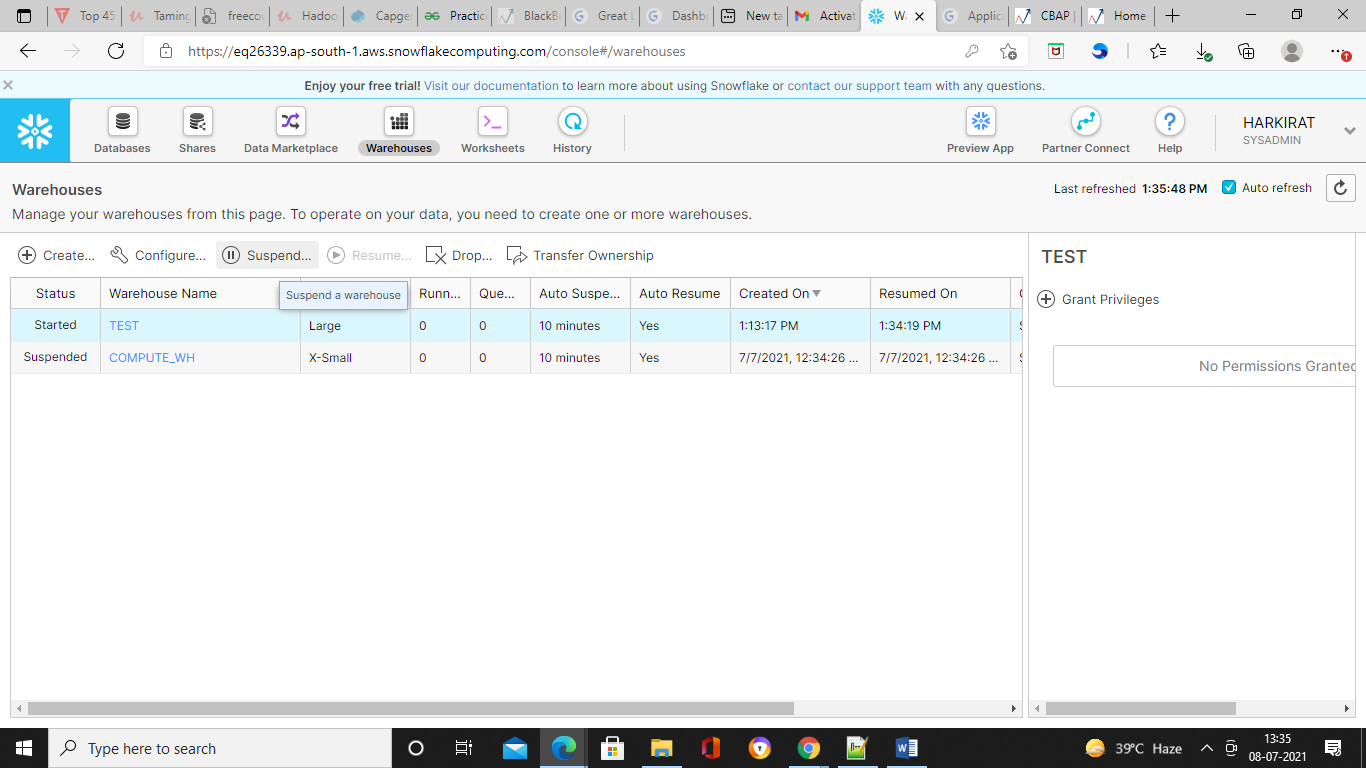


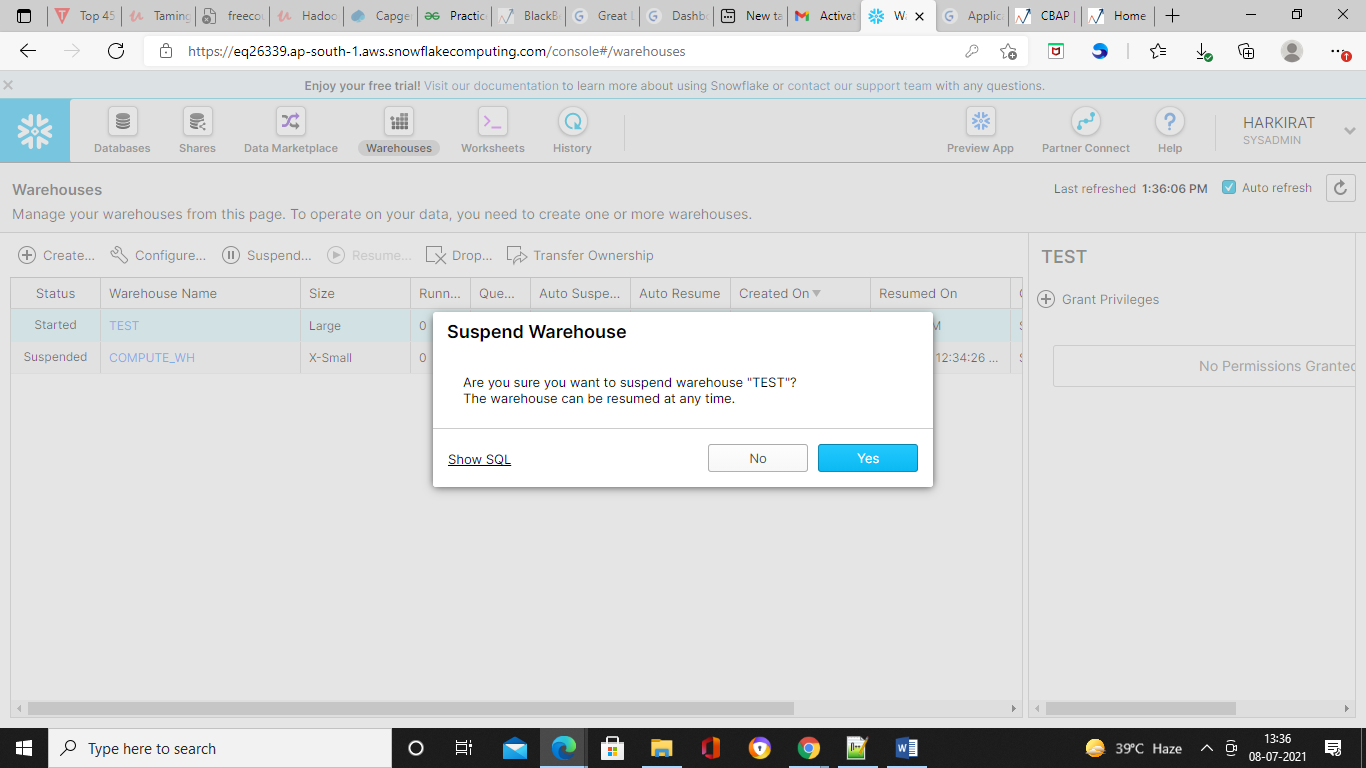
In case you want to change the status of your warehouse,

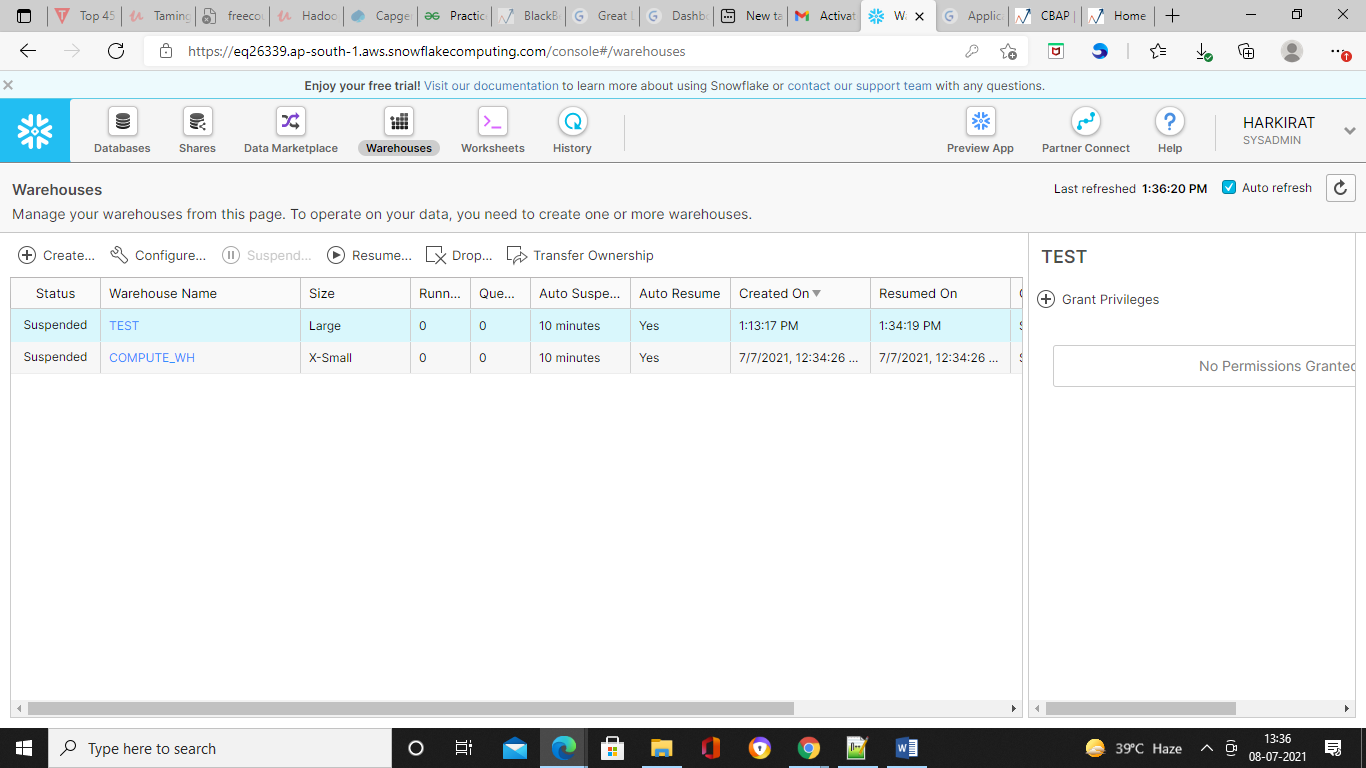
* select the warehouse and click on suspend.
* This will give us a pop up message saying that “Are you sure you want to suspend the warehouse”.
* Click YES.

The status will change to suspended.



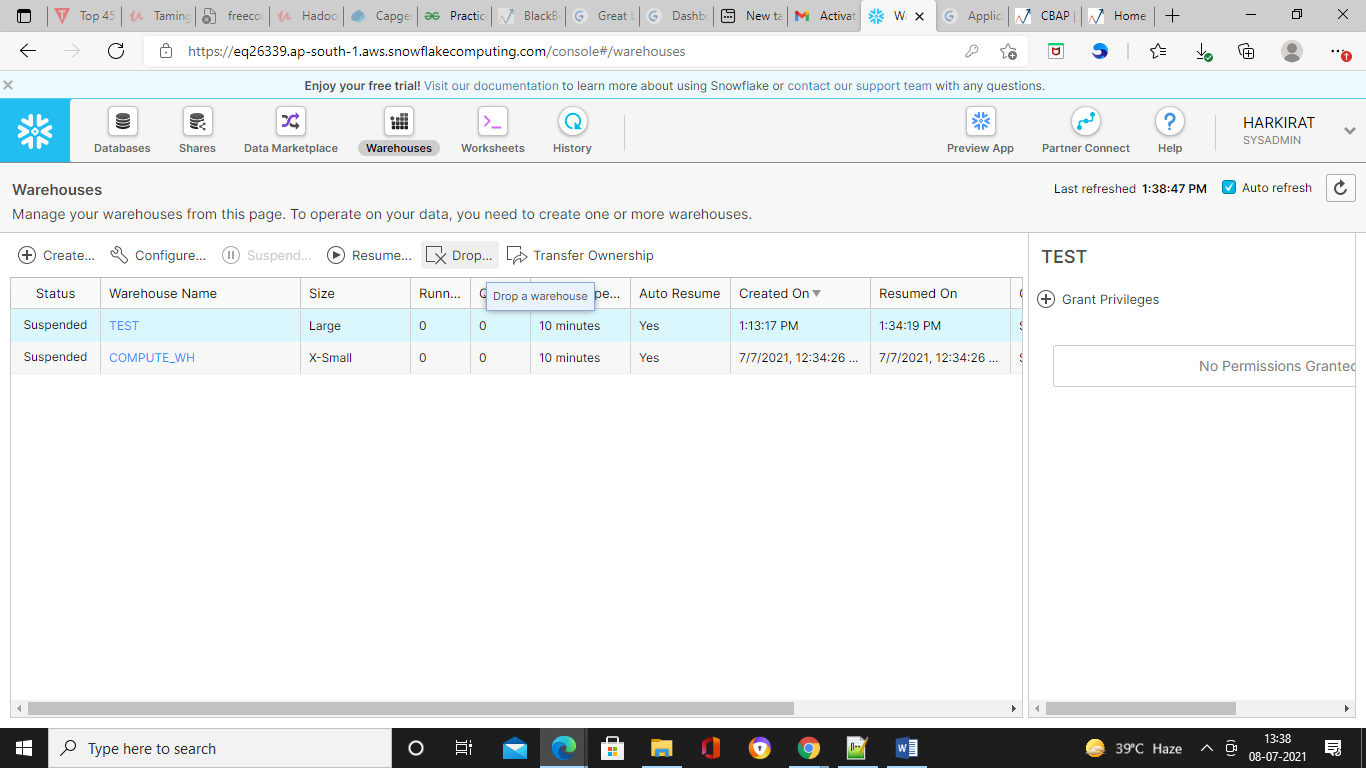


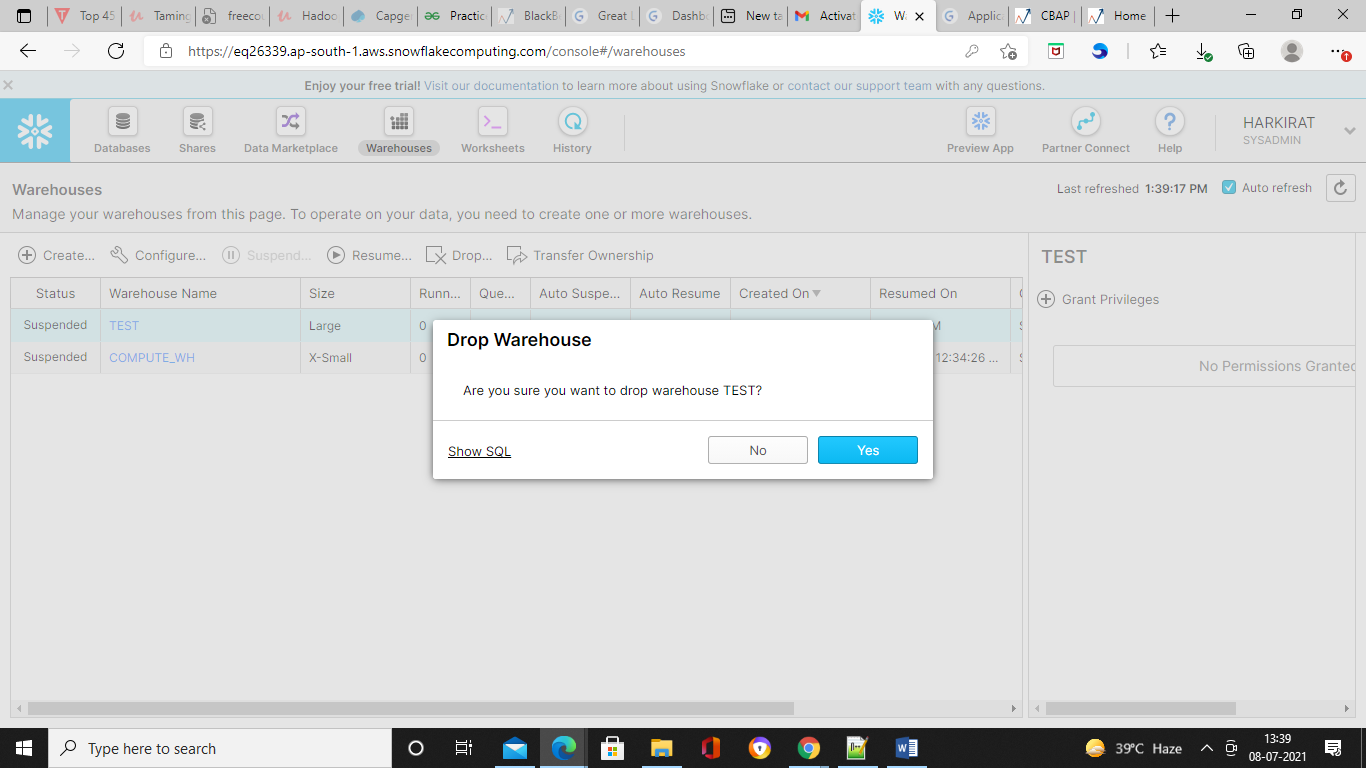




In case you want to drop the warehouse,

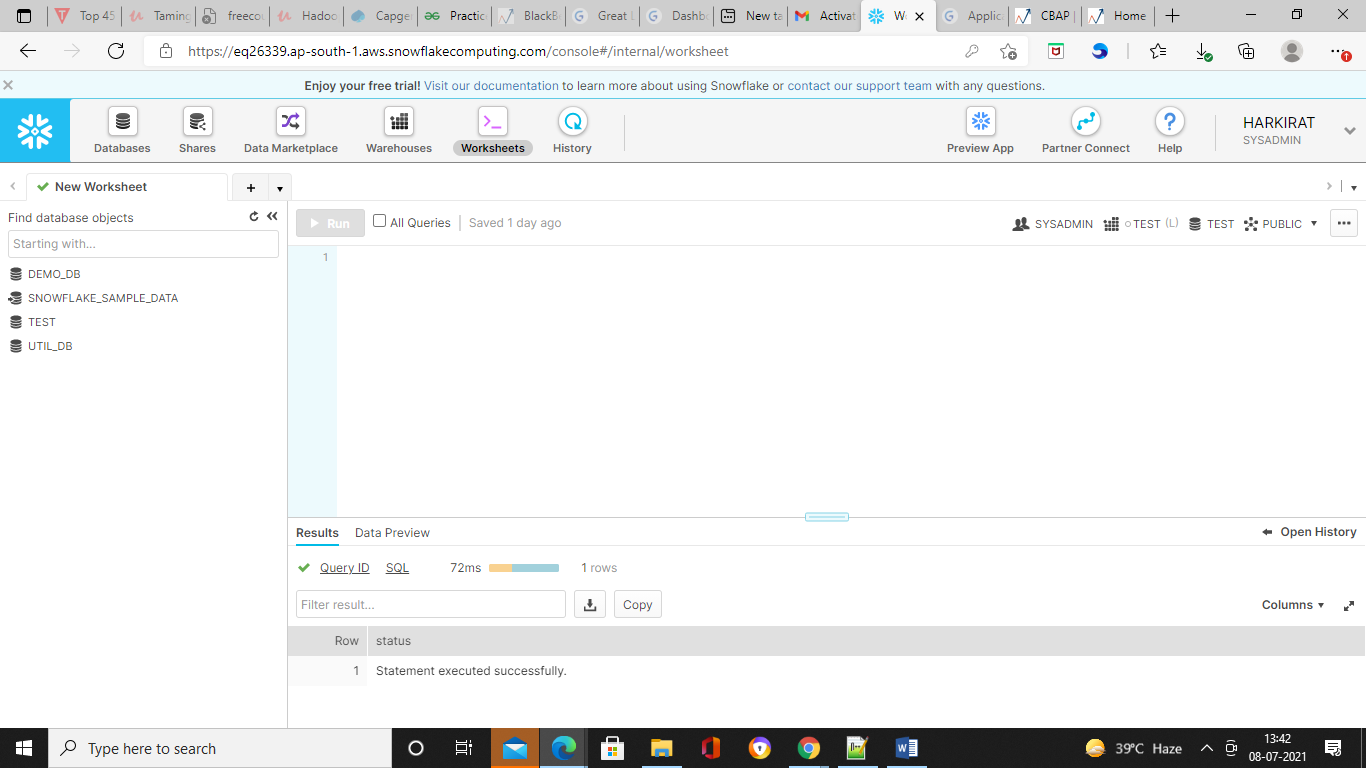
* Select the warehouse.
* Click drop
* A pop up message will appear. Click Yes.





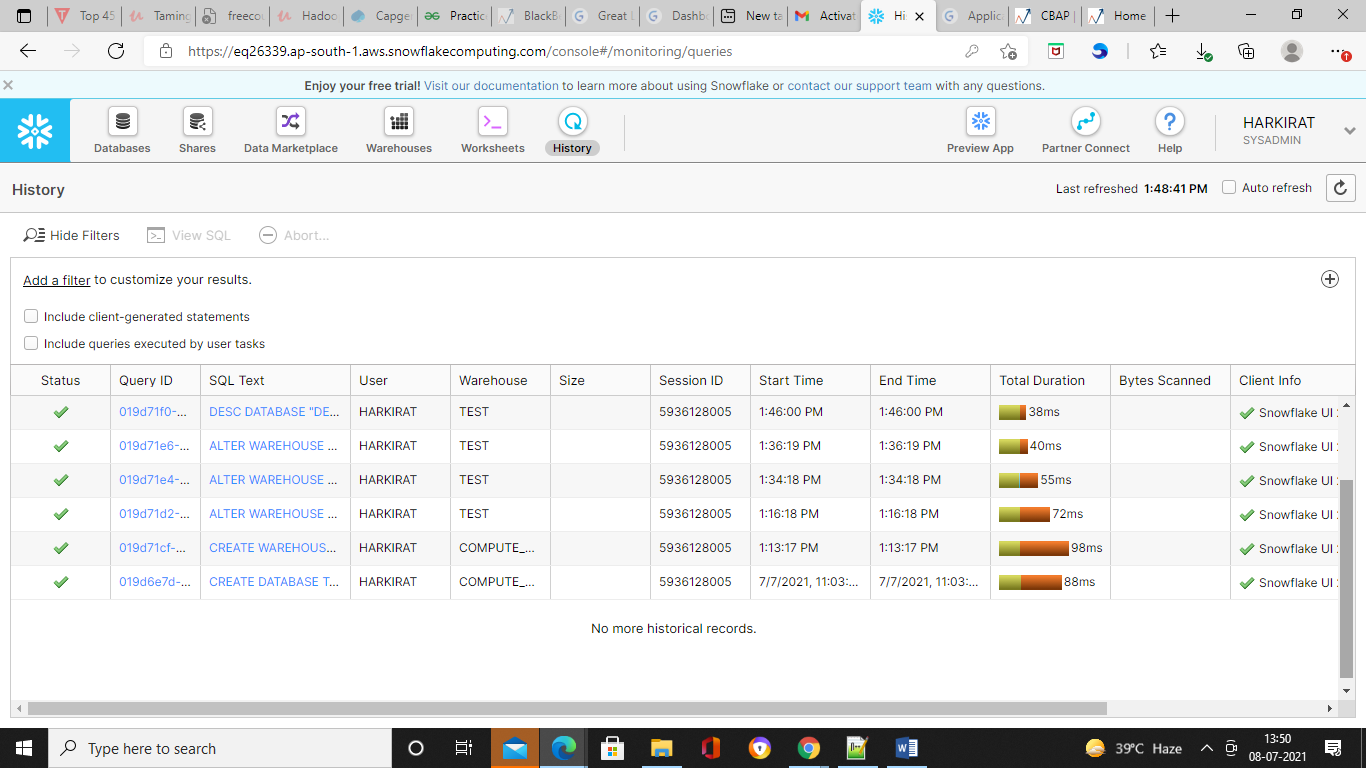
**Worksheet**

Worksheet is an actual SQL client where you can write query and execute it.

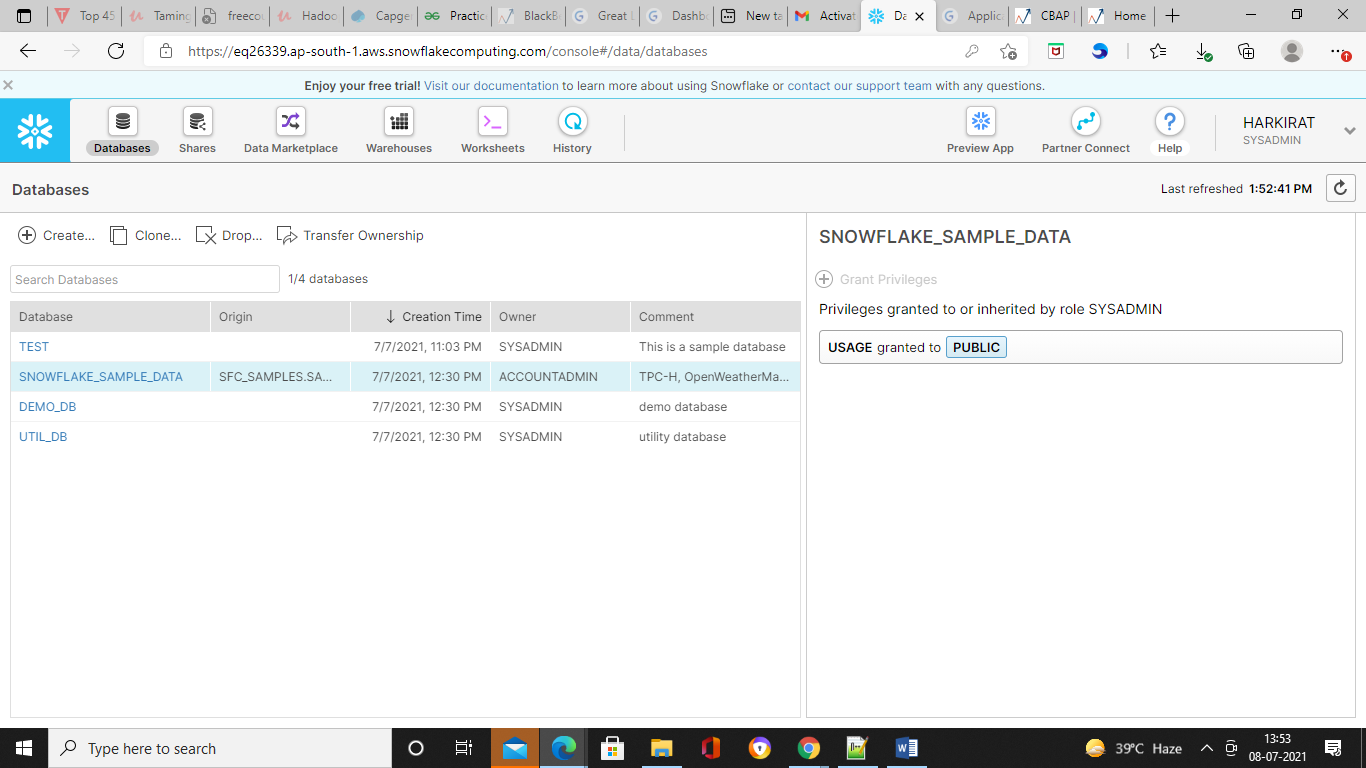


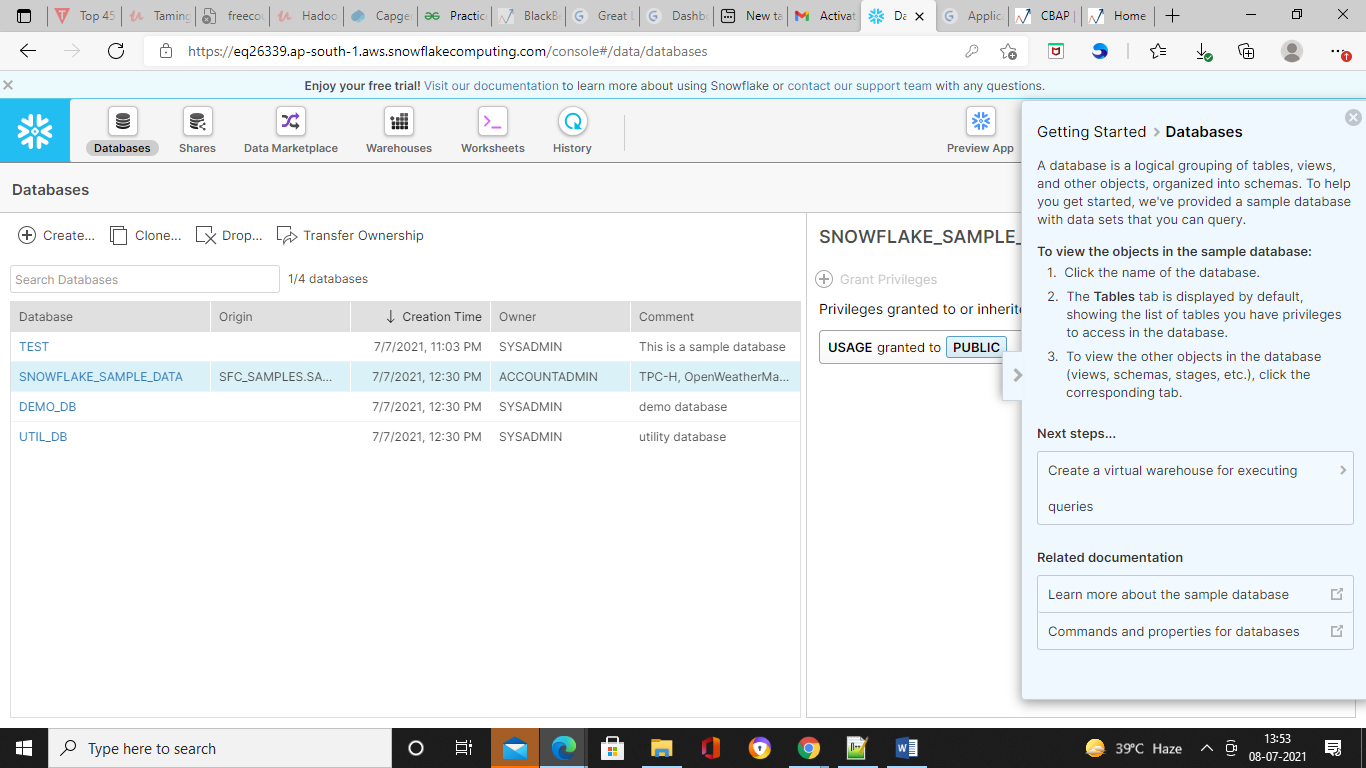
**History**

When we execute any query to create or change the database or warehouse, everything gets locked in the History section.

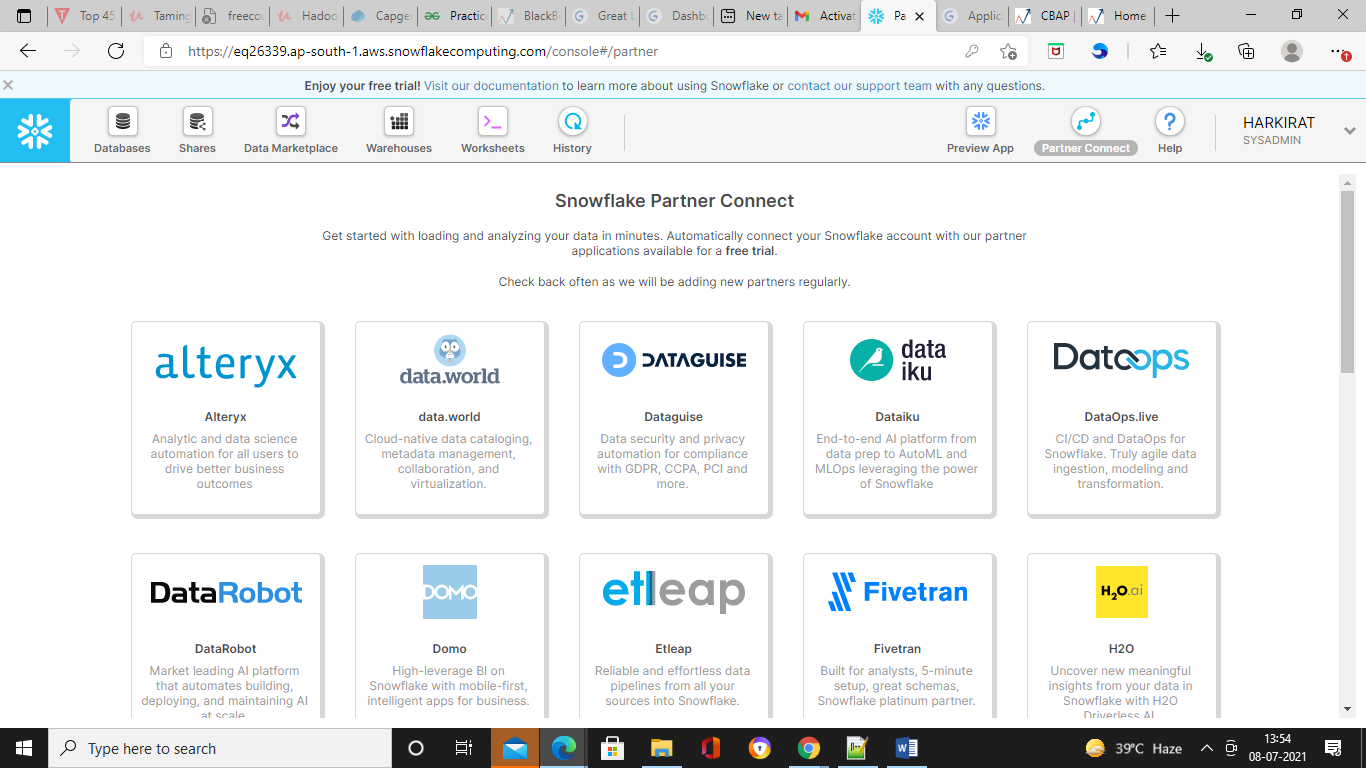


**Help**





**Partner connect**



**Why we need to create database and tables?**

As we know that snowflake is a data warehouse so it need to use the data from persistent layer and that persistent layer is nothing but our databases so these databases are no other like the relational database so we know the oracle or sql server. These are kind of relational databases. Snowflake do not use any underline databases, so it uses its own databases structure and for that reason we need to create a database and within that database we will have the schema and within that schema we can create tables. So once we create the tables we can load the data in those tables so that we can perform the computational operations.

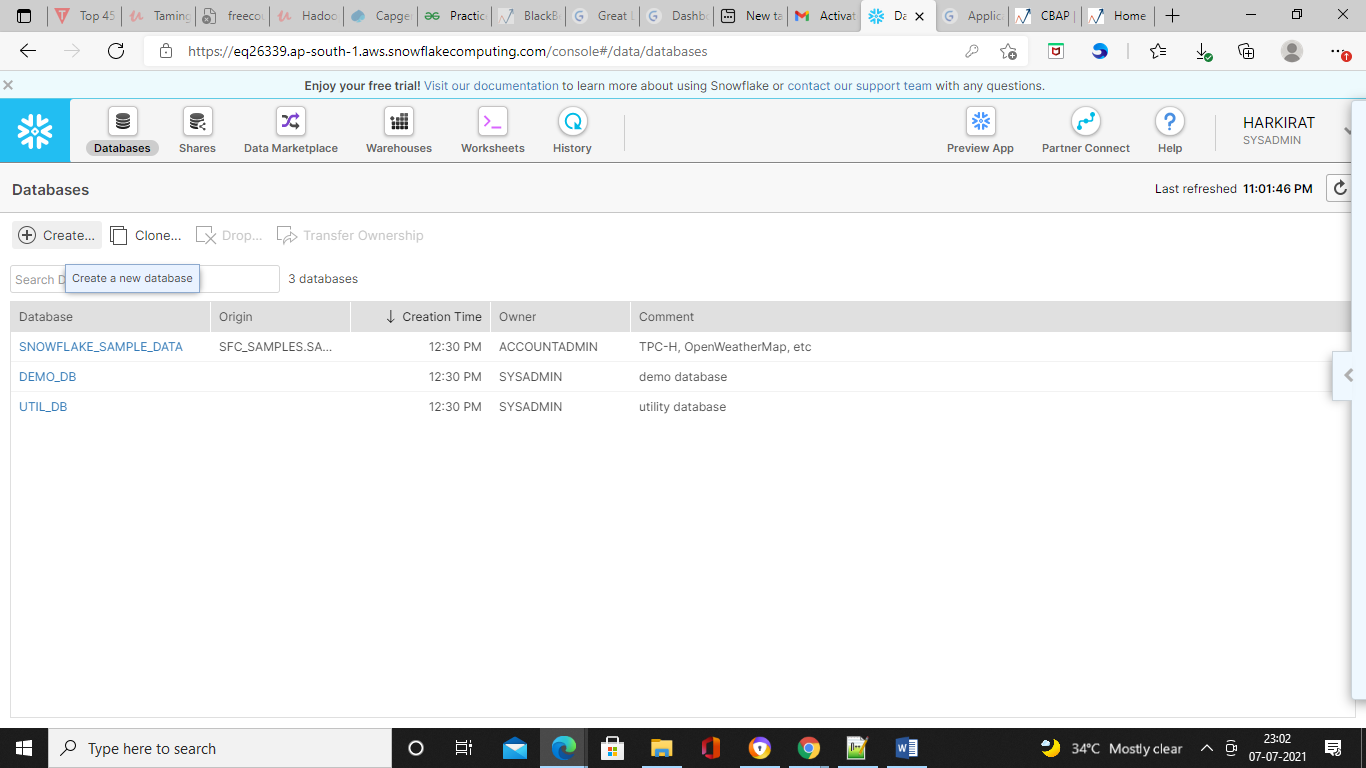
Steps involved during this data load process:

1. Creating a database and table
2. Create an external stage
3. Create file format for data
4. Load data

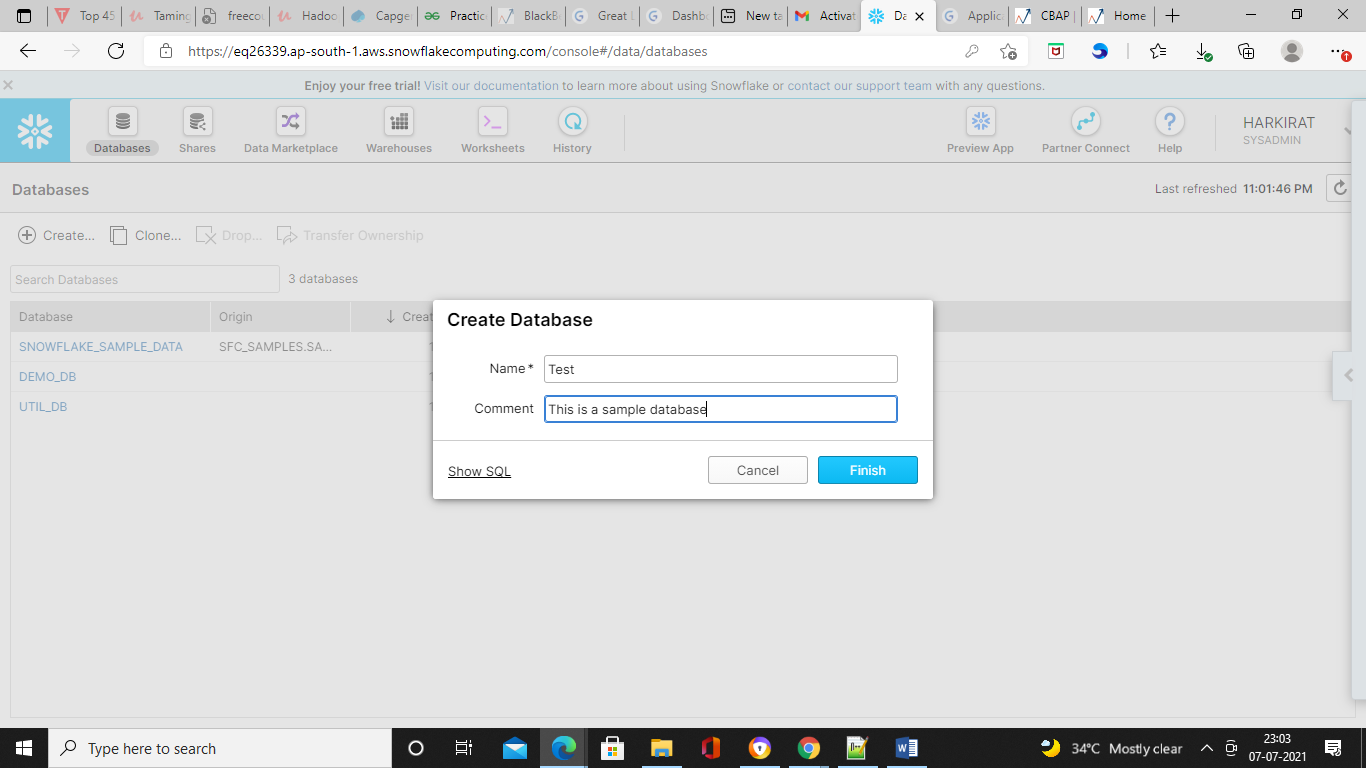
**Creating a database and table**

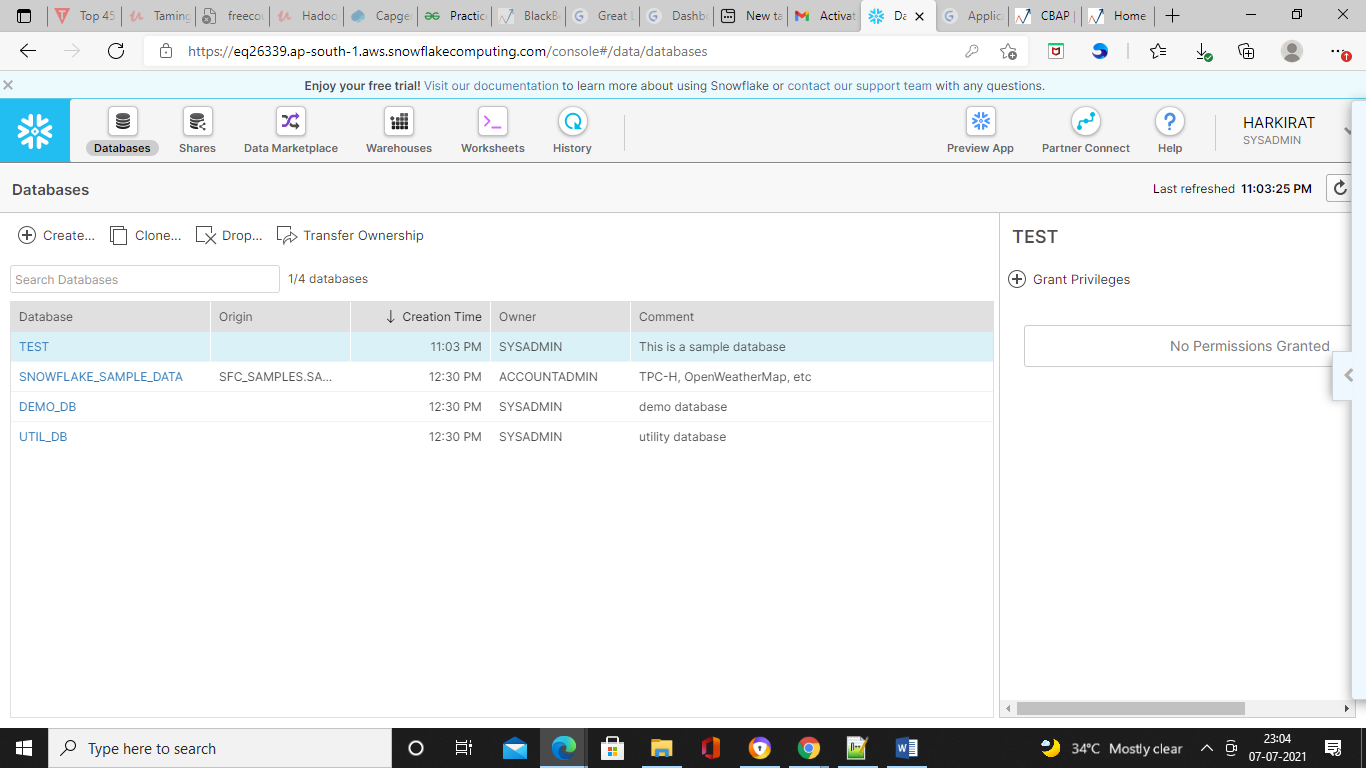
Creating a new database

1. Go to the databases.
2. You can create multiple databases. Click on Create option.

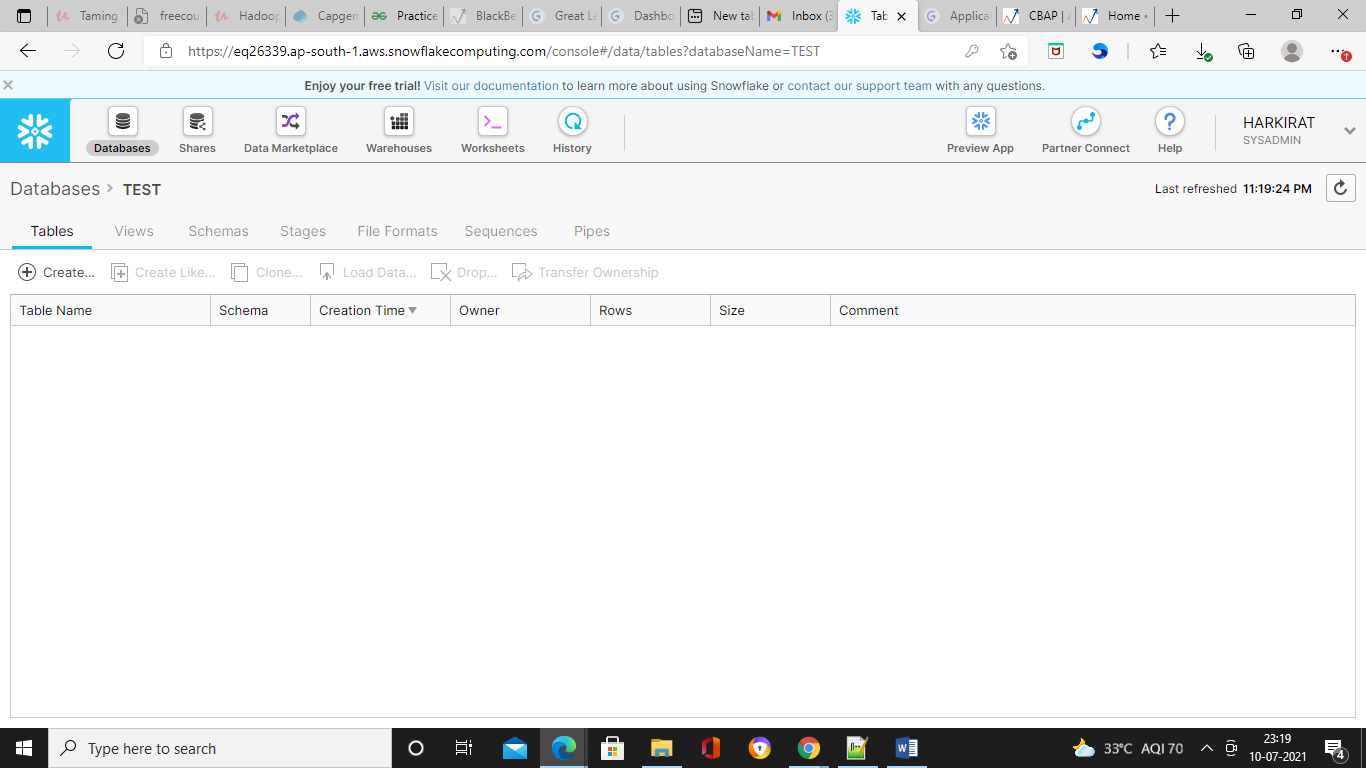


1. Provide the name of the database. We will use the name as Test. We can also provide the comment to the database.



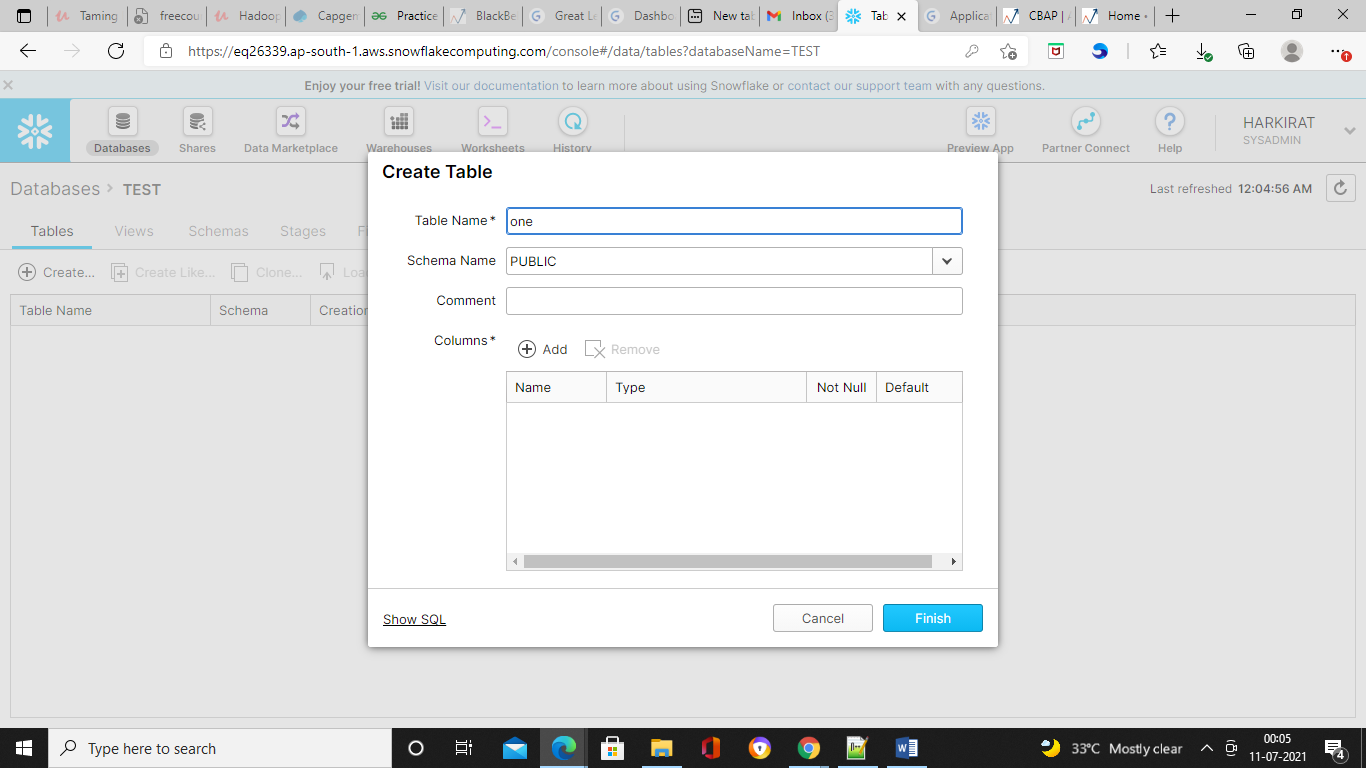


Here our database Test is created. Now the next thing to do is creating tables. For that we have to select the particular database.



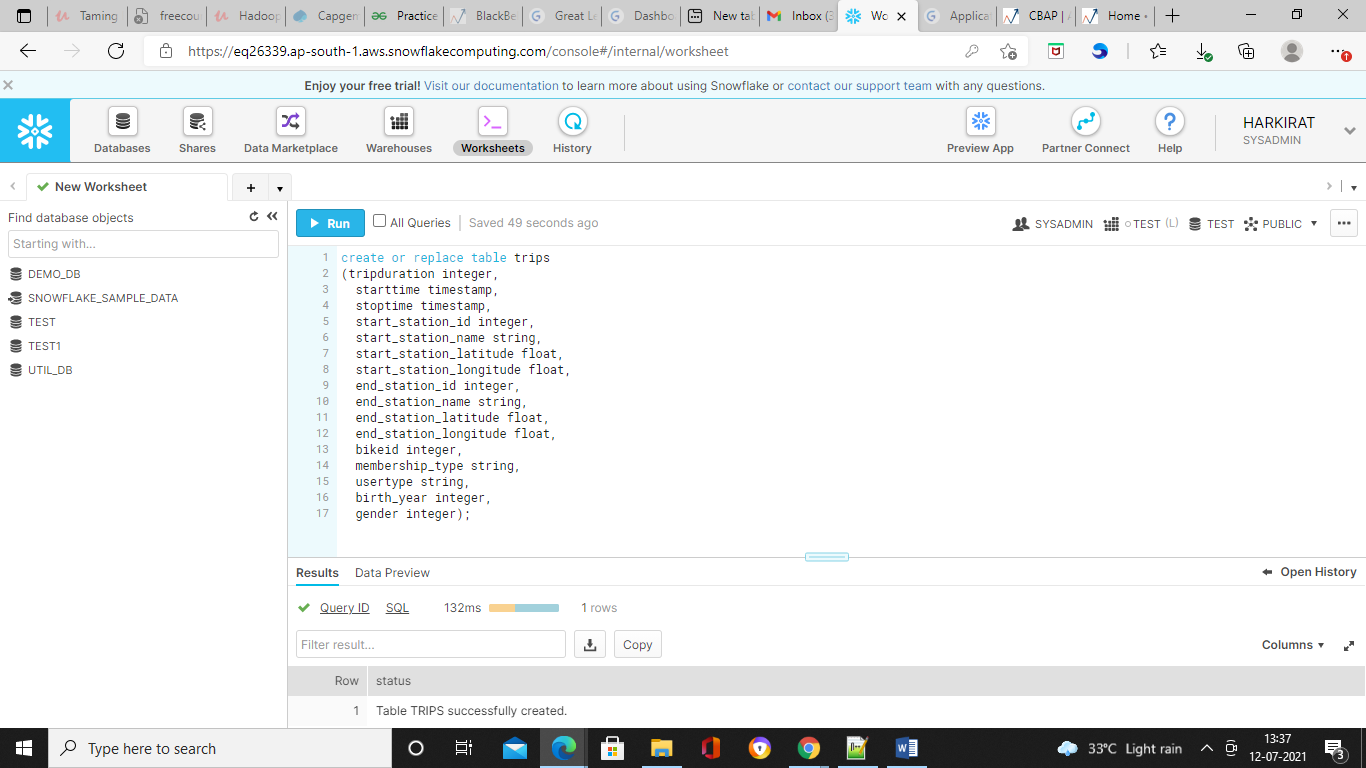
Creating a table

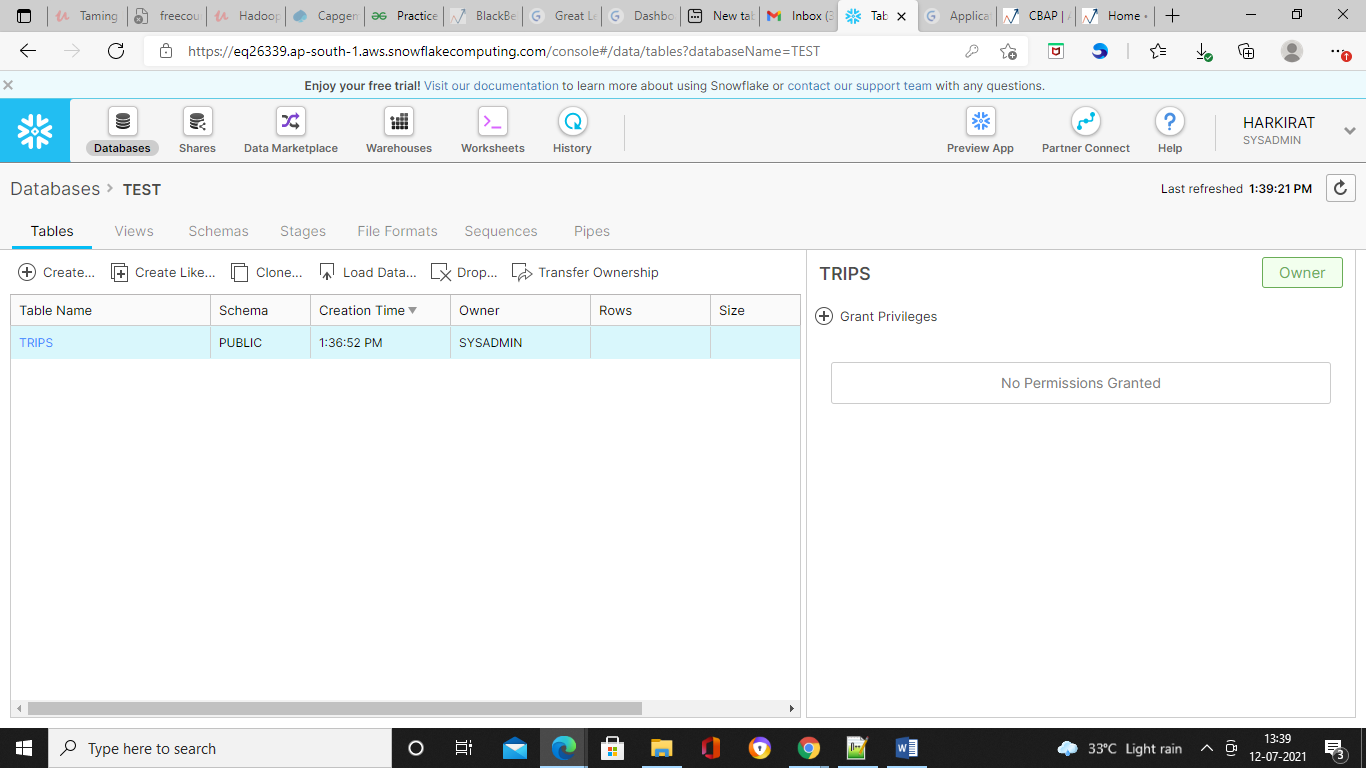
There are two ways to create the table. Either you can come to the user interface and click on create table option. Here you can provide the name of the table and you can select the appropriate schema and you can provide the comment and then add the column one by one.

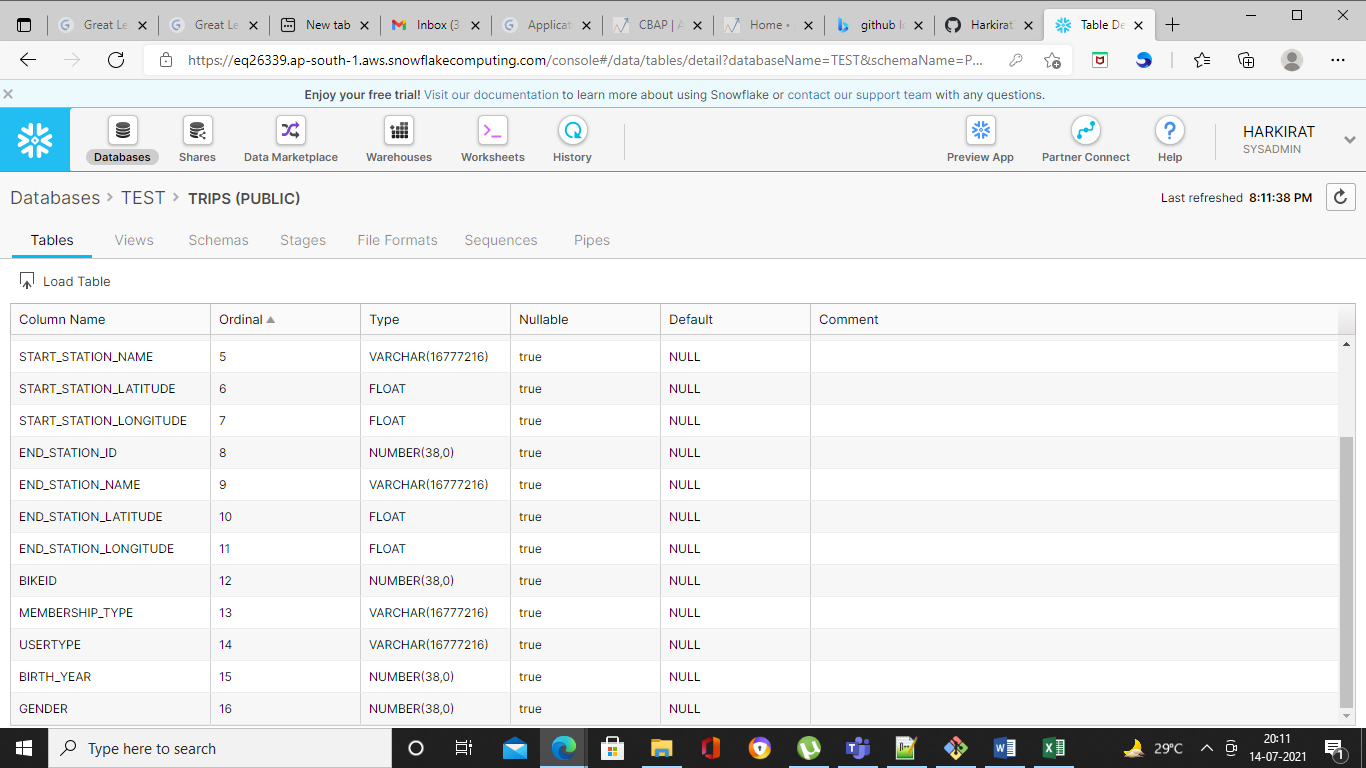


This is more tedious process. Normally in real time, this is going to take a long time if you have the multiple tables with multiple columns.

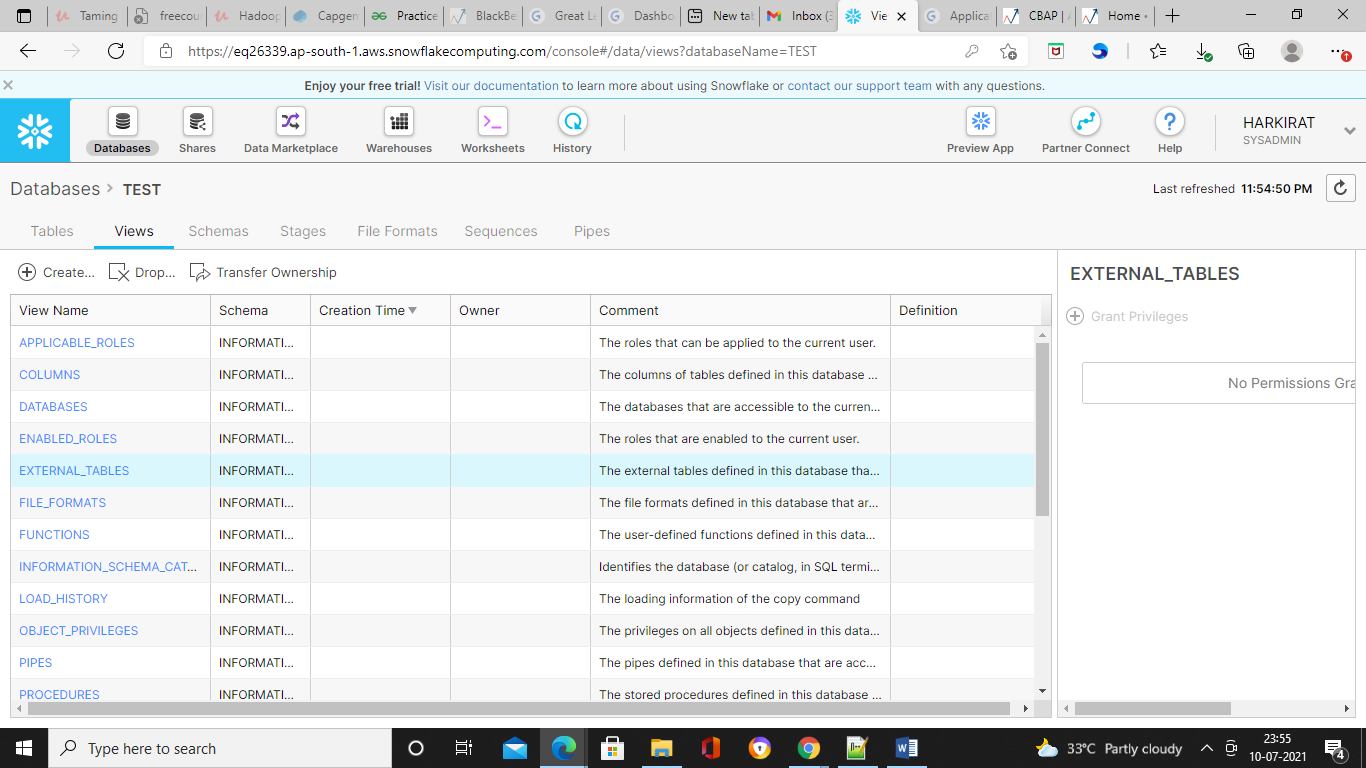
So the other option is you can use the worksheet to create the tables. You can just write your SQL statement and you can execute it. So it is same as the create table option which you can use in your relational database such as Oracle or Sql server. Now to write that query there is a particular syntax comes with the snowflake.



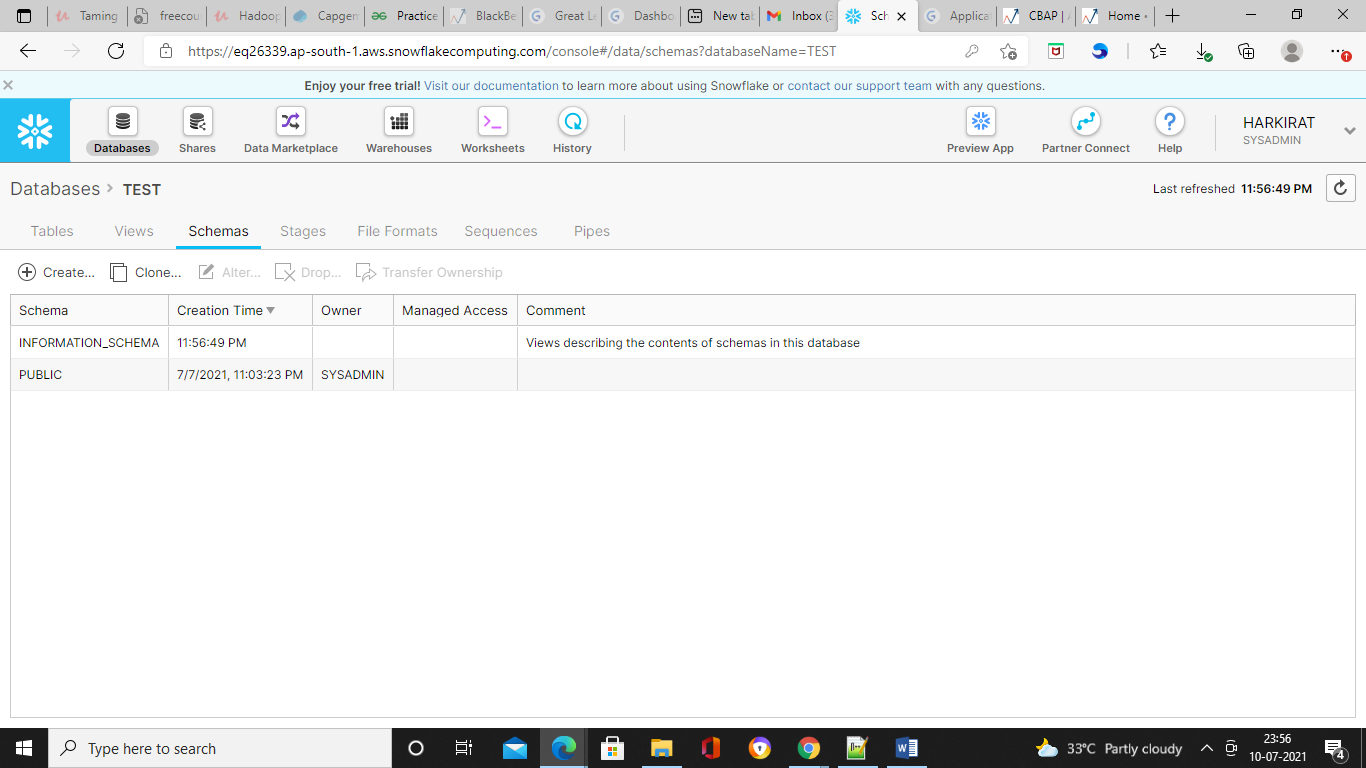




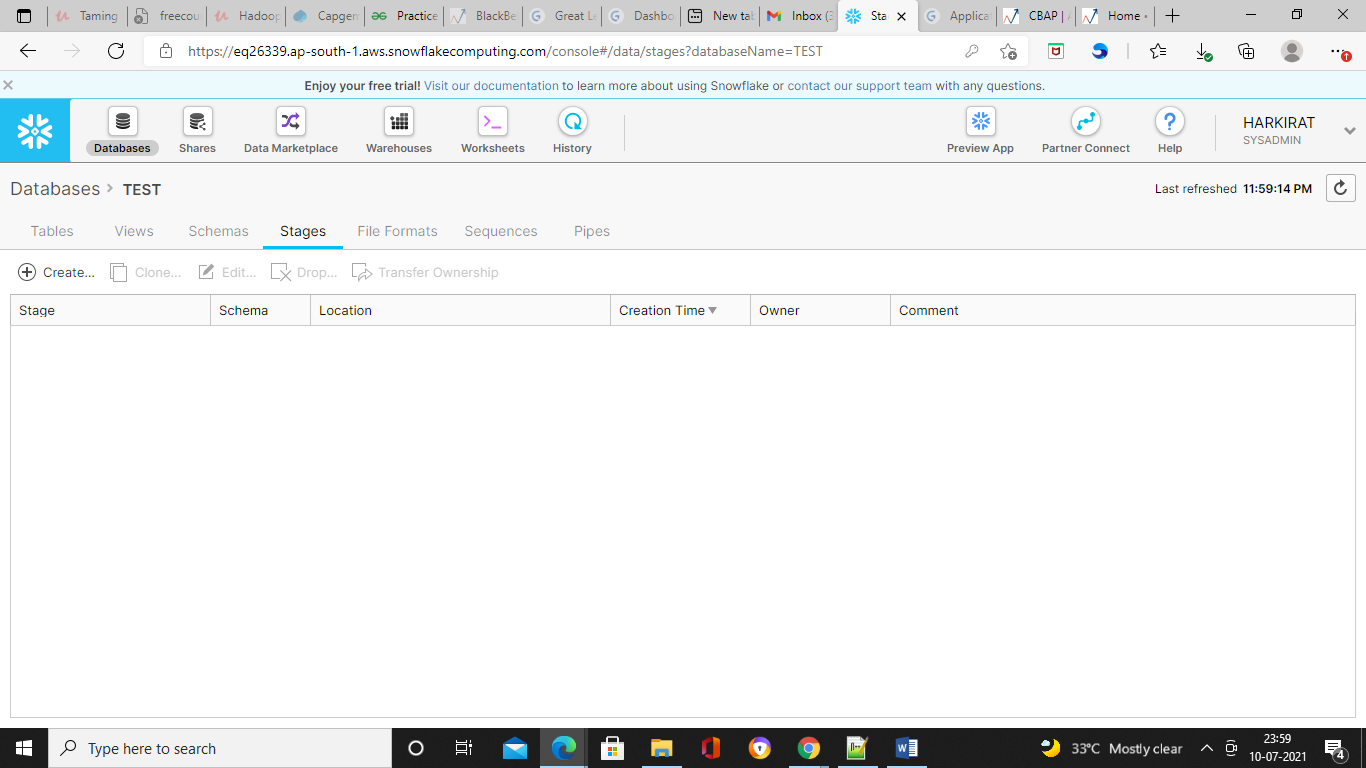
You can also clone the existing table if it is already there. Then you can load the data. You can drop the table, you can also transfer the ownership on that table. Apart from that there are several other tabs available. We have views tab to view all the existing tables.



We have the schemas option.



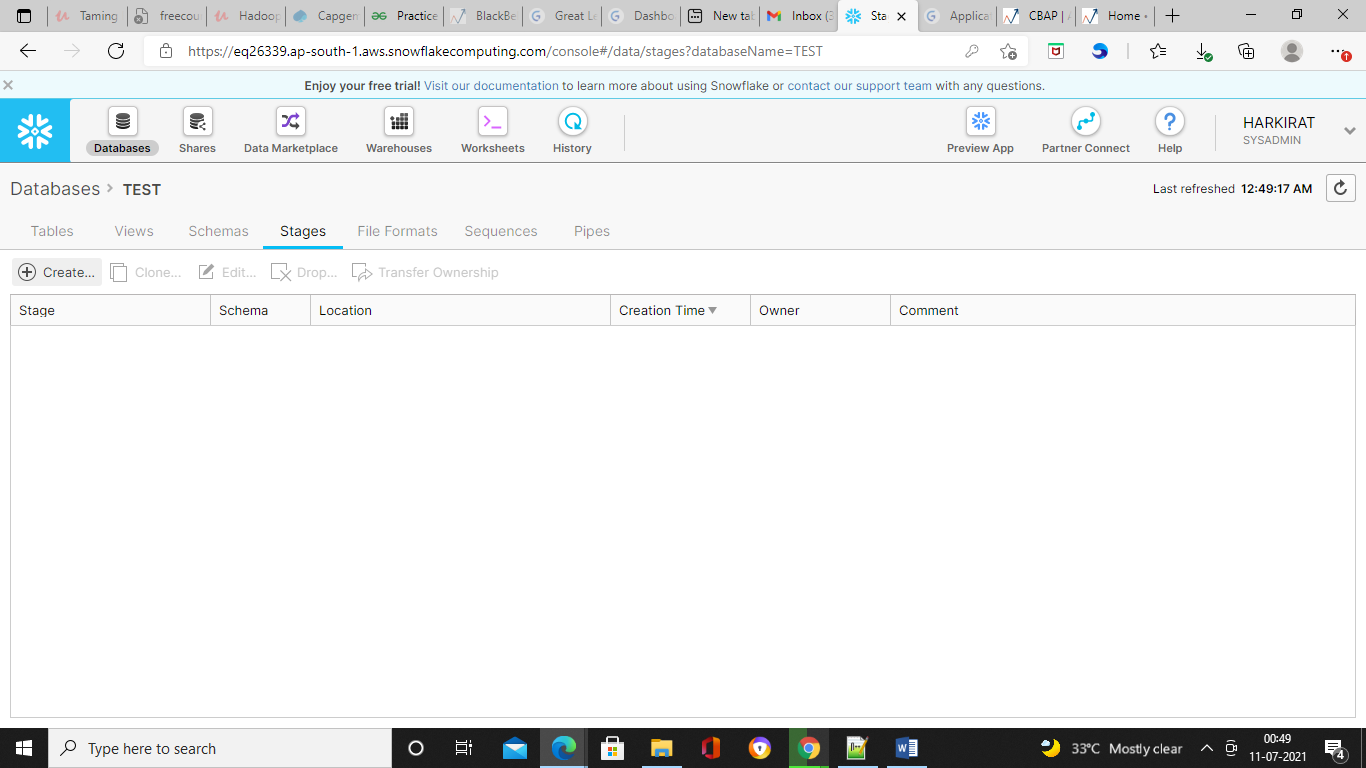
Then we have the Stages option.

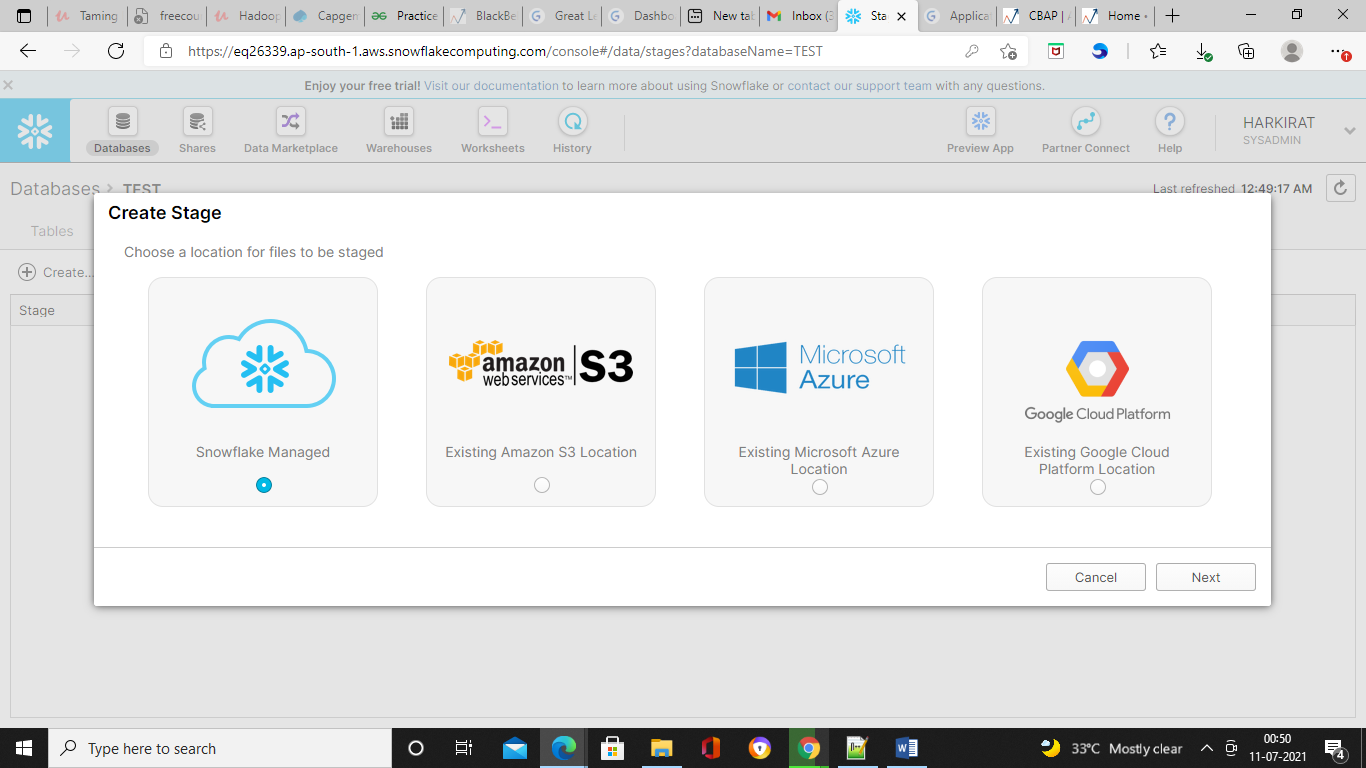


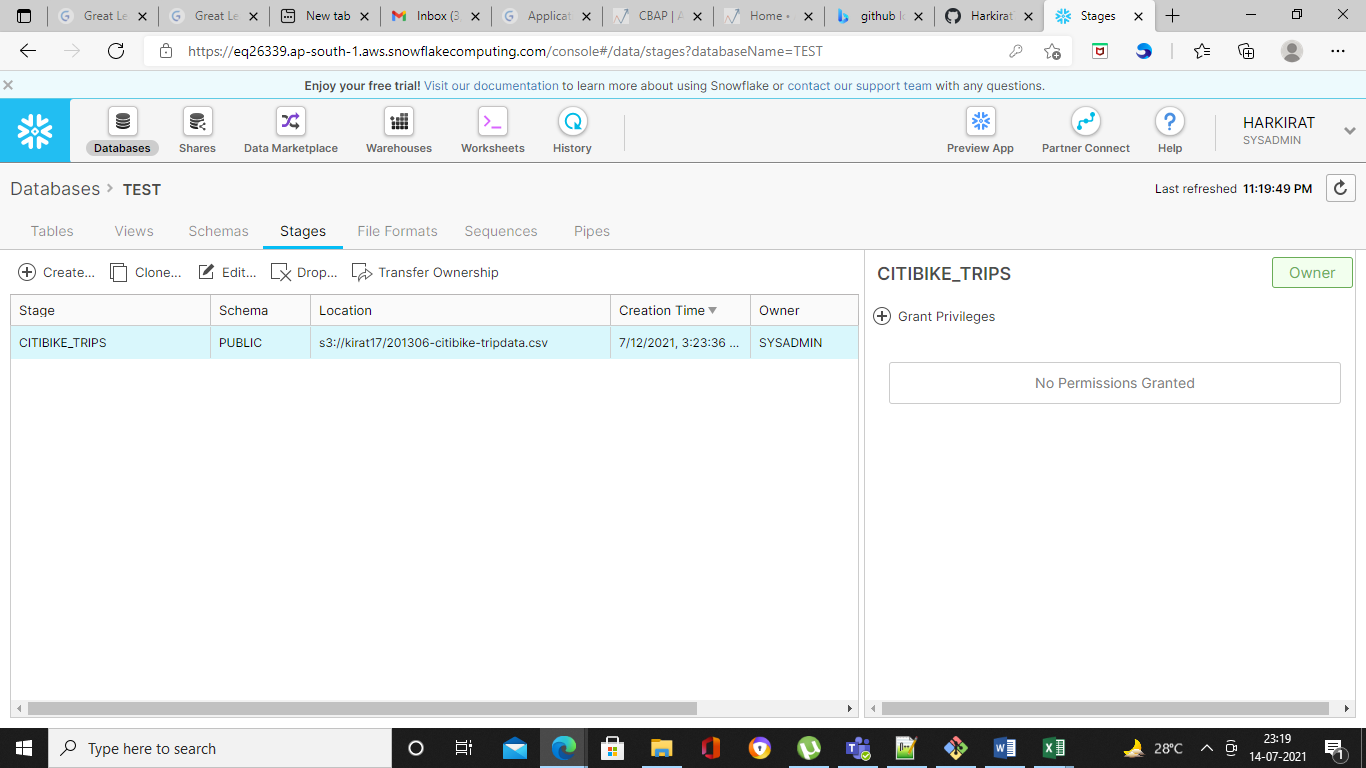
Then we have the file formats option, Sequences and Pipes option. If we have the continuously loading data then we have to use this pipes.

**Creating an external stage**

The stage is used to onboard the data in snowflake.



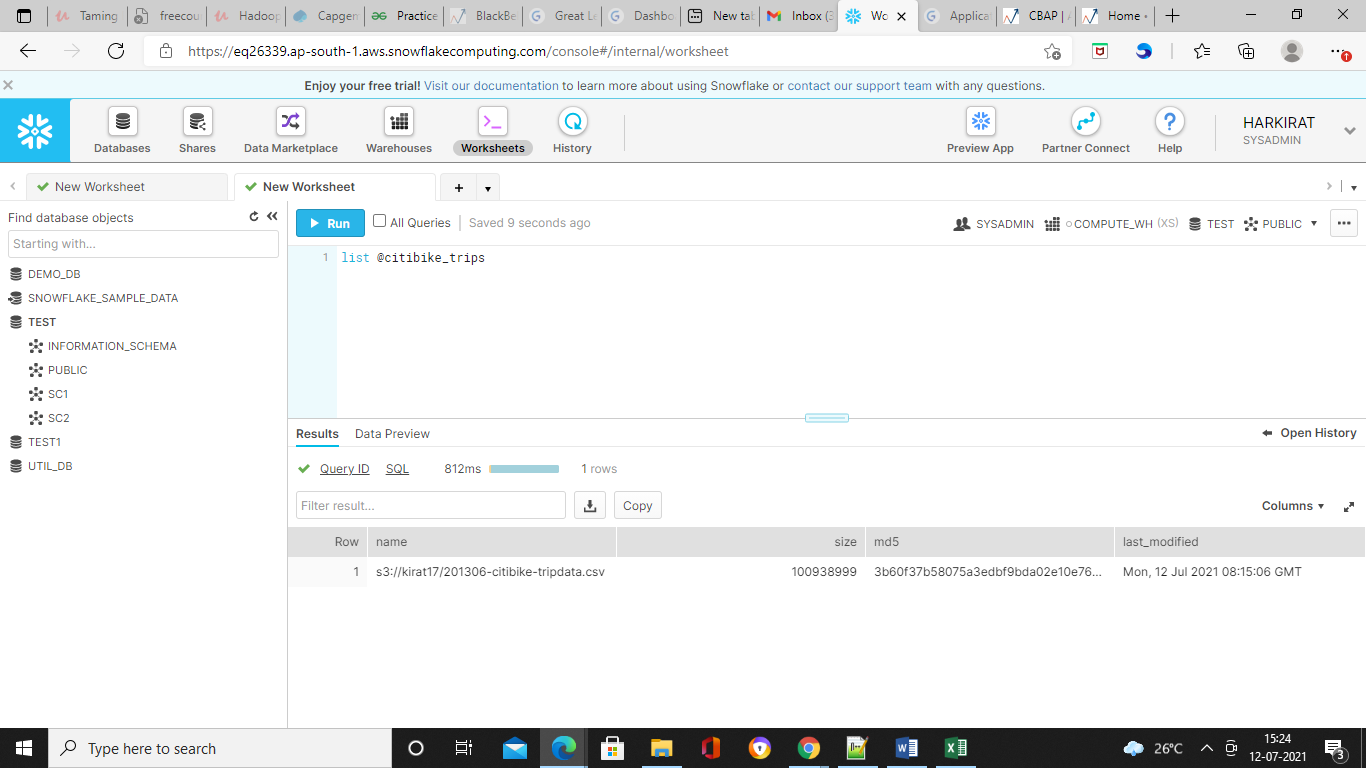




Stage connection is ready. Now to see all the information this stage contains, there is a command we can execute and verify what information does this stage contain. For that purpose, we will go to the worksheet. In order to verify we use list command.

list @citibike\_trips

here, citibike\_trips is the stage name.



This stage contains the Name, size, md5 and when it was last modified. So all these details tell us the s3 bucket is ready to use through the stage connection.

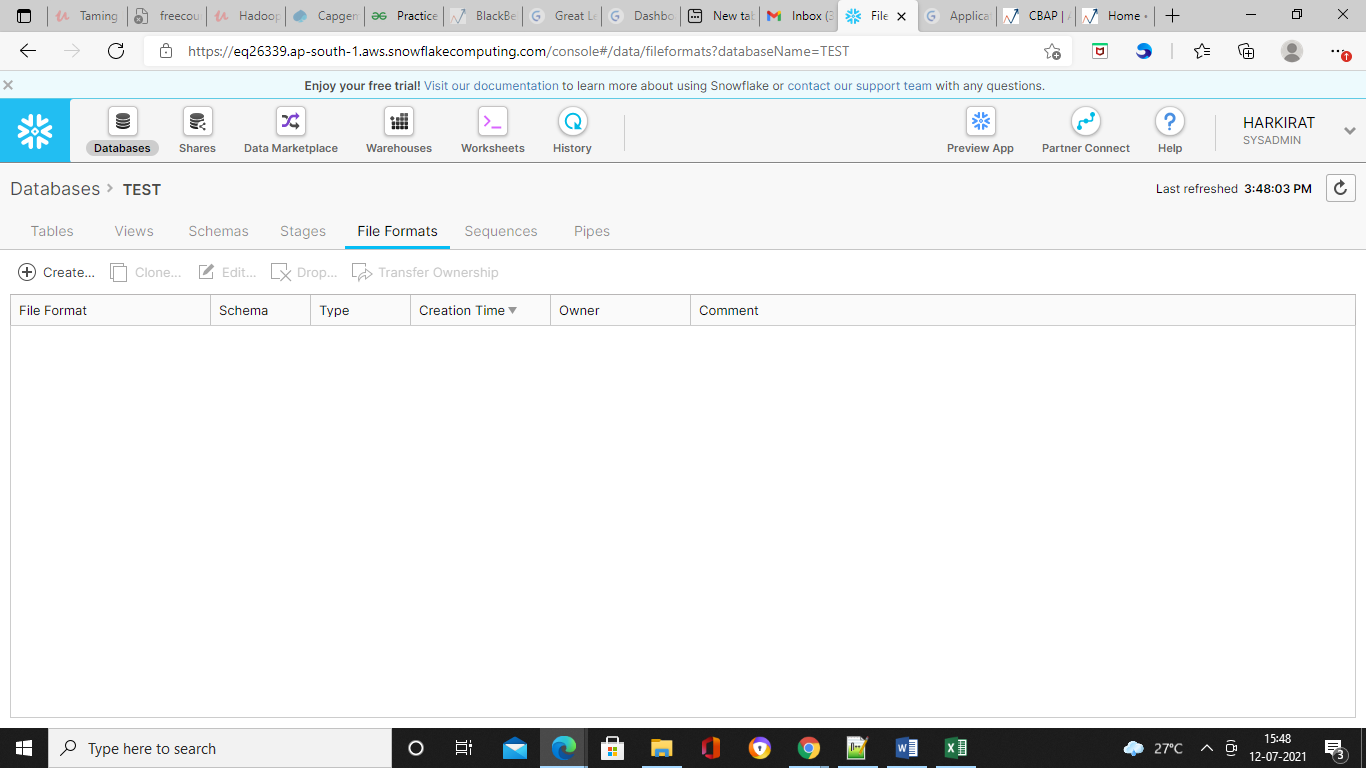
**Creating the file format for data**

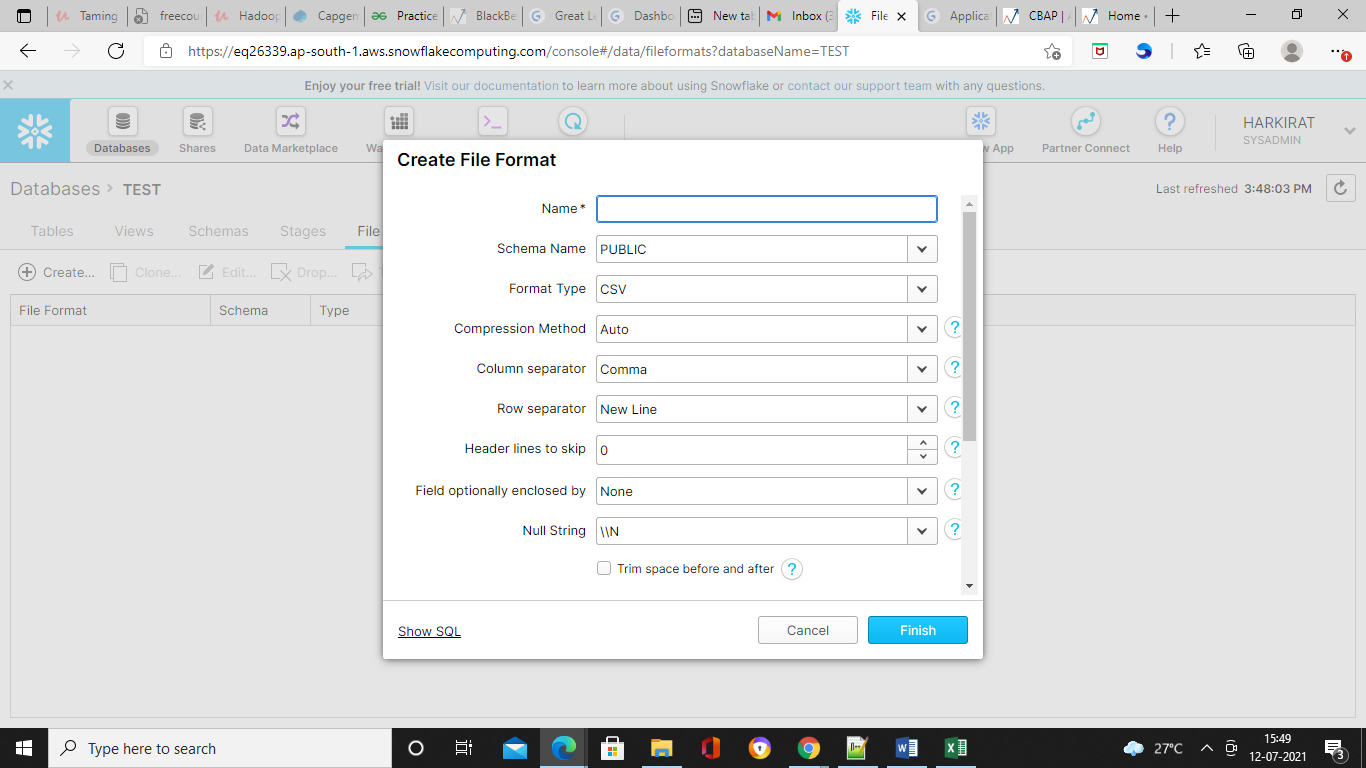
We loaded the csv file in staging platform but it will not be possible to load the data directly into the snowflake data layer without understanding the structure of the data and hence we next to create the

File format.

In our case we have a csv file so we will create a csv file format.

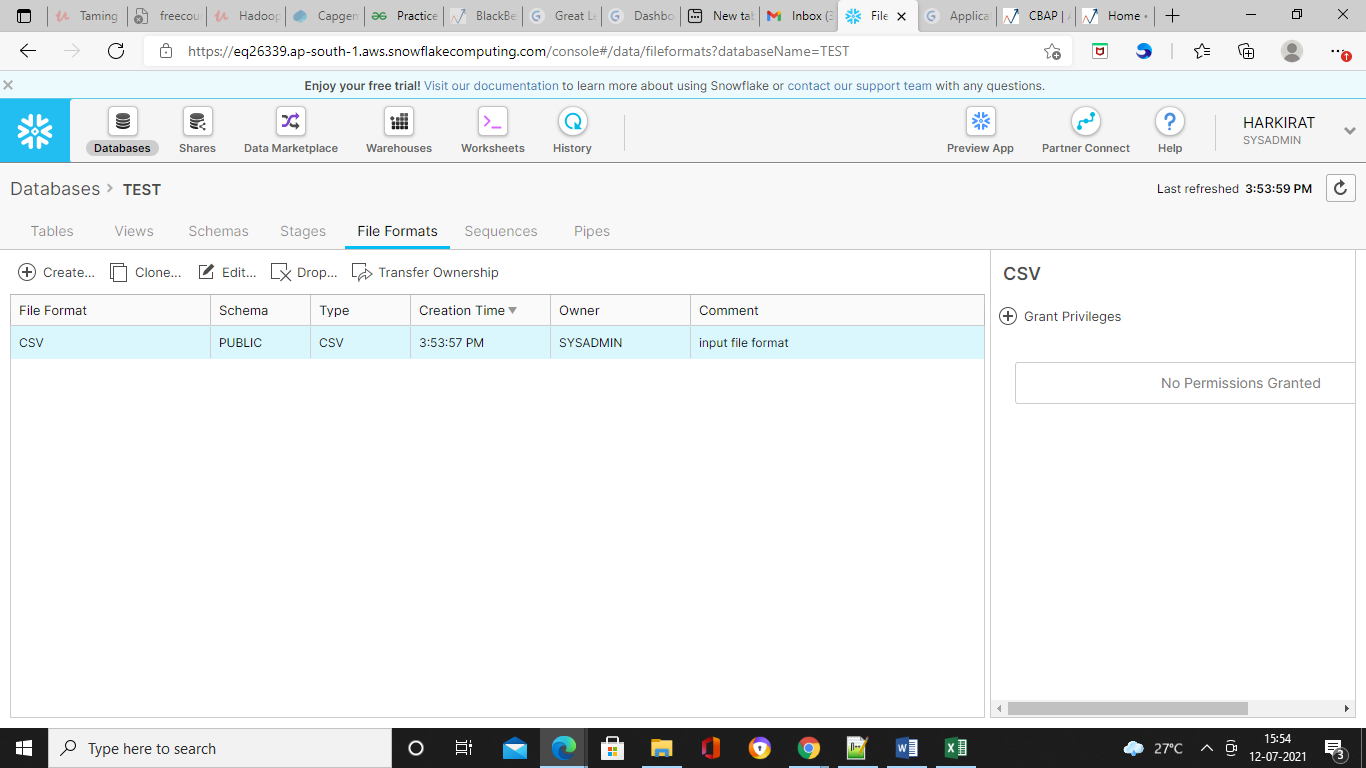
1. Go to databases and select the database for which you are going to create a file format. We are going to use test as a database.
2. Within the database, select the table which we have created.
3. Go to file formats.
4. You will see the multiple options. Select create option.





1. Set the parameters and click on finish.

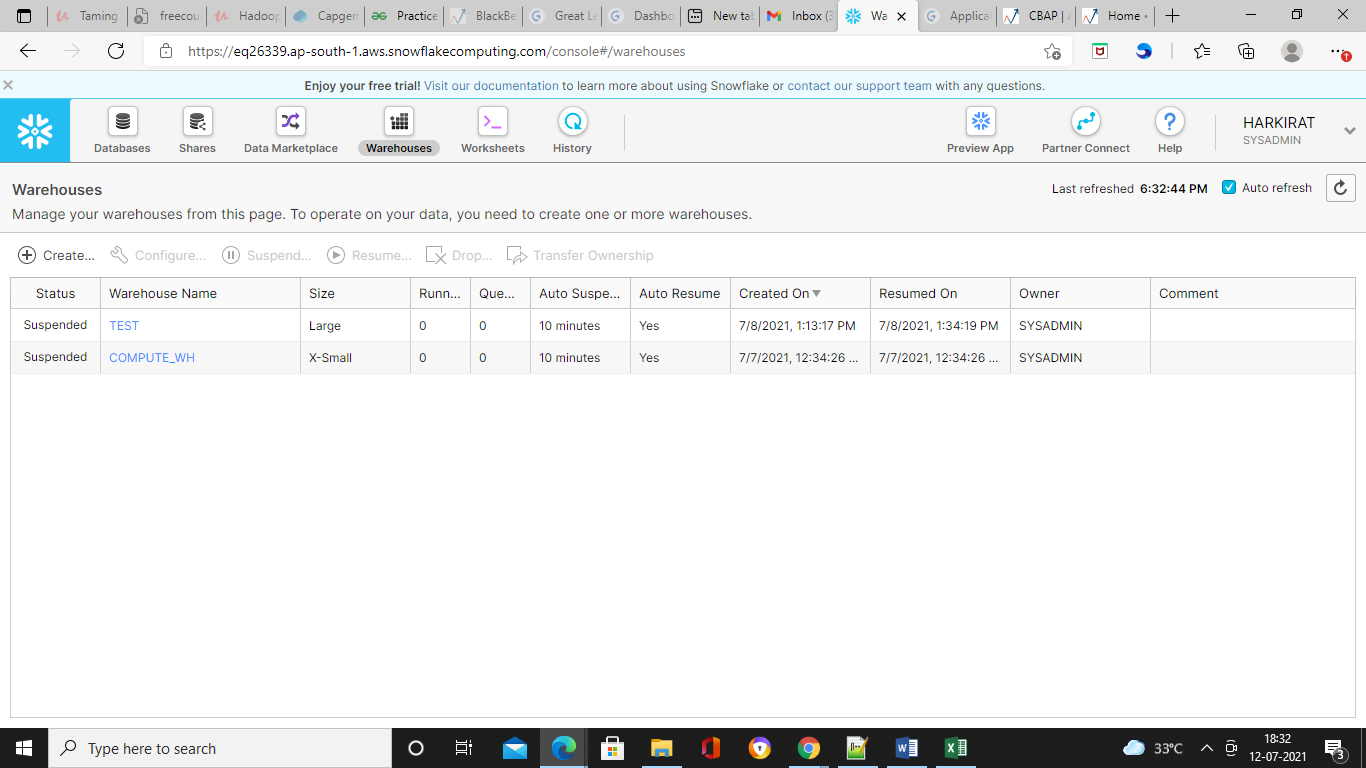
Here what we are trying to do is, when we load the data from our s3 bucket to our snowflake database the file format will be internally detected based on the what certificate file is present and when we actually start the process this file format we need to pass an input. So this is very important step for loading the data in the snowflake.



**Load data**

When we run the load script or there is a particular command we need to execute with which the data will be pulled from the s3 bucket, translated from the what file format we are going to use and will be loaded into the tables which will be created in the snowflake.

To start the actual load process, we need a processor or someone to handle the actual processing i.e. a node. And that node or process node is named as a warehouse.



In order to load the data into the snowflake then use the term into. And then after that we have to give the name of the table. In our case it’s trips. Now we give the table so we are inserting into or loading into the trips table but we need to provide the source and hence the from clause will come and then we have to provide the name of our staging. So if we go back to the databases and go to the stages the name of the stage is citibikes. Before that we will use symbol @

copy into trips from @citibike\_trips

file\_format=CSV;