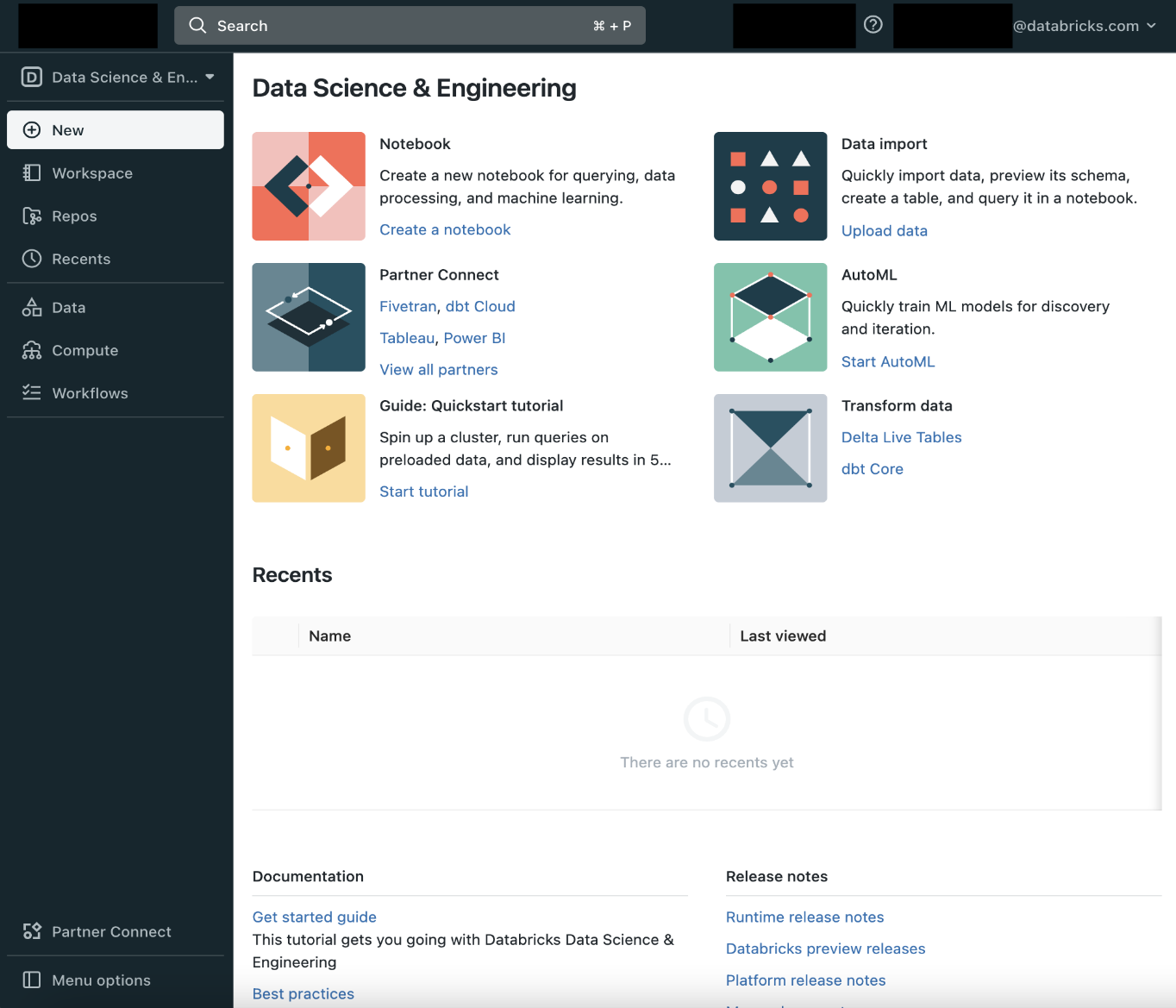
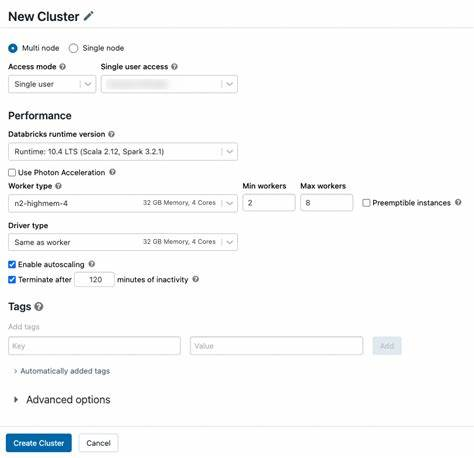
Databricks interface



When you click workspace there is shared and user's workspace

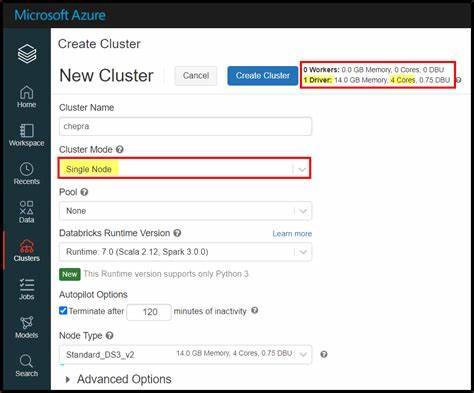
Create Cluster

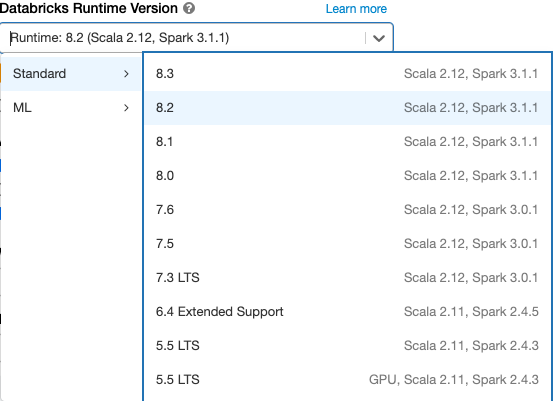
[Create a cluster - Azure Databricks | Microsoft Learn](https://learn.microsoft.com/en-us/azure/databricks/clusters/configure)



If you unclicked the autoscaling then there is no min and max workers node and the typical use cases for this is streaming workload in which the time it takes to scaling up a cluster might not be acceptable for the real time workload

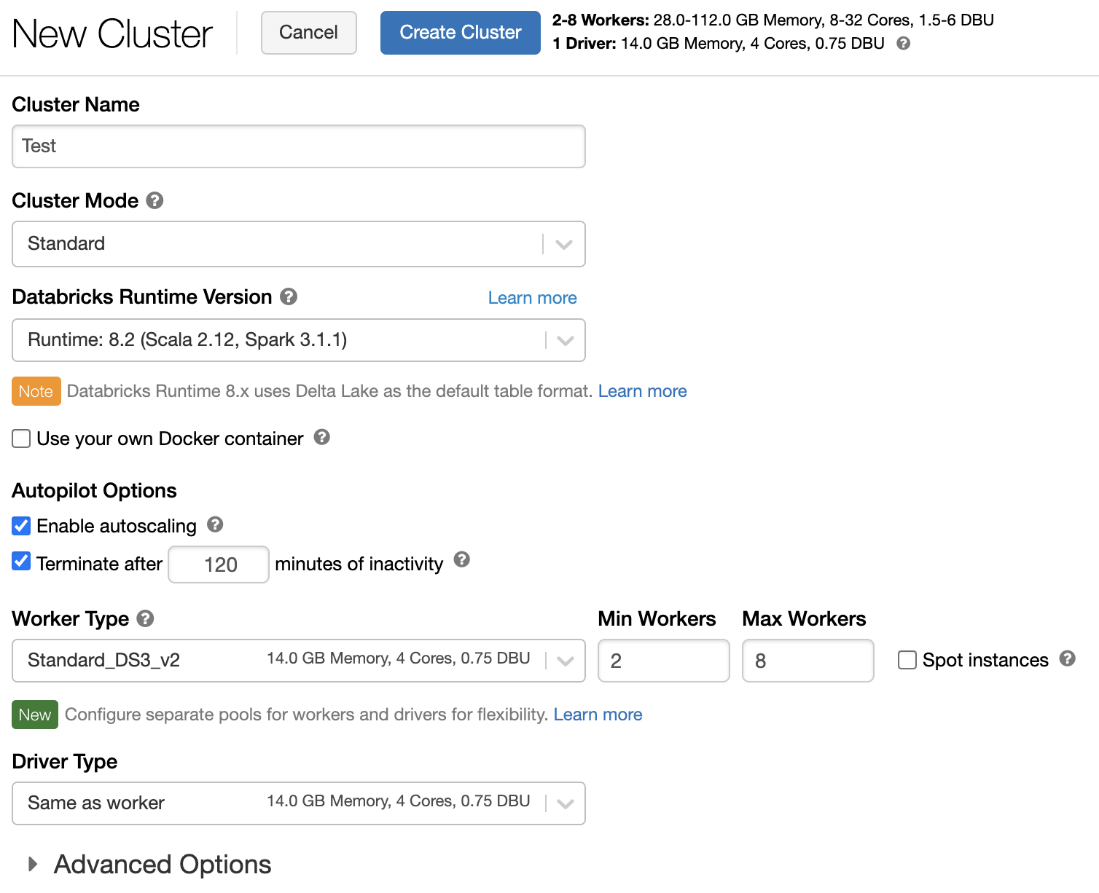
Single node Cluster



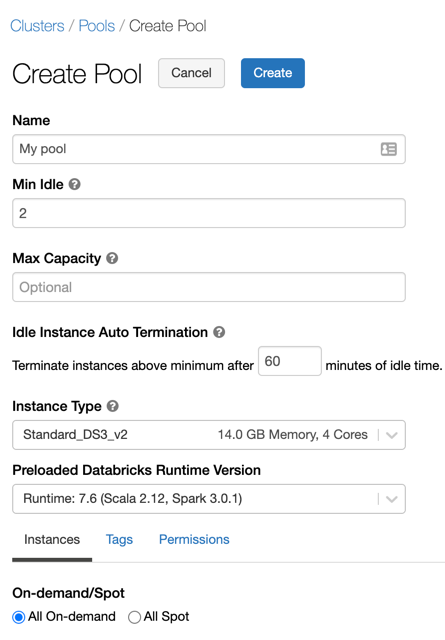


The above image is new UI and below image is old UI

[Configure clusters - Azure Databricks | Microsoft Learn](https://learn.microsoft.com/en-us/azure/databricks/archive/compute/configure)



Cluster pool



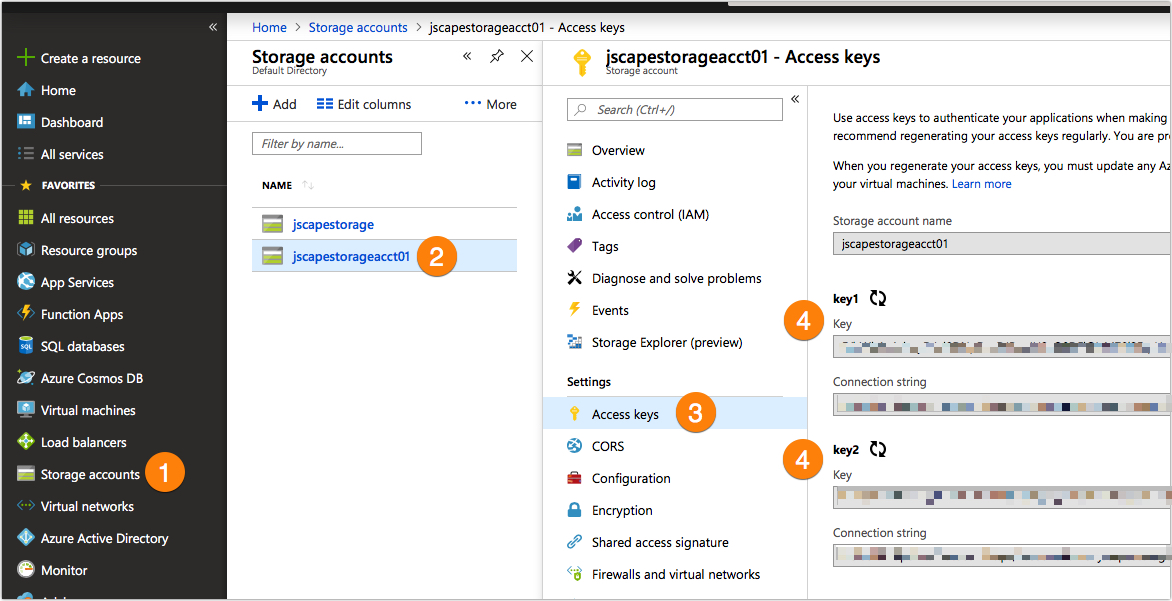
Access Azure Data Lake through access key

Spark.conf.set(“fs.azure.account.key.<storage-account>.dfs.core.windows.net”, “<access key>”)

To access the data/file from azure data lake storage gen 2

d[butils.fs.ls(“abfss://demo@formula1dl.dfs.core.windows.net/”)=example](mailto:Dbutils.fs.ls()

abfss://container@storageAccount.dfs.core.windows.net/folder\_path/file\_name



Access Azure Data lake through SAS Token

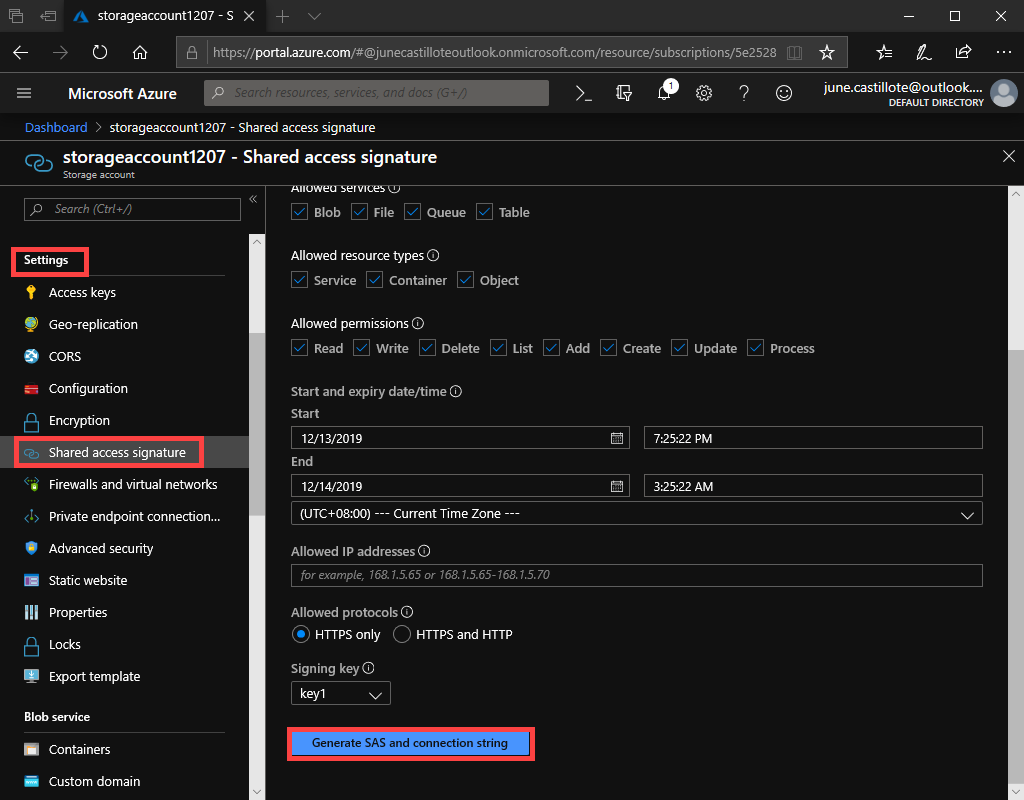
spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "SAS")

spark.conf.set("fs.azure.sas.token.provider.type.<storage-account>.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.sas.FixedSASTokenProvider")

spark.conf.set("fs.azure.sas.fixed.token.<storage-account>.dfs.core.windows.net", "<token>")

To access the data/file from azure data lake storage gen 2

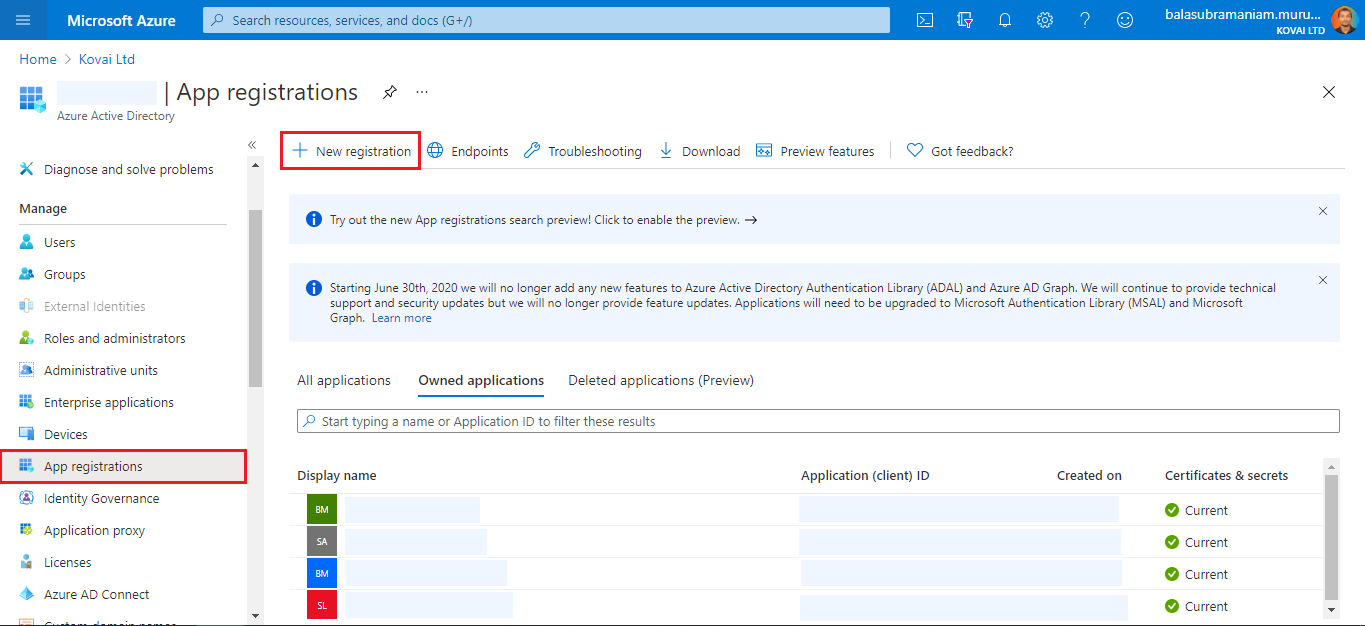
abfss://container@storageAccount.dfs.core.windows.net/folder\_path/file\_name

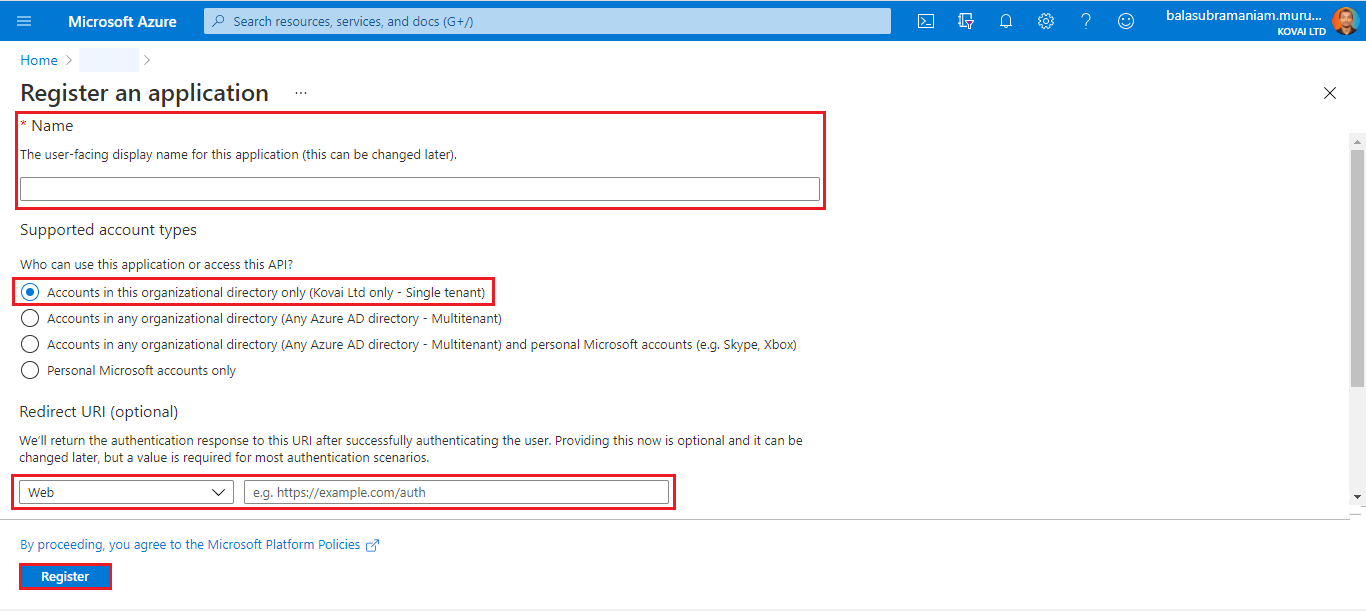


Access Azure Data lake through Service Principal

## Create an Azure service principal

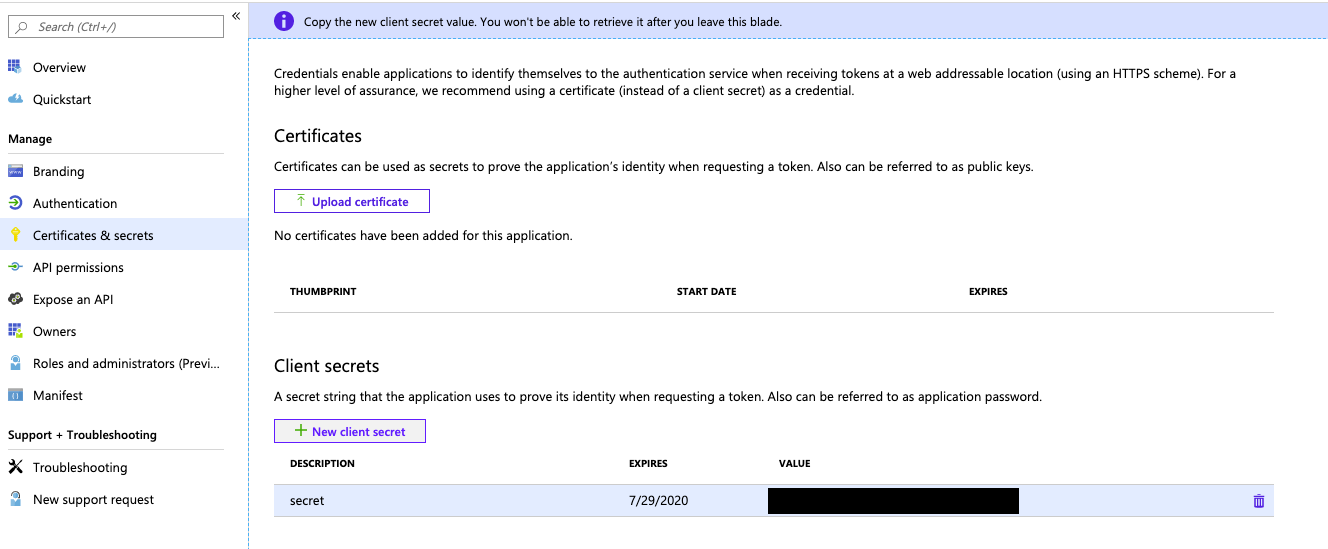
1. Sign in to the Azure portal.
2. If you have access to multiple tenants, subscriptions, or directories, click the **Directories + subscriptions** (directory with filter) icon in the top menu to switch to the directory in which you want to provision the service principal.
3. Search for and select **Azure Active Directory**.
4. In **Manage**, click **App registrations > New registration**.
5. For **Name**, enter a name for the application.
6. In the **Supported account types** section, select **Accounts in this organizational directory only (Single tenant)**.
7. Click **Register**.





## Create a client secret for your service principal

1. In **Manage**, click **Certificates & secrets**.
2. On the **Client secrets** tab, click **New client secret**. Give description and expires

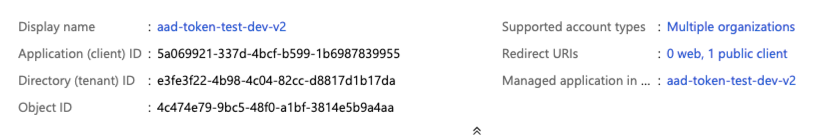


1. Copy and store the client secret’s **Value** in a secure place, as this client secret is the password for your application.

On the application page’s **Overview** page, in the **Essentials** section, copy the following values:

* **Application (client) ID**
* **Directory (tenant) ID**

Example



## Connect to Azure Data Lake Storage Gen2 using python

spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account>.dfs.core.windows.net", "client\_id")

spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account>.dfs.core.windows.net", client\_secret)

spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-account>.dfs.core.windows.net",f“https://login.microsoftonline.com/{tenant\_id}/oauth2/token”)

## Grant the service principal access to Azure Data Lake Storage Gen2

1. In the Azure portal, go to the **Storage accounts** service.
2. Select an Azure storage account to use.
3. Click **Access Control (IAM)**.
4. Click **+ Add** and select **Add role assignment** from the dropdown menu.
5. Set the **Select** field to the Azure AD application name that you created in step 1 and set **Role** to **Storage Blob Data Contributor**.

When you select a member you have to search the name which you give during app registration

1. Click **Save**

To access the data/file from azure data lake storage gen 2

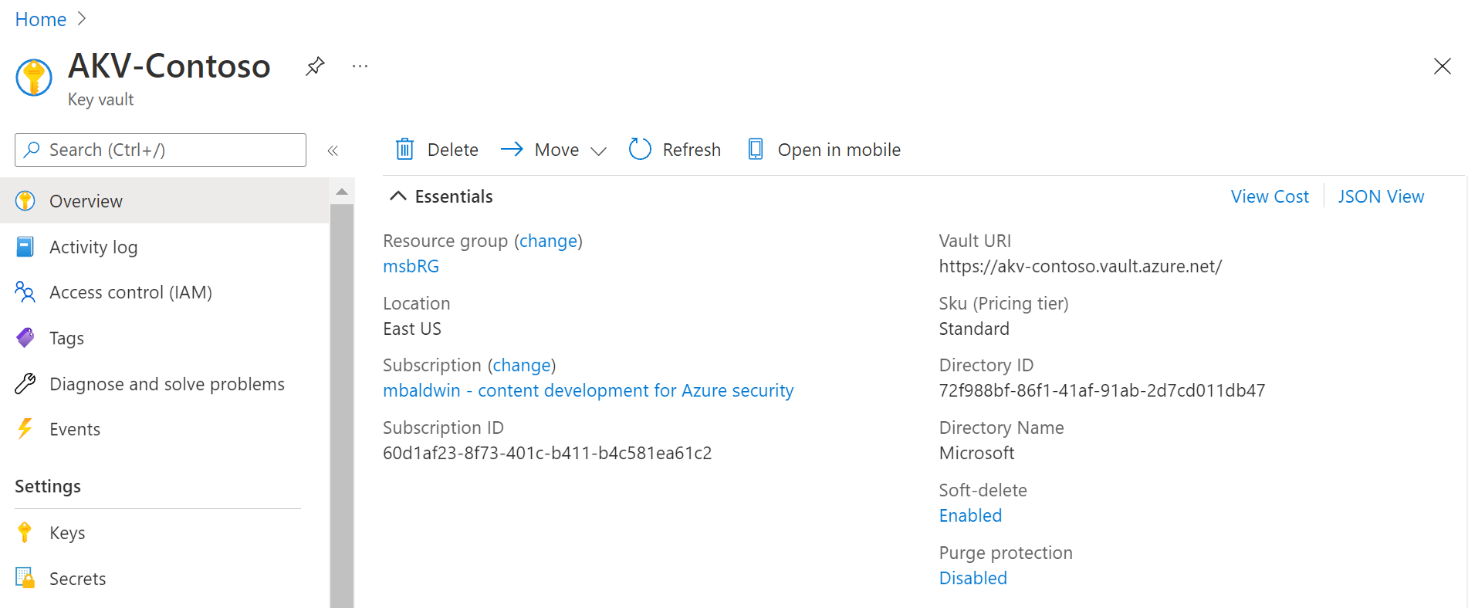
abfss://container@storageAccount.dfs.core.windows.net/folder\_path/file\_name

Mounting Datalake using secret

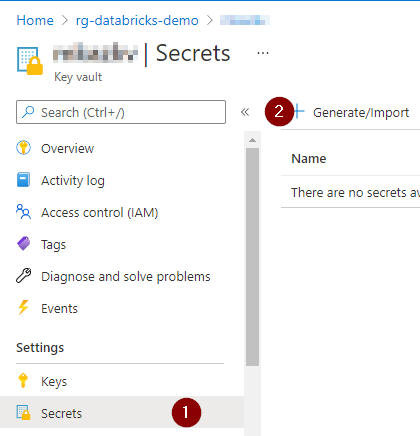
Create Azure Key vault

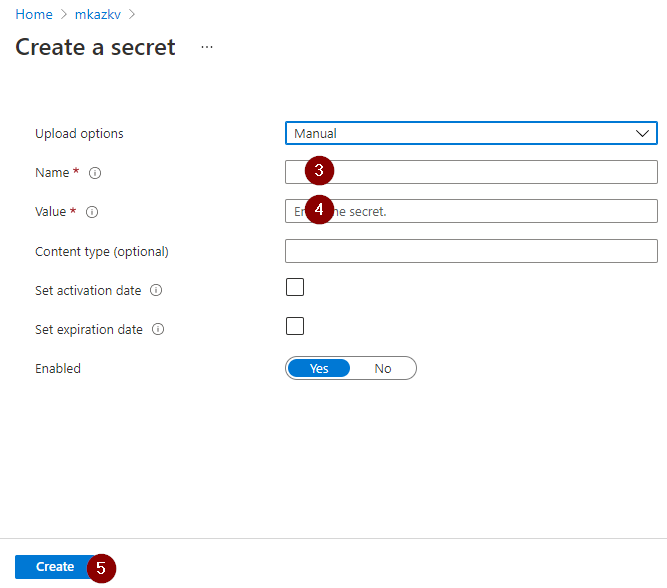
1. From the Azure portal menu, or from the **Home** page, select **Create a resource**.
2. In the Search box, enter **Key Vault**.
3. From the results list, choose **Key Vault**.
4. On the Key Vault section, choose **Create**.
5. On the **Create key vault** section provide the following information:
   * **Name**: A unique name is required. For this quickstart, we use **Contoso-vault2**.
   * **Subscription**: Choose a subscription.
   * Under **Resource Group**, choose **Create new** and enter a resource group name.
   * In the **Location** pull-down menu, choose a location.
   * Leave the other options to their defaults.
6. Select **Create**.

Your Key vault is created



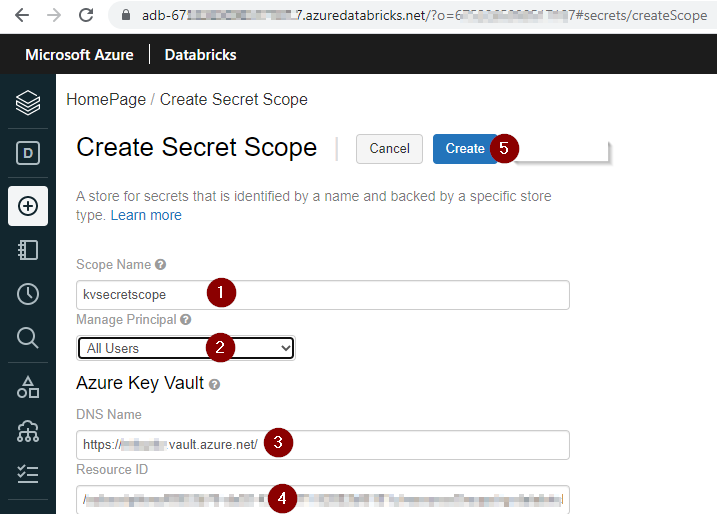
Go to the secrets



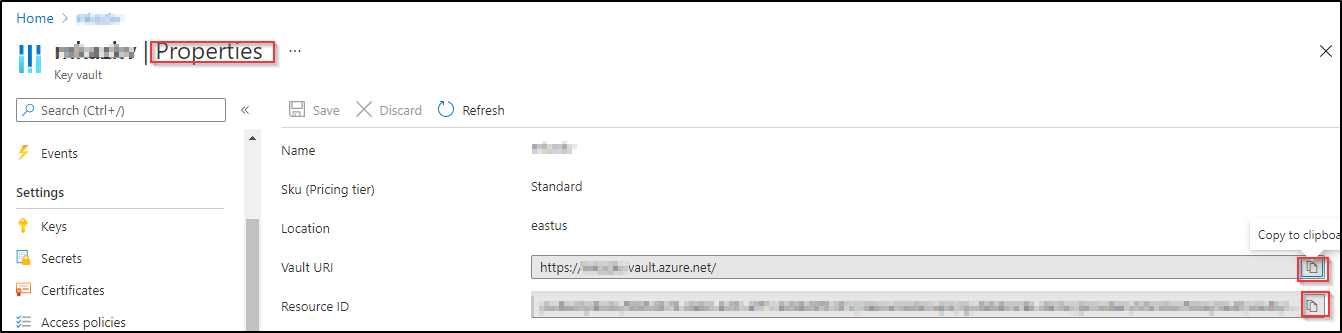


Do this step for all three(client\_id, tenant\_id, client\_secreate)

Came to databricks homepage and search URL which is #secrets/createScope and after that press enter one page is open which is



After press the enter the secrete, scope will open in that you have to provide the DNS name, Resource ID, and for that go the azure key vault



Then create

Group By:-

Demo\_df \

.groupby(“driver\_name”) \

.agg(sum(“points”).alias(“total\_point”),countDistinct(“race\_name”).alias(“number\_of\_race”)) \

.show()



To perform an operation on a group first, we need to partition the data using **Window.partition By (),** and for row number and rank function we need to additionally order by on partition data using orderBy clause.

Window function: -

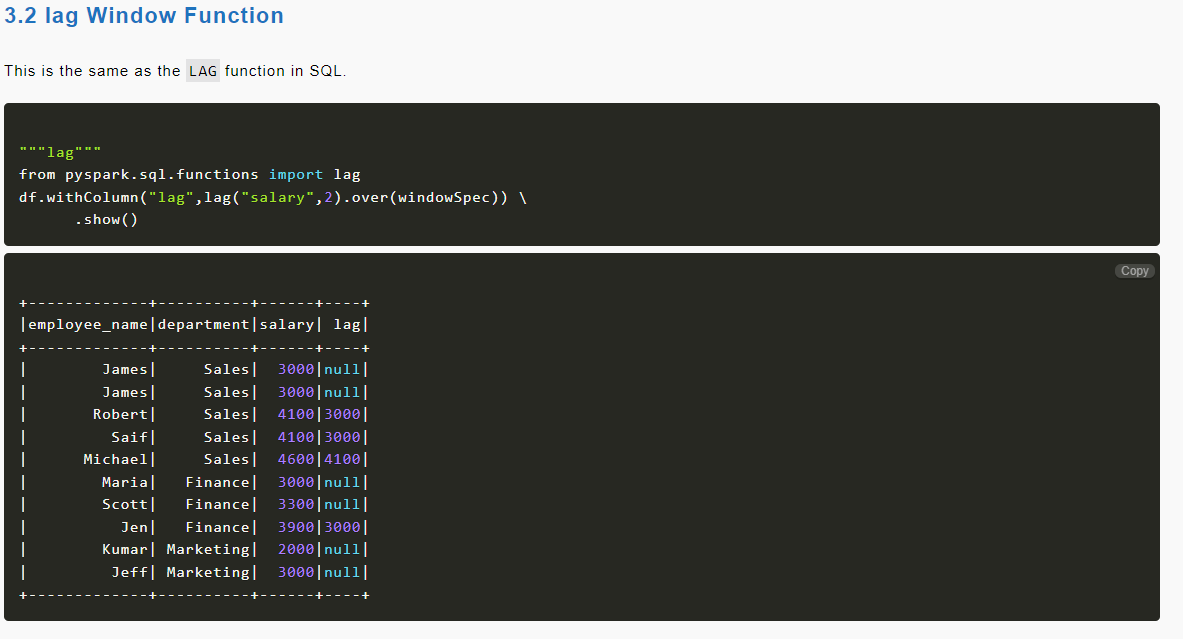
From pyspark.sql.window import window

From pyspark.sql.function import desc,rank

driverRankSpec=window.partitionBy(“race\_year”).orderBy(desc(“total\_points”))

Demo\_gtouped\_df.withcolumn(“rank”,rank().over(driverRankSpec)).show(100)

Dense\_rank:- The difference between rank and dense\_rank is that dense\_rank leaves no gaps in ranking sequence when there are ties. That is, if you were ranking a competition using dense\_rank and had three people tie for second place, you would say that all three were in second place and that the next person came in third. Rank would give me sequential numbers, making the person that came in third place (after the ties) would register as coming in fifth.







So, if you are comfortable with SQL, you can create a temporary view on DataFrame/Dataset by using createOrReplaceTempView() and using SQL to select and manipulate the data.

View in spark

In order to access a dataframe from SQL spark give you two options create temporary view and global view

race\_result\_df = spark.read.parquet(f“{presentation\_folder\_path}/race\_result”)

Race\_result\_df.createOrReplaceTempView(“v\_race\_results”)

%sql(Temporary)

SELECT COUNT (1)

FROM V\_race\_results(view name)

WHERE race\_year =2019

Iska through ma view ko access karunga jo mana create kea ha

Upper statement will be in a SQL cell

If we want to execute SQL statement from python cell you can use sql function, so you access it by spark.sql

Spark.sql(“SELECT\* FROM v\_race\_results WHERE race\_year = 2019”)

Basically, the temporary view only available within the spark session, in the data bricks context That is the current notebook that you have created that is the one notebook which has got access to the view

Global view

Race\_result\_df.createOrReplaceGlobalTempView(“gv\_race\_results”)

%sql

SELECT COUNT(1)

FROM global\_temp.global\_viewname;(global\_temp.gv\_race\_results)

If we want to execute SQL statement from python cell you can use sql function, so you access it by spark.sql

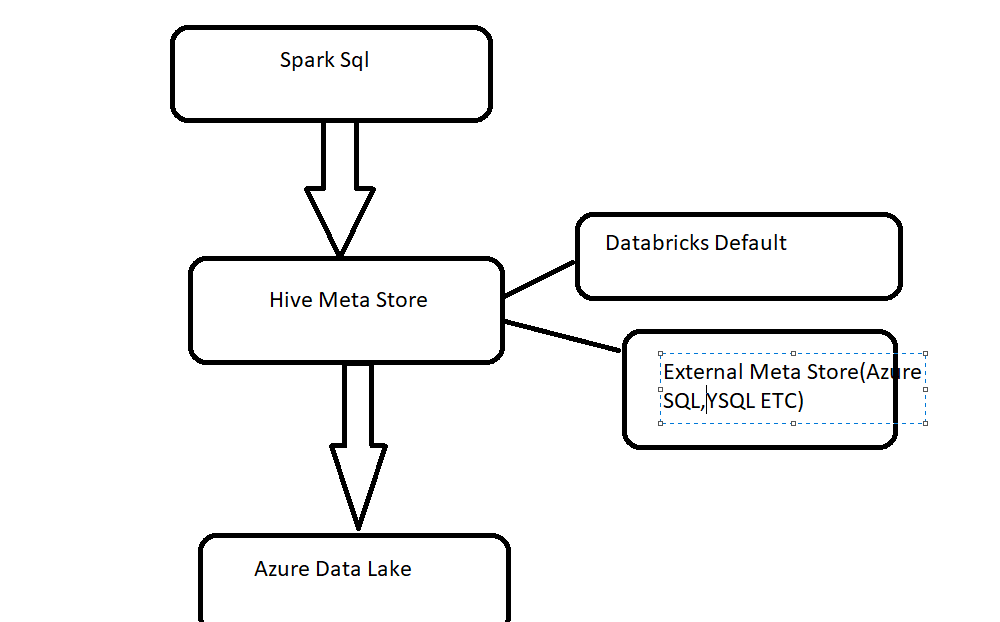
Spqrk.sql(“select \* FROM global\_temp.gv\_race\_results”)

When to use local\_view and when to use global\_view the scenario where you would use the local temp view actually your scope is just a notebook so you create the view and then you access it and then after that you don’t need that view then you go to the temporary view but whereas actually, if you wanted to create all your temporary view with in a notebook and then you have other notebook working on those views, then you would create a global view so that you can create the view up front and then your other notebooks could be using those views

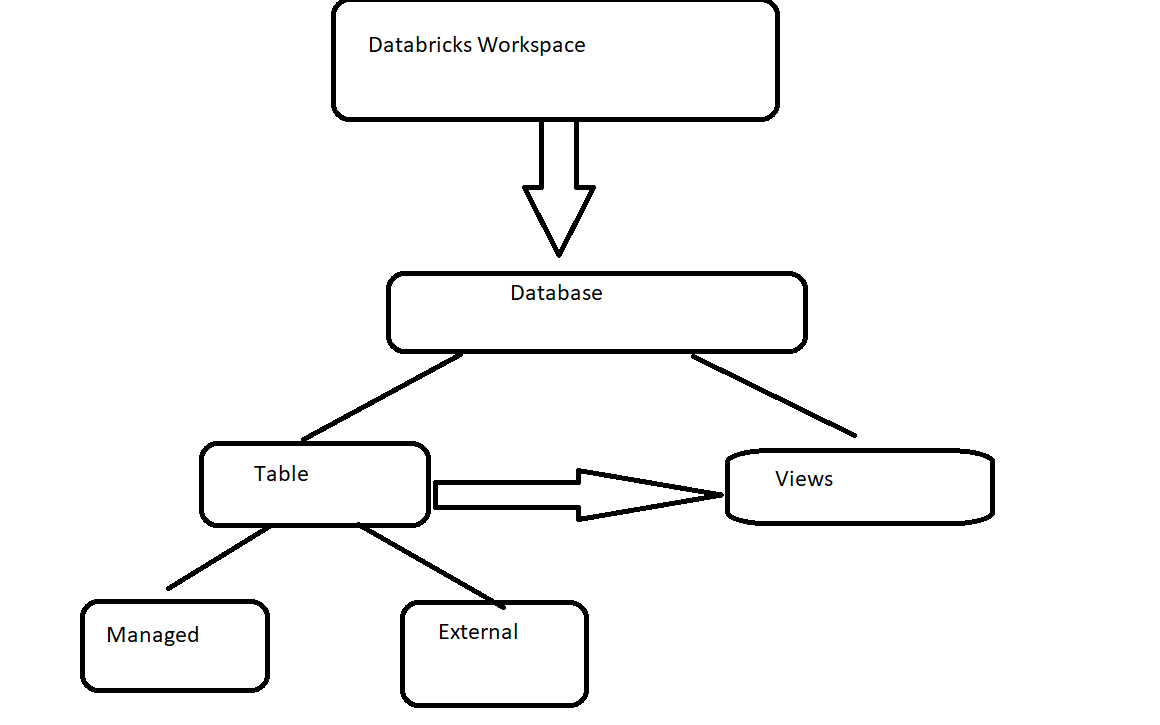
SPARK-SQL-DATABASE/TABLE/VIEW

Hive meta store

Our data is stored in our data lake as files of varying type such as CSV, JSON and parquet in order for spark to treat this data as table and columns we need to register this data in a meta store, meta store is nothing but a storage for storing the metadata about the data files example like location of the file, the format of the data, column name etc. Spark use a method still provide by the Apache Hive project for this which is called Hive meta store. When it comes to choosing the storage for Hive meta store, we have a choice we can either choose the default data Bricks managed meta store or the External storage one we register our table in the Hive meta store we can use spark sql to access these table



Let’s look at how Databricks organize database, table and view At Databricks workspace can have number of databases They are also referred as schema within a database they can have a number of table and views, Table are basically structures given to the data stored in an object storage as we said ADLs is our object storage there are two type of table managed, external. Views can be built on the tables with a selection of data for example you could apply a filter to a table, or you could select just a certain number of columns



In case of managed table, sparks maintain both the metadata in Hive meta store and also the data files associated with the table which is stored in ADL, In the case of managed table, it deletes the files as well as drop the table

In case of external table sparks manages the metadata and we manage the datafiles, the meaning of this is external tables we specify the location of the data files and sparks doesn’t decide if it for itself Dropping the table for an external table doesn’t delete the files

Managed table using python=

%python

race\_results\_df=spark.read.parquet(f “{presentation\_folder\_path}/race\_result”)

Dataframe ma data aa Gaya ha abb hum usa write kar na ha table ka andar nicha wala wo kam kar raha ha (write data in the table)

%python

race\_results\_df.write.format(“parquet”).saveAsTable(“demo.race\_results\_python”) = Managed Table

Managed table using SQL

CREATE TABLE demo.race\_results\_sql

AS

SELECT \* <------This statement will perform and the result store

FROM demo.race\_results\_python<---- in the table we create

WHERE race\_year = 2020;

Isma kay hoga ke select ka through data select hogay aur jo exernal tabel bana ha usma jaya ga

In the managed table if you drop the table, it will be delete from ever where

External Table using python

%python

race\_results\_df.write.format(“parquet”). option(“path”,f”{Presentation\_folder\_path}/race\_results\_ext\_python”}.saveAsTable(“demo.race\_result\_ext\_python”)

External Table using sql

CREATE TABLE demo.race\_results\_ext\_sql (mana yaha kay kea ha ke ek tabel create kea ha)

(

race\_year INT,

race\_name STRING,

Asa he sub variable ka samna unka data type likan ga

)

USING parquet (format of file we mention)

LOCATION “/mnt/formula1dl/presentation/race\_results\_ext\_sql (ya hum log managed tabel ma nahi karta ha)

But in this the Table is empty so we have to insert

INSERT INTO demo.race\_result\_ext\_sql

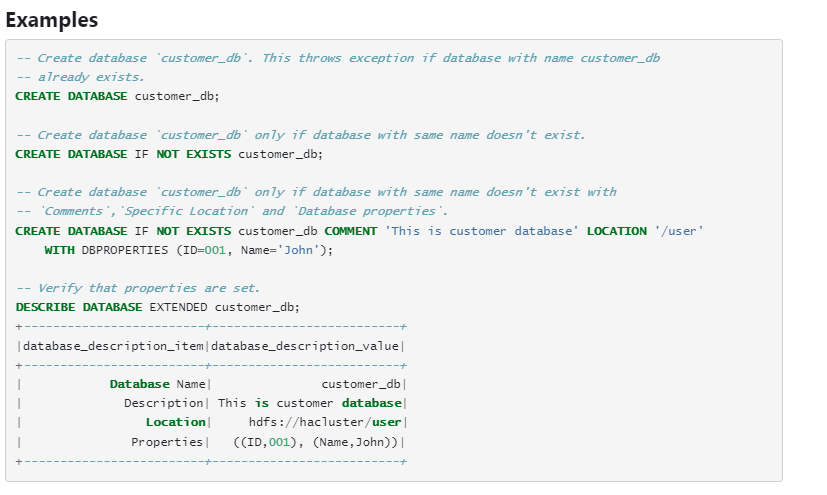
SELECT \* FROM demo.race\_results\_ext\_py WHERE race\_year = 2020;

When you drop the external table it being remove from metadata but if you check in the azure storage explorer in the presentation container you will see that file

In the case of external table, you can controlling the data so if you wanted to remove this data you will have to run commands to remove data from the file

MetaStor=MetaStor is nothing but a storage for storing the metadata about the file such as location of the file, the format of the data, column names, etc.

Database=



In order to see the content of Data menu you need to have a cluster running and database is by default which is default database

DESCRIBE DATABASE EXTENDED database\_name; = It is optional to add extended

Create a table

DROP TABLE IF EXISTS f1\_raw.circuits;

CREATE TABLE IF NOT EXISTS f1\_raw.circuits(circuitId INT,

CircuitRef STRING,

Name STRING,  **DDL FORMAT**

Location STRING,

LAT DOUBLE,

LNG DOUBLE,

Alt INT,

Url STRING

)

USING csv

OPTIONS(path “/mnt/formula1dl/raw/circuits.csv”)

Permanent view=

CREATE OR REPLACE VIEW demo.race\_results

AS

SELECT \*

FROM demo.race\_result\_python

WHERE race\_year = 2000;

SQL BASIC

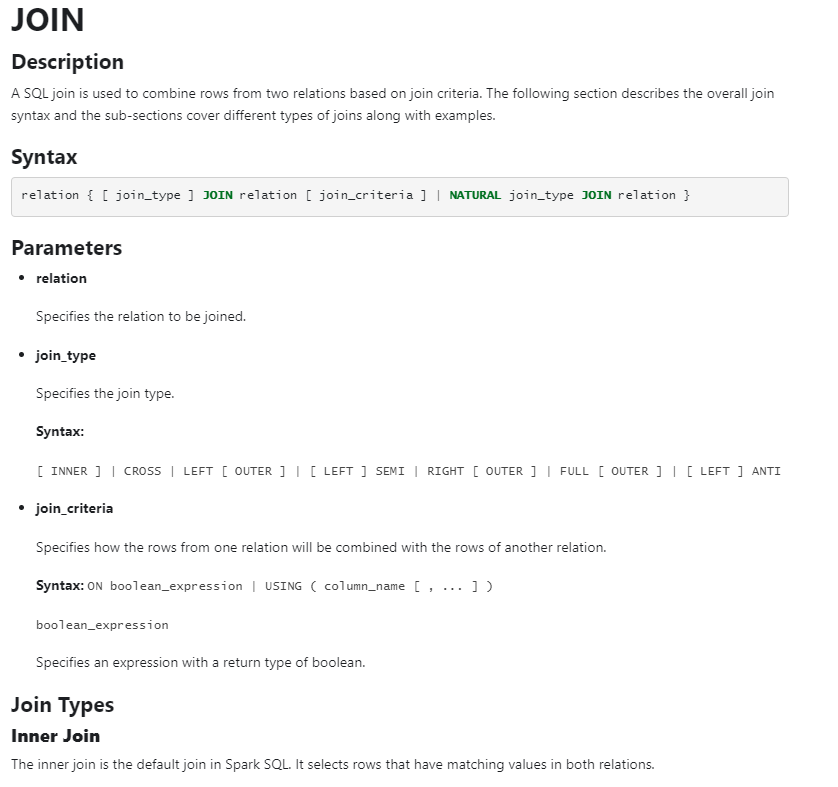
SELECT \*

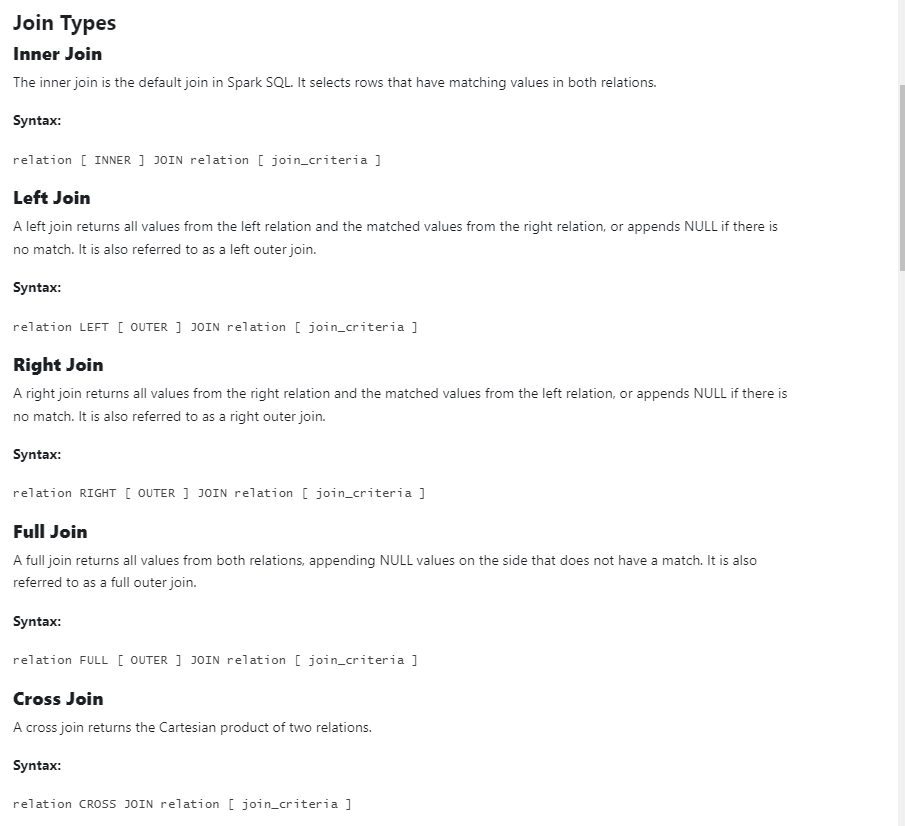
FROM drivers

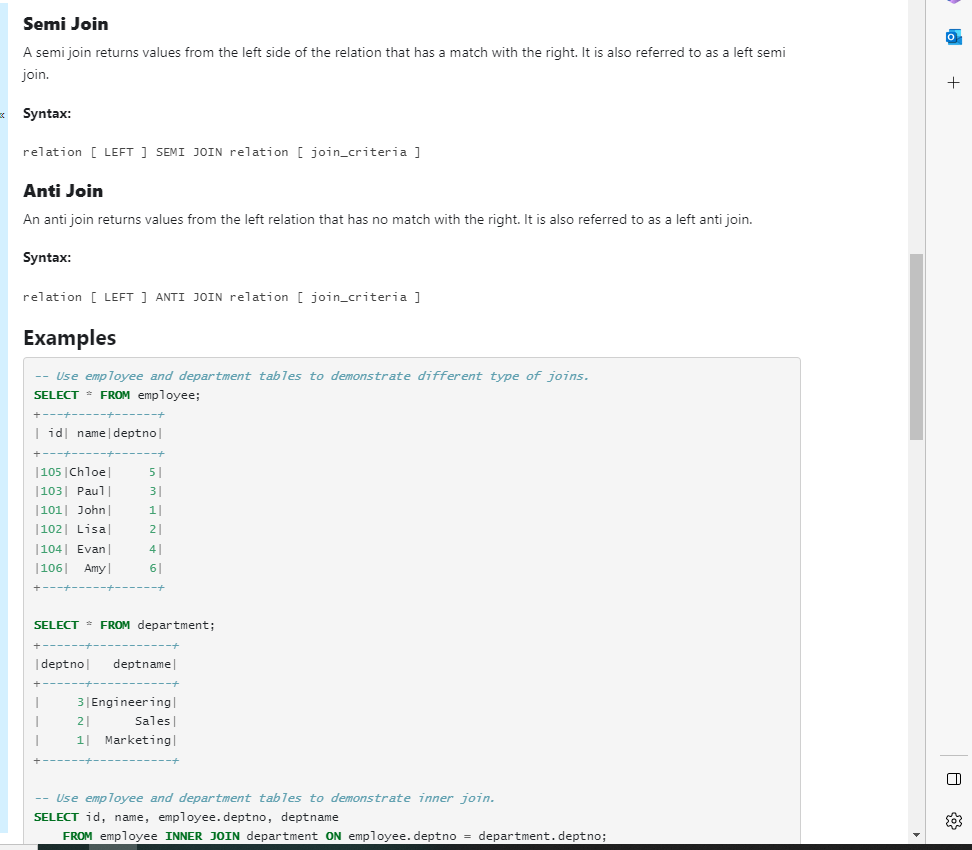
ORDER BY nationality ASC,

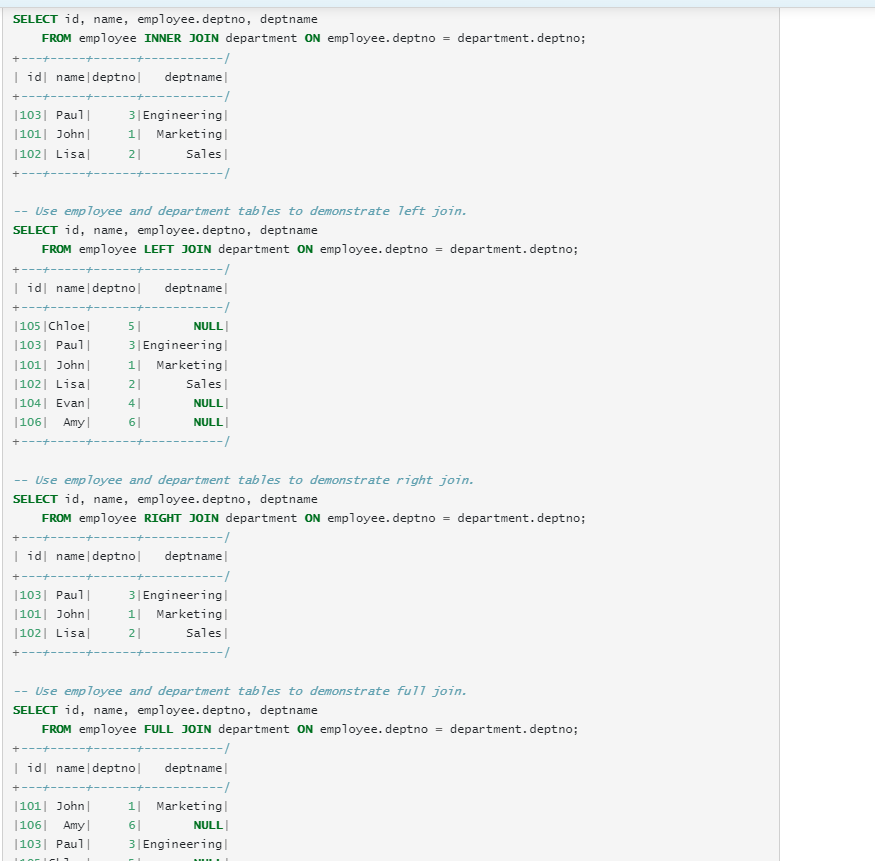
dob DESC;

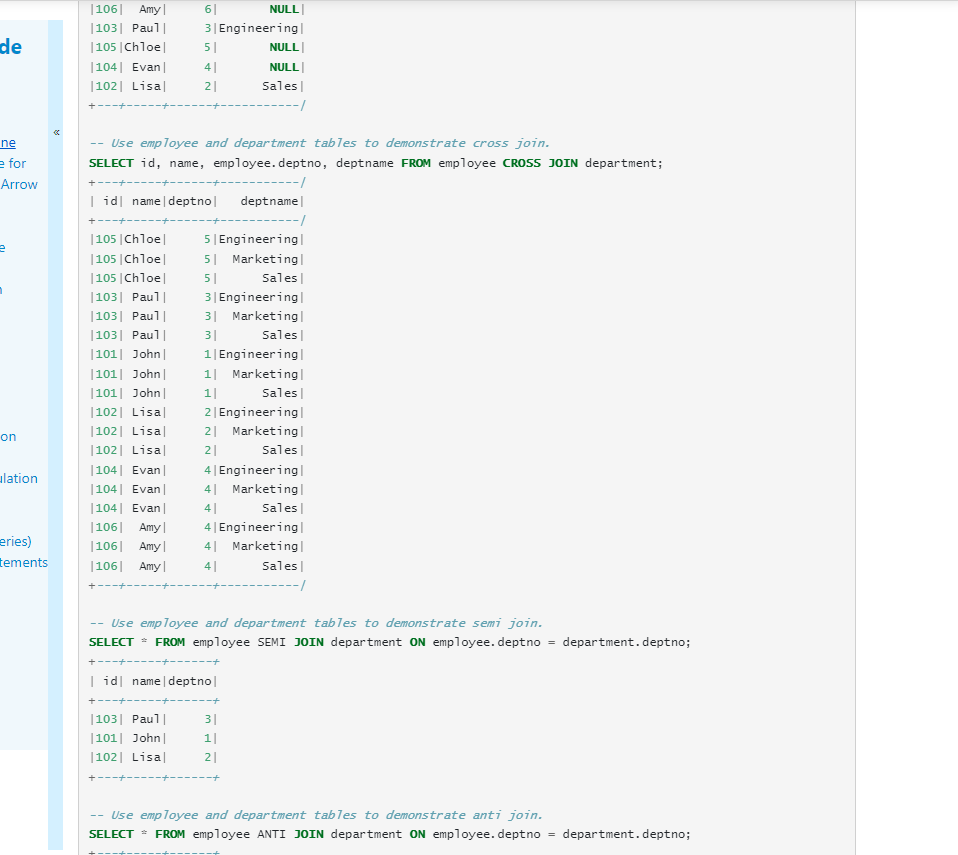
Join

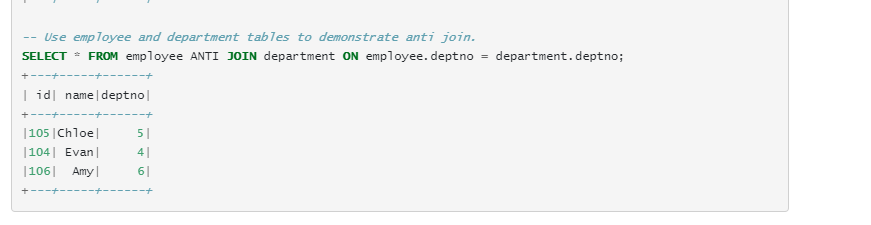












The CONCAT function adds two or more strings together.

CONCAT(*string1*, *string2*, *....*, *string\_n*)

SELECT \*, CONCAT(driver\_ref, ‘\_’,code) AS new\_driver\_ref

FROM drivers = isma kay hoga ke driver\_df and code ka data merage hoga and new\_driver\_ref ka andar show hoga

The Split() function splits a string into an array of strings.

Split(*string*, *separator*, *limit*, *compare*)

SELECT \*, SPLIT(name, ‘ ’)

FROM drivers = isma kay hoga ke name jo ha who split hojaya ga based on space

The DATE\_FORMAT() function formats a date as specified.

DATE\_FORMAT(*date*, *format*)

SELECT \*, date\_format(dob, ‘dd-MM-yyyy')

FROM drivers

The HAVING clause is used to apply a filter on the result of GROUP BY based on the specified condition. SQL as the WHERE keyword failed when we use it with aggregate expressions.

Window function

rank() - Computes the rank of a value in a group of values.

SELECT nationality,name,dob,RANK() OVER(PARTITION BY nationality ORDER BY dob DESC) AS age\_rank

FROM drivers

ORDER BY nationality ,age\_rank

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

### HAVING Syntax

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)  
HAVING condition  
ORDER BY column\_name(s);

Data Load Type

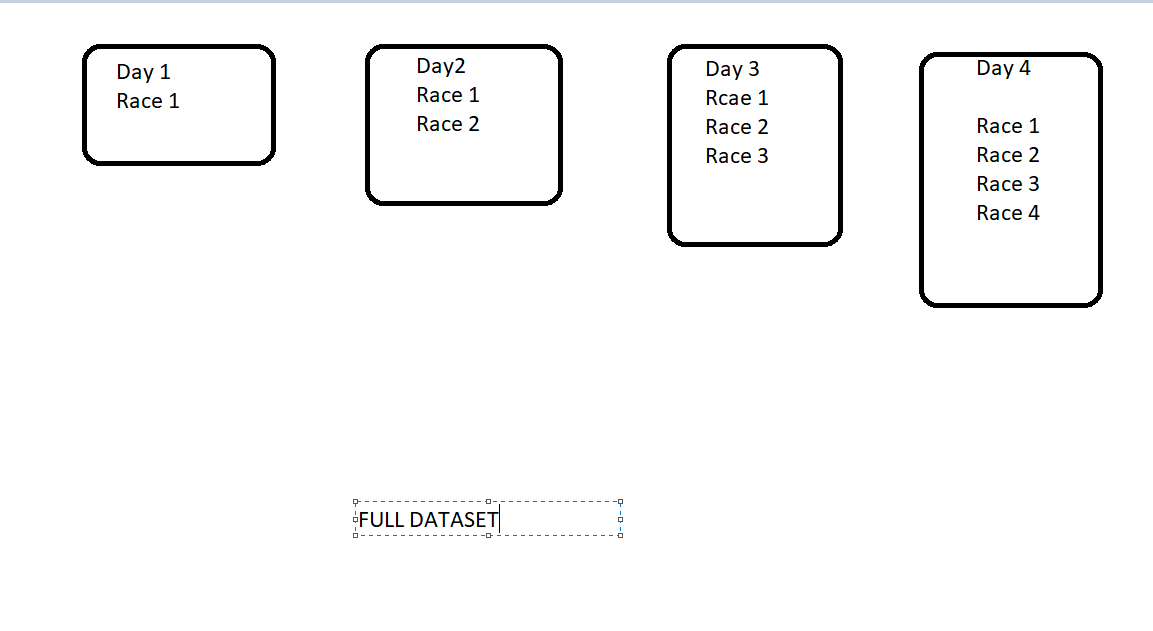
Full Load

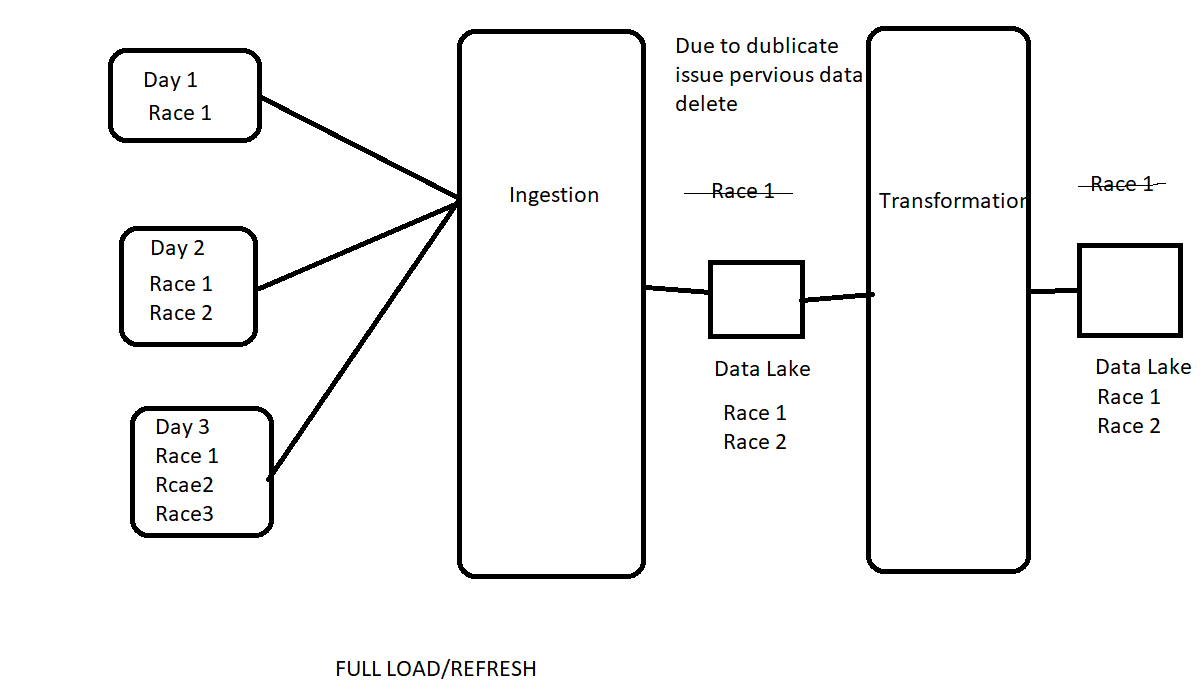
In our design we took all the data that we recevied from the raw container,processed all of the data and replaced everthing in our processed and the presentation container every time this work well when you have a small data this is generally called as full load

Incremental Load

It won’t be suitable for large data pipelines, mainly due to the amount of processing we have to perform in order to process all the data every time. Those pipeline are generally designed to load and process only the data that has changed between the current run and the pervious run this is generally called as incremental load

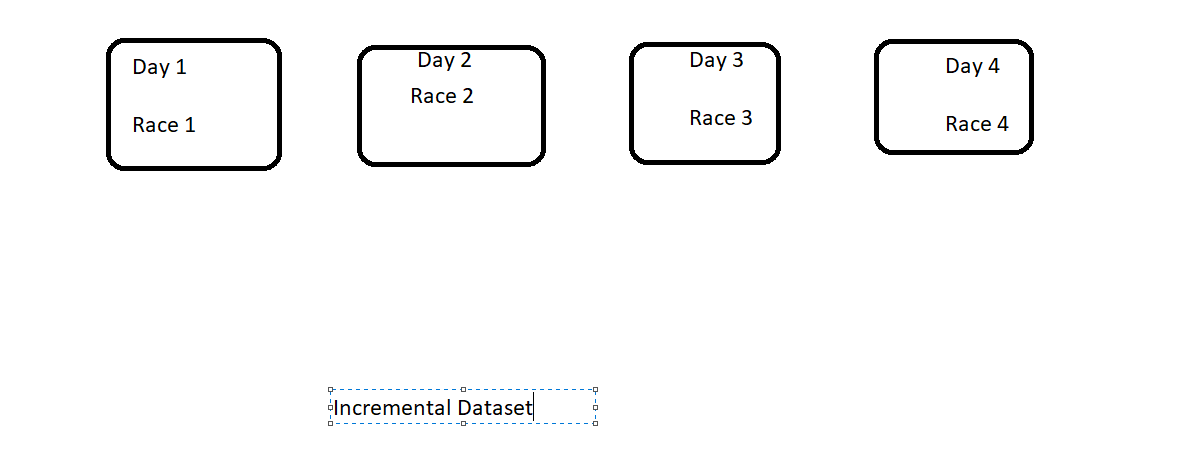
For example, if we receive all the data every time it is a good candidate for full load but if we instead receive only the data that’s changed since the last load, then that is a candidate for an incremental load

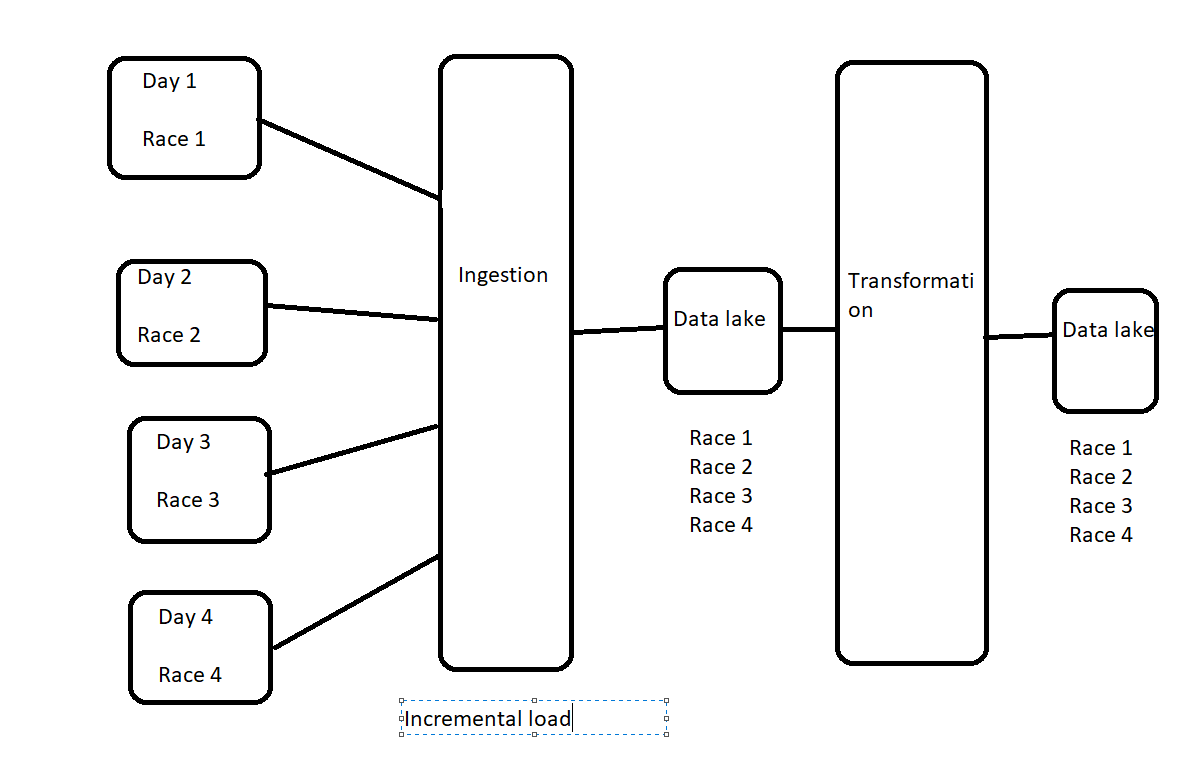




The problem here is that we are processing the data from all races on every run, even though only the data from the recent day or recent race has changed This adds a significant amount of processing required for the ingestion as well as the transformation thus potentially adding latency to the data being consumed and also more processing power to be required and resulting potentially in higher cost for your application as well.

Both Ingestion and transformation is empty to start





On day 1 there is no change between a full load and the incremental load both the processed and the presentation containers are empty to start with, so day 1 data is returned to those. day 2 data is going to be vastly different. on day 2 we only receive the data from race 2 as we

said before so the ingestion process will append the race 2 data to the data lake. the transformation process will now need to identify the changes from the ingested data and process only that and this will result in the data lake for the presentation layer to be added with the race 2 data load same for day 3

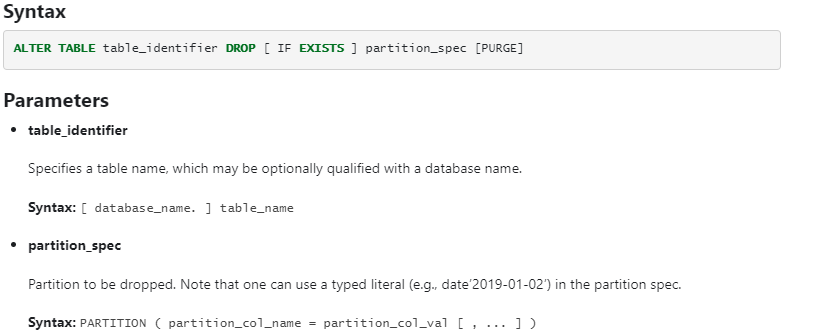
Hybrid scenario

DROP DATABASE IF EXISTS f1\_processed CASCADE; = This will drop all the table under the database and database itself.

Input widgets allow you to add parameters to your notebooks and dashboards. The widget API consists of calls to create various types of input widgets, remove them, and get bound values.

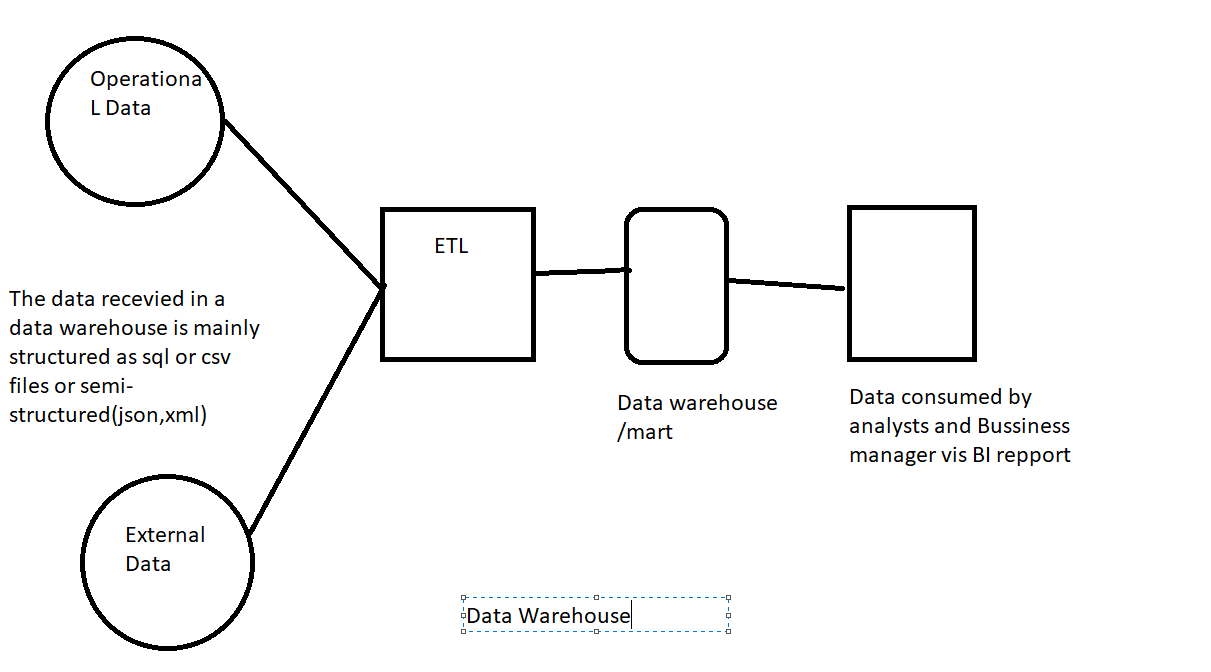
Collect = it will give you a list

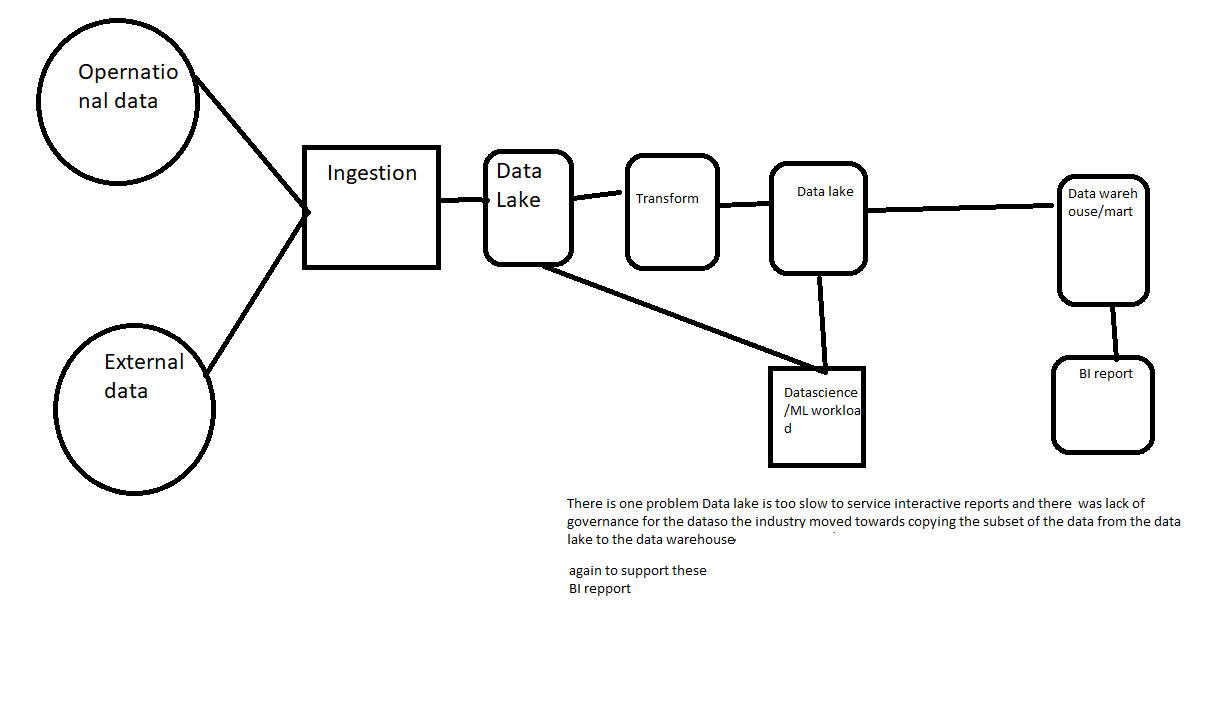
Catalog =



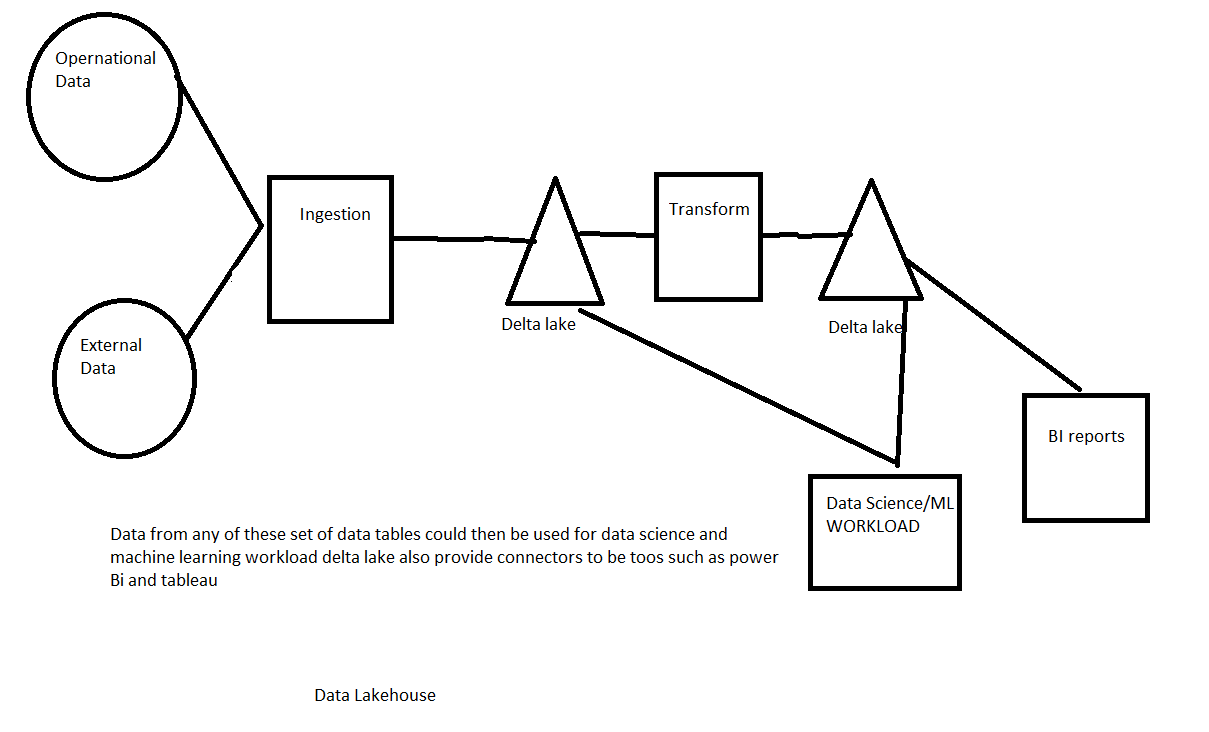
Incremental load-method 2, Incremental load improvement, Incremental load-Notebook, Incremental load – race, Incremental load-Driver, Incremental load- constructer(NOT UNDERSTAND)

Delta Lake





Delta lake can handle not only structured and semi-structured data, but they can also handle unstructured



Create Delta tabel

Read and write in delta lake

%sql (ya islea ku ke cell jo ha who python ha)

CREATE DATABASE IF NOT EXISTS d1\_demo

LOCATION ‘/mnt/formula1dl/demo’

result\_df = spark.read.option(“inferschema”, True).json(“/mnt/formula1dl/raw/2021-03-28/results.json”)

Isko khali example consider karna





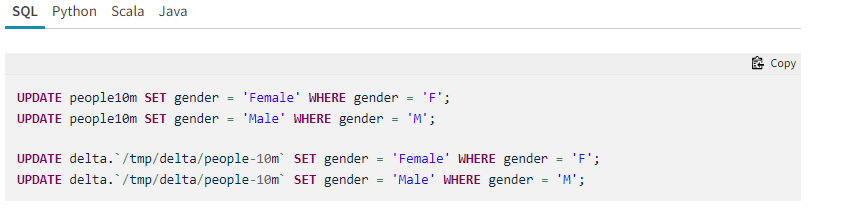
Update and delete in Delta Lake

%sql

UPDATE f1\_demo.results\_managed (Table)

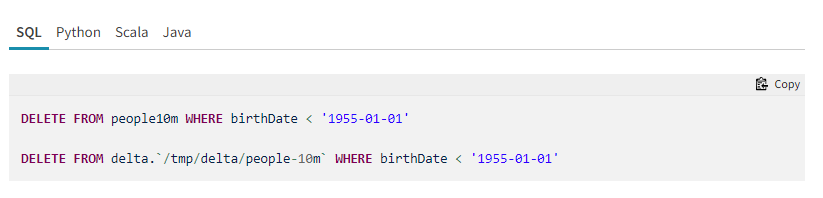
SET points = 11-position (what we have to set)

WHERE position <=10 (condition)



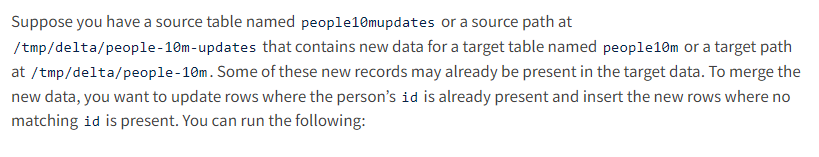


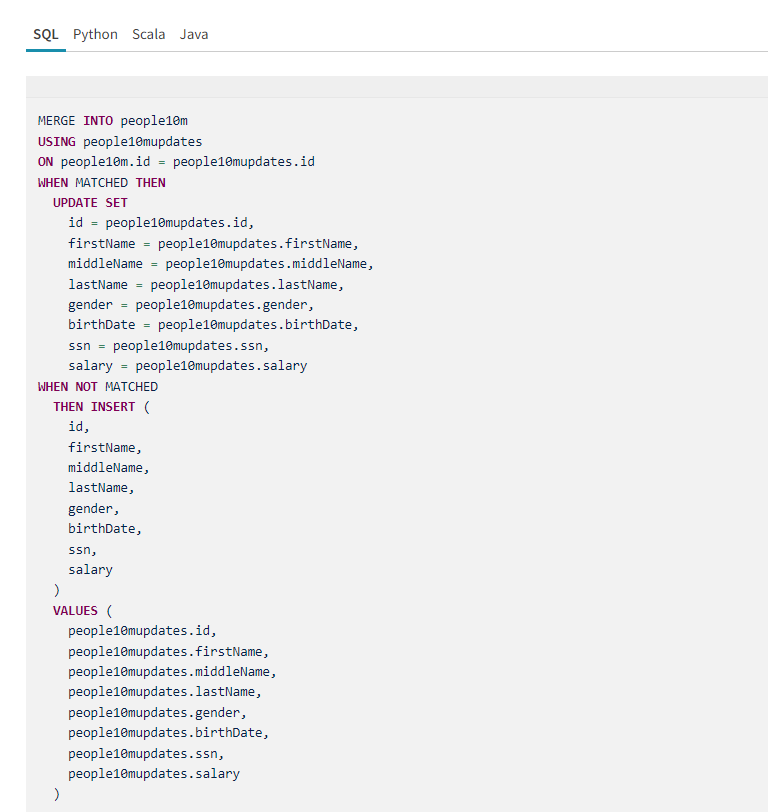




Merage

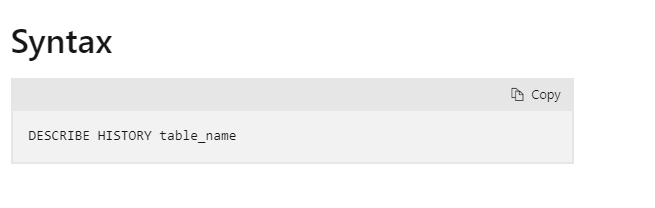
So basically, a merge statement gives you the ability to insert any new records being received, update any existing records for which new data has been received and if you had a delete request, then you can apply the delete as well and you do all three of these in one statement



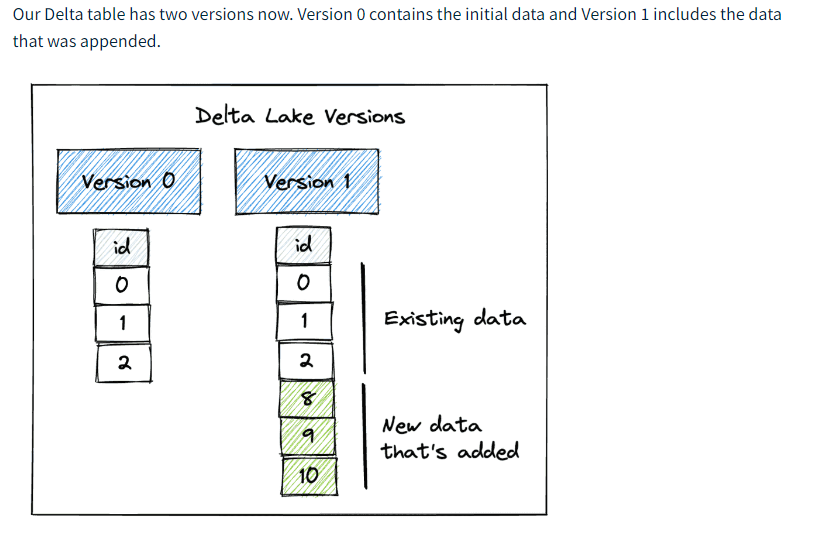


History

%sql



Versioning





If I have a legal requirement to delete the data for someone, I should be physically delete the data altogether and we shouldn’t be able to see the data and for that we use Vacuum and Usually Vacuum removes the history which is older than seven days, vacuum is not triggered automatically the default retention threshold for data files is 7 days. (isko assan language ma samaj legea ke 7 ka baad delete kar ta ha)

%sql

VACUUM f1\_demo.drivers\_merge (Table)

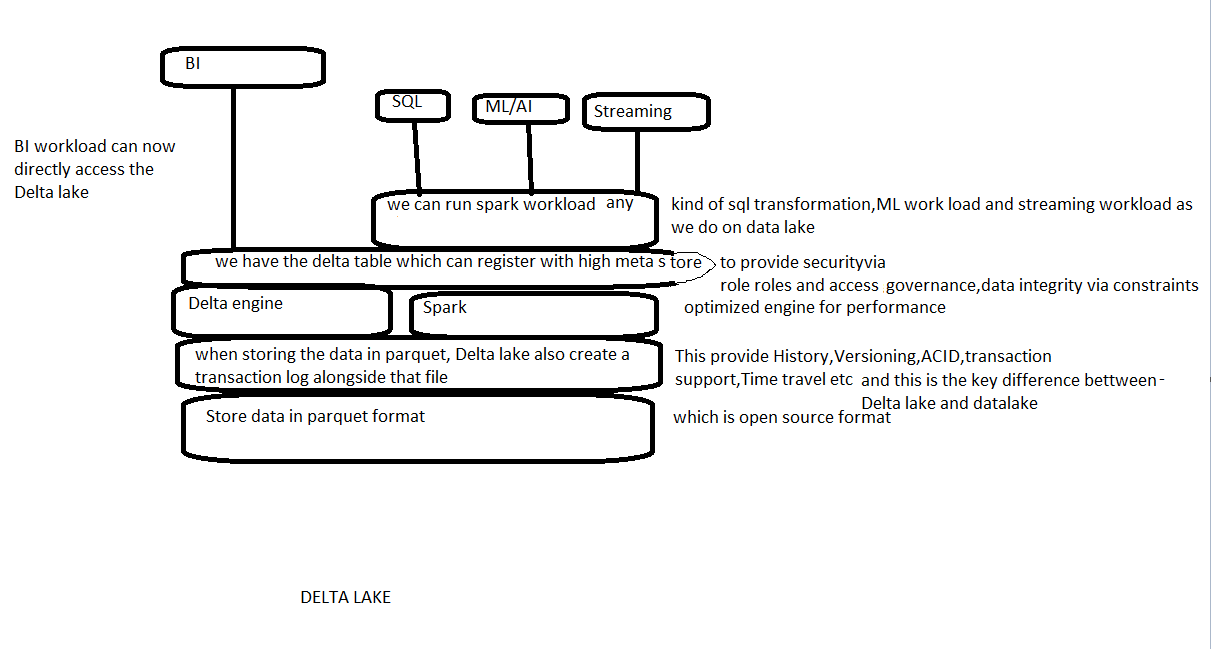
If we need to remove the data immediately what we do is set the retention to 0 hours

%sql

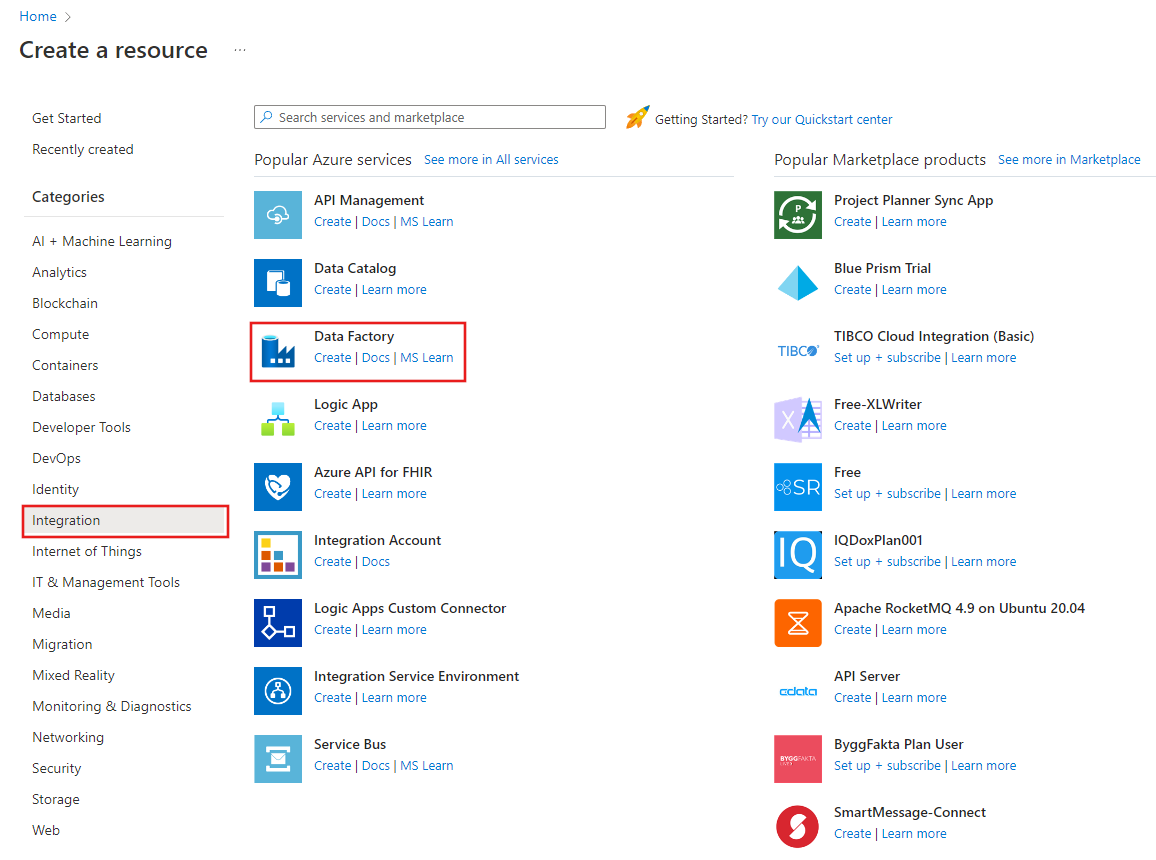
SET spark.databricks.delta.retentionDurationCheck.enabled = false

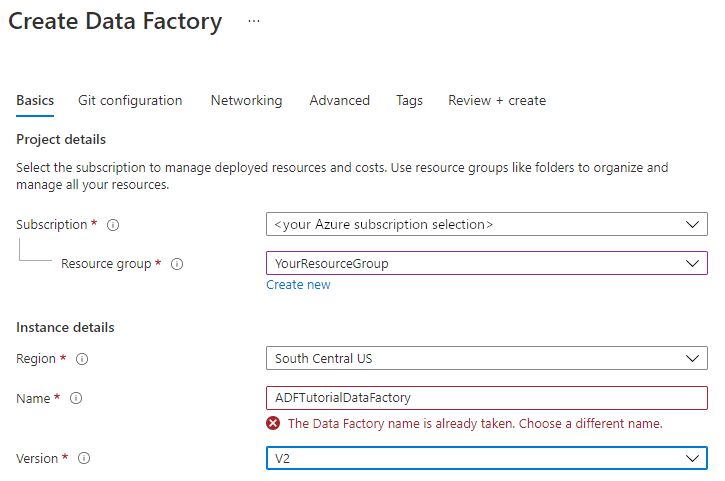
VACUUM f1\_demo.drivers\_merge RETAIN 0 HOURS

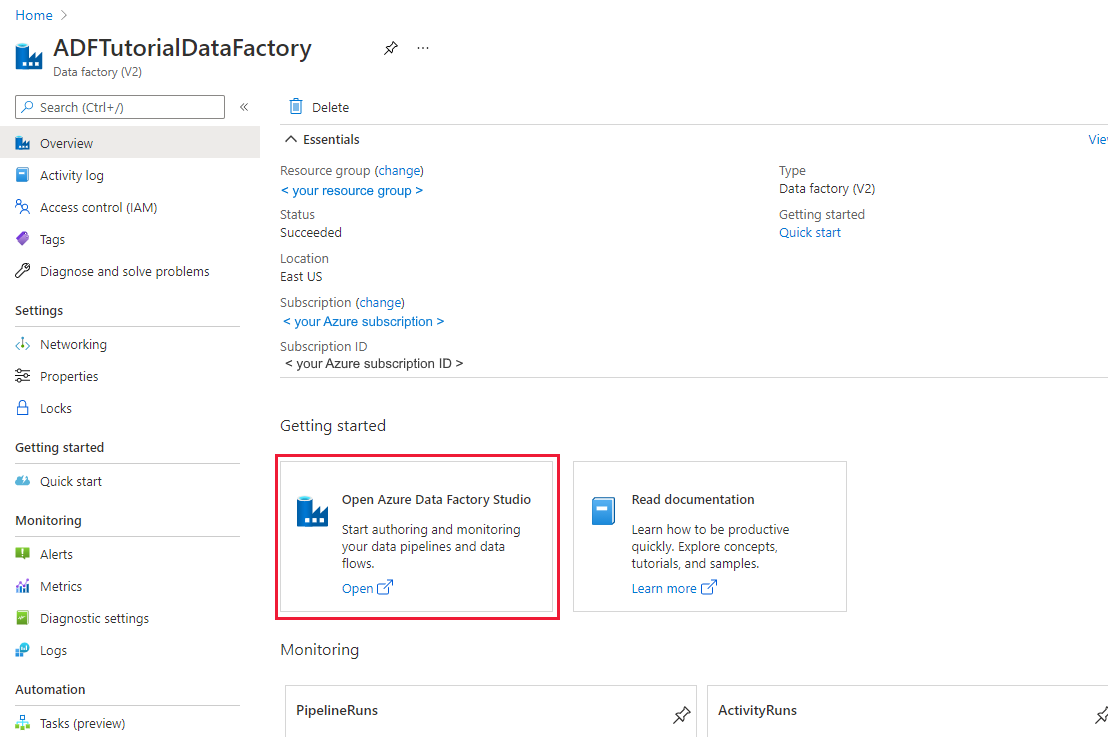
(When you run this query, it shows error like are you sure you would like to vacuum files with such a low retention period?)



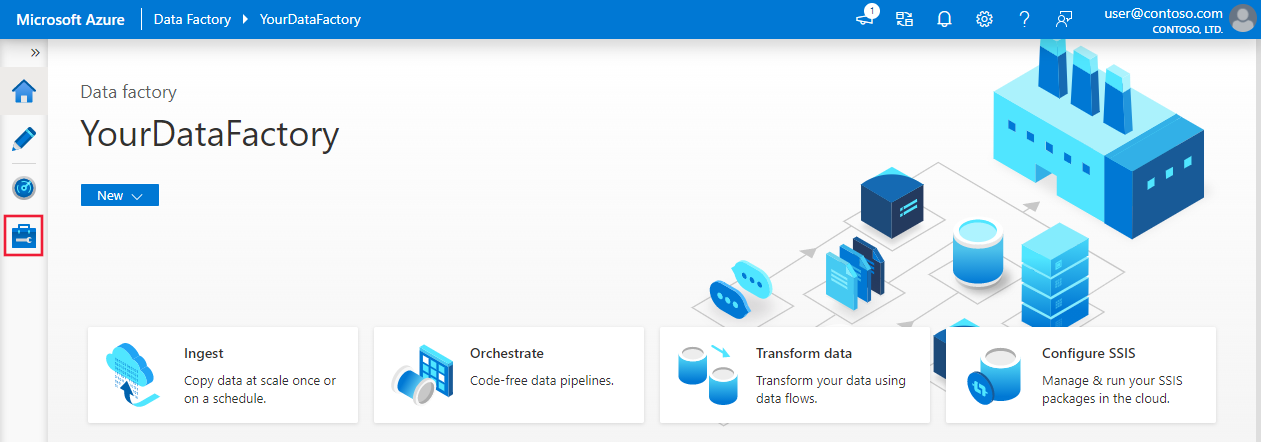
# Run a Databricks notebook with the Databricks Notebook Activity in Azure Data Factory

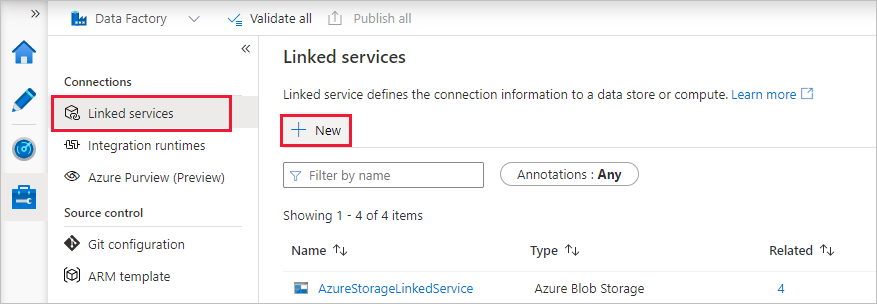


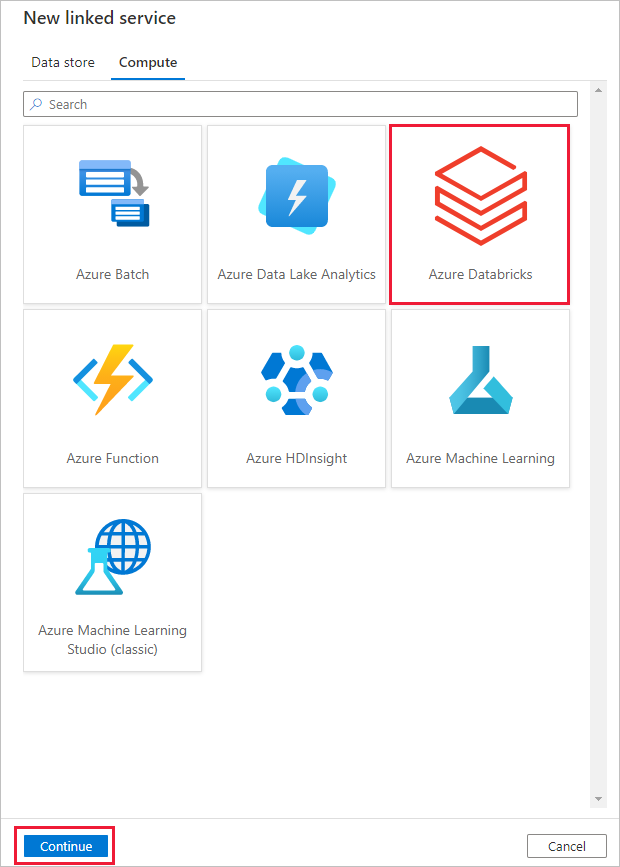


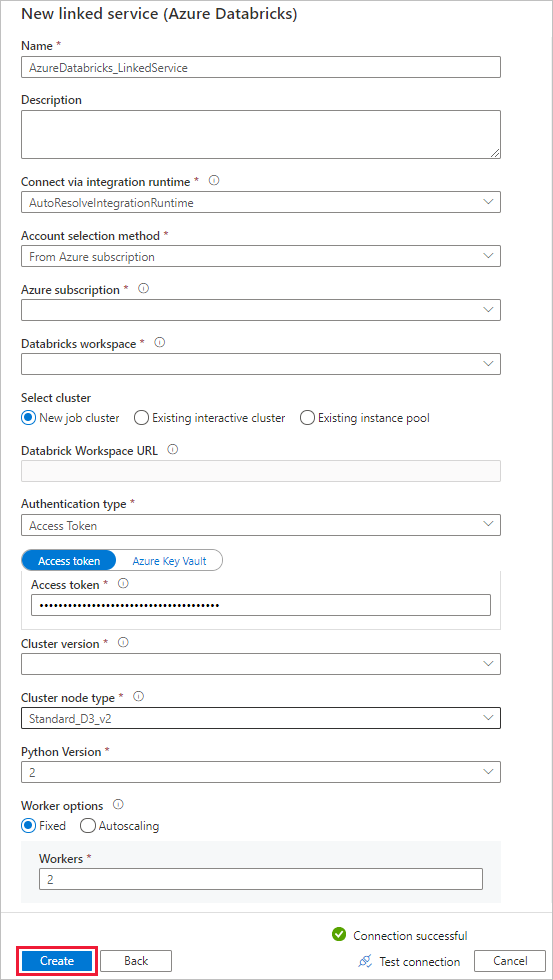


### Create an Azure Databricks linked service

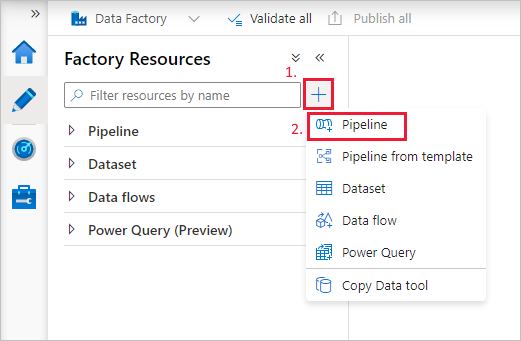




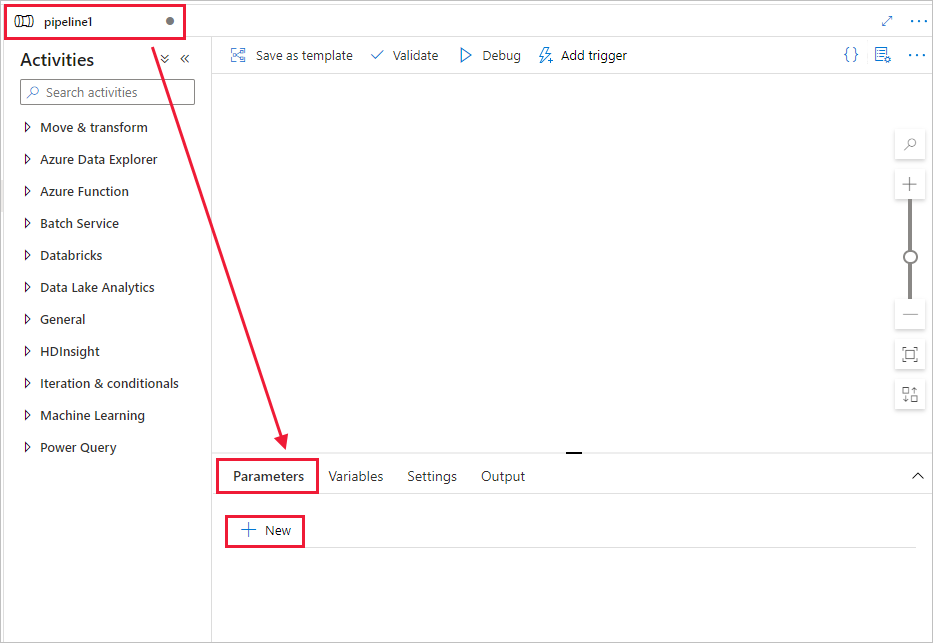


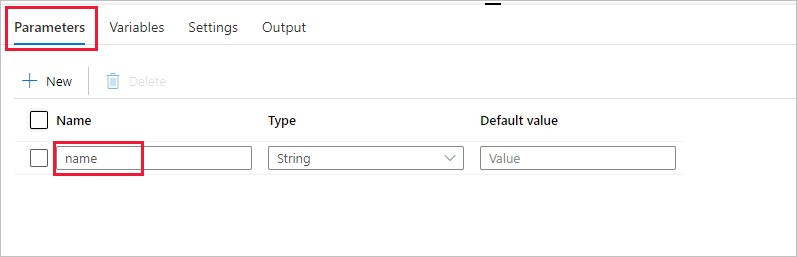


## Create a pipeline

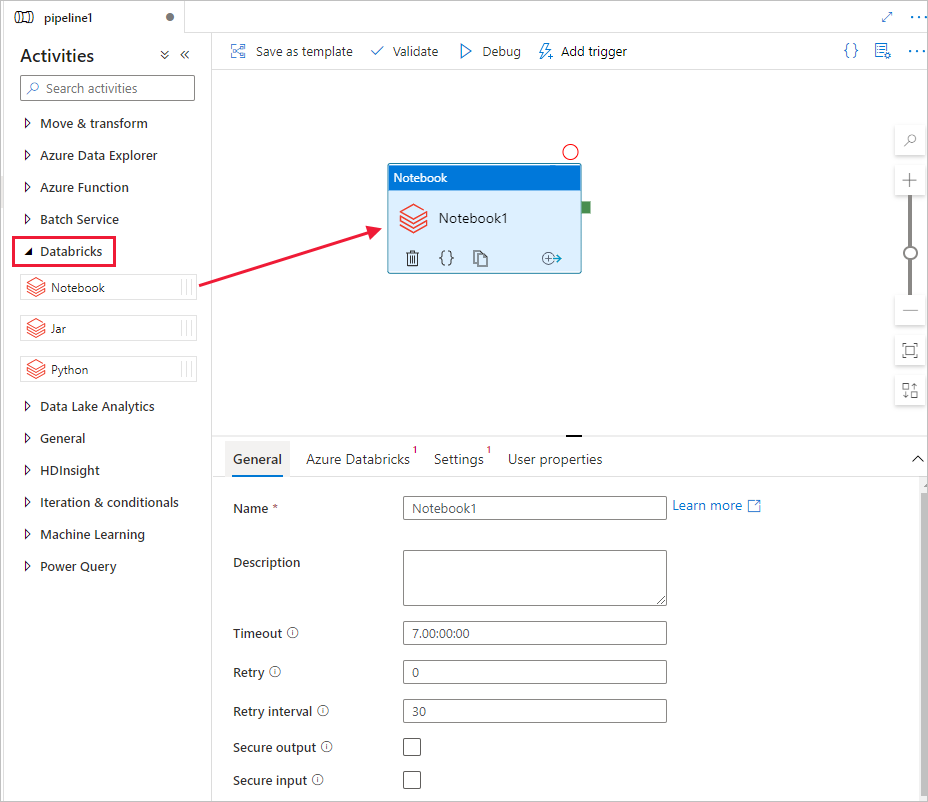


Create a **parameter** to be used in the **Pipeline**. Later you pass this parameter to the Databricks Notebook Activity. In the empty pipeline, select the **Parameters** tab, then select **+ New** and name it as '**name**'.





In the **Activities** toolbox, expand **Databricks**. Drag the **Notebook** activity from the **Activities** toolbox to the pipeline designer surface.

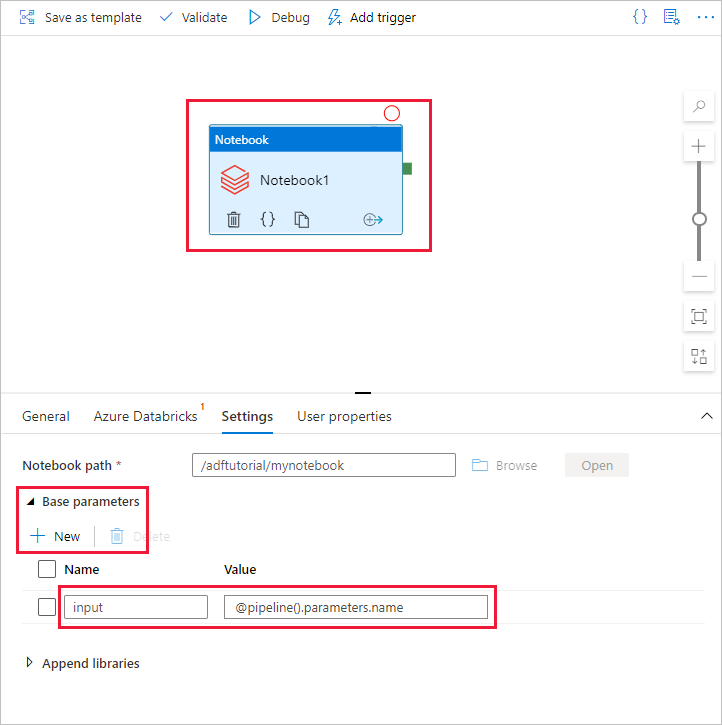


In the properties for the **Databricks** **Notebook** activity window at the bottom, complete the following steps:

1. Switch to the **Azure Databricks** tab.
2. Select **AzureDatabricks\_LinkedService** (which you created in the previous procedure).
3. Switch to the **Settings** tab.
4. Browse to select a Databricks **Notebook path**. Let’s create a notebook and specify the path here. You get the Notebook Path by following the next few steps

Switch back to the **Data Factory UI authoring tool**. Navigate to **Settings** Tab under the **Notebook1** activity.

a. Add a **parameter** to the Notebook activity. You use the same parameter that you added earlier to the **Pipeline**.



In a Base parameter I am using the parameter which is used by in the Udemy course

dbutils.widgets.text(“p\_data\_source”, “ ”)

v\_data\_source = dbutils.widgets.get(“p\_data\_source”)

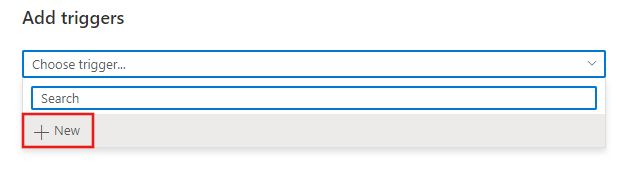
dbutils.widgets.text(“p\_file\_date”, “ ”)

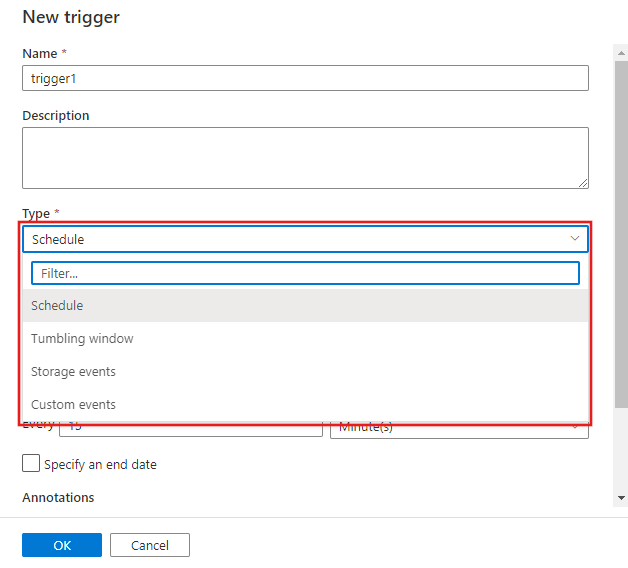
v\_file\_date = dbutils.widgets.get(“p\_file\_date”)

For data source came outside the activity and click on the pipeline and go into the variable tab and give the name of the variable and default value (in a project we use Ergast which is your data source) hum log na jo variable ka use kea ha us variable ka through hum parameter ma value dalan ga

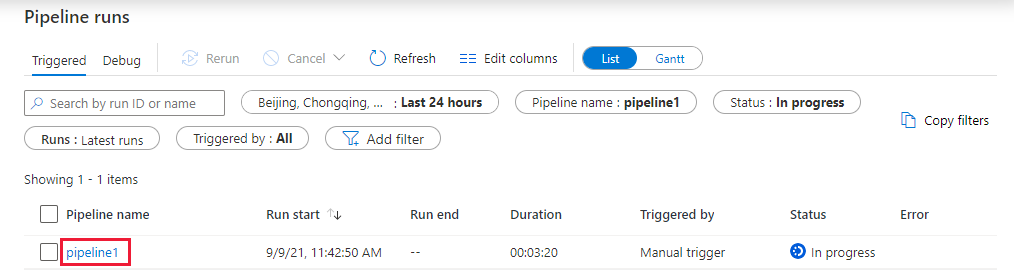
And for the file date we again came outside the activity and click on the pipeline and click on the parameter tab and in that we give p\_window\_end\_date that because when we implement the solution, we are get out trigger to send something called a window end date, p\_window\_end\_date will be in the timestamp we have to convert them in date for that we use function.

## Trigger a pipeline run



Click kar na NO par, phila jis pipeline ko, app ko activate kar na wha jao and then click on new trigger and select from drop down and then click YES to activate the trigger after that it ask the value of the parameter trigger won’t run unless you are not publish all the pipeline

## Monitor the pipeline run



[Run a Databricks Notebook with the activity - Azure Data Factory | Microsoft Learn](https://learn.microsoft.com/en-us/azure/data-factory/transform-data-using-databricks-notebook)

**DEBUG THE PIPELINE**

[How to Debug a Pipeline in Azure Data Factory (serverlessnotes.com)](https://www.serverlessnotes.com/docs/how-to-debug-a-pipeline-in-azure-data-factory#:~:text=In%20the%20Azure%20Data%20Factory%20Studio%2C%20click%20in,completely%20and%20the%20ETL%20process%20will%20be%20performed%3A)