Practical F01
Introduction to Databricks Notebook

Objectives of this practical

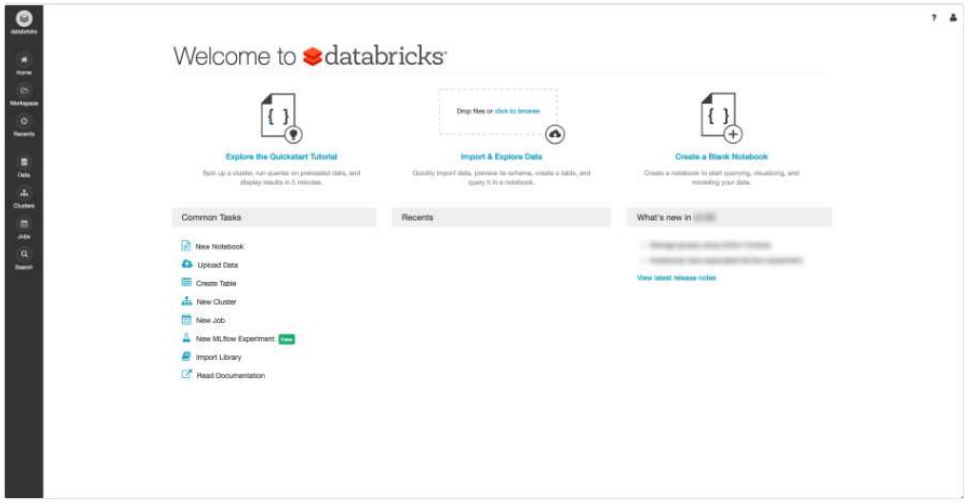
1. Get familiar with the Databricks Community Edition
2. Learn how to import data to Databricks
3. Write Spark SQL to create and query tables

Section 1 Overview Databricks Community Ed

databricks

Databricks is a managed plateform of Apche Spark.

First, **create an account** with the Databricks Community Edition and login. The following is the welcome page.



Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a notebook to start querying, visualizing, and modeling your data.

Let start by getting familiar with some terminologies used in the Databricks Community Edition.

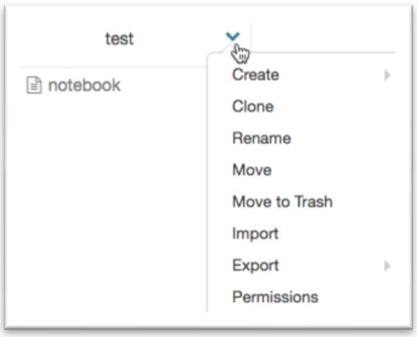
You can also view an online tutorial (Overview of Databricks) at https://www.linkedin.com/learning/apache-spark-essential-training/overview-of-databricks?u=2122804

Workspace

A Databricks Workspace is an environment for accessing all of your Databricks assets. The Workspace organizes objects (notebooks, libraries, and experiments) into folders, and provides access to data objects and computational resources.

Folders contain all static assets within a Workspace: notebooks, libraries, experiments, and other folders. Icons indicate the type of the object contained in a folder. A Databricks Workspace has three special folders: *Workspace, Shared, and Users*. You cannot rename or move a special folder.

Click a folder name to open or close the folder and view its contents. To perform an action on a folder, click the Down Caret at the right side of a folder and select a menu item.



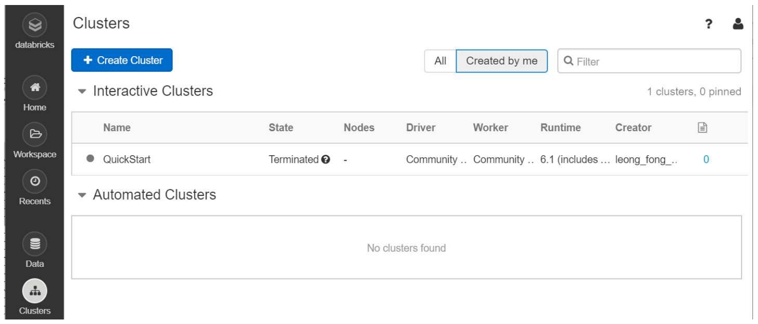
Workspace Object: Notebook
A notebook is a web-based interface to a document that contains runnable code, visualizations, and narrative text. For this module, we will using either **Python Notebook or SQL Notebook**.

Databricks File System (DBFS)
Databricks File System (DBFS) is a **distributed file system** mounted into a Databricks workspace and available on Databricks clusters. DBFS is an abstraction on top of scalable object storage.

The default storage location in DBFS is known as the DBFS root. Several types of data are stored in the following DBFS root locations:

- /FileStore: Imported data files, generated plots, and uploaded libraries. See FileStore.
- /databricks-datasets: Sample public datasets.
- /databricks-results: Files generated by downloading the full results of a query.
- /databricks/init: Global and cluster-named (deprecated) init scripts.
- /user/hive/warehouse: Data and metadata for non-external Hive tables.

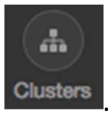
Clusters
A Databricks cluster is a set of computation resources and configurations on which you run data engineering, data science, and data analytics workloads, such as production ETL pipelines, streaming analytics, ad-hoc analytics, and machine learning.



Name	State	Nodes	Driver	Worker	Runtime	Creator	
QuickStart	Terminated	-	Community ..	Community ..	6.1 (includes ...	leong_fong_..	0

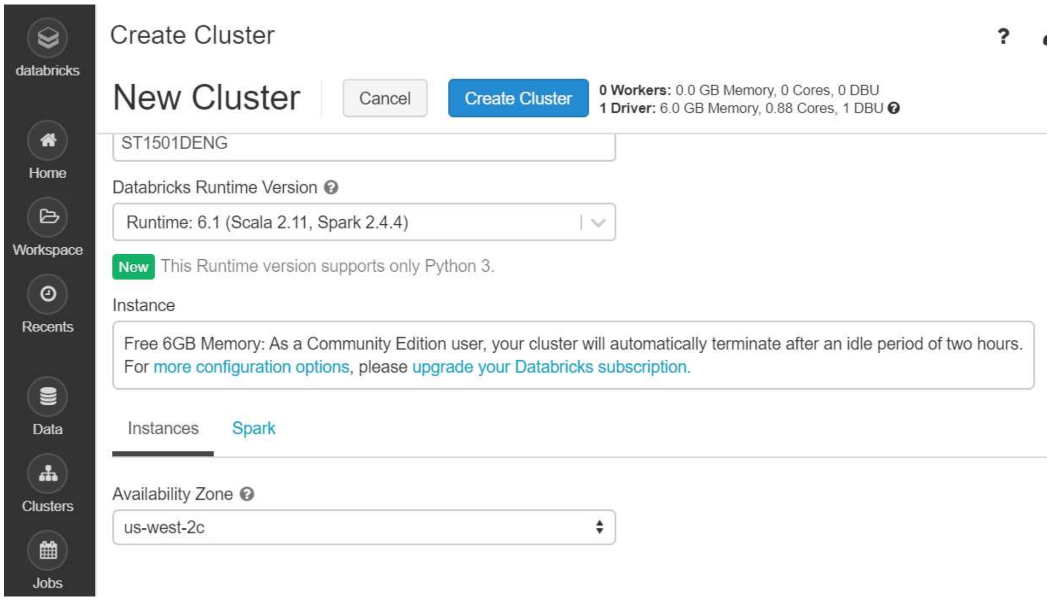
Section 2 Runing Databrick Notebook (SQL)

Task A: Create a Cluster

a) To create a cluster, in the sidebar, click the **Clusters** button .
On the Clusters page, click **Create Cluster.**

b) On the Create Cluster page, specify the cluster name **ST1501DENG** and select from the Databricks Runtime Version drop-down, **6.1 (Scala 2.11, Spark 2.4.4)** or similar.

Then, create the 'Create Cluster'.



0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

ST1501DENG

Databricks Runtime Version
Runtime: 6.1 (Scala 2.11, Spark 2.4.4)

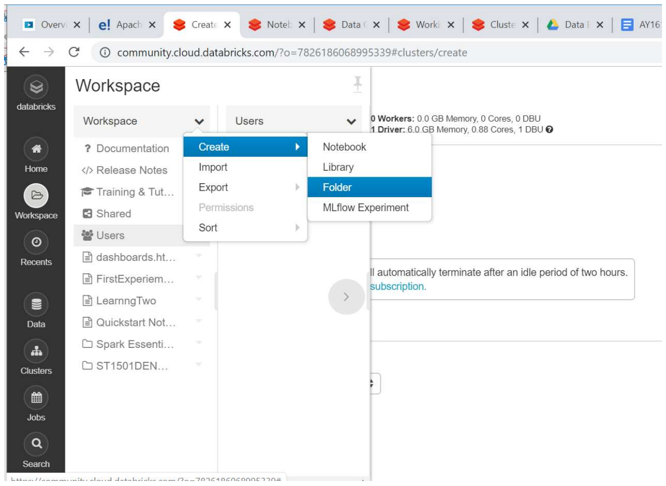
New This Runtime version supports only Python 3.

Instance
Free 6GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.
For more configuration options, please upgrade your Databricks subscription.

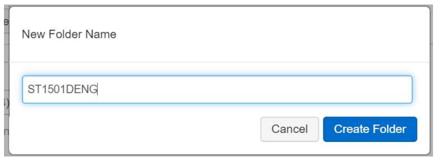
Availability Zone
us-west-2c

Task B: Create a Folder name ST1501DENG and SQL Notebook

c) To create the folder **ST1501DENG**, click the Down Caret at the right side of the 'Work Space'. Select 'Create', 'Folder'.

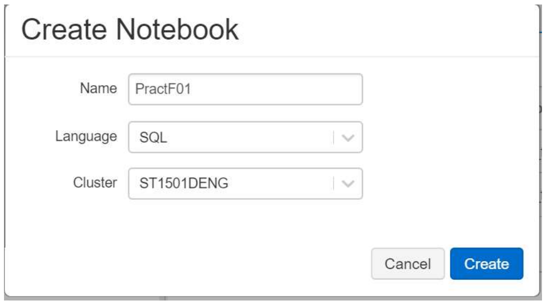


d) In the pop-up dialog box, enter 'ST1501DENG' and click the "Create Folder" button.


New Folder Name: ST1501DENG

e) If you have completed all the above steps correctly, a "ST1501DENG' folder should be created in the Workspace.

Click the Workspace button and click the 'ST1501DENG' folder. Click the Down Caret at the right side of the 'folder'. Select 'Create', 'Notebook'. Enter Name, Language and Cluster as shown in the following.


Create Notebook

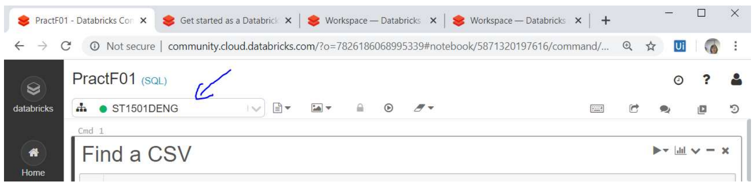
Name: PractF01

Language: SQL

Cluster: ST1501DENG

Task C: Create Tables using Spark SQL

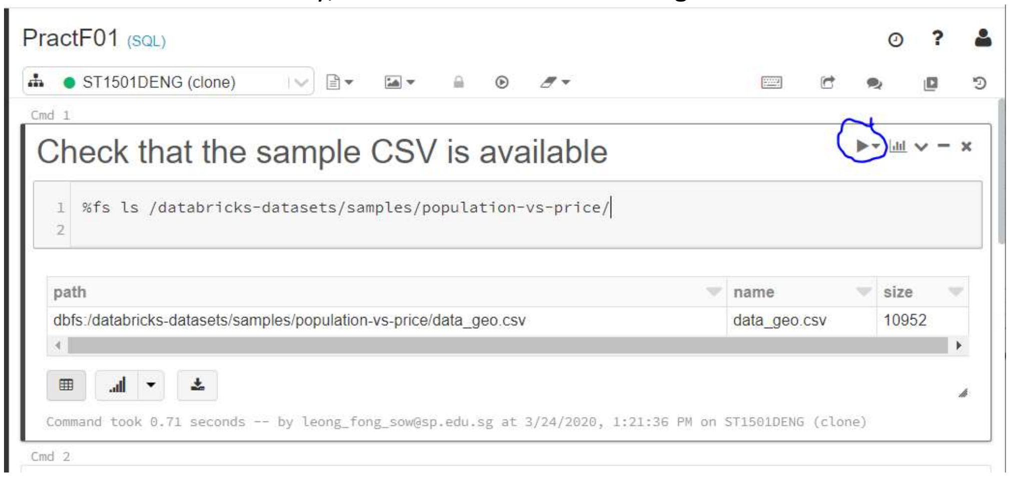
f) An SQL Notebook named PractF01 should open.
Ensure that the cluster ST1501DENG created in Task A is shown In the SQL Notebook.


PractF01 - Databricks Co
Get started as a Databrick
Workspace — Databricks
Workspace — Databricks
Not secure | community.cloud.databricks.com/?o=7826186068995339#notebook/5871320197616/command/...
databricks
Home
PractF01 (SQL)
ST1501DENG
Cmd 1
Find a CSV

g) To confirm that the Databricks sample 'data_geo.csv' is available for our practical, enter the following code into the first cell. Click the 'triangle' (highlighted in blue) to execute the command.

```
%fs ls /databricks-datasets/samples/population-vs-price/
```

If the cell runs successfully, it should show the following.



```
%fs ls /databricks-datasets/samples/population-vs-price/
```

path	name	size
dbfs:/databricks-datasets/samples/population-vs-price/data_geo.csv	data_geo.csv	10952

```
Command took 0.71 seconds -- by leong_fong_sow@sp.edu.sg at 3/24/2020, 1:21:36 PM on ST1501DENG (clone)
```

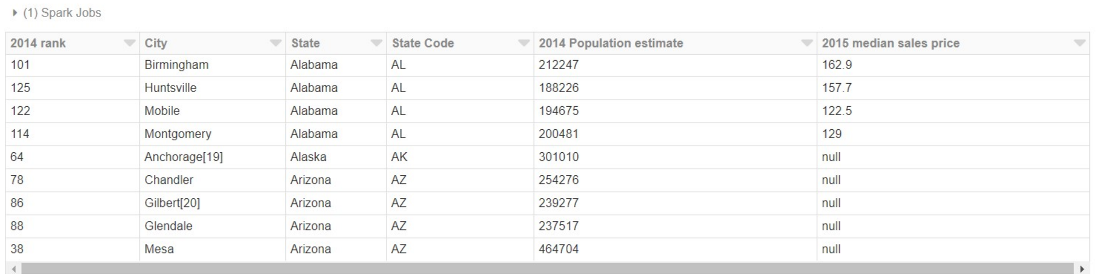
h) Enter the following command in a new cell.

The command creates a table 'population_v_price' by inferring the schema of the csv file.

```
CREATE TABLE  IF NOT EXISTS population_v_price
USING CSV
OPTIONS (path "/databricks-datasets/samples/population-vs-price/data_geo.csv",
header "true",
inferSchema "true");

/* check results */
select * from population_v_price limit 20;
```

Run the cell, you should see 20 rows of data display.



(1) Spark Jobs

2014 rank	City	State	State Code	2014 Population estimate	2015 median sales price
101	Birmingham	Alabama	AL	212247	162.9
125	Huntsville	Alabama	AL	188226	157.7
122	Mobile	Alabama	AL	194675	122.5
114	Montgomery	Alabama	AL	200481	129
64	Anchorage[19]	Alaska	AK	301010	null
78	Chandler	Arizona	AZ	254276	null
86	Gilbert[20]	Arizona	AZ	239277	null
88	Glendale	Arizona	AZ	237517	null
38	Mesa	Arizona	AZ	464704	null

i) Enter the following command in a new cell. Excute the cell and you should see 10 rows of records.

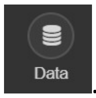
The command creats a table 'online_retail' by defning a schema from another sample data.

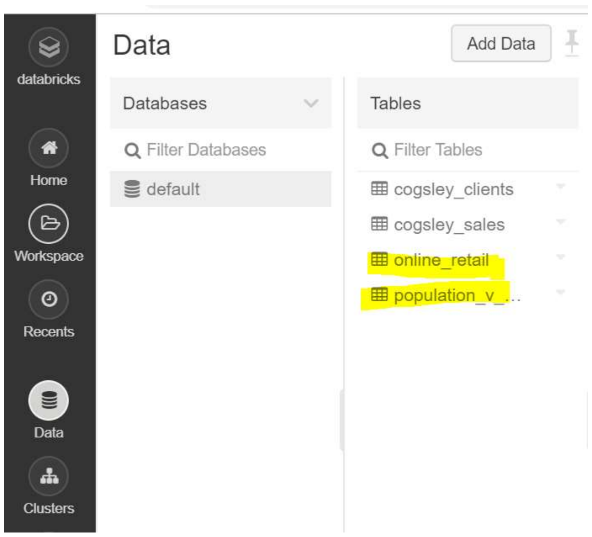
```
CREATE TABLE IF NOT EXISTS online_retail(
InvoiceNo string,
StockCode string,
Description string,
Quantity int,
InvoiceDate string,
UnitPrice double,
CustomerID int,
Country string)
USING CSV
OPTIONS (path "/databricks-datasets/online_retail/data-001/data.csv", header "true");

/* check results */
select * from online_retail limit 10;
```

j) We have just created two tables, one without defining the schema and one with a defined schema.

If you have completed the task correctly, the two tables ‘population_v_price” and ‘online_retail’ should be saved in the default database.

Check that two tables are created by clicking .



Section 3 Using Spark SQL to Query

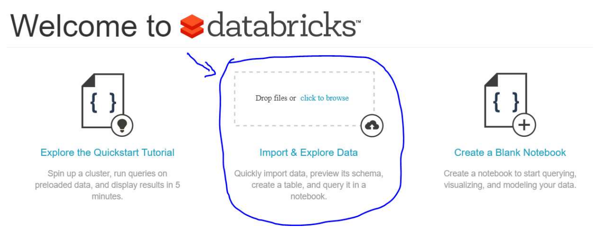
If you have completed Section 2, you should be now more familiar with the Spark Community Edition and to run SQL Notebook. You should also notice that Spark SQL allows us to create table easily with CSV file as a data source.

In this section, you are going to use **Spark SQL** to create a database with two tables, and write SQL queries.

Spark SQL supports both ANSI SQL and HiveQL queries. The power of Spark SQL derives from several key facts: SQL analysts can now take advantage of Spark's computation abilities, data engineers can use Spark SQL where appropriate in any data flow. Please note that Spark SQL is intended to operate as an OLAP database, not an OLTP database.

No. **Task A: Create Database, Tables and Write Spark SQL**

a) Import **cogsley_clients.csv** and **cogsley_sales.csv** to the Databricks filestore using the Databricks UI.

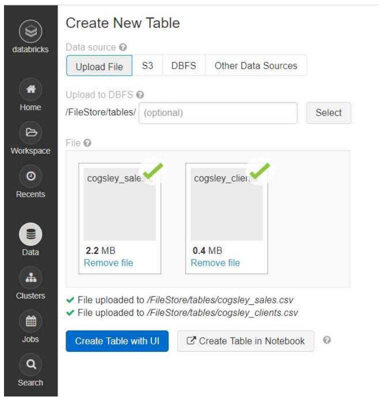


Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a notebook to start querying, visualizing, and modeling your data.

Drag files to the File dropzone or click the dropzone to browse to and choose files. After upload, a path displays for each file. The path will be something like /FileStore/tables/<filename>-<random-number>.<file-type> and you use this path in a notebook to read data.



✓ File uploaded to /FileStore/tables/cogsley_sales.csv
✓ File uploaded to /FileStore/tables/cogsley_clients.csv

No. Task A: Create Database, Tables and Write Spark SQL

b) Create a new SQL Notebook, called it **PractF01_Sect3**. Refer to Section 2 Task B if necessary.

c) Check that the two CSV files are uploaded by using the following python code. Note **‘%python**’ is a magic command to switch the SQL cell to a Python cell.

```
%python
dbutils.fs.ls("/FileStore/tables")
```

d) Create a database called PractF01.

```
Create database PractF01.
```

e) Write the Spark SQL to create **cogsley_clients** table, using cogsely_client.csv as the data source.

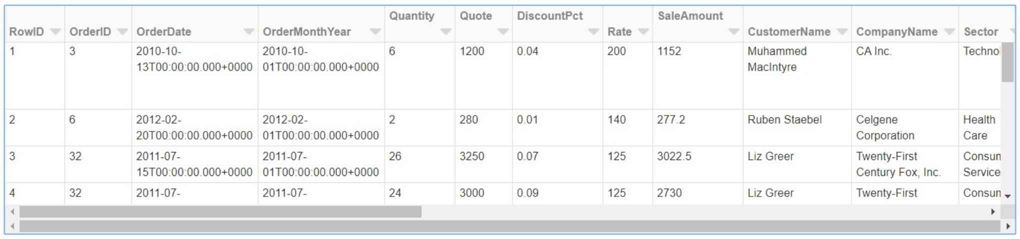
```
Use PractF01;

CREATE TABLE IF NOT EXISTS cogsley_clients
USING CSV
OPTIONS (path "/FileStore/tables/cogsley_clients.csv", header "true", inferSchema
"true");
```

Check that it works by entering the following:

```
select * from cogsley_clients limit 10;
```

f) Write the Spark SQL to create **cogsley_sales** table using cogsely_sales.csv as the data source. Check that it works by displaying its first 4 records. (SQL Hint: Refer to the above step.)



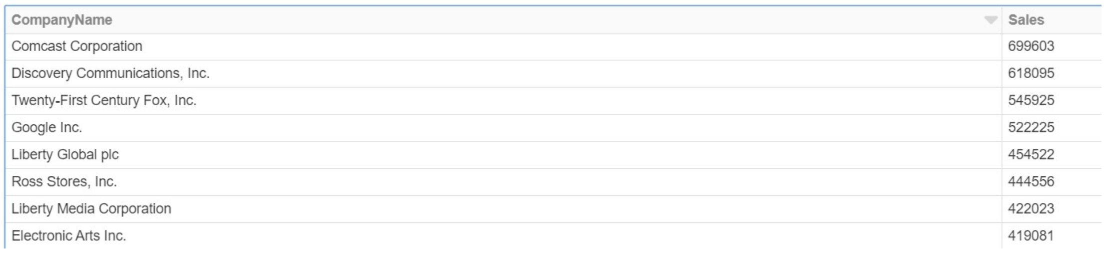
RowID	OrderID	OrderDate	OrderMonthYear	Quantity	Quote	DiscountPct	Rate	SaleAmount	CustomerName	CompanyName	Sector
1	3	2010-10-13T00:00:00.000+0000	2010-10-01T00:00:00.000+0000	6	1200	0.04	200	1152	Muhammed MacIntyre	CA Inc.	Techno
2	6	2012-02-20T00:00:00.000+0000	2012-02-01T00:00:00.000+0000	2	280	0.01	140	277.2	Ruben Staebel	Celgene Corporation	Health Care
3	32	2011-07-15T00:00:00.000+0000	2011-07-01T00:00:00.000+0000	26	3250	0.07	125	3022.5	Liz Greer	Twenty-First Century Fox, Inc.	Consun Service
4	32	2011-07-	2011-07-	24	3000	0.09	125	2730	Liz Greer	Twenty-First	Consun

g) You can display the table schema using the ‘describe’ command. For example:

Describe cogsley_sales;

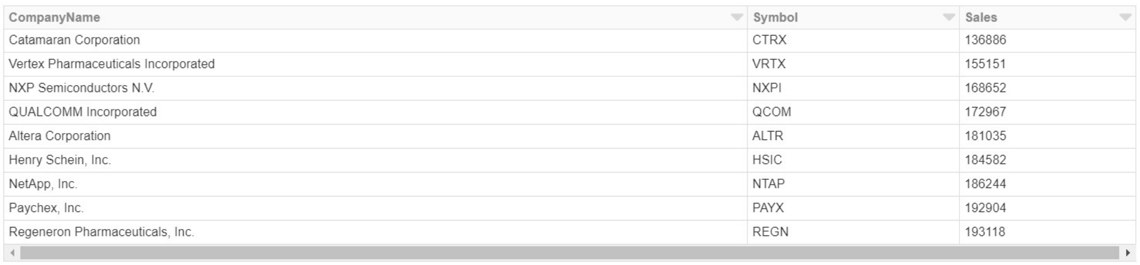
h) Write the Spark SQL to calculate the **total ‘SalesAmount’** of each company**.** Show the result in **descending order of the total sales.** (SQL Hint: Use sum() function and group by clause.)

The result should be similar to the following:



CompanyName	Sales
Comcast Corporation	699603
Discovery Communications, Inc.	618095
Twenty-First Century Fox, Inc.	545925
Google Inc.	522225
Liberty Global plc	454522
Ross Stores, Inc.	444556
Liberty Media Corporation	422023
Electronic Arts Inc.	419081

i) Write the Spark SQL to calculate the total 'SalesAmount' of each company. Show each company with its corresponding ‘symbol’ available from the cogsley_client table. (SQL Hint: Use inner join.)



CompanyName	Symbol	Sales
Catamaran Corporation	CTRX	136886
Vertex Pharmaceuticals Incorporated	VRTX	155151
NXP Semiconductors N.V.	NXPI	168652
QUALCOMM Incorporated	QCOM	172967
Altera Corporation	ALTR	181035
Henry Schein, Inc.	HSIC	184582
NetApp, Inc.	NTAP	186244
Paychex, Inc.	PAYX	192904
Regeneron Pharmaceuticals, Inc.	REGN	193118

j) Write the Spark SQL to list the **top 3 sectors** with the hightest total 'SalesAmount'.

-- End of Practical --