Connect SQL to Databricks

We need to creates SQL keys in Key vault for that we need to get these details from SQL database

SQL Database-> Settings -> Connection strings -> JDBC

jdbc:sqlserver://deepthi.database.windows.net:1433;database=dbnew;

user=admin12@deepthi;password={your_password_here};encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;

here URL will be jdbc:sqlserver://deepthi.database.windows.net:1433;database=dbnew;

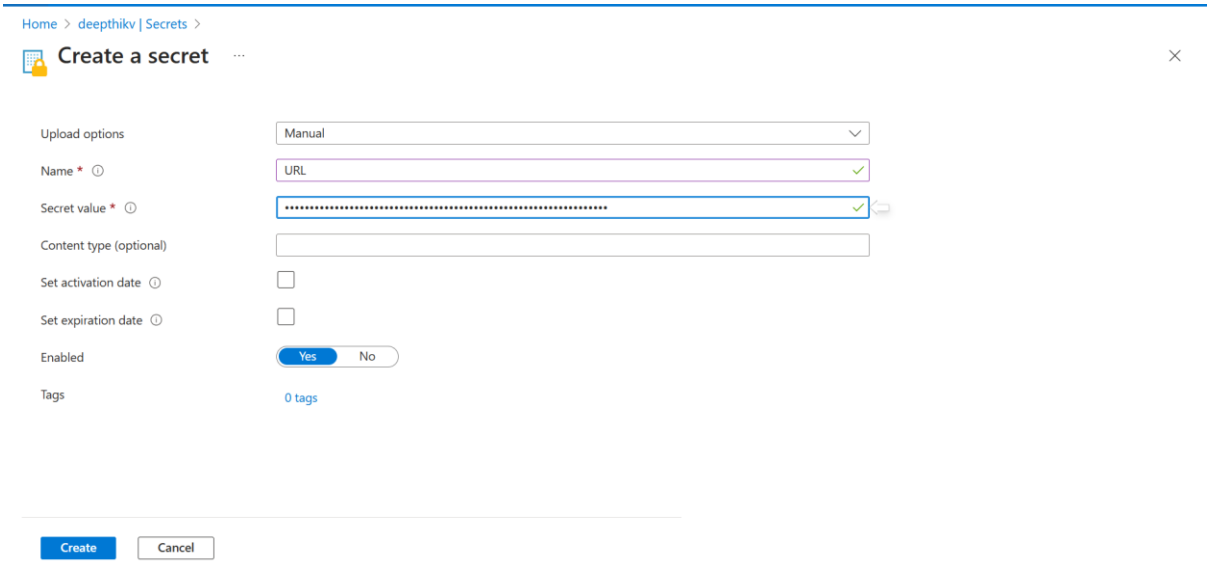
username is admin12

password is **********

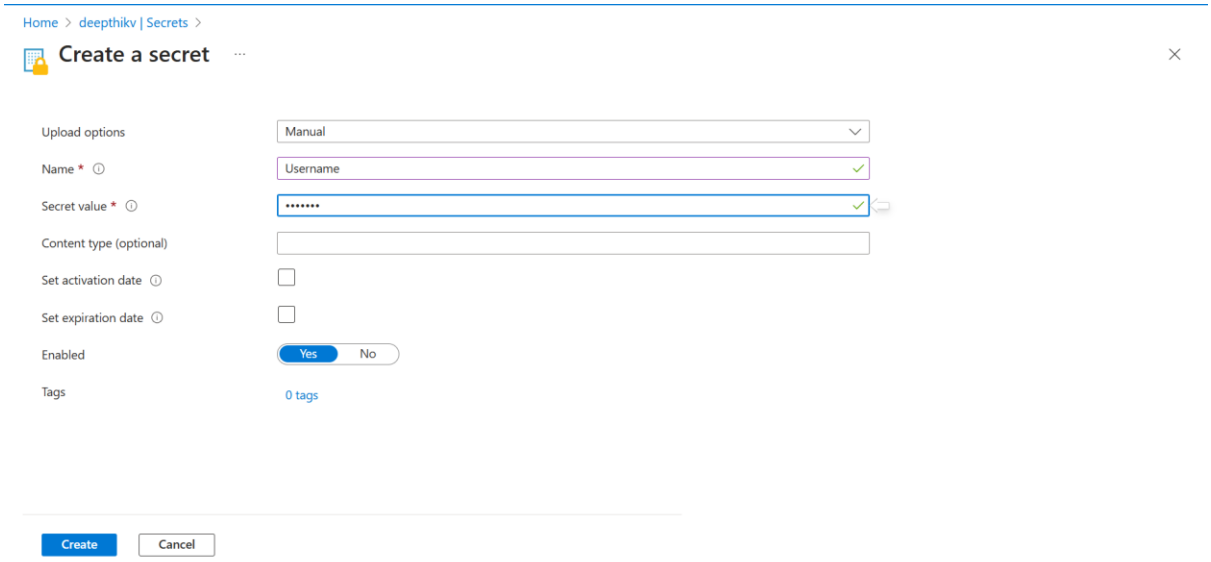
as per my account details

Now got to key vault -> Objects-> Secrets-> creates keys

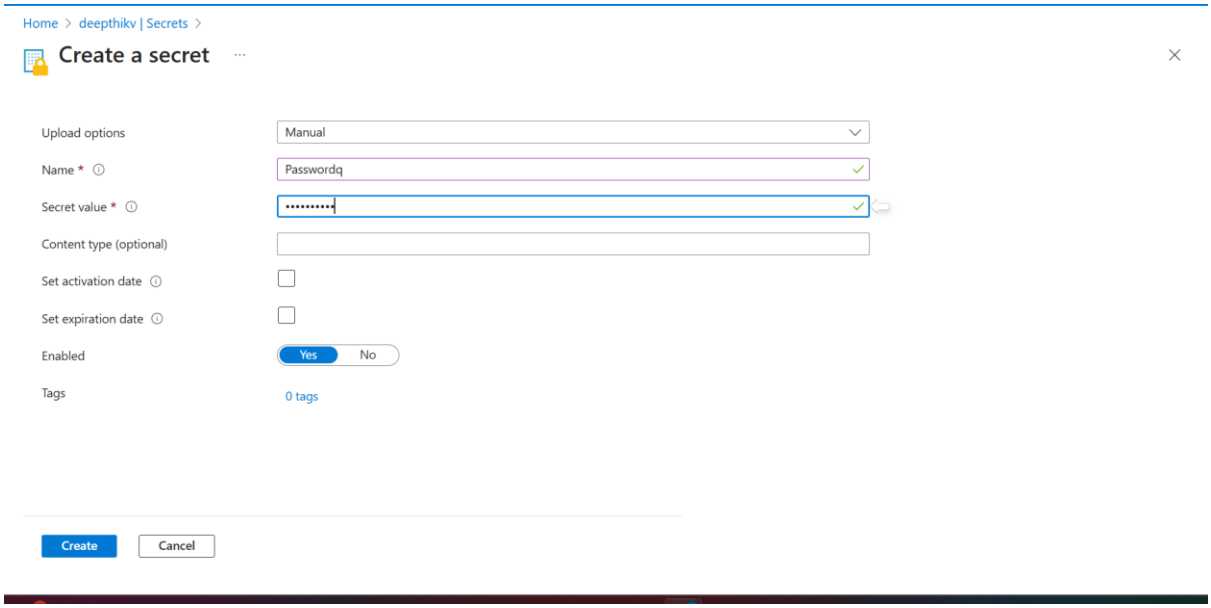
URL key:



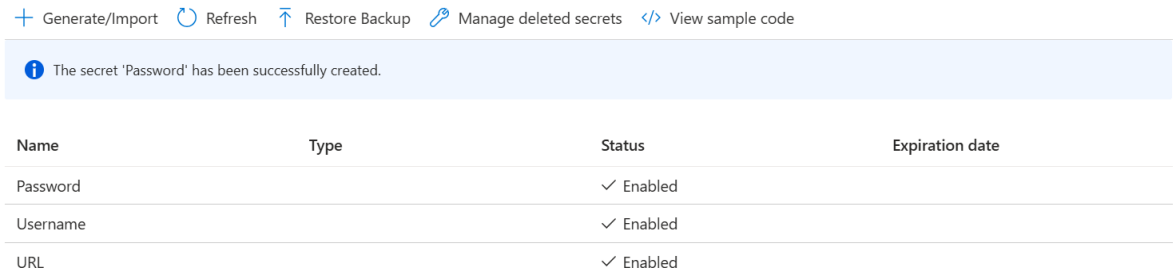
Username:



Password:

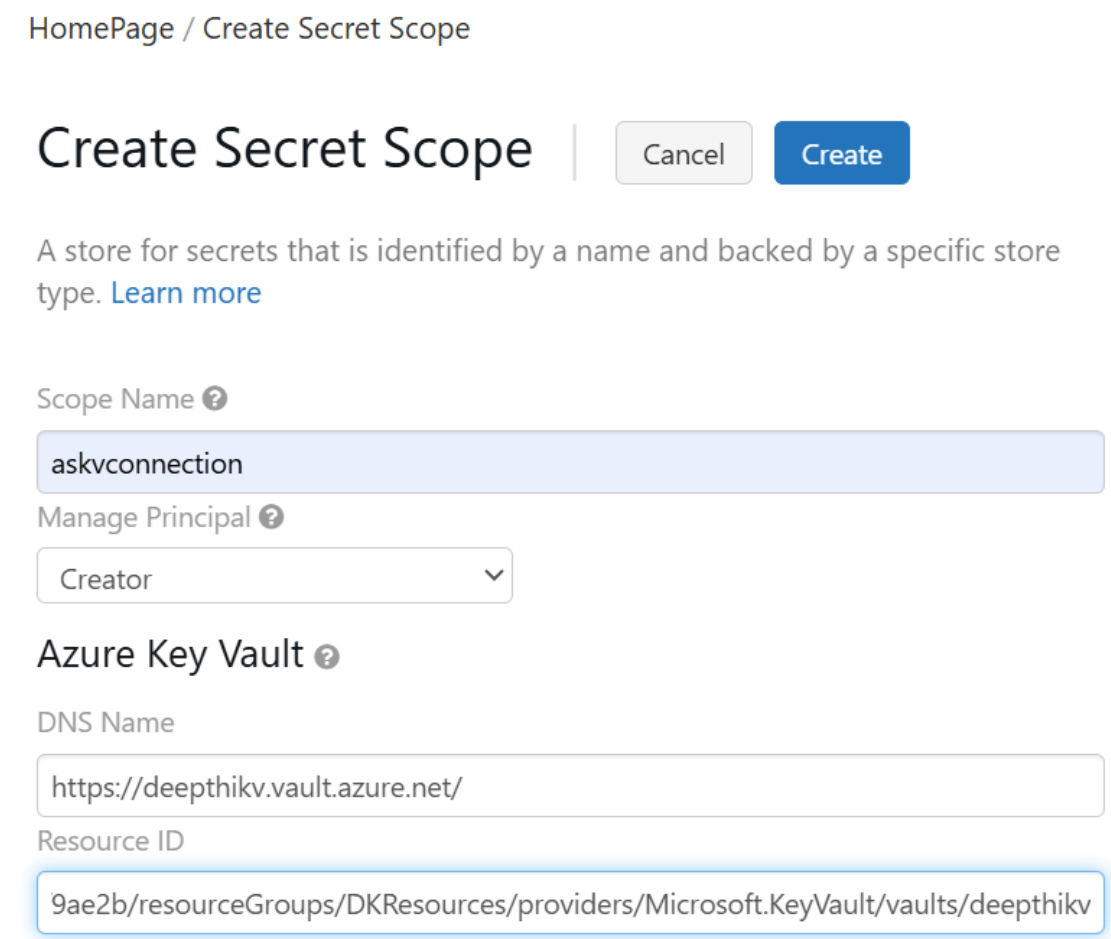


3 keys are created



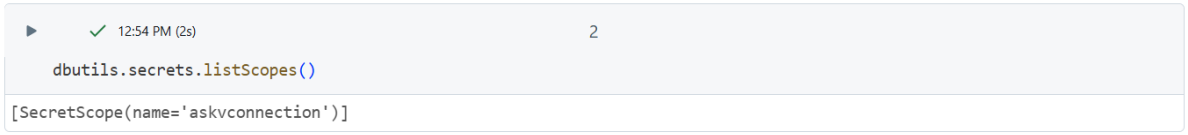
Now go to data bricks and create scope is it is not created

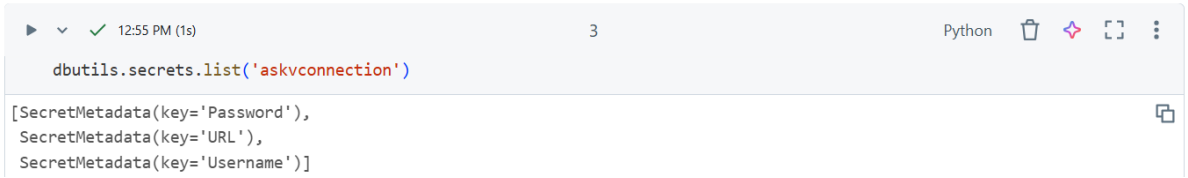
Using: https://adb-3370721355401501.1.azuredatabricks.net/?o=3370721355401501#secrets/createScope



Now open a notebook and check if the scope is created correctly using

Dbutils.secrets.listScope() command





Now we need to make a connection with SQL using the key which we created

Using code

username = dbutils.secrets.get(scope = "askvconnection", key = "Username")

password = dbutils.secrets.get(scope = "askvconnection", key = "Password")

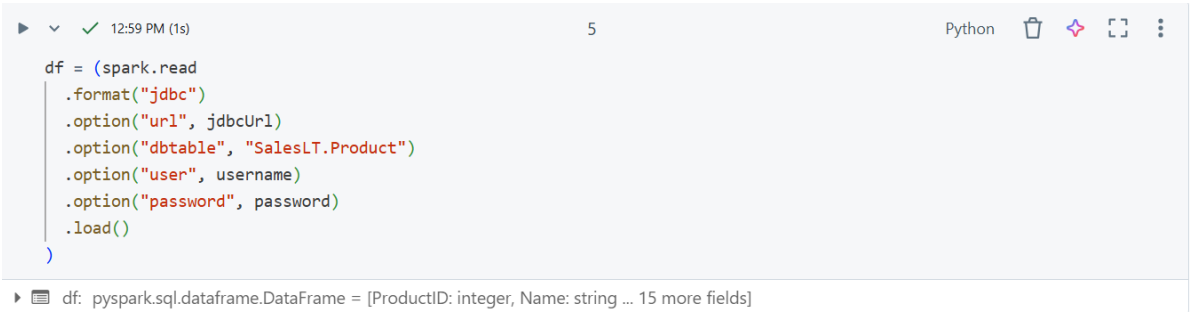
```
jdbcUrl = dbutils.secrets.get(scope = "askvconnection", key = "URL")


df = (spark.read
  .format("jdbc")
  .option("url", jdbcUrl)
  .option("dbtable", "SalesLT.Product")
  .option("user", username)
  .option("password", password)
  .load()
)
```

Instead of creating variables separately we directly give the key values in read code

Like below

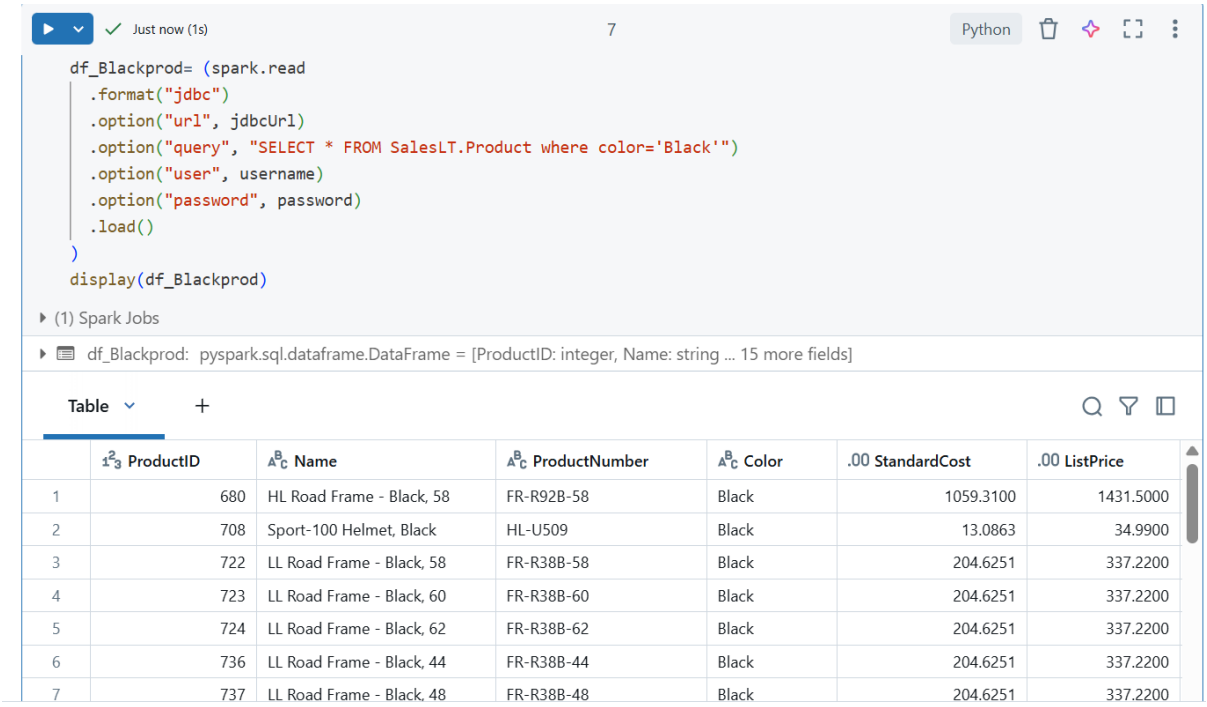
```
df = (spark.read
  .format("jdbc")
  .option("url", dbutils.secrets.get(scope = "askvconnection", key = "URL"))
  .option("dbtable", "SalesLT.Product")
  .option("user", dbutils.secrets.get(scope = "askvconnection", key = "Username"))
  .option("password", dbutils.secrets.get(scope = "askvconnection", key = "Password"))
  .load()
)
```



In the above code we are using table name directly instead we can use a query to get the data as well

```
df_Blackprod= (spark.read
  .format("jdbc")
```

```
  .option("url", jdbcUrl)
  .option("query", "SELECT * FROM SalesLT.Product where color='Black'")
  .option("user", username)
  .option("password", password)
  .load()
)
display(df_Blackprod)
```



	ProductID	Name	ProductNumber	Color	StandardCost	ListPrice
1	680	HL Road Frame - Black, 58	FR-R92B-58	Black	1059.3100	1431.5000
2	708	Sport-100 Helmet, Black	HL-U509	Black	13.0863	34.9900
3	722	LL Road Frame - Black, 58	FR-R38B-58	Black	204.6251	337.2200
4	723	LL Road Frame - Black, 60	FR-R38B-60	Black	204.6251	337.2200
5	724	LL Road Frame - Black, 62	FR-R38B-62	Black	204.6251	337.2200
6	736	LL Road Frame - Black, 44	FR-R38B-44	Black	204.6251	337.2200
7	737	LL Road Frame - Black, 48	FR-R38B-48	Black	204.6251	337.2200

We can also write back to SQL using code:

```
(df_Blackprod.write
  .format("jdbc")
  .option("url", jdbcUrl)
  .option("dbtable", "dbo.BlackProduct")
  .option("user", username)
  .option("password", password)
  .save()
)
```

Here we are creating a table dbo.BlackProduct with only black products data

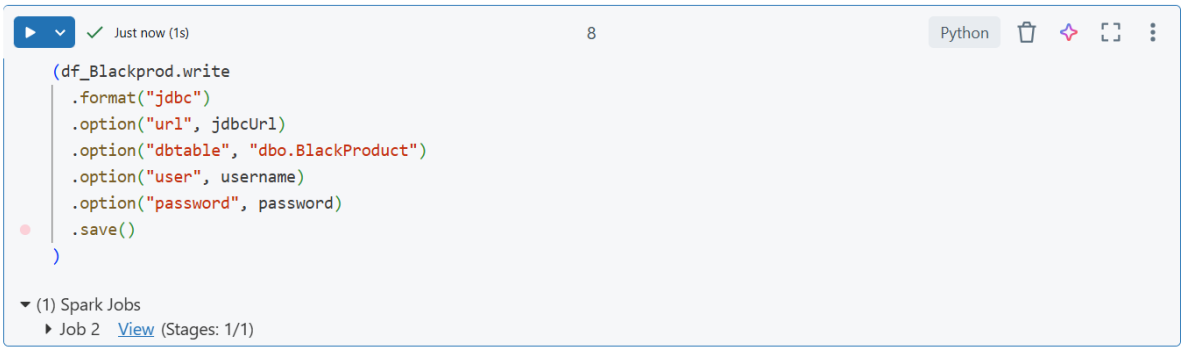
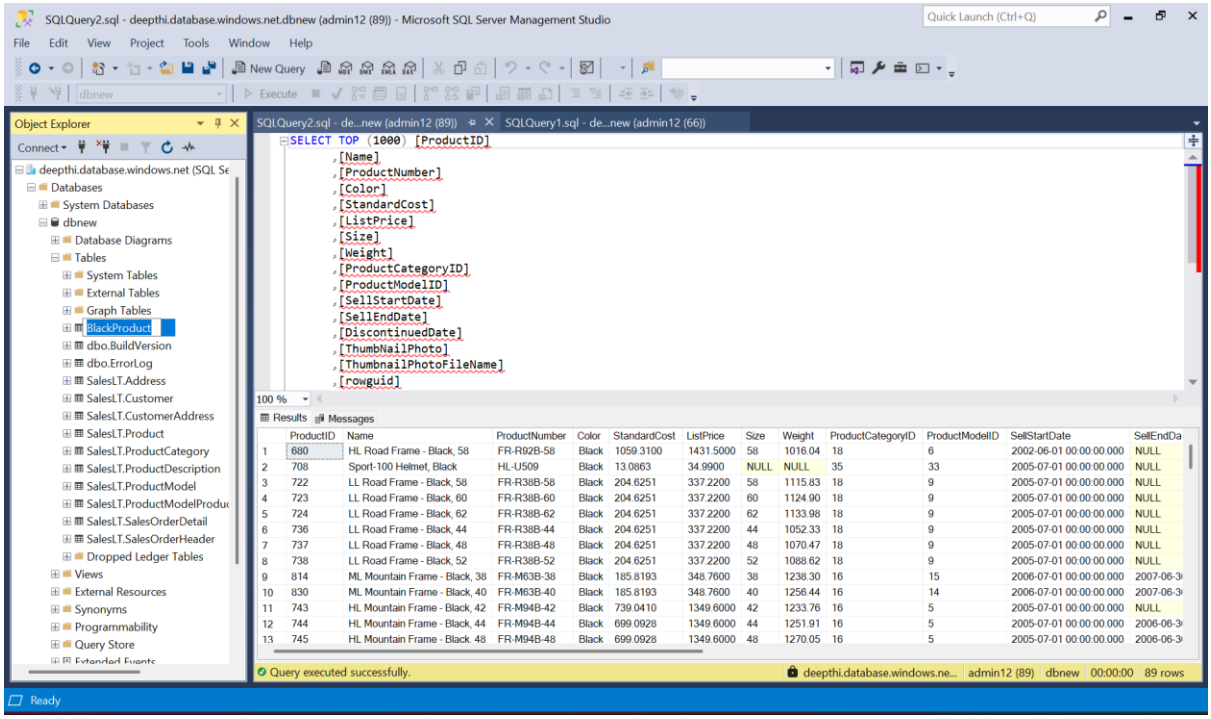


Table is created in SQL database we can check by using SSMS



As read and write codes can be used multiple times for different tables, we can use functions to reuse the same code multiple times, we can achieve this by writing the reuseable code in different notebook and use it in any notebook.

We define function suing def key word.

```
def read_sql(tablename):
  df = (spark.read
    .format("jdbc")
    .option("url", dbutils.secrets.get(scope = "askvconnection", key = "URL"))
    .option("dbtable", tablename)
    .option("user", dbutils.secrets.get(scope = "askvconnection", key = "Username"))
    .option("password",dbutils.secrets.get(scope = "askvconnection", key = "Password"))
```

```
  .load()
 )
 return(df)
```

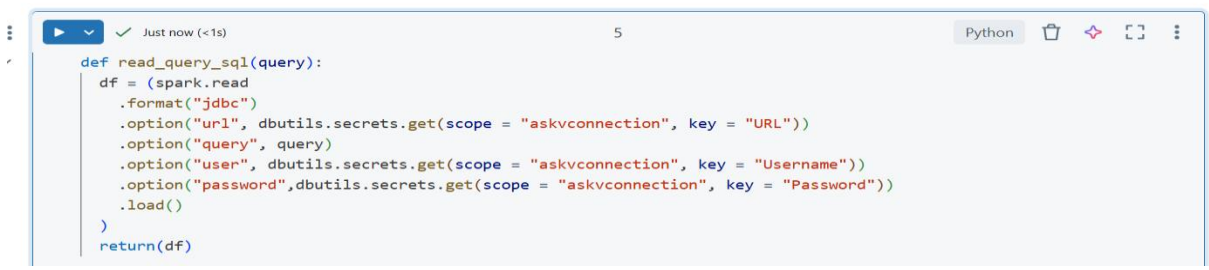
This is how a function is defined, run this code and can call this function.

Read using table name

Read data using query

```
def read_query_sql(query):
 df = (spark.read
  .format("jdbc")
  .option("url", dbutils.secrets.get(scope = "askvconnection", key = "URL"))
  .option("query", query)
  .option("user", dbutils.secrets.get(scope = "askvconnection", key = "Username"))
  .option("password",dbutils.secrets.get(scope = "askvconnection", key = "Password"))
  .load()
 )
 return(df)
```

Read using Query



Write to SQL

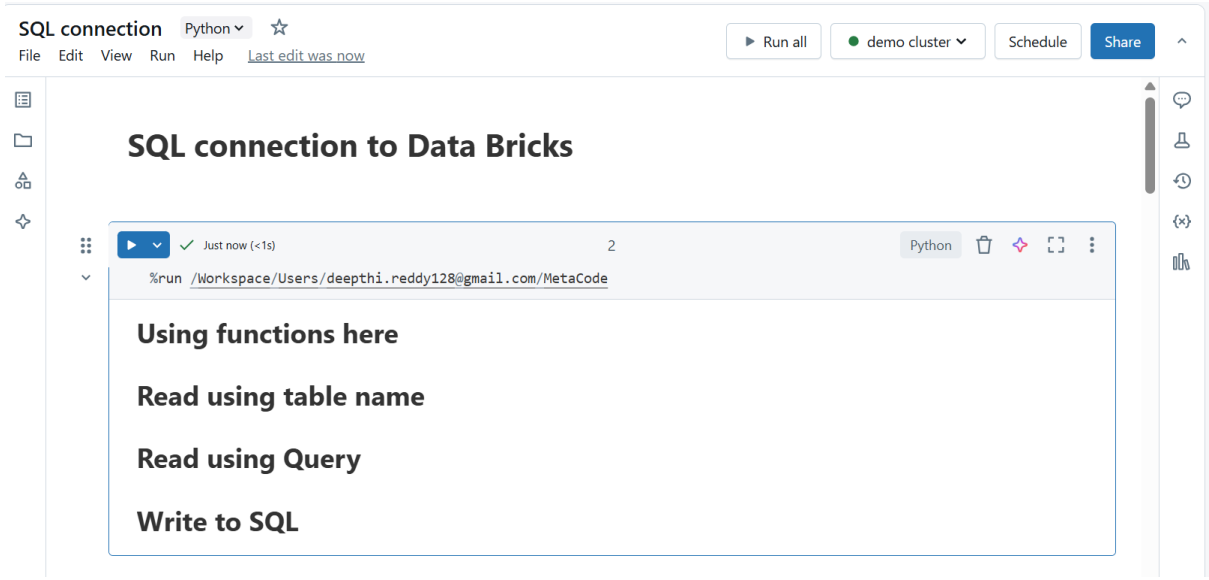
Write to SQL

```
def write_sql(df,mode, tablename):
    (df.write
        .mode(mode)
        .format("jdbc")
        .option("url", dbutils.secrets.get(scope = "askvconnection", key = "URL"))
        .option("dbtable", tablename)
        .option("user", dbutils.secrets.get(scope = "askvconnection", key = "Username"))
        .option("password",dbutils.secrets.get(scope = "askvconnection", key = "Password"))
        .save()
    )
```

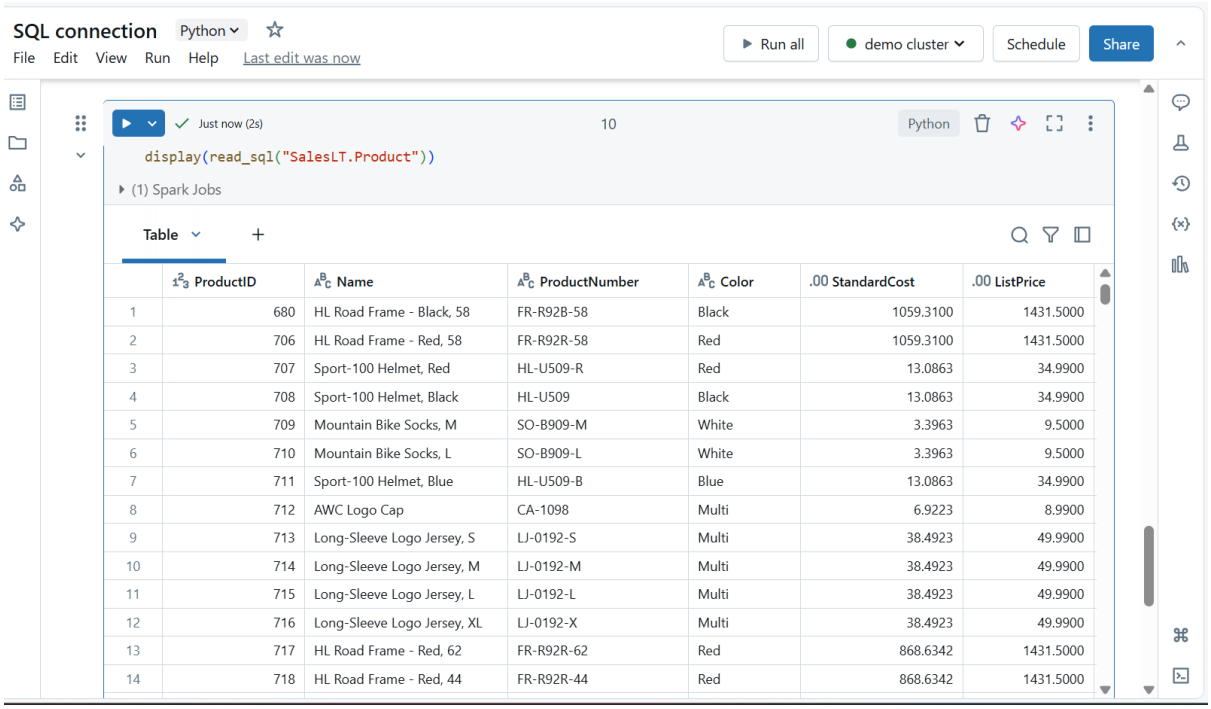
Now to use these functions code in another notebook we need to use line command %run/notebook path

%run//Workspace/Users/deepthi.reddy128@gmail.com/MetaCode

Use this code in the notebook where we want to use the reusable code

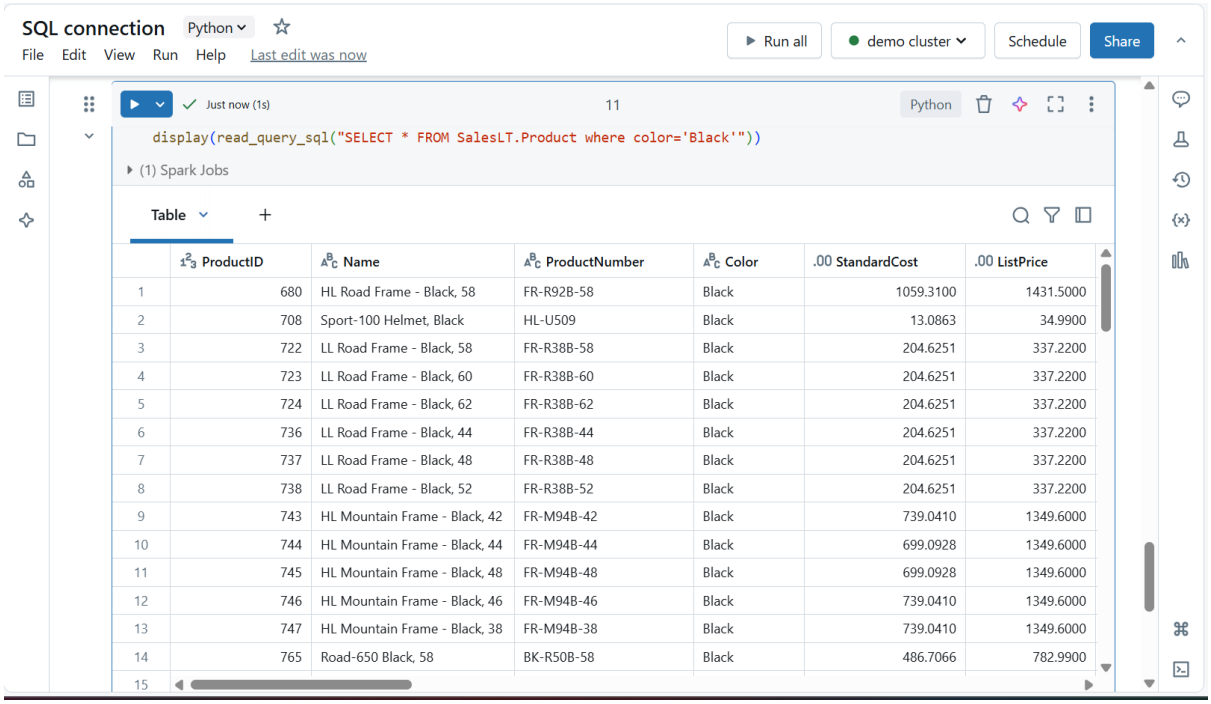


Using read_sql() function in another notebook to read data



	ProductID	Name	ProductNumber	Color	StandardCost	ListPrice
1	680	HL Road Frame - Black, 58	FR-R92B-58	Black	1059.3100	1431.5000
2	706	HL Road Frame - Red, 58	FR-R92R-58	Red	1059.3100	1431.5000
3	707	Sport-100 Helmet, Red	HL-U509-R	Red	13.0863	34.9900
4	708	Sport-100 Helmet, Black	HL-U509	Black	13.0863	34.9900
5	709	Mountain Bike Socks, M	SO-B909-M	White	3.3963	9.5000
6	710	Mountain Bike Socks, L	SO-B909-L	White	3.3963	9.5000
7	711	Sport-100 Helmet, Blue	HL-U509-B	Blue	13.0863	34.9900
8	712	AWC Logo Cap	CA-1098	Multi	6.9223	8.9900
9	713	Long-Sleeve Logo Jersey, S	LJ-0192-S	Multi	38.4923	49.9900
10	714	Long-Sleeve Logo Jersey, M	LJ-0192-M	Multi	38.4923	49.9900
11	715	Long-Sleeve Logo Jersey, L	LJ-0192-L	Multi	38.4923	49.9900
12	716	Long-Sleeve Logo Jersey, XL	LJ-0192-X	Multi	38.4923	49.9900
13	717	HL Road Frame - Red, 62	FR-R92R-62	Red	868.6342	1431.5000
14	718	HL Road Frame - Red, 44	FR-R92R-44	Red	868.6342	1431.5000

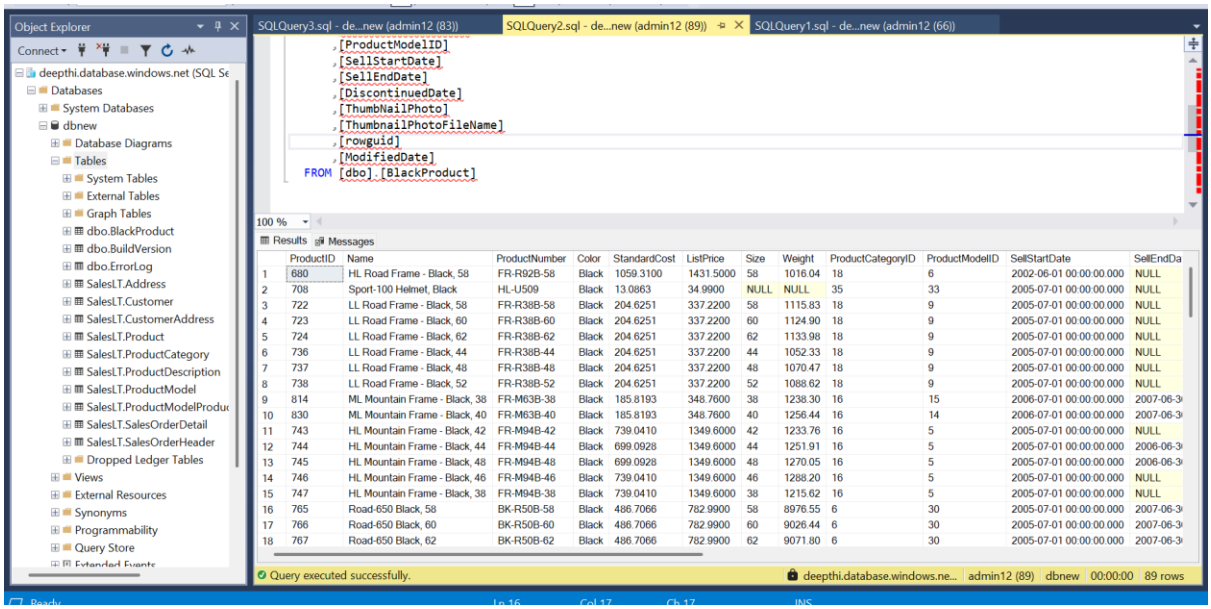
Using read_query_sql() function



	ProductID	Name	ProductNumber	Color	StandardCost	ListPrice
1	680	HL Road Frame - Black, 58	FR-R92B-58	Black	1059.3100	1431.5000
2	708	Sport-100 Helmet, Black	HL-U509	Black	13.0863	34.9900
3	722	LL Road Frame - Black, 58	FR-R38B-58	Black	204.6251	337.2200
4	723	LL Road Frame - Black, 60	FR-R38B-60	Black	204.6251	337.2200
5	724	LL Road Frame - Black, 62	FR-R38B-62	Black	204.6251	337.2200
6	736	LL Road Frame - Black, 44	FR-R38B-44	Black	204.6251	337.2200
7	737	LL Road Frame - Black, 48	FR-R38B-48	Black	204.6251	337.2200
8	738	LL Road Frame - Black, 52	FR-R38B-52	Black	204.6251	337.2200
9	743	HL Mountain Frame - Black, 42	FR-M94B-42	Black	739.0410	1349.6000
10	744	HL Mountain Frame - Black, 44	FR-M94B-44	Black	699.0928	1349.6000
11	745	HL Mountain Frame - Black, 48	FR-M94B-48	Black	699.0928	1349.6000
12	746	HL Mountain Frame - Black, 46	FR-M94B-46	Black	739.0410	1349.6000
13	747	HL Mountain Frame - Black, 38	FR-M94B-38	Black	739.0410	1349.6000
14	765	Road-650 Black, 58	BK-R50B-58	Black	486.7066	782.9900
15						

Using write_sql() function

To check if the data is written to SQL need to check using SSMS



Only black color data is stored in dbo.BlackProduct table