IBM

ICP for Data Data Virtualization Demo: Stock Transactions

Prepared for Fast Start 2019 Hands On Lab Session
January 17, 2019

IBM CLOUD PRIVATE FOR DATA - DATA VIRTUALIZATION

Objective

With data virtualization, you can query data across many systems without having to copy and replicate data, which increases productivity reduces cost and complexity. It also ensures that Analytics are performed on accurate and up-to-date data (vs. data that was copied at some prior time).

What Will This Demo Show?

1. **Data Virtualization** over widely distributed data
2. **SQL editor** editing and running live queries
3. **Jupyter Notebook** connecting, querying and plotting

In this IBM Cloud Private for Data demo you will:

- Provision Data Virtualization
- Create a new Project
- Add a new Data Source
- Add Virtualized tables + Assign to a Project
- Create a Virtualized Join View + Assign to a Project
- Assign existing Virtualized tables to a Project
- Add a Notebook to a Project + Run SQL Queries
- Tour additional menus in Virtualized data (i.e. SQL Editor, Manage Users, etc.)

Set Up

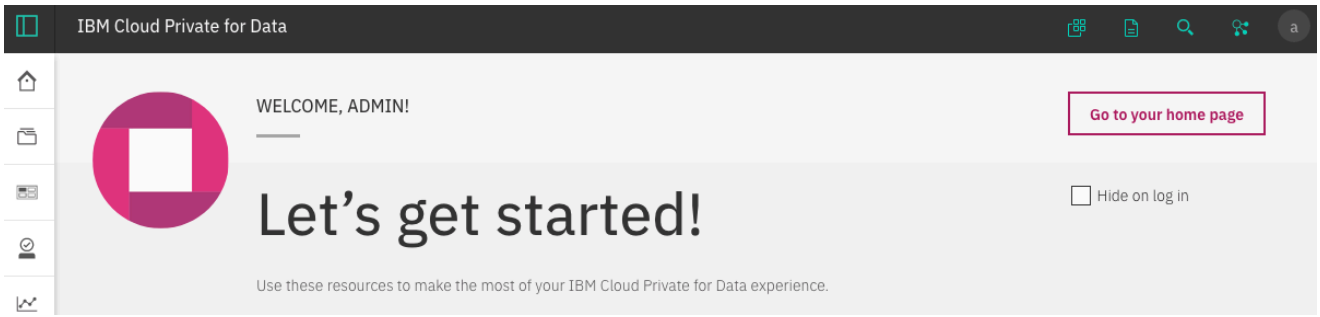
- All of these items should come prepared for the demo to function accordingly
- **Scale:** 4 distributed data sources
- **Database:** Db2 Family (3 Db2 on Cloud sources on the IBM Cloud), Db2 Warehouse (AWS)

Transactional Data

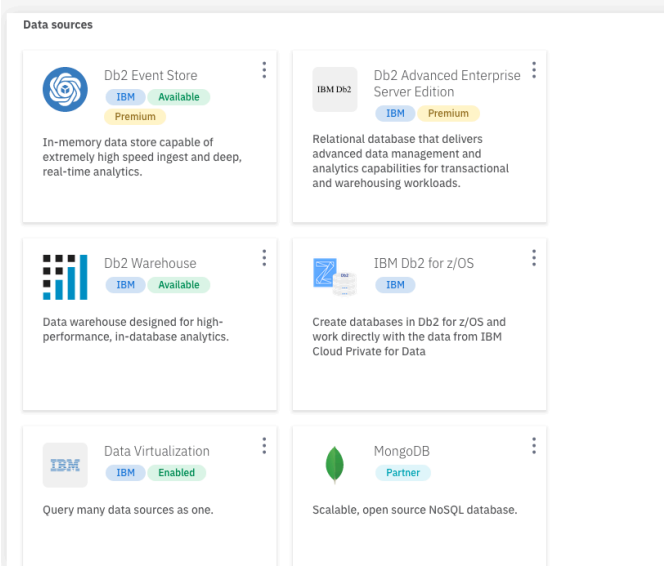
- In this demo we are utilizing stock market transaction data

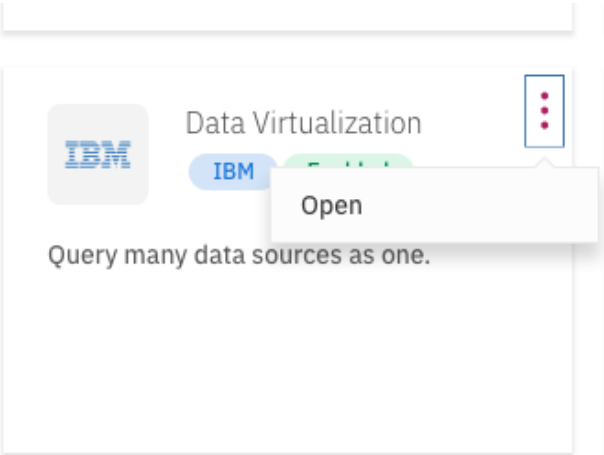
Provision Data Virtualization

1) After logging into your ICP for Data cluster, select the **"Add-ons"** button from the menu bar at the top of the window. It is the left most teal icon button on the top right of your screen.



2) Scroll down into the **Data Sources** section until you see **Data Virtualization**. Click the 3 vertical menu dots and click on the **"Open"** button.

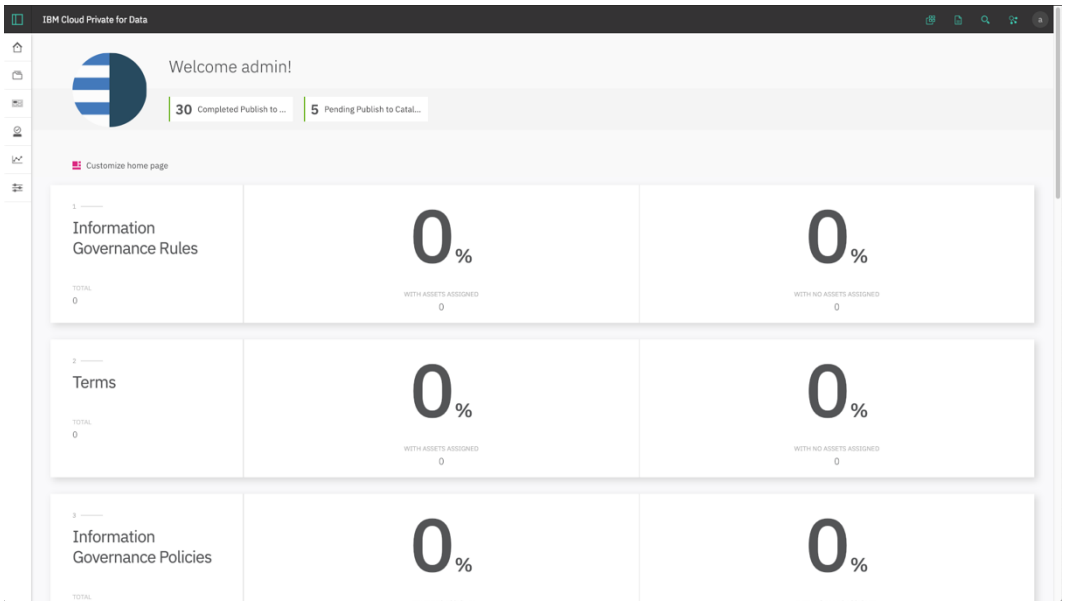




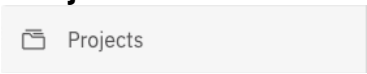
3) The service is now being provisioned and this may take several minutes. Exit from ICP for Data and then login again with the credentials that you have been provided with. This is to allow for the Data Virtualization menu option to be created and accessible.

Create a new Project

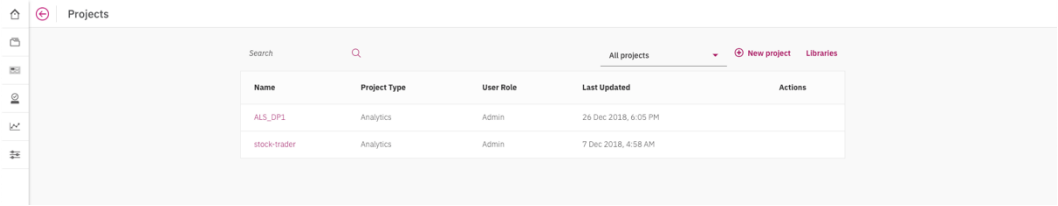
4) Start from the ICP for Data Home Screen



5) We need to navigate to the Project section to create a new project which later will contain multiple datasets and a Jupyter notebook for SQL analysis. Click on the "**Projects**" icon from the left menu, followed by "**Projects**".

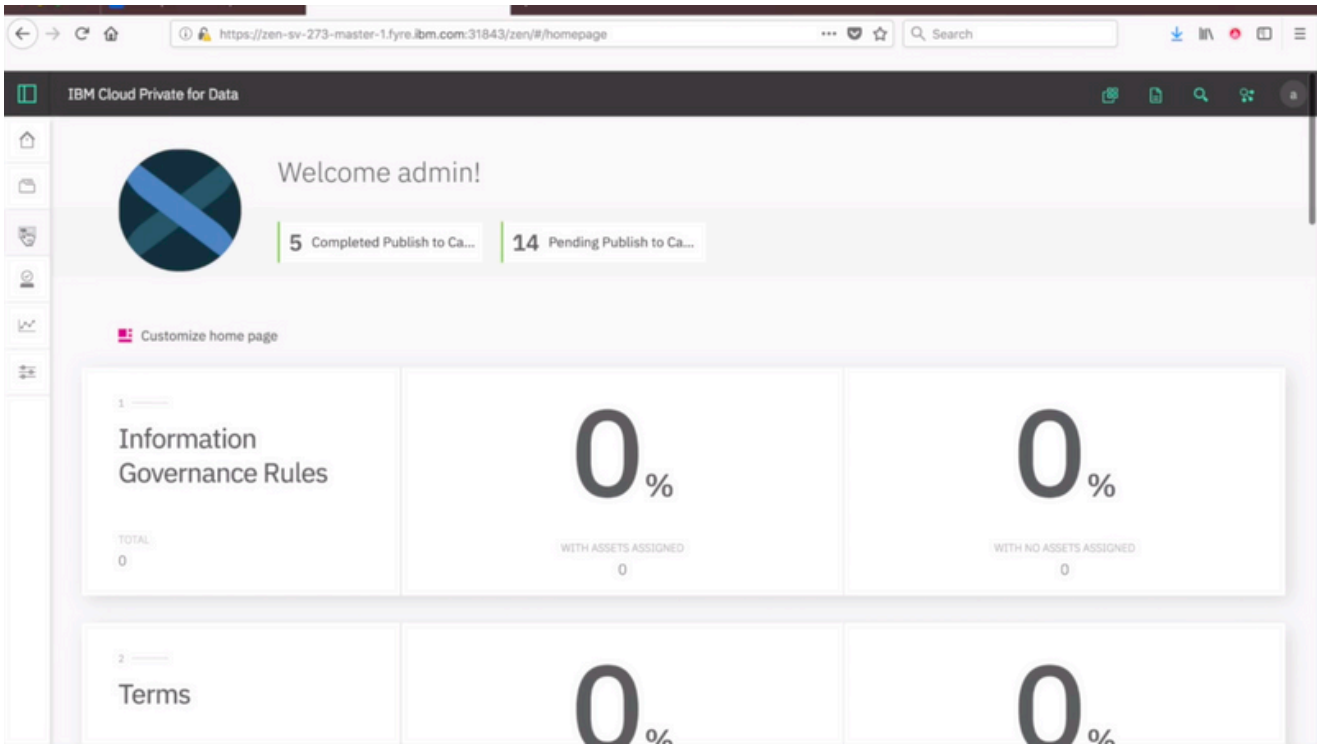


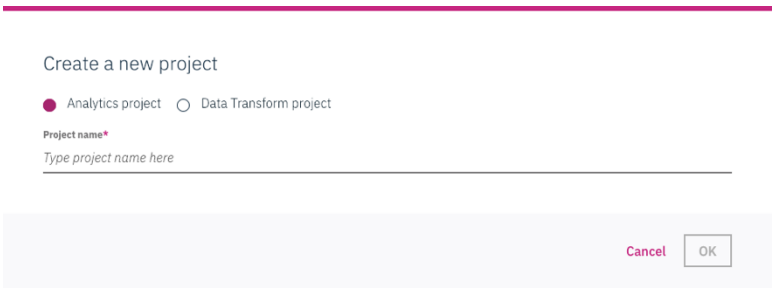
6) The "Projects" screen opens:



7) In "Projects", you can see the already defined projects and their types. Click on New project "**New project**" to begin.

8) Once the "Create a new project" dialog opens, click into the Project name box and "**stock-history**" will automatically be entered for you. This project will be created with the default type of "Analytics project".

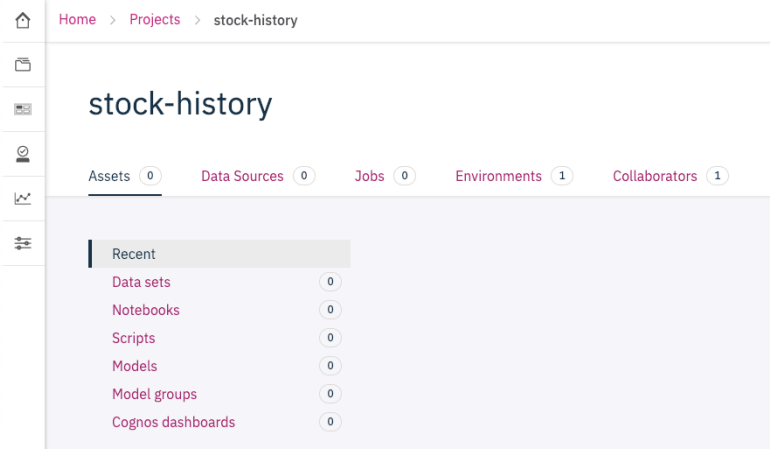




9) Click on the "**Ok**" button to continue.

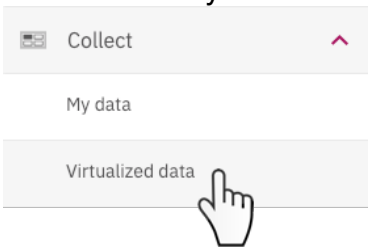
10) From the summary screen you are provided options to enter a description, load an existing project from a file or create a project from a Git repository. We are just going to create a blank project so click "**Create**" to continue.

11) The “stock-history” project is now complete. But as you can see it does not contain any assets. Next, we will add data sets.

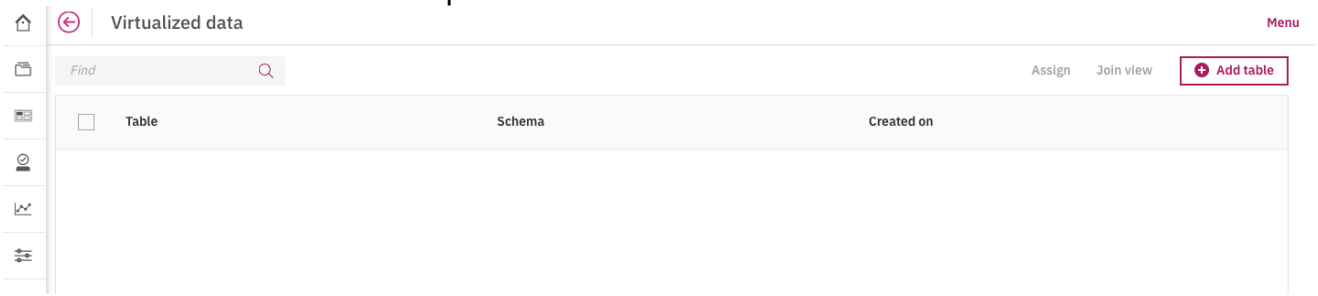


Add a new Data Source

1) This section begins from the “stock-history” project screen. We need to navigate to the virtualized data section to add a new data source. Click on the ”**Collect**” icon on the left followed by “**Virtualized data**”:



2) The “Virtualized data” screen opens and will show no table entries.

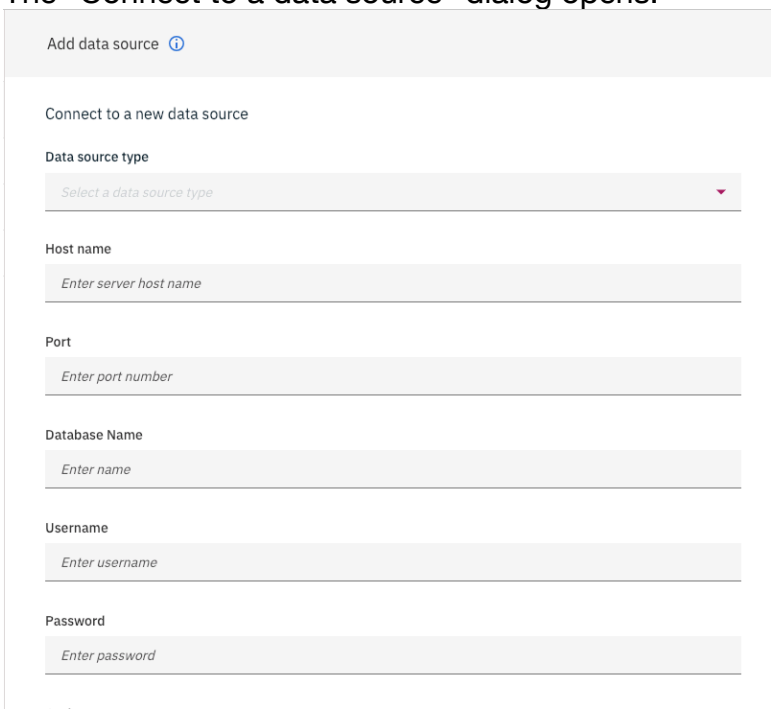


3) We need to add our data sources from AWS and the IBM Cloud. Click on the “**Menu**” button in the upper right followed by “**Data sources**”.

4) We now want to add a new data source. Click on the Add data source “**+ Add data source**” button to begin.

5) From the Add data source screen you are prompted to find an existing data source, we do not have one so click on the “**New data source**” link to continue.

6) The “Connect to a data source” dialog opens:



7) First we will add a data source from AWS. Click the “**Select a data source type**” dropdown, followed by “**Db2 Family**”.

   a. Click into the “**Enter server host name**” field. **18.216.114.67**

   b. Click into the “**Enter port number**” field. **50000**

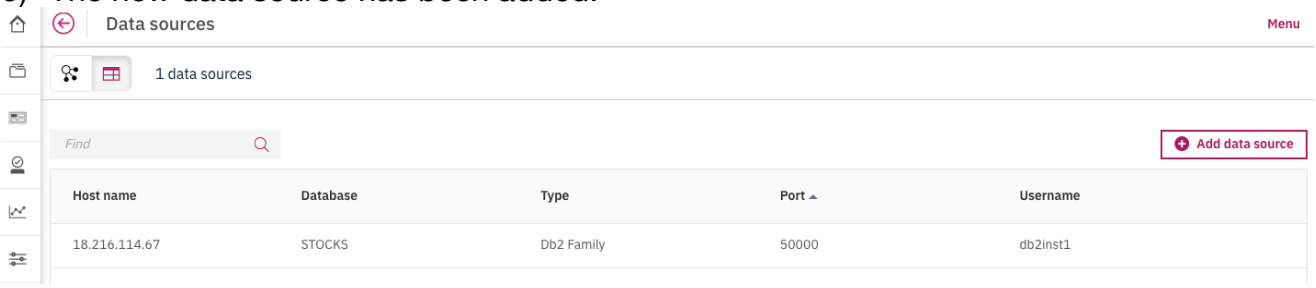
   c. Click into the “**Enter name**” field. **STOCKS**

   d. Click into the “**Enter username**” field. **db2inst1**

   e. Click into the “**Enter password**” field. **thinkfast2019**

8) Finally, click on the “**Add**” button to continue.

9) The new data source has been added:



10) We need to add the additional data sources. First Db2 on Cloud in the Dallas data center.

    a. Click the "**Select a data source type**" dropdown, followed by "**Db2 Family**".

    b. Click into the "**Enter server host name**" field. **dashdb-txn-flex-yp-dal10-522.services.dal.bluemix.net**

    c. Click into the "**Enter port number**" field. **50000**

    d. Click into the "**Enter name**" field. **BLUDB**

    e. Click into the "**Enter username**" field. **bluadmin**

    f. Click into the "**Enter password**" field. **MGEyMWQ1ZmU1MDlj**

    g. Finally, click on the "**Add**" button to continue.

11) Next another Db2 on Cloud in the Dallas data center.

    a. Click the "**Select a data source type**" dropdown, followed by "**Db2 Family**".

    b. Click into the "**Enter server host name**" field. **dashdb-txn-flex-yp-dal10-437.services.dal.bluemix.net**

    c. Click into the "**Enter port number**" field. **50000**

    d. Click into the "**Enter name**" field. **BLUDB**

e. Click into the “**Enter username**” field. **bluadmin**

f. Click into the “**Enter password**” field. **ODdmMmEyN2RlNDJl**

g. Finally, click on the “**Add**” button to continue.

12) And a third Db2 on Cloud in the Dallas data center.

a. Click the “**Select a data source type**” dropdown, followed by “**Db2 Family**”.

b. Click into the “**Enter server host name**” field. **dashdb-txn-flex-yp-wdc04-10.services.dal.bluemix.net**

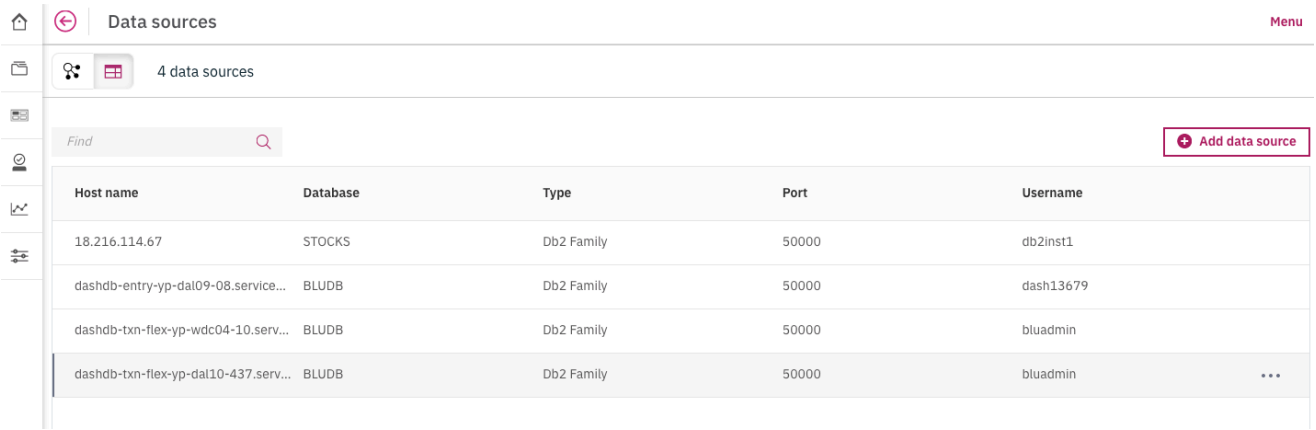
c. Click into the “**Enter port number**” field. **50000**

d. Click into the “**Enter name**” field. **BLUDB**

e. Click into the “**Enter username**” field. **bluadmin**

f. Click into the “**Enter password**” field. **OWZiMWI3YzdmYzU0**

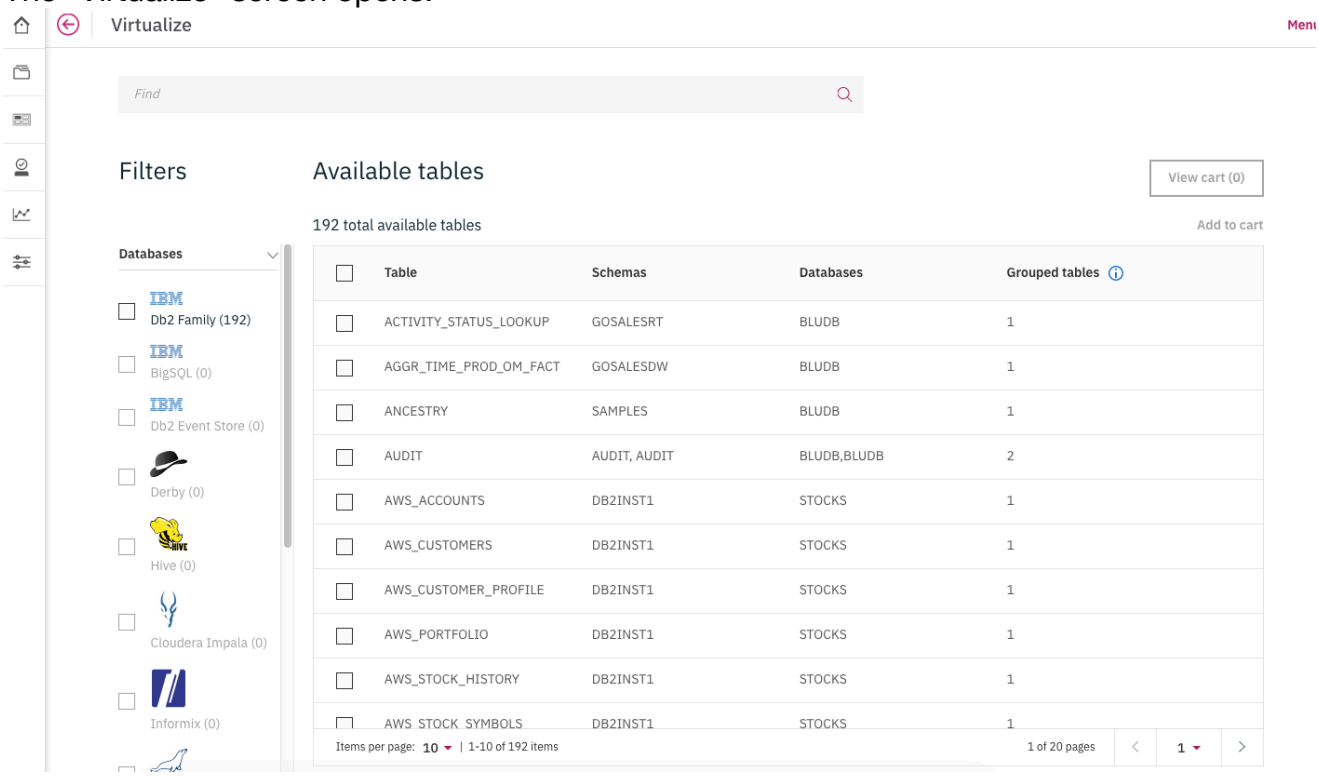
g. Finally, click on the “**Add**” button to continue.



Add Virtualized tables + Assign to a Project

1) This section begins from the “Data sources” screen. We need to navigate to the Virtualize section to add tables from the new data source. Click on the “**Menu**” button in the upper right followed by “**Virtualize**”.

2) The “Virtualize” screen opens:



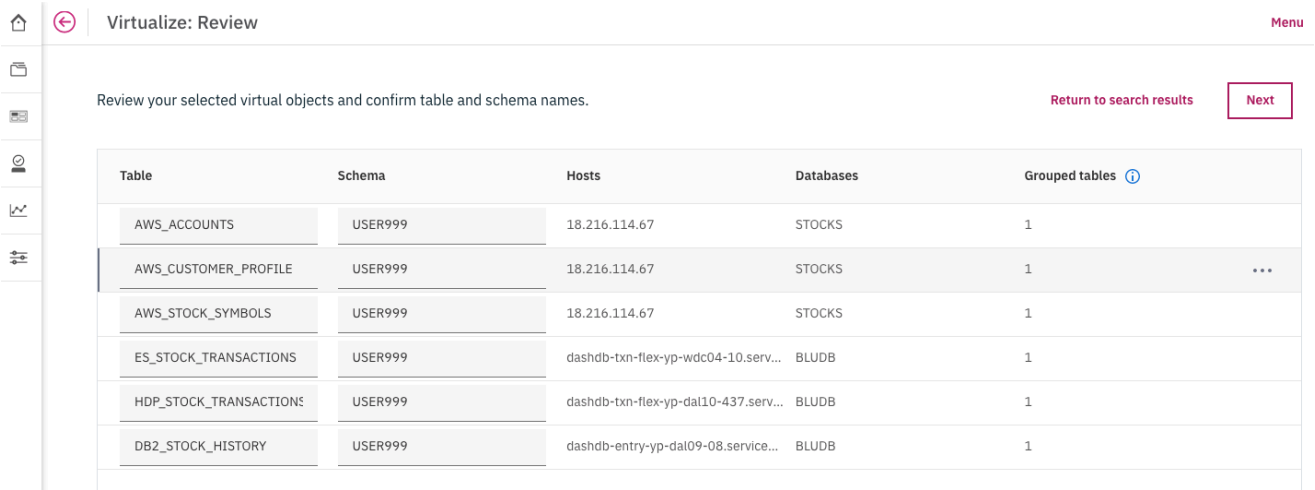
3) This display shows all physical tables across all of the data sources that are available for virtualization. Note the various filters available to narrow a search and also the search field to enter a specific physical table to search for.

4) Select the following checkboxes:
“AWS_ACCOUNTS”
“AWS_CUSTOMER_PROFILE”
“AWS_STOCK_SYMBOLS”
“DB2_STOCK_HISTORY”
“HDP_STOCK_TRANSACTIONS”
“ES_STOCK_TRANSACTIONS”

5) Once the above tables are selected, click on the "**Add to cart**" link in the upper right.

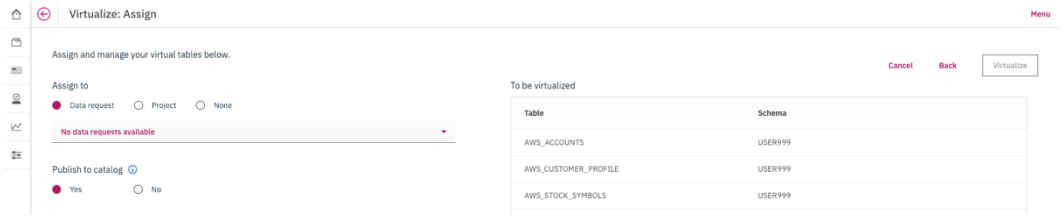
6) Click on the View cart (3) "**View Cart (6)**" button to continue.

7) The "Virtualize: Review" screen opens:



8) On the "Virtualize: Review" screen you have the option of assigning each of the virtual tables a new Name and/or change the Schema. Select one of the tables and the "**...**" table actions button to the right and then "**Edit column names**" to view any of the tables and see how they can be edited then **Cancel** to return to the Review screen. We are going to leave the settings as is, click on the Next "**Next**" button in the upper right to continue.

9) The "Virtualize: Assign" screen opens:



10) On the "Virtualize: Assign" screen you have multiple options to assign the virtualized assets including whether or not to publish to catalog. More information about each of these options below. For this demo, click on the "**Project**" radio button in the Assign to section.

Assign to	When to use this option
Data request	Select **Data request** if you created the virtualized table in response to a data request. Then, choose the appropriate request.
Project	Select **Project** if you created the virtualized table to use in a specific analytics project. Then, choose the appropriate project.
None	Select **None** if the table was not created in response to a data request or to use in a specific project. This is the default setting if no data requests or projects exist.

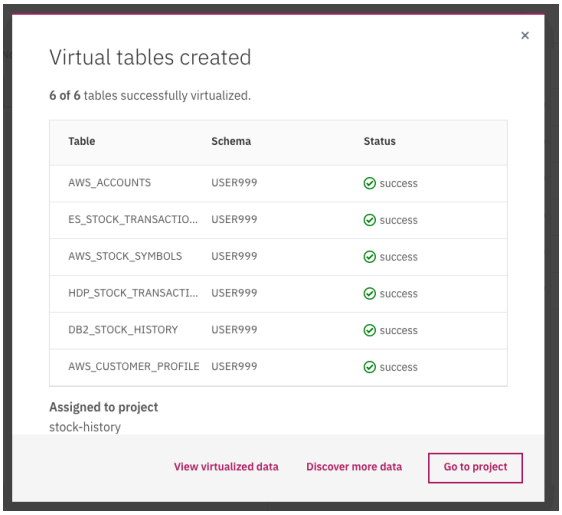
Publish to Catalog	A request is sent for approval to publish to the data catalog. A user with Manage catalog permissions, such as a data steward, will see Pending Publish to Catalog requests on their home page so they can approve it. When approved, the join view will be added to the data catalog.

11) Click on the "**Select a project**" dropdown list followed by "**stock-history**".

12) Click on the "**No**" radio button in the Publish to catalog section.

13) Click on the Virtualize "**Virtualize**" button to continue.

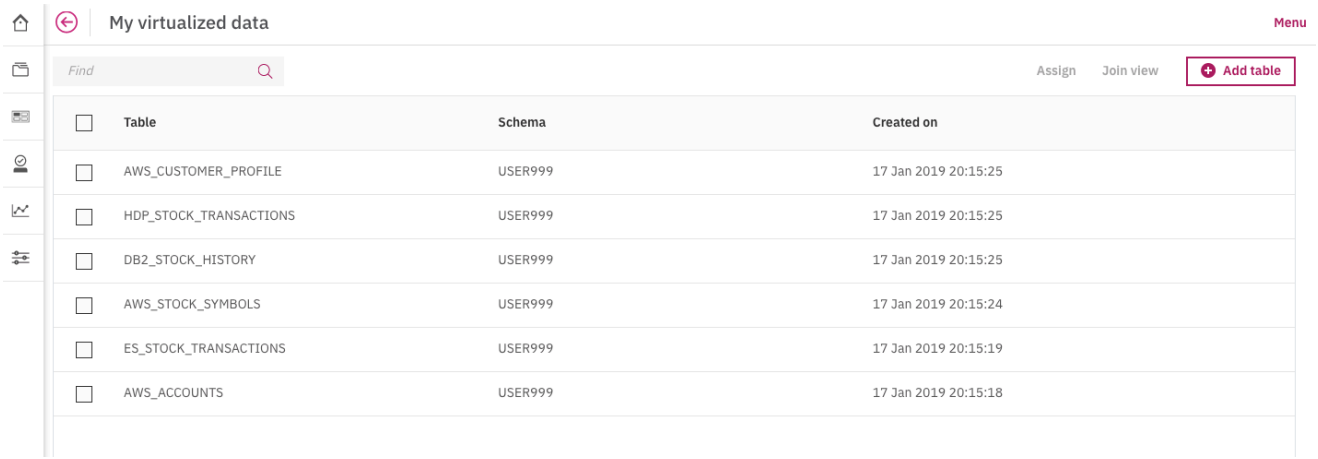
14) The "Virtual tables created" dialog opens:



15) On the “Virtual tables created” dialog, the tables were successfully created and assigned to the “stock-history” project. Click on “**View virtualized data**” link to complete.

Create a Virtualized Join View + Assign to a Project

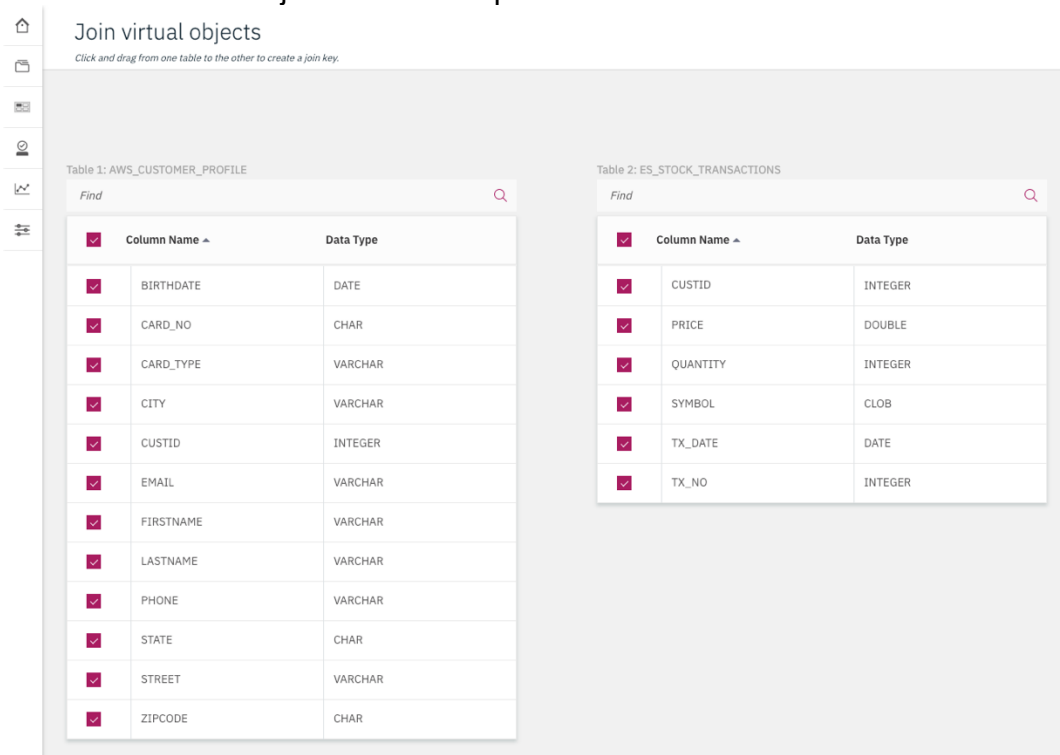
1) This section begins from the "Virtualized data" screen:



2) Select the "**AWS_CUSTOMER_PROFILE**" and "**ES_STOCK_TRANSACTIONS**" checkboxes (in this order).

3) Click on the "**Join view**" link in the upper right to continue.

4) The “Join virtual objects” screen opens:

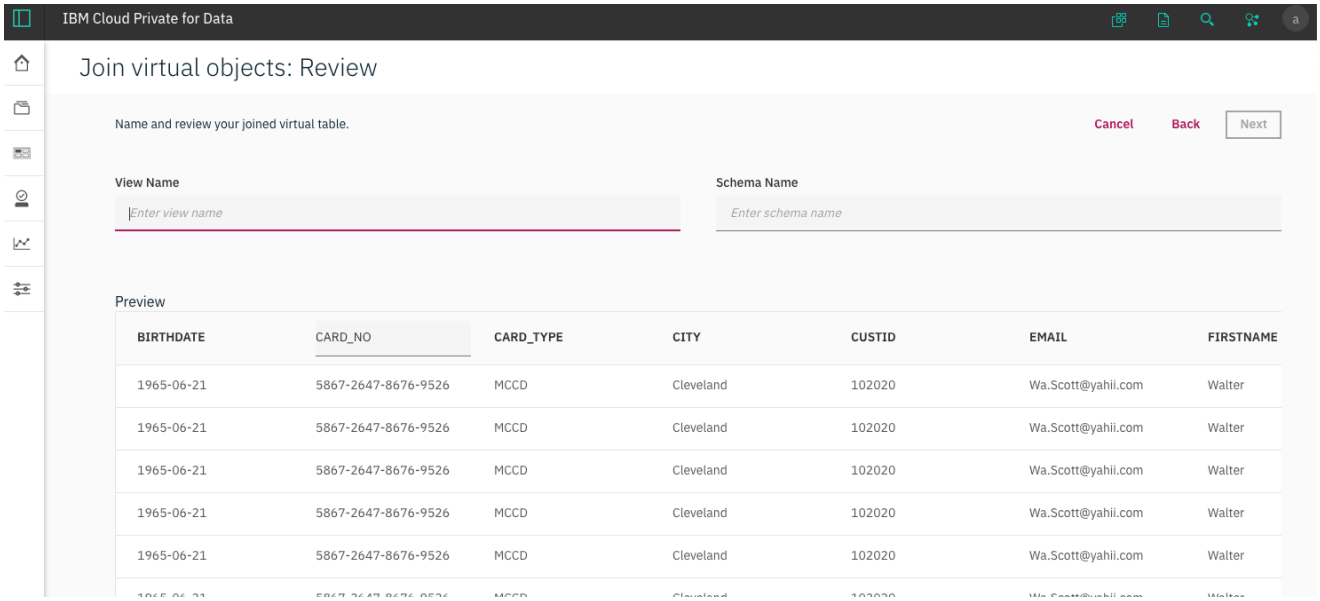


5) On the “Join virtual objects” screen, the two selected tables appear side-by-side and you can choose which columns will be included in the new Join view table. Click the “**CARD_NO**” checkbox to unselect it. This will remove and mask out this column from the virtual view that is created for users.

6) Click and hold anywhere on the “**CUSTID**” column in Table 1 AWS_CUSTOMER_PROFILE and drag your cursor across to the other table to join to the “**CUSTID**” column in Table 2 ES_STOCK_TRANSACTIONS:

7) Click the “**Join**” button and the connection link will disappear with “**CUSTID**” key in both tables. Click on the “**Preview**” link in the lower right to continue. It may then take several minutes for the preview of the Join operation to execute.

8) The “Join virtual objects: Review” page will be displayed.



9) On the “Join virtual objects: Review” screen, click into the “**Enter view name**” field and enter **“JV-AWS_CUSTOMER_PROFILE-ES_STOCK_TRANSACTIONS”**

10) Click into the “**Enter schema name**” field and enter **“USER999”**

11) Click on the Next “**Next**” button to continue.

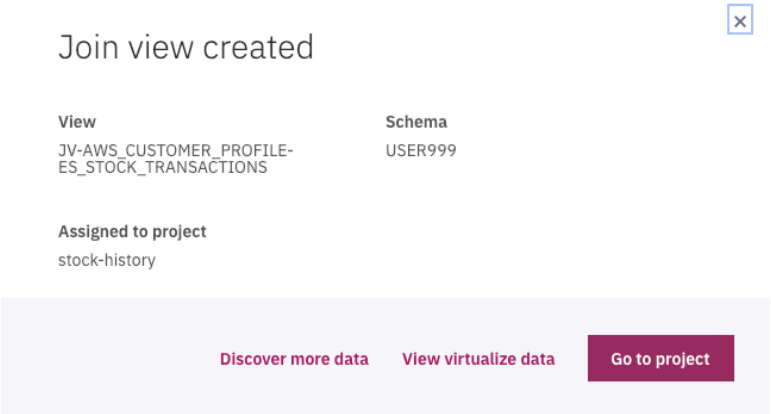
12) The “Join virtualize objects: Assign” screen opens:

13) On the “Join virtualize objects: Assign” screen, click on the “**Select a project**” dropdown list followed by “**stock-history**”.

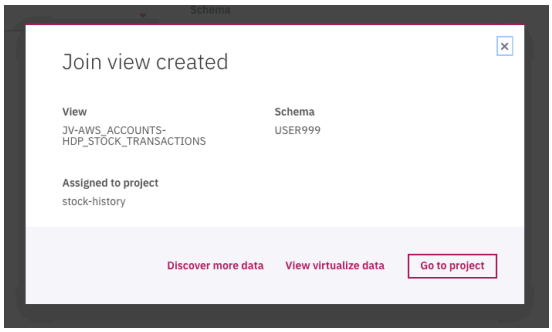
14) Click on the "**No**" radio button in the Publish to catalog section.

15) Click on the Create view "**Create view**" button to continue.

16) The "Join view created" dialog opens:

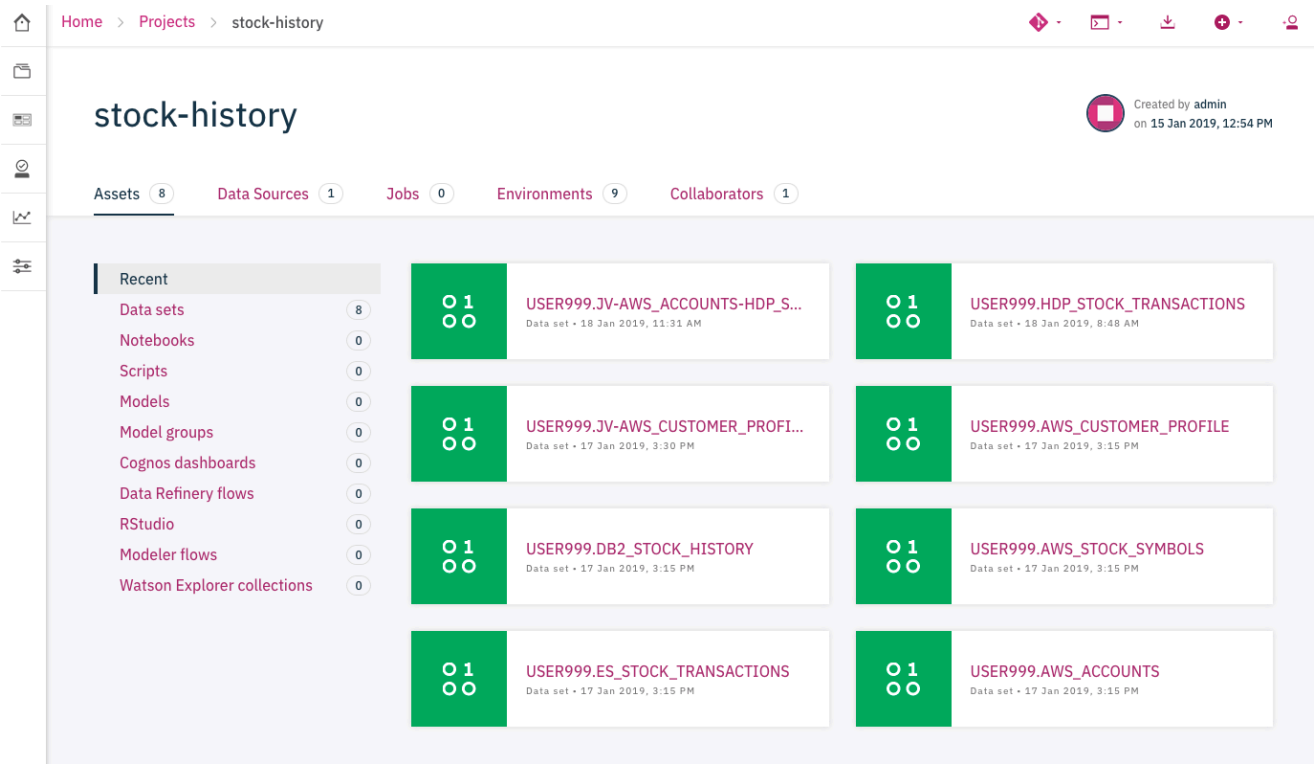


17) Return to the "Virtualized data" page and repeat these steps and in a similar fashion create a second Join View of the "**AWS_ACCOUNTS"** and **"HDP_STOCK_TRANSACTIONS"** tables with a Key of **"CUSTID"** and create a Join Table View called **"JV-AWS_ACCOUNTS-HDP_STOCK_TRANSACTIONS"** with schema **"USER999"** and add it to the **stock-history** project.



18) On the "Join view created" dialog, click on Go to project "**Go to project**" button to continue.

19) The "stock-history" project screen opens with 8 Data sets. We now have all of our virtual data views ready for use.

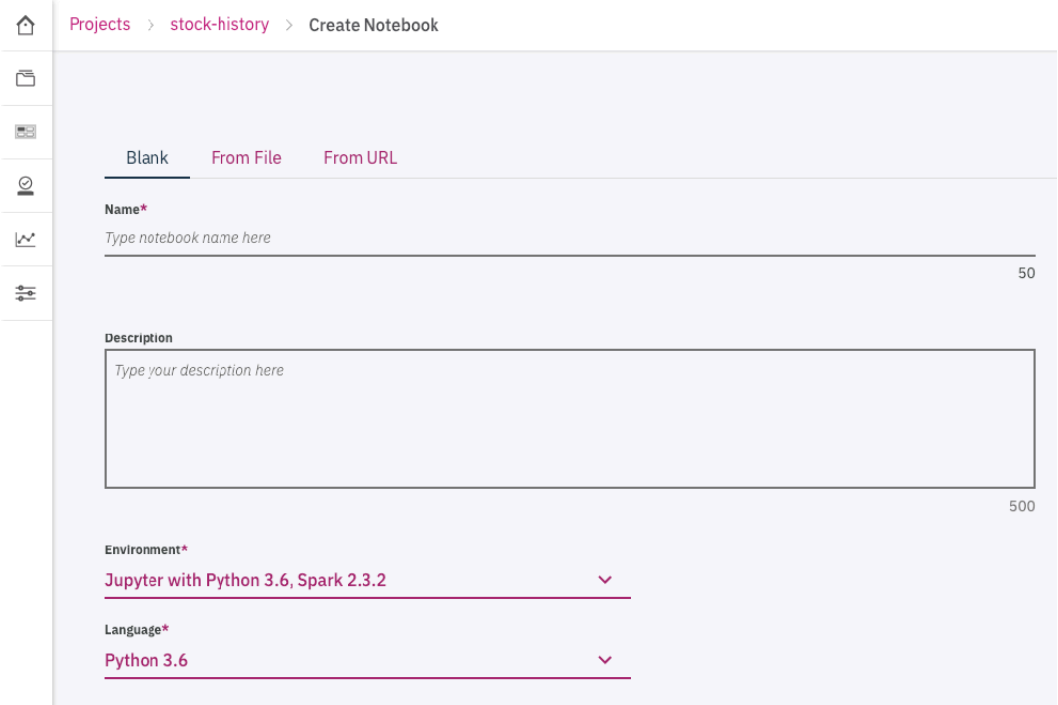


Add a Notebook to a Project + Run SQL Queries

1) This section begins from the “stock-history” project screen. Click on the “Notebooks” link along the left.

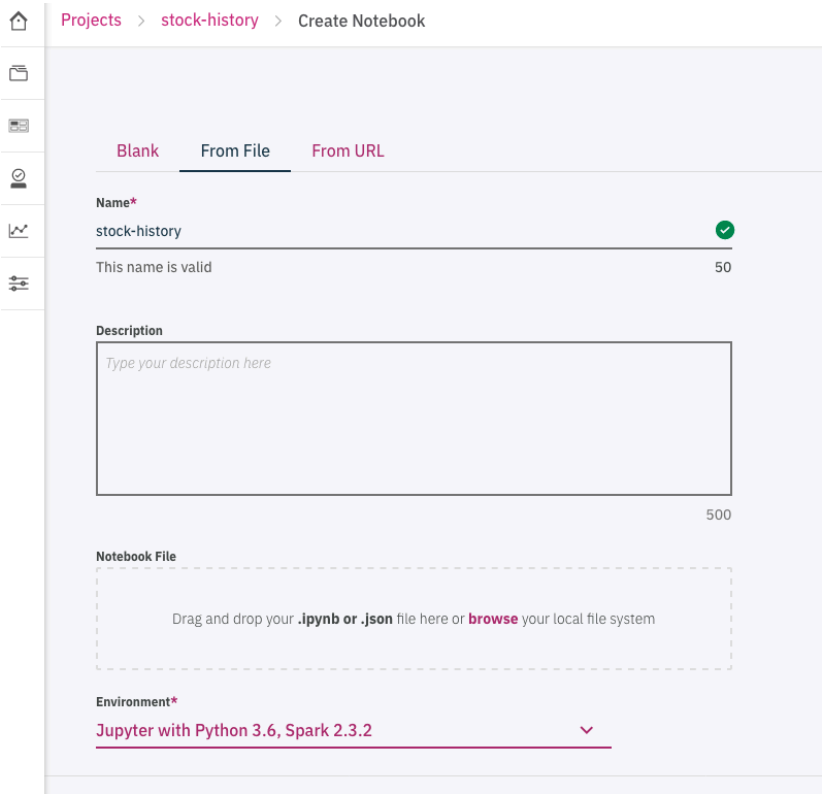
2) On "Notebooks", you can see there are currently none. Click on the Add Notebook “**+ Add Notebook”** button to begin.

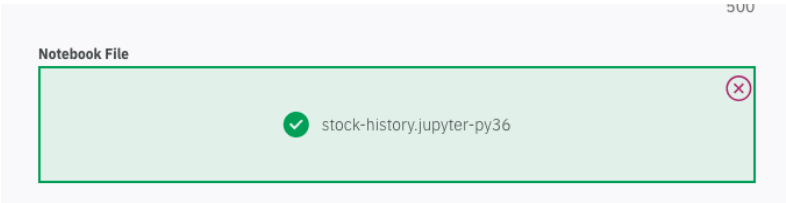
3) The “Create Notebook” screen opens:



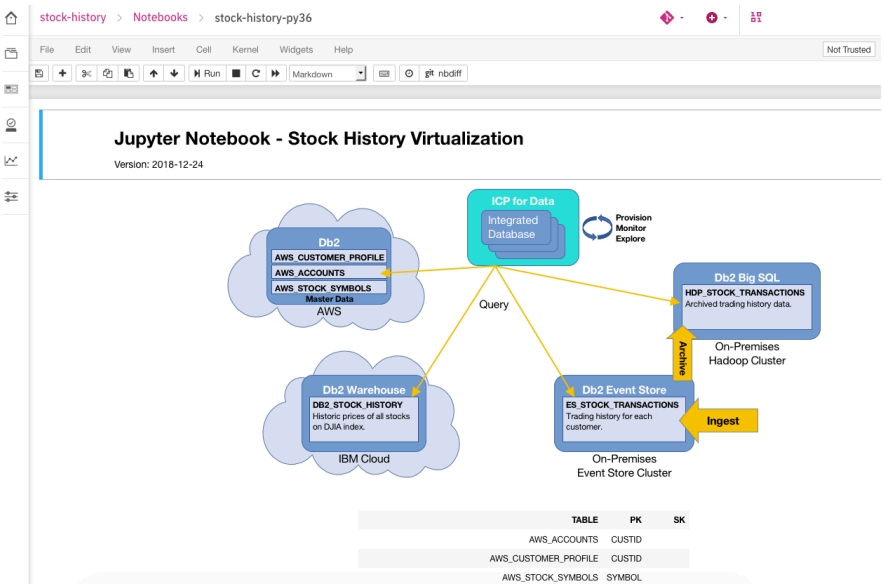
4) On the “Create Notebook” screen, click into the “**Type notebook name here**” field and enter **stock-history**

5) A notebook has already been created for you in the file **stock-history.jupyter-py36.ipynb** and will need to be copied to your desktop. Click the “**From File**” and then the “**browse**” link. Select the file from your desktop and click Open. The filename should be populated into the Notebook file window with a green checkmark. Click on the “**Create**” button.





6) You will be launched into the the Jupyter Notebook capability of ICP for Data with the "Jupyter Notebook – Stock History Virtualization" notebook open.



7) There are a number of scripts for execution within the Notebook. We will execute a few of them in this session and skip over some that take a longer time to execute. Await the output on the execution of each query before proceeding to the next step.

8) Scroll down in the Notebook and click on the **"Simple Query with 1 Virtual Table (1 Db2)"** markdown cell. Click on the Run "**> Run**" button to advance to the code frame.

9) Click on the Run "**> Run**" button to execute the code in the cell. Ignore the pink warning that is displayed. The "**Out[1]:**" will be rendered after 30 seconds or so.

Out[1]:

	CUSTID	TX_COUNT	BALANCE
0	100000	48	10036.56
1	100001	63	-17543.91
2	100002	60	8204.60
3	100003	71	9241.59
4	100004	60	1063.10

10) Scroll down in the Notebook and click on the "**Complex Query with 1 Virtual Table (1 Db2) plus Plotting**" markdown cell. It should highlight in blue on the left. Click on the Run "**> Run**" button to step over the "**Complex Query with 1 Virtual Table (1 Db2) plus Plotting**" to advance to the code cell.

11) Click on the Run "**> Run**" button to the code to import the Python libraries to plot using MATLAB.

```
In [ ]: import matplotlib
        import matplotlib.pyplot as plt
```

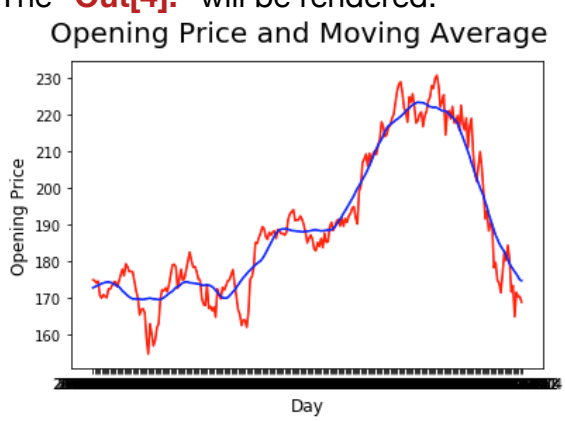
12) Click on the Run "**> Run**" button to execute the third query "SELECT TX_DATE, OPENING, AVG(OPENING) OVER (ORDER BY TX_DATE ROWS BETWEEN 15 PRECEDING AND 15 FOLLOWING) AS MOVING_AVG FROM USER999.JV-AWS_CUSOMER_PROFILE-ES_STOCK_TRANSACTIONS WHERE SYMBOL = 'AAPL' ORDER BY TX_DATE" which is our, "Complex Query with 1 Virtual Table (1 Db2) plus Plotting".

```
In [ ]: df = None
        dataSet = dsx_core_utils.get_remote_data_set_info('USER999.DB2_STOCK_HISTORY')
        dataSource = dsx_core_utils.get_data_source_info(dataSet['datasource'])
        if (sys.version_info >= (3, 0)):
          conn = jaydebeapi.connect(dataSource['driver_class'], dataSource['URL'], [dataSource['user'], dataSource['password']])
        else:
          conn = jaydebeapi.connect(dataSource['driver_class'], [dataSource['URL'], dataSource['user'], dataSource['password']])
        #query = 'select * from "' + (dataSet['schema'] + '"."' if (len(dataSet['schema'].strip()) != 0) else '') +  dataSet['table'] + '"'
        query = \
        """
        SELECT TX_DATE, OPENING,
             AVG(OPENING) OVER (
                ORDER BY TX_DATE
             ROWS BETWEEN 15 PRECEDING AND 15 FOLLOWING) AS MOVING_AVG
          FROM USER999.DB2_STOCK_HISTORY
             WHERE SYMBOL = 'AAPL'
          ORDER BY TX_DATE
        """


        if (dataSet['query']):
            query = dataSet['query']
        df = pd.read_sql(query, con=conn)
        txdate= df['TX_DATE']
        opening = df['OPENING']
        avg = df['MOVING_AVG']

        plt.xlabel("Day", fontsize=12);
        plt.ylabel("Opening Price", fontsize=12);
        plt.suptitle("Opening Price and Moving Average", fontsize=20);
        plt.plot(txdate, opening, 'r');
        plt.plot(txdate, avg, 'b');
        plt.show();
```

The "**Out[4]:**" will be rendered.



13) Click on the Run "**> Run**" button again to step over the "**Simple Query Joining 2 Virtual Tables (1 Db2 and 1 Hadoop)**" markdown cell.

14) Click on the Run "**> Run**" button to execute the forth query "SELECT A.CUSTID, STH.TX_DATE, STH.SYMBOL, STH.PRICE, STH.QUANTITY FROM AWS_ACCOUNTS A INNER JOIN HDP_STOCK_TRANSACTIONS_HISTORY STH ON A.CUSTID = STH.CUSTID WHERE SYMBOL = 'AAPL' ORDER BY A.CUSTID = 108277 ORDER BY STH.QUANTITY DESC" which is our, "Simple Query Joining 2 Virtual Tables (1 Db2 and 1 Hadoop)".

```
In [ ]: df = None
        dataSet = dsx_core_utils.get_remote_data_set_info('USER999.AWS_ACCOUNTS')
        dataSource = dsx_core_utils.get_data_source_info(dataSet['datasource'])
        if (sys.version_info >= (3, 0)):
          conn = jaydebeapi.connect(dataSource['driver_class'], dataSource['URL'], [dataSource['user'], dataSource['password']])
        else:
          conn = jaydebeapi.connect(dataSource['driver_class'], [dataSource['URL'], dataSource['user'], dataSource['password']])
        #query = 'select * from "' + (dataSet['schema'] + '"."' if (len(dataSet['schema'].strip()) != 0) else '') +  dataSet['table'] + '"'
        query = \
        """
        SELECT A.CUSTID, STH.TX_DATE, STH.SYMBOL, STH.PRICE, STH.QUANTITY
          FROM AWS_ACCOUNTS A
            INNER JOIN HDP_STOCK_TRANSACTIONS_HISTORY STH
              ON A.CUSTID = STH.CUSTID
          WHERE A.CUSTID = 108277
            ORDER BY STH.QUANTITY DESC
        """

        if (dataSet['query']):
            query = dataSet['query']
        df = pd.read_sql(query, con=conn)
        df.head()
```

The "**Out[5]:**" will be rendered.

Out[5]:

	CUSTID	TX_DATE	SYMBOL	PRICE	QUANTITY
0	108277	2017-08-08	KO	46.0	86
1	108277	2017-03-21	PFE	35.0	85
2	108277	2017-10-06	XOM	81.0	72
3	108277	2017-01-25	DWDP	60.0	69
4	108277	2017-03-06	MRK	67.0	65

15) Click on the Run "**> Run**" button again to step over the "**Complex Query with Multiple Joins using 5 Virtual Tables (3 Db2, 1 Event Store, 1 Hadoop)**" markdown cell.

16) Click on the Run "**> Run**" button to execute the forth query "SELECT C.LASTNAME, P2018.SYMBOL FROM PURCHASES_2018 P2018, PURCHASES_2017 P2017, CUSTOMERS_IN_CA C WHERE C.CUSTID = P2017.CUSTID AND C.CUSTID = P2018.CUSTID AND P2017.SYMBOL = P2018.SYMBOL ORDER BY 1,2" which is our, "Complex Query with Multiple Joins using 5 Virtual Tables (3 Db2, 1 Event Store, 1 Hadoop)".

```
In [ ]: df = None
        dataSet = dsx_core_utils.get_remote_data_set_info('USER999.AWS_ACCOUNTS')
        dataSource = dsx_core_utils.get_data_source_info(dataSet['datasource'])
        if (sys.version_info >= (3, 0)):
          conn = jaydebeapi.connect(dataSource['driver_class'], dataSource['URL'], [dataSource['user'], dataSource['password']])
        else:
          conn = jaydebeapi.connect(dataSource['driver_class'], [dataSource['URL'], dataSource['user'], dataSource['password']])
        #query = 'select * from "' + (dataSet['schema'] + '"."' if (len(dataSet['schema'].strip()) != 0) else '') +  dataSet['table'] + '"'
        query = \
        """
        WITH
        CUSTOMERS_IN_CA(CUSTID, LASTNAME) AS
          (
          SELECT P.CUSTID, P.LASTNAME
          FROM AWS_CUSTOMER_PROFILE P, AWS_ACCOUNTS A
          WHERE P.STATE = 'CA' AND
                A.BALANCE > 0 AND
                A.CUSTID = P.CUSTID
          ),
        TOP_STOCKS(SYMBOL, VOLUME) AS
          (
          SELECT SYMBOL, MAX(VOLUME)
          FROM DB2_STOCK_HISTORY
          WHERE SYMBOL <> 'DJIA' AND MONTH(TX_DATE) = 10
          GROUP BY SYMBOL
          ORDER BY 2 ASC
          FETCH FIRST 3 ROWS ONLY
          ),
        PURCHASES_2018(CUSTID, SYMBOL) AS
          (
          SELECT UNIQUE CUSTID, CAST(SYMBOL AS VARCHAR(4))SYMBOL
          FROM ES_STOCK_TRANSACTIONS ES
          WHERE MONTH(ES.TX_DATE) = 10 AND
                ES.CUSTID IN (SELECT CUSTID FROM CUSTOMERS_IN_CA) AND
                ES.SYMBOL IN (SELECT SYMBOL FROM TOP_STOCKS) AND
                ES.QUANTITY > 0
          ),
        PURCHASES_2017(CUSTID, SYMBOL) AS
          (
          SELECT UNIQUE CUSTID, SYMBOL
          FROM HDP_STOCK_TRANSACTIONS_HISTORY SH
          WHERE SH.CUSTID IN (SELECT CUSTID FROM CUSTOMERS_IN_CA) AND
                SH.SYMBOL IN (SELECT SYMBOL FROM TOP_STOCKS) AND
                SH.QUANTITY > 0
          )
        SELECT C.LASTNAME, P2018.SYMBOL
        FROM PURCHASES_2018 P2018, PURCHASES_2017 P2017, CUSTOMERS_IN_CA C
        WHERE C.CUSTID = P2017.CUSTID AND
              C.CUSTID = P2018.CUSTID AND
              P2017.SYMBOL = P2018.SYMBOL
        ORDER BY 1,2
        """
```

The "**Out[6]:**" will be rendered.

Out[6]:

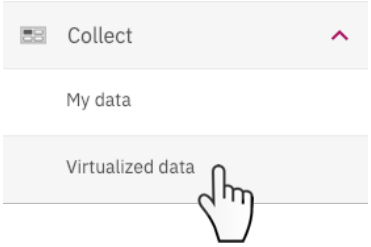
	LASTNAME	SYMBOL
0	Ayers	AXP
1	Bonner	AXP
2	Case	AXP
3	Collins	GS
4	Collins	GS

17) If you would like you can scroll back up and execute the "**Simple Query with 1 Join View Virtual Table**" code frame, but it may take several minutes to complete. You can click on the "**Stop"** button to end execution if it is taking too long.

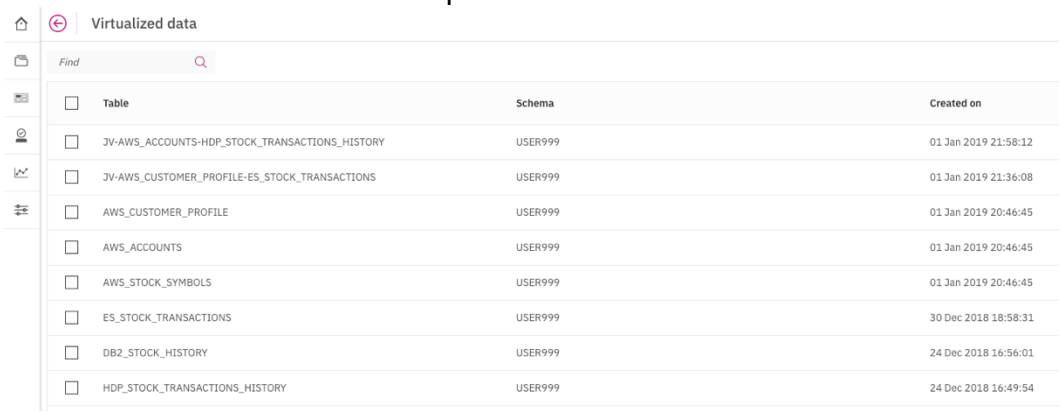
This concludes our work with the Virtualized tables that we created. We will return to the Data Virtualization feature to explore the remaining capabilities and menu options.

Tour additional menus in Virtualized data (i.e. SQL Editor, Manage Users, etc.)

1) Navigate to the virtualized data section to tour the additional menus. Click on the "**Collect**" icon on the left followed by "**Virtualized data**":

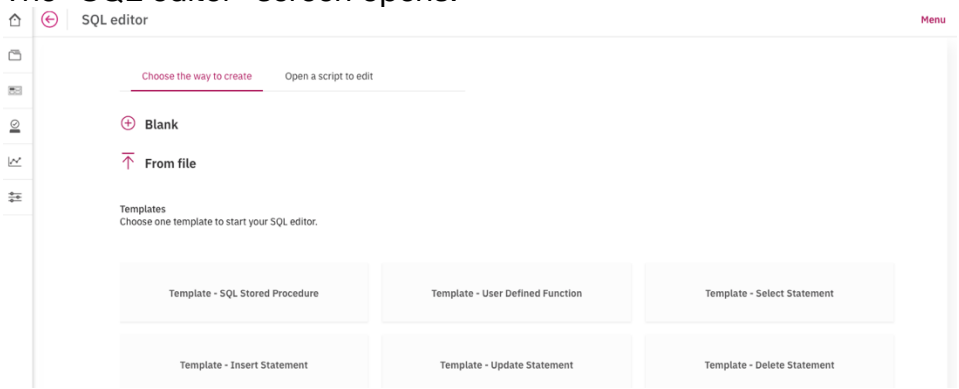


2) The “Virtualized data” screen opens:

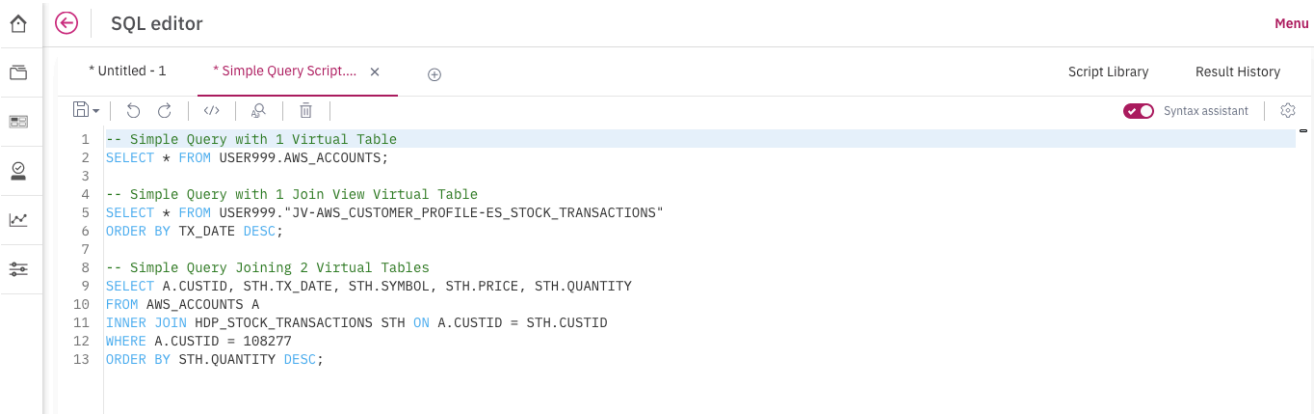


3) In “Virtualized data”, click on the “**Menu**” button in the upper left followed by “**SQL editor**”.

4) The “SQL editor” screen opens:

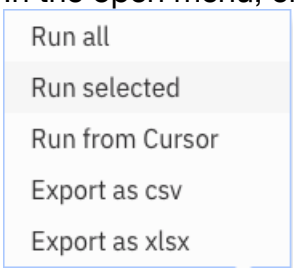


5) Copy the **“Simple SQL Query.sql”** file to your desktop. Select the “**From file**” button. Find the file that you just copied, select it and click **“Open”**. The SQL editor will open with the SQL script from the file populating the editor window.

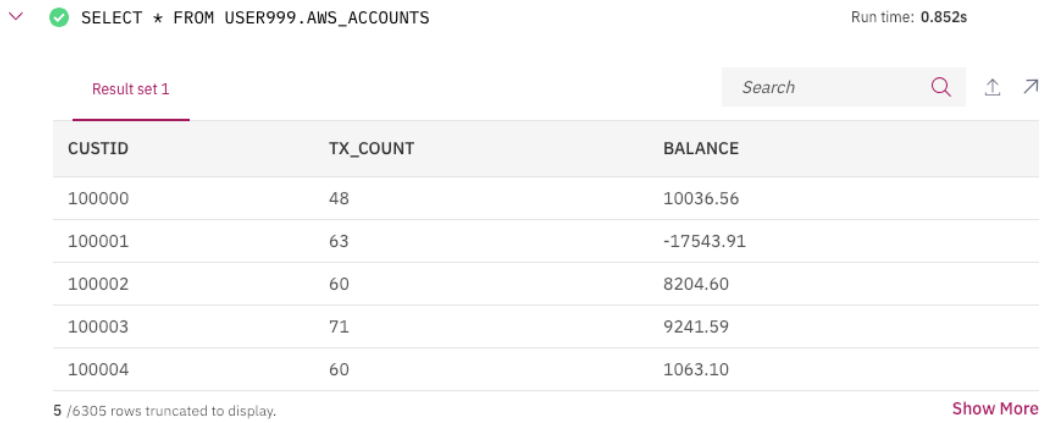


6) Place your cursor and click anywhere on line 2 to highlight the "**SELECT * FROM USER999.AWS_ACCOUNTS;**" statement, click on the button at the bottom next to "**Run all**" to open the menu.

7) In the open menu, click "**Run selected**" to execute the first query.



8) The results of the first query are rendered in the right panel:

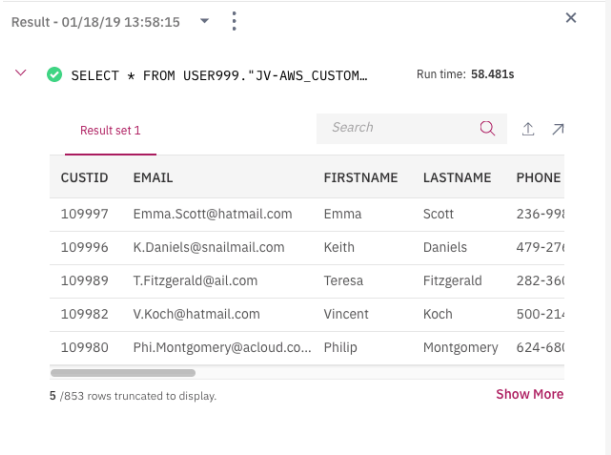


CUSTID	TX_COUNT	BALANCE
100000	48	10036.56
100001	63	-17543.91
100002	60	8204.60
100003	71	9241.59
100004	60	1063.10

9) Click anywhere in lines 5 or 6 to select the second query, "**SELECT * FROM USER999.JV-AWS_CUSTOMER_PROFILE-ES_STOCK_TRANSACTIONS ORDER BY TX_DATE DESC;**".

10) Click on the Run selected "**Run selected**" button to execute the second query. This query will take a minute or two to execute.

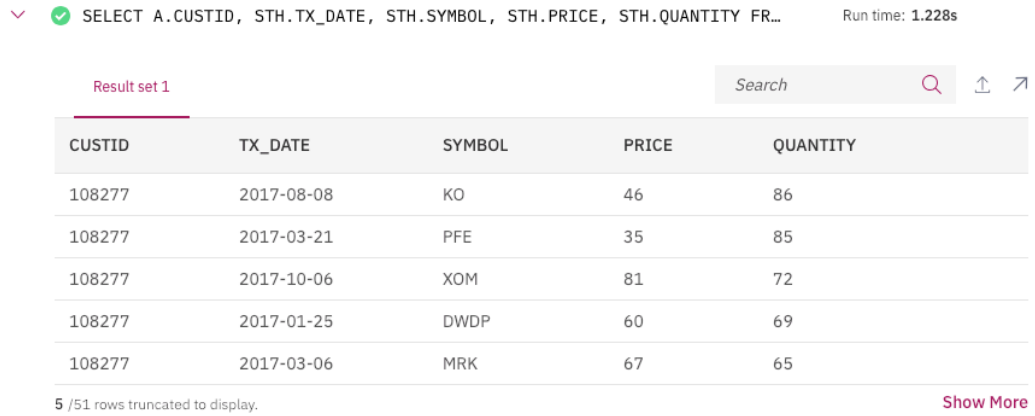
11) The results of the second query are rendered in the right panel:



12) Click anywhere in line 9 through 13 to select the third query, "**SELECT A.CUSTID, STH.TX_DATE, STH.SYMBOL, STH.PRICE, STH.QUANTITY FROM AWS_ACCOUNTS A INNER JOIN HDP_STOCK_TRANSACTIONS_HISTORY STH ON A.CUSTID = STH.CUSTID WHERE A.CUSTID = 108277 ORDER BY STH.QUANTITY DESC;**".

13) Click on the Run selected "**Run selected**" button to execute the third query.

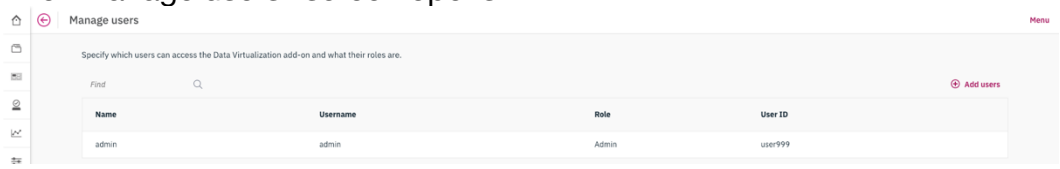
14) The results of the third query are rendered in the right panel:



15) Note the "**Result History**" and "**Script Library**" buttons in the top right and browse through both. The SQL editor provides the ability to develop and test SQL queries for virtual views and tables without having to leave the platform. A set of standard SQL query templates are already loaded in the Script Library and can be used as the basis to create new queries in the editor.

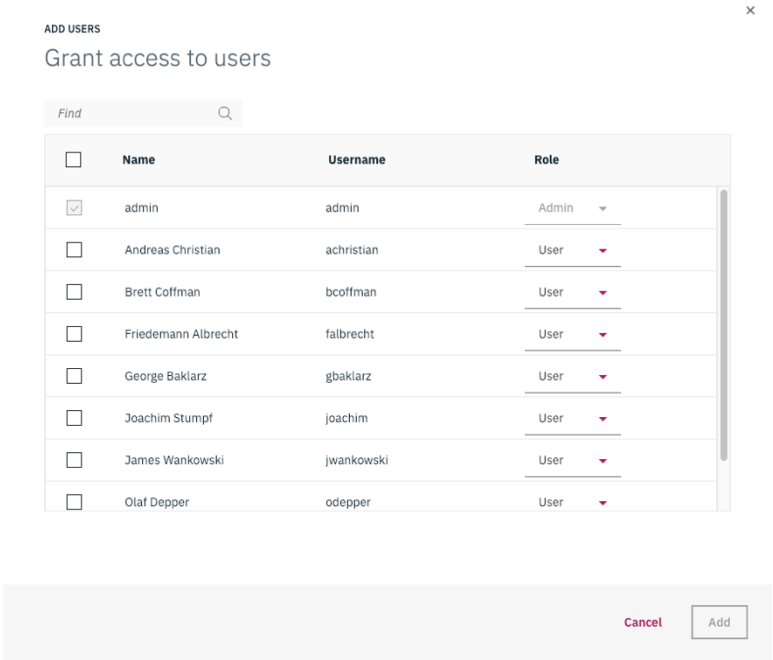
16) Click on the "**Menu**" button in the upper left followed by "**Manage users**" to continue.

17) The "Manage users" screen opens:



18) In the "Manage users" screen there is currently only the "Admin" account, we are going to add a new user. Click on the Add users "**+ Add users**" button.

19) The “Grant access to users” dialog opens:

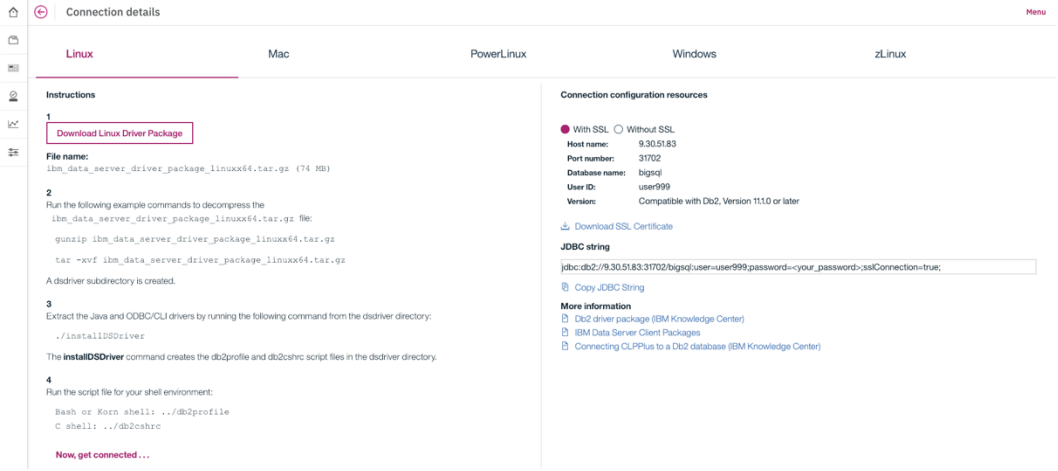


20) On the “Grant access to users” dialog platform users can be granted access to Data Virtualization with 3 possible user roles. You may not have additional platform users other than Admin at this point. Roles and their feature access are highlighted below.

Data virtualization features	Admin	Engineer	User
Provision Data Virtualization	√		
Manage users	√		
Data sources	√	√	
Virtualize	√	√	
Virtualized data	√	√	√
Connection details	√	√	√
About	√	√	√
SQL Editor	√	√	√

21) Manage users is complete, click on the “**Menu**” button in the upper left followed by “**Connection details**” to continue.

22) The “Connection details” screen opens:

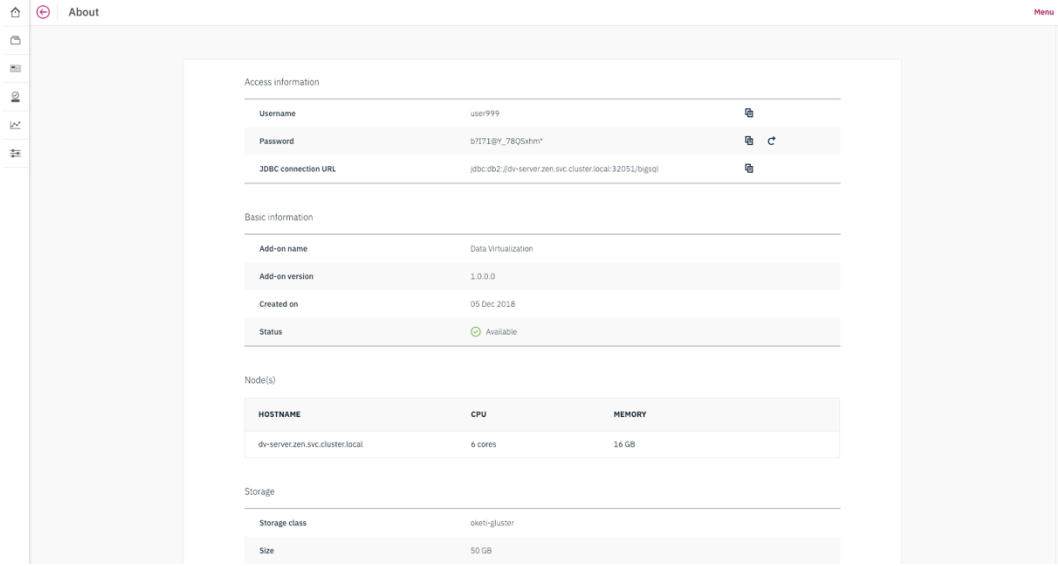


23) On the “Connection details” screen, there are tabs for Linux, Mac, PowerLinux, Windows, zLinux which provide the details on connection to the Virtualized Data server with and without SSL. Click on the “**Mac**” tab to continue.

24) **Prerequisites for SSL for Db2, Big SQL, and data virtualization**
If you plan to use SSL for a Db2 for Linux, Unix and Windows or a Big SQL connection that uses a self-signed certificate or a certificate that is signed by a local certificate authority (CA), you need to import the SSL certificate to the Spark trust store. If you provisioned the Data Virtualization add-on, and you plan to use SSL to connect to the Data Virtualization server, a self-signed certificate is available from the Connection details page (go to Collect > Virtualized data and then open the Connection details page). You need download the certificate and import it to the Spark trust store.

25) Connection details is complete, click on the “**Menu**” button in the upper left followed by “**About**” to continue.

26) The ”About” screen opens:



27) On the “About” screen shows Data Virtualization connection details, version info, node info and storage info.