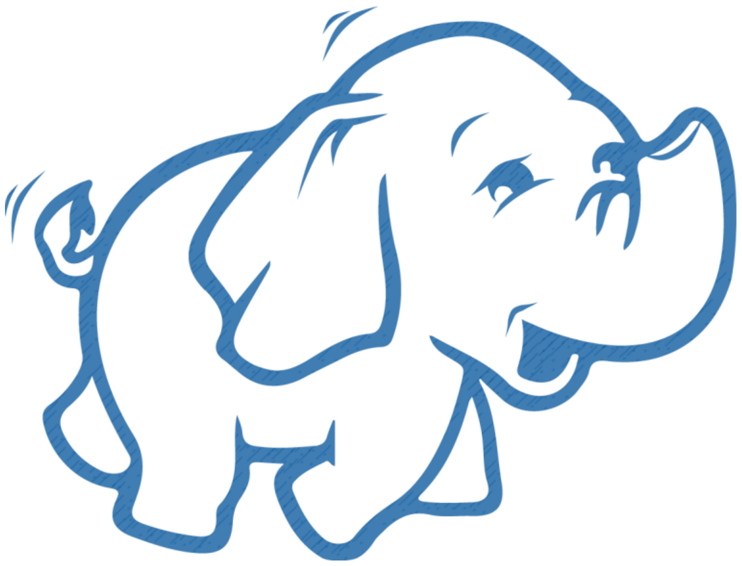
Microsoft

Exploring Hive in Apache Hadoop with Microsoft Azure HDInsight



Introduction

Hive is a data warehousing system that simplifies analyzing large datasets stored in Hadoop clusters, using SQLLike language known as HiveQL. Hive converts queries to either map/reduce, Apache Tez or Apache Spark jobs.

To highlight how customers can efficiently leverage HDInsight Hive to analyze big data stored in Azure Blob Storage, this document provides an end-to-end walkthrough of analyzing a web transaction log of an imaginary book store using Hive.

This walkthrough highlights the performance difference between MR, Tez and Tez + LLAP execution engine.

Takeaways

After completing this lab, you will learn,

1. Different ways to execute hive queries on an HDInsight cluster
2. Tez performance graph
3. To use join, aggregates, analytic function, ranking function, group by and order by in Hive QL 4. Analyze hive tables in PowerBI desktop

Prerequisites

Azure Account Requirements

While carrying out all exercises within this hands-on lab, you will make use of the **Azure Preview portal** from https://portal.azure.com/.

To perform this lab, you will require a Microsoft Azure account.

If you do not have an Azure account, you can request for a free trial version by visiting http://azure.microsoft.com/en-us/pricing/free-trial/.

Within the one-month trial version, you can perform other SQL Server 2014 hands-on labs along with other tutorials available on Azure.

NOTE: - To sign up for a free trial, you will need a mobile device that can receive text messages and a valid credit card.

Make sure you follow the ***Roll back Azure changes*** *section at the end of this exercise after creating the Azure database so that you can make the most of your $200 free Azure credit*

Special instructions

HDInsight cluster usually take 15-20 minutes to create. These credentials are shared in good faith and the understanding is that attendees will not misuse these for any purposes, including but not limited to this lab. If you have any concerns, please close this lab now and do not proceed any further.

Cluster Credentials:

<u>Generic information:</u>

Resource Group: mtllab

*Storage Account : mtllab**<number 1 through 12>***

Location : East US

<u>Hive Cluster:</u>

*Cluster Name for Hive Cluster: mtllab**<number 1 through 12>***

Cluster URL (Ambari) for Hive Cluster: https://*mtllab**<number 1 through 12>.***azurehdinsight.net/

Username: admin

Password: HDItut@123

<u>Hive LLAP CLuster</u>

*Cluster Name for Hive (LLAP) Cluster: mtllabllap**<number 1 through 12>***

Cluster URL (Ambari) for Hive (LLAP) Cluster: https://*mtllab*llap***<number 1 through 12>***.azurehdinsight.net/

Username: admin

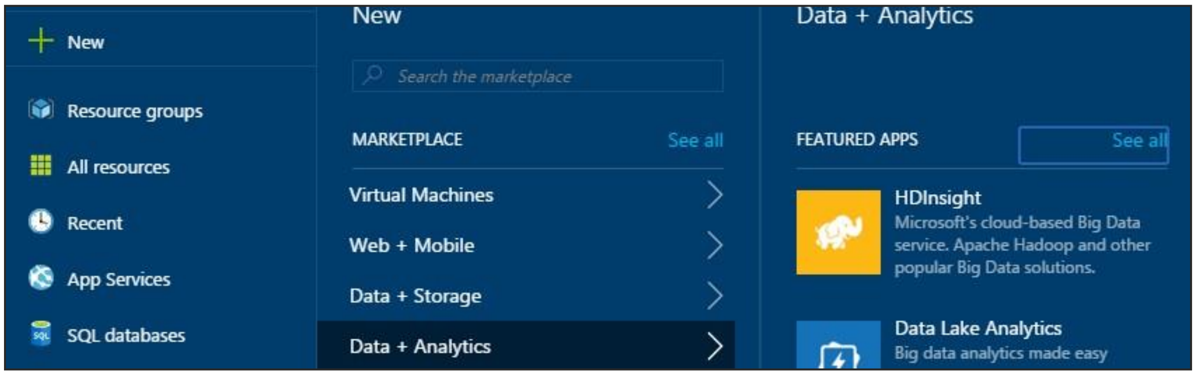
Password: HDItut@123

Class

Provision HDInsight Linux Hadoop cluster with Azure Management Portal

To provision HDInsight Hadoop cluster with Azure Management Portal, perform the below steps.

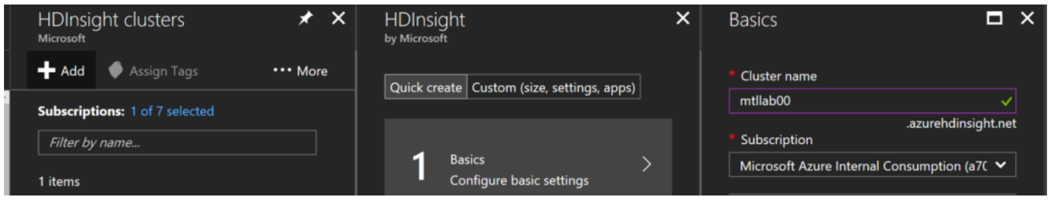
1. Go to the Azure Preview Portal by clicking the **Preview Portal** link Preview Portal on the IE favorites bar. Login using your azure account credentials.
2. Select **NEW -> Data Analytics-> HDInsight**



3. Enter or select the following values.

   a. **Cluster Name:** Enter the cluster name. A green tick will appear if the cluster name is available.

   b. **Resource Group:** Select an existing resource group or create a new resource group.



Cluster name: mtllab00 .azurehdinsight.net

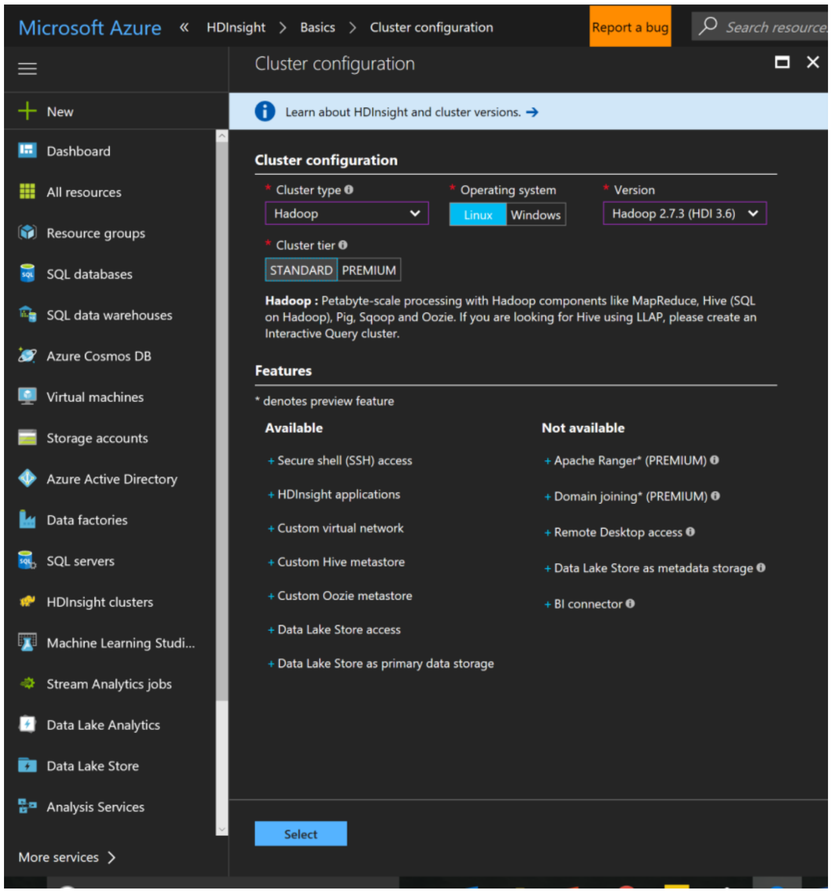
Subscription: Microsoft Azure Internal Consumption (a7(

   c. **Cluster Type:** Select Hadoop as the cluster type.

   d. **Cluster Operating System:** Select Linux as the cluster operating system

   e. **Version:** Select 3.6 as the cluster version.

   f. **Cluster Tier:** Select the **Standard** cluster tier



Cluster configuration

Learn about HDInsight and cluster versions.

Cluster configuration

* Cluster type: Hadoop
* Operating system: [✓ Linux] Windows
* Version: Hadoop 2.7.3 (HDI 3.6)
* Cluster tier: [✓ STANDARD] PREMIUM

Hadoop : Petabyte-scale processing with Hadoop components like MapReduce, Hive (SQL on Hadoop), Pig, Sqoop and Oozie. If you are looking for Hive using LLAP, please create an Interactive Query cluster.

Features

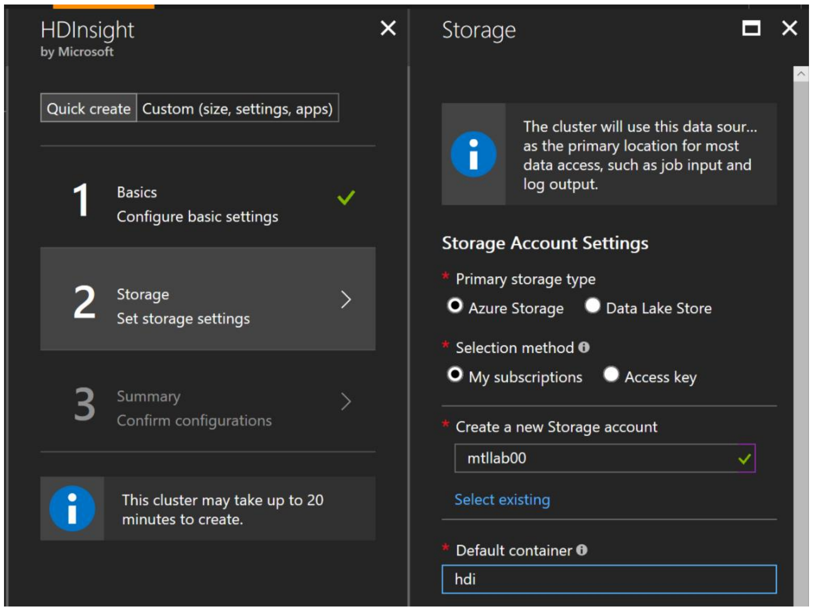
* denotes preview feature

Available	Not available
+ Secure shell (SSH) access	+ Apache Ranger* (PREMIUM)
+ HDInsight applications	+ Domain joining* (PREMIUM)
+ Custom virtual network	+ Remote Desktop access
+ Custom Hive metastore	+ Data Lake Store as metadata storage
+ Custom Oozie metastore	+ BI connector
+ Data Lake Store access	
+ Data Lake Store as primary data storage	

g. **Subscription:** Select the Azure subscription to create the cluster.

h. **Credentials:** Configure the username and password for HDInsight cluster and the SSH connection. SSH connection is used to connect to HDInsight cluster through a SSH client such as Putty.

i. **Storage:** Create a new storage account and a default container named **hdi**.



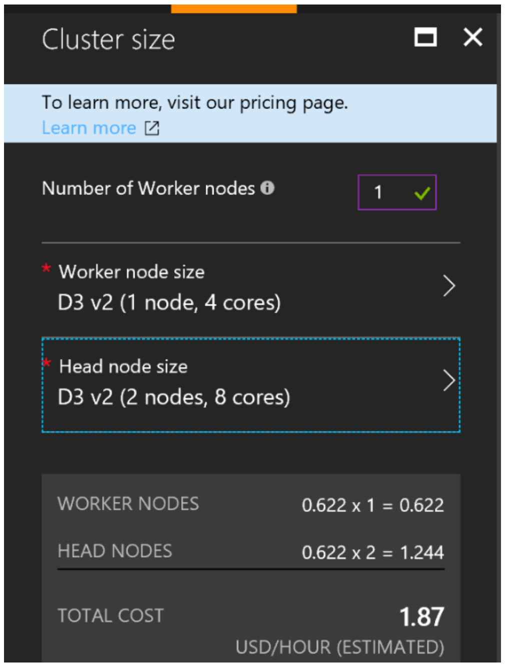
This cluster may take up to 20 minutes to create.

The cluster will use this data sour... as the primary location for most data access, such as job input and log output.

Storage Account Settings

* Primary storage type: [✓ Azure Storage] Data Lake Store
* Selection method: [✓ My subscriptions] Access key
* Create a new Storage account: mtllab00
* Default container: hdi

j. **Cluster Size:** Set the head node and worker nodes to D3 v2 and the number or worker nodes to 1 as shown below.


Cluster size

To learn more, visit our pricing page.
Learn more

Number of Worker nodes: 1

Worker node size
D3 v2 (1 node, 4 cores)

Head node size
D3 v2 (2 nodes, 8 cores)

WORKER NODES	0.622 x 1 = 0.622
HEAD NODES	0.622 x 2 = 1.244
TOTAL COST	1.87 USD/HOUR (ESTIMATED)

Note: *You can select lowest pricing tier A3 nodes or reduce the number of worker nodes decrease the cluster cost.*

k. Leave other configuration options as default and click **Create** to provision HDInsight Hadoop cluster. It will take 15-20 minutes for cluster provisioning.

The HDInsight Linux Hadoop cluster is now ready to work with.

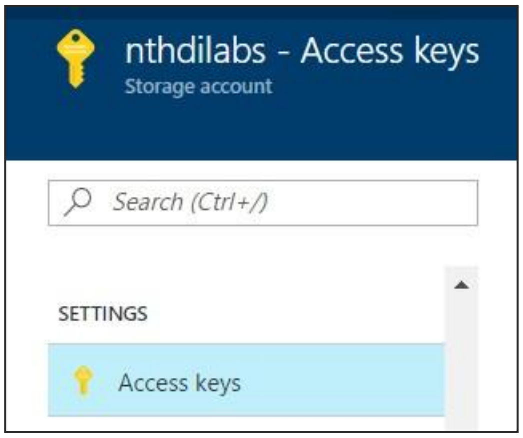
You can install HDInsight tools for visual studio using Web Platform Installer (http://go.microsoft.com/fwlink/?linkid=255386&clcid=0x409). HDInsight tools for visual studio are packaged with the Azure SDK for .Net. Install the one that matches your visual studio version.

Copy lab data to the storage account

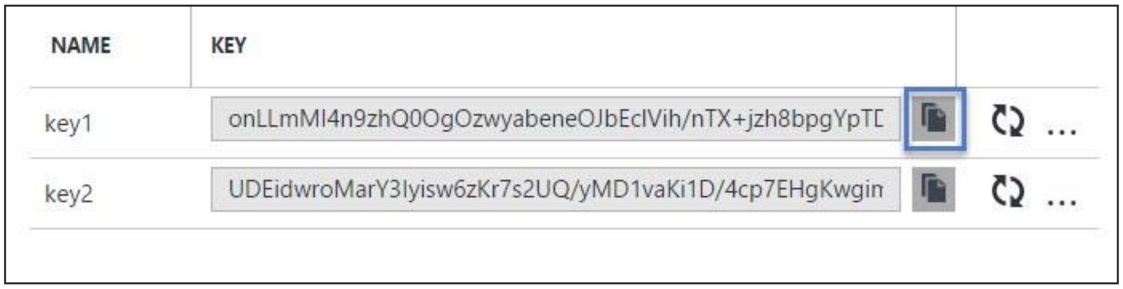
In this section, you'll copy the files required for the lab to your storage account. You'll copy the files between two storage account with the help of AzCopy utility. You can download the utility from here http://aka.ms/downloadazcopy

To copy the files, follow the below steps.

1. Copy your Azure Storage account access keys. This is required to copy data from the source Azure Storage account to your Azure Storage account. To get your storage account access key, navigate to your storage account on the Azure Management Portal and select **Access keys** under **Settings**.



2. Click on the copy icon to copy **Key1** from the **Access Keys** pane.



NAME	KEY
key1	onLLmMI4n9zhQ0OgOzwyabeneOJbEclVih/nTX+jzh8bpgYpTD
key2	UDEidwroMarY3lyisw6zKr7s2UQ/yMD1vaKi1D/4cp7EHgKwgin

3. Press Window + R to open the run window. Type cmd and press enter to open a new command console window.

4. Change the directory to C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy.

5. Copy and paste the following command on the console window to transfer **weblogs.csv** file from the source storage account to your storage account.

```
AzCopy /Source:https://mtlworkshop.blob.core.windows.net/hdi/Hive/
/Dest:https://mtllab<1-12>.blob.core.windows.net/hdi/bookstore/weblogs/
/SourceKey:s2eYqDZOhBOOv0glTcneXYC7t5jld58rP28BddfdP4Mv4/hI+pArEUAFjVgWORdyUC1Kx
xro0o144vmm8QfKRw== /DestKey:<KEY1> /Pattern:weblogs.csv
```

*Note: Replace **<1-12>** with your name or a unique ID, so that your files do not get processed by other jobs.*

Copy and paste the following command to copy the **HiveApp.exe** from the source storage account to your storage account.

```
AzCopy /Source:https://mtlworkshop.blob.core.windows.net/hdi/Hive/
/Dest:https://mtllab<1-12>.blob.core.windows.net/hdi/
/SourceKey:s2eYqDZOhBOOv0glTcneXYC7t5jld58rP28BddfdP4Mv4/hI+pArEUAFjVgWORdyUC1Kx
xro0o144vmm8QfKRw== /DestKey:<KEY1> /Pattern:Hiveapp.exe
```

Note: Replace **`<1-12>`** *with your unique ID and* ***<KEY1>*** *with your storage account key.*

Different ways to execute Hive queries on HDInsight cluster

HDInsight provides different platforms to execute a Hive query. In this section we'll learn to execute hive queries with

1. Ambari Hive view
2. Hive command line interface
3. Beeline command line tool
4. PowerShell
5. Microsoft Windows Azure Management Libraries
6. Visual Studio tools for HDInsight

Ambari Hive View

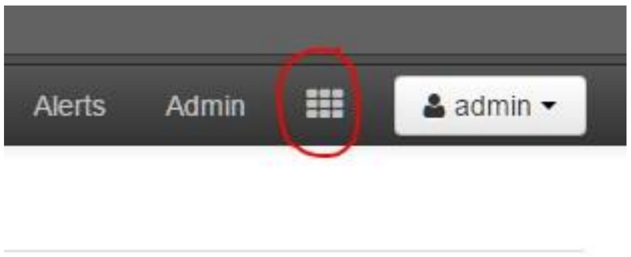
To run a hive job through Ambari Hive View, follow the below steps.

1. Navigate to cluster dashboard on Azure Portal. Select **Ambari Views** under **Quick Links** section. For

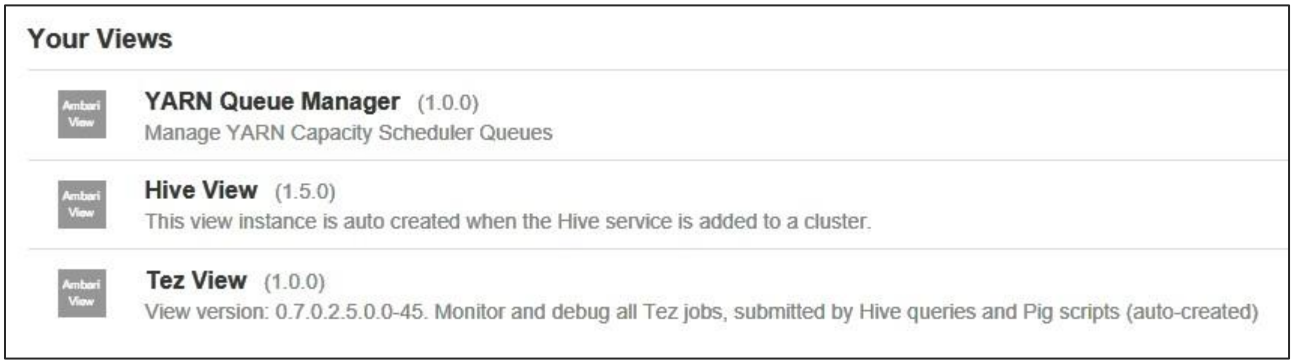
   Username: admin

   Password: HDItut@123

2. Click on the ellipses to explore views:



3. Click **Hive View** to open the Ambari Hive View.


Your Views

YARN Queue Manager (1.0.0)
Manage YARN Capacity Scheduler Queues

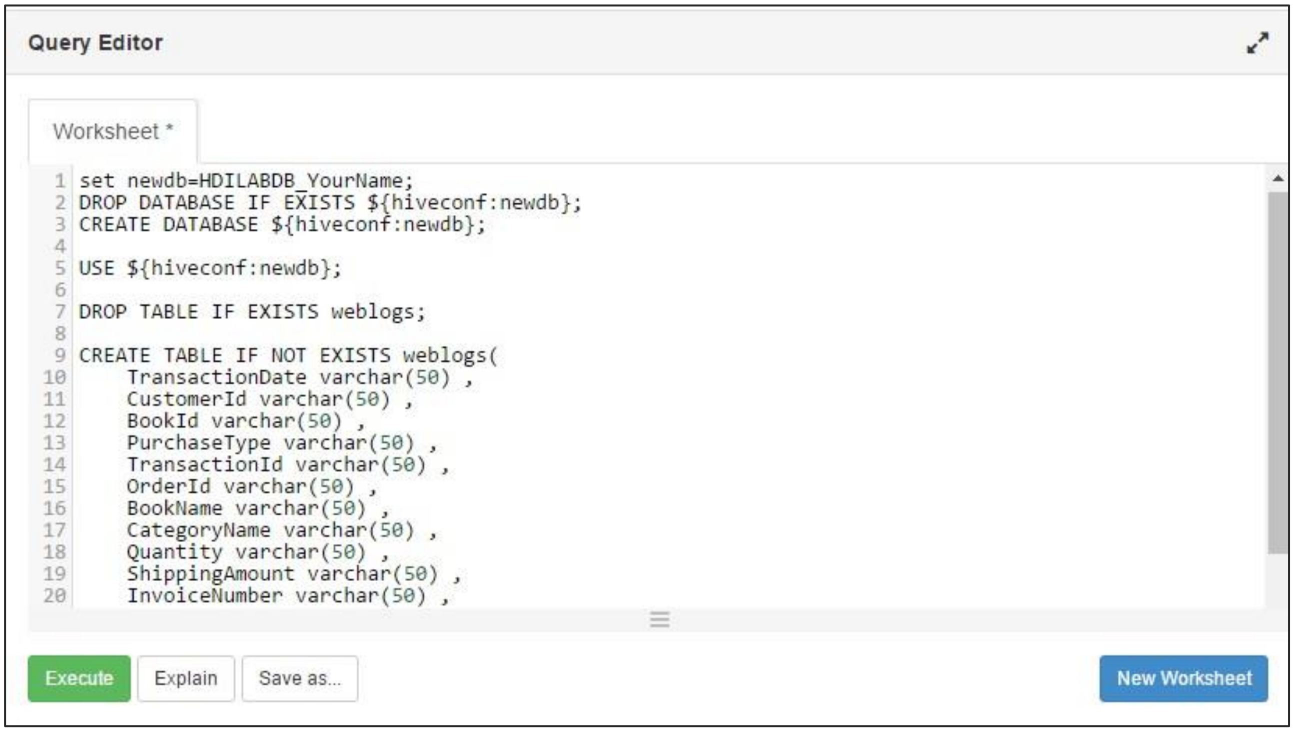
Hive View (1.5.0)
This view instance is auto created when the Hive service is added to a cluster.

Tez View (1.0.0)
View version: 0.7.0.2.5.0.0-45. Monitor and debug all Tez jobs, submitted by Hive queries and Pig scripts (auto-created)

4. Copy and paste the following query in the **Query Editor**. *Do not execute yet.*

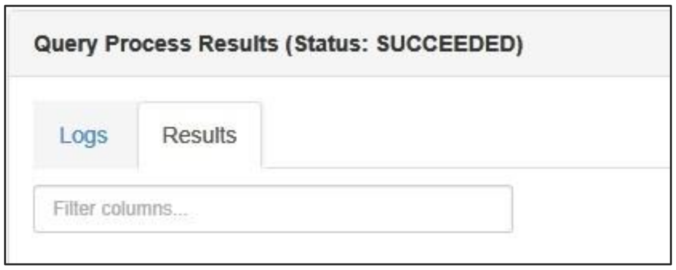
```
set newdb=HDILABDB;
DROP DATABASE IF EXISTS ${hiveconf:newdb};
CREATE DATABASE ${hiveconf:newdb};

USE ${hiveconf:newdb};

DROP TABLE IF EXISTS weblogs;

CREATE TABLE IF NOT EXISTS weblogs(
  TransactionDate varchar(50) ,
  CustomerId varchar(50),
  BookId varchar(50),
  PurchaseType varchar(50),
  TransactionId varchar(50),
  OrderId varchar(50),
  BookName varchar(50),
  CategoryName varchar(50),
  Quantity varchar(50),
  ShippingAmount varchar(50),
  InvoiceNumber varchar(50),
  InvoiceStatus varchar(50),
  PaymentAmount varchar(50)
) ROW FORMAT DELIMITED FIELDS TERMINATED by ',' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION '/bookstore/weblogs/';
```

*Note: The above query creates a new database, HDILABDB and creates a hive external table, **weblogs**, for the weblogs data which we uploaded to Azure Storage Account in the previous step.*



```
set newdb=HDILABDB_YourName;
DROP DATABASE IF EXISTS ${hiveconf:newdb};
CREATE DATABASE ${hiveconf:newdb};

USE ${hiveconf:newdb};

DROP TABLE IF EXISTS weblogs;

CREATE TABLE IF NOT EXISTS weblogs(
    TransactionDate varchar(50) ,
    CustomerId varchar(50) ,
    BookId varchar(50) ,
    PurchaseType varchar(50) ,
    TransactionId varchar(50) ,
    OrderId varchar(50) ,
    BookName varchar(50) ,
    CategoryName varchar(50) ,
    Quantity varchar(50) ,
    ShippingAmount varchar(50) ,
    InvoiceNumber varchar(50) ,
```

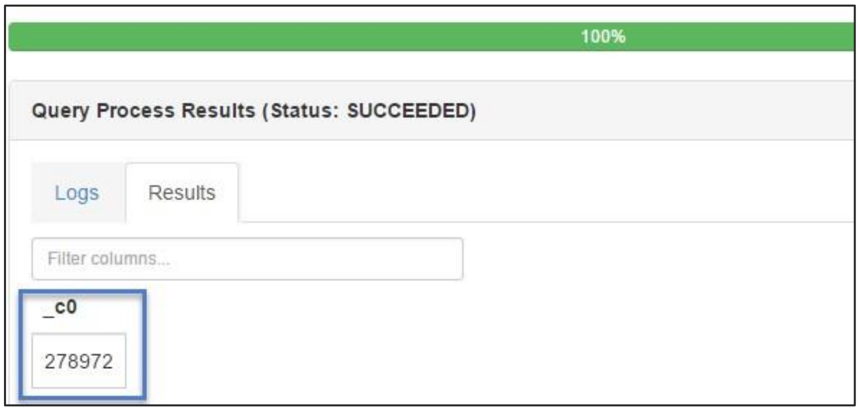
5. Click Execute to run the query. Once the query complete, the Query Process Results, status will change to **SUCCEEDED**.



6. To confirm the table creation, execute the following query in the **Query Editor**.

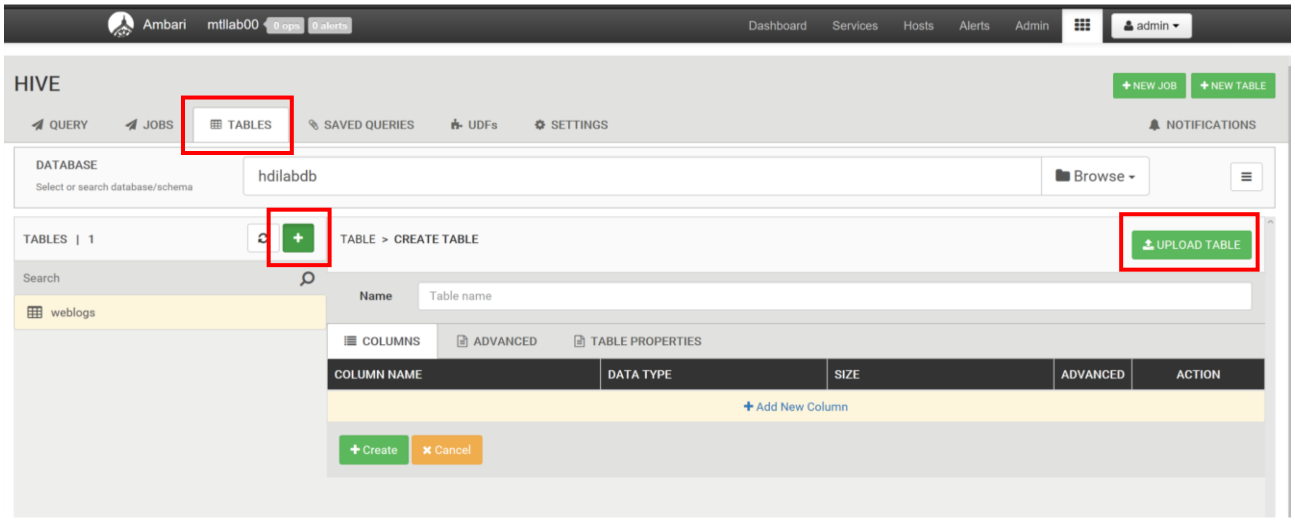
```
SELECT COUNT(*) FROM HDILABDB.weblogs;
```

You should get the following output. The weblogs table has **278972** records.

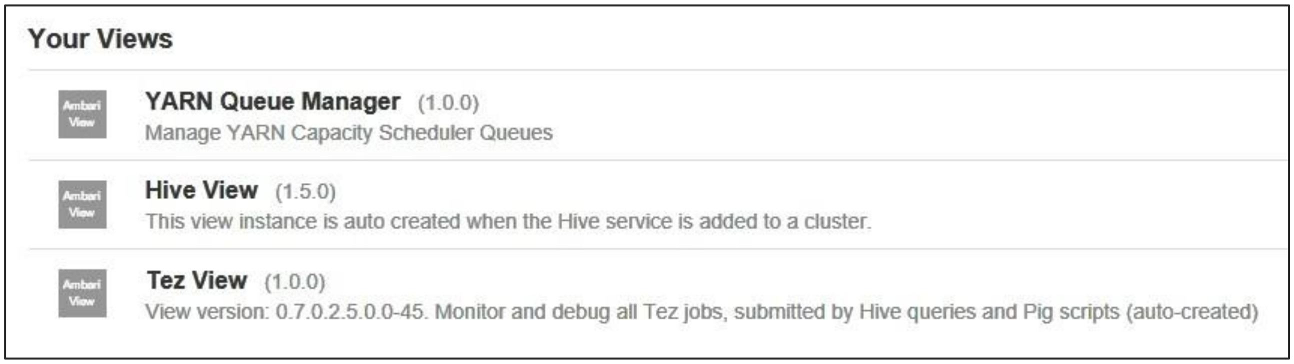


```
_c0
278972
```

Hive View can also be used to create an external table by uploading a csv, xml or json file from a local system or from the Azure storage account. To create an external table from the *Hive View*, select Upload Table option from the top menu and provide the required details.


Ambari mtllab00 0 ops 0 alerts
Dashboard Services Hosts Alerts Admin admin
HIVE
NEW JOB NEW TABLE
QUERY JOBS TABLES SAVED QUERIES UDFs SETTINGS NOTIFICATIONS
DATABASE
Select or search database/schema
hdilabdb
Browse
TABLES | 1
Search
weblogs
TABLE > CREATE TABLE
UPLOAD TABLE
Name Table name
COLUMNS ADVANCED TABLE PROPERTIES
COLUMN NAME DATA TYPE SIZE ADVANCED ACTION
Add New Column
Create Cancel

To view the details of the queries executed, select the **Tez View**, from the Ambari Views page



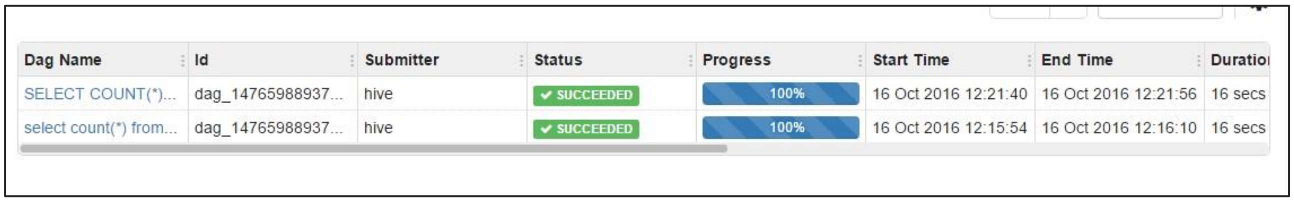
Your Views

YARN Queue Manager (1.0.0)
Manage YARN Capacity Scheduler Queues

Hive View (1.5.0)
This view instance is auto created when the Hive service is added to a cluster.

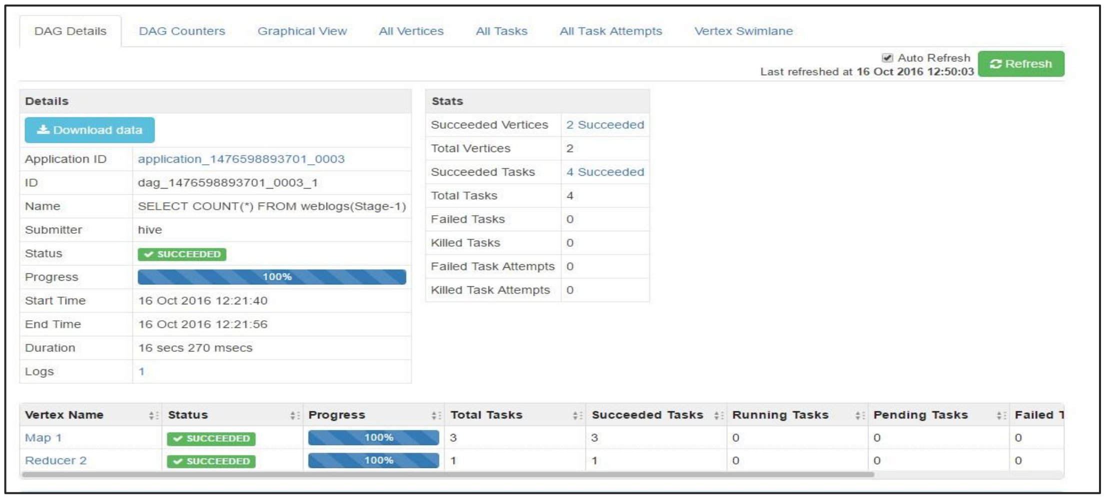
Tez View (1.0.0)
View version: 0.7.0.2.5.0.0-45. Monitor and debug all Tez jobs, submitted by Hive queries and Pig scripts (auto-created)

7. The **Tez View** displays the summary of all the hive jobs executed on the Hadoop cluster.



Dag Name	Id	Submitter	Status	Progress	Start Time	End Time	Duratio
SELECT COUNT(*)...	dag_14765988937...	hive	✓ SUCCEEDED	100%	16 Oct 2016 12:21:40	16 Oct 2016 12:21:56	16 secs
select count(*) from...	dag_14765988937...	hive	✓ SUCCEEDED	100%	16 Oct 2016 12:15:54	16 Oct 2016 12:16:10	16 secs

8. To get detailed information of a particular job, click on the query under the **Dag Name** column.


DAG Details DAG Counters Graphical View All Vertices All Tasks All Task Attempts Vertex Swimlane
Auto Refresh
Refresh
Download data

Last refreshed at 16 Oct 2016 12:50:03

Details	
Application ID	application_1476598893701_0003
ID	dag_1476598893701_0003_1
Name	SELECT COUNT(*) FROM weblogs(Stage-1)
Submitter	hive
Status	✓ SUCCEEDED
Progress	100%
Start Time	16 Oct 2016 12:21:40
End Time	16 Oct 2016 12:21:56
Duration	16 secs 270 msecs
Logs	1

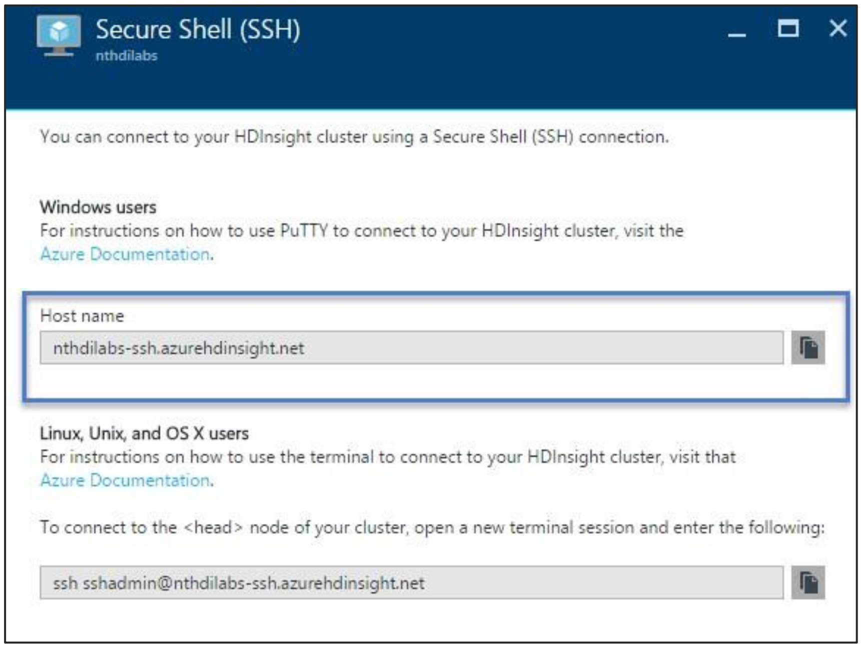
Stats	
Succeeded Vertices	2 Succeeded
Total Vertices	2
Succeeded Tasks	4 Succeeded
Total Tasks	4
Failed Tasks	0
Killed Tasks	0
Failed Task Attempts	0
Killed Task Attempts	0

Vertex Name	Status	Progress	Total Tasks	Succeeded Tasks	Running Tasks	Pending Tasks	Failed T
Map 1	✓ SUCCEEDED	100%	3	3	0	0	0
Reducer 2	✓ SUCCEEDED	100%	1	1	0	0	0

Hive command line interface

To run a hive query from Hive CLI, follow the below steps.

1. Navigate to the Hadoop cluster on the Azure Management Portal, and select **Secure Shell** from the cluster menu. Copy the host name from the **Secure Shell** pane



Secure Shell (SSH)
nthdilabs

You can connect to your HDInsight cluster using a Secure Shell (SSH) connection.

Windows users
For instructions on how to use PuTTY to connect to your HDInsight cluster, visit the Azure Documentation.

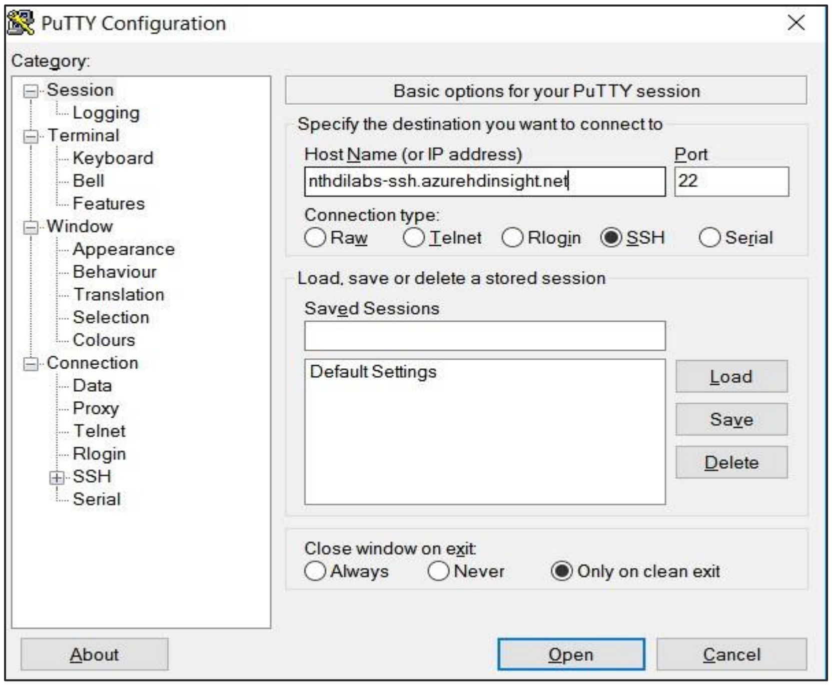
Host name
nthdilabs-ssh.azurehdinsight.net

Linux, Unix, and OS X users
For instructions on how to use the terminal to connect to your HDInsight cluster, visit that Azure Documentation.

To connect to the <head> node of your cluster, open a new terminal session and enter the following:

```
ssh sshadmin@nthdilabs-ssh.azurehdinsight.net
```

2. Click **putty** icon on the desktop, and enter the host name copied in step 1 under **Host Name** text box and Select **Open**.



PuTTY Configuration

Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address): nthdilabs-ssh.azurehdinsight.net
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

3. Enter the SSH username and password to connect to the HDInsight Hadoop cluster. In the console window, type **hive** and press enter to connect to hive CLI.

Username: sshuser

Password: HDItut@123

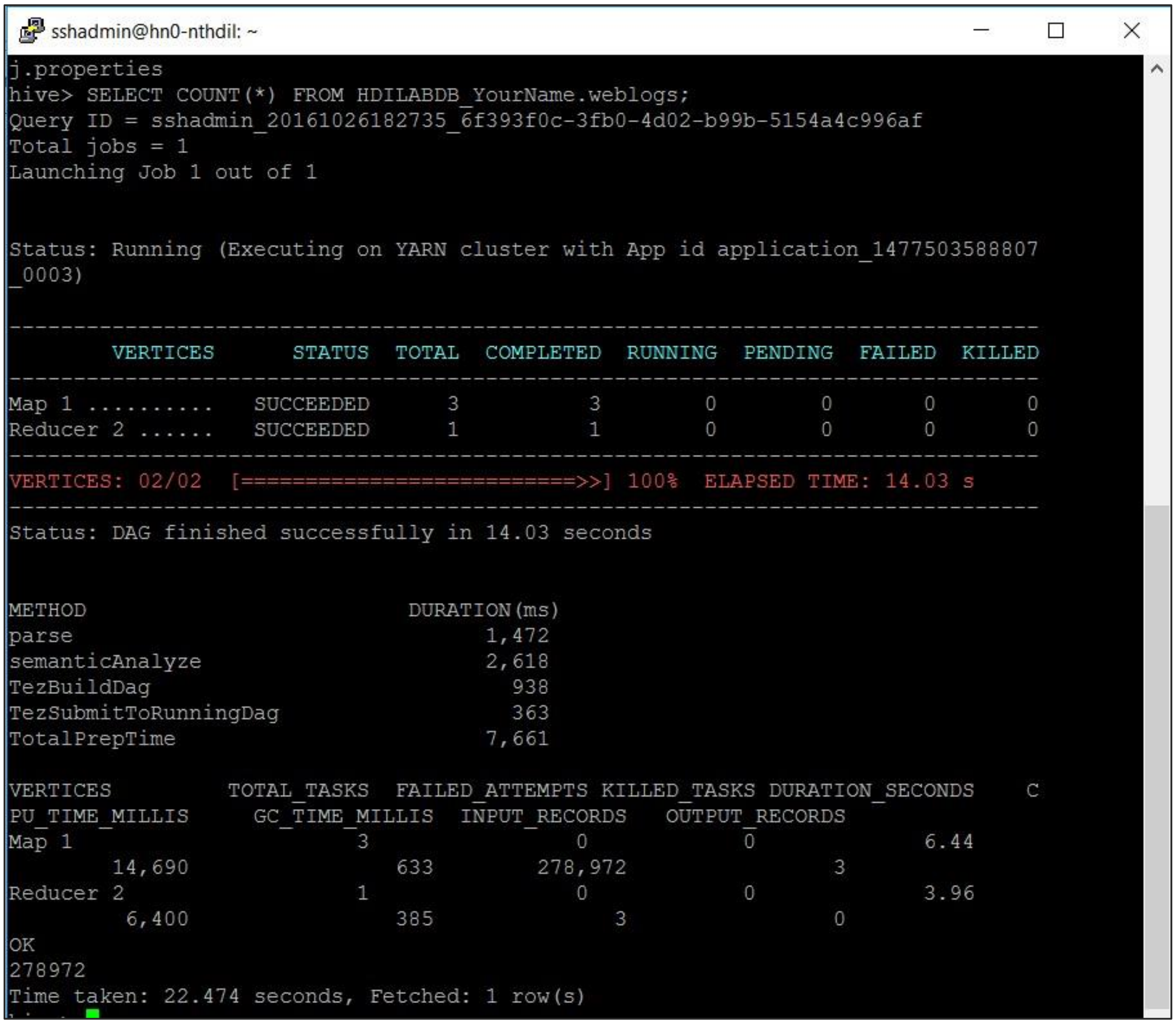
4. Start Hive.

```
sshuser@hn0-MTLHDi:~$ hive
```

5. Using the CLI, execute the following query to count the number of rows in the weblogs table.

```
SELECT COUNT(*) FROM HDILABDB.weblogs;
```

You should get the following output.



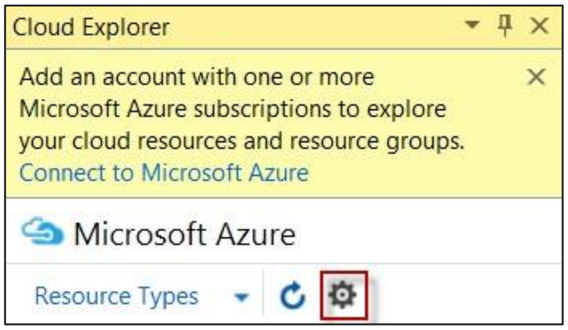
```
j.properties
hive> SELECT COUNT(*) FROM HDILABDB_YourName.weblogs;
Query ID = sshadmin_20161026182735_6f393f0c-3fb0-4d02-b99b-5154a4c996af
Total jobs = 1
Launching Job 1 out of 1


Status: Running (Executing on YARN cluster with App id application_1477503588807
_0003)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      3          3        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 14.03 s
--------------------------------------------------------------------------------
Status: DAG finished successfully in 14.03 seconds


METHOD                         DURATION(ms)
parse                                 1,472
semanticAnalyze                       2,618
TezBuildDag                             938
TezSubmitToRunningDag                   363
TotalPrepTime                         7,661

VERTICES         TOTAL_TASKS  FAILED_ATTEMPTS KILLED_TASKS DURATION_SECONDS    C
PU_TIME_MILLIS     GC_TIME_MILLIS  INPUT_RECORDS   OUTPUT_RECORDS
Map 1                      3                0            0             6.44
        14,690                633        278,972                3
Reducer 2                  1                0            0             3.96
         6,400                385              3                0
OK
278972
Time taken: 22.474 seconds, Fetched: 1 row(s)
```

6. Exit from the hive command line by typing 'exit;'

Execute Hive query with Visual Studio tools for HDInsight

To run a hive job through Visual Studio, follow the below steps.

1. Open Visual Studio and select View from the top menu and then select Cloud Explorer. Click on "Connect to Microsoft Azure" in the yellow tool tip or click on the settings icon (highlighted in red) to login to your Azure account.



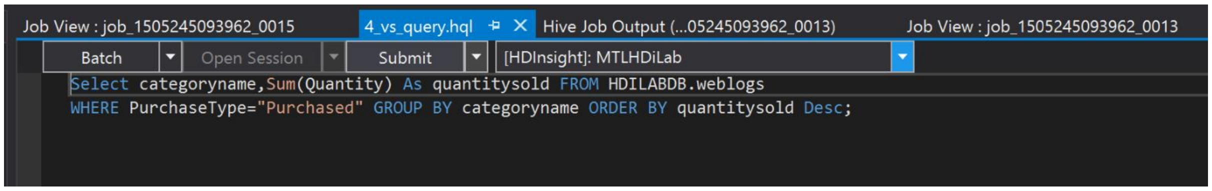
Add an account with one or more Microsoft Azure subscriptions to explore your cloud resources and resource groups.
Connect to Microsoft Azure

2. Navigate to **File -> Open ->File**. In the file open dialog box, browse to **\LabAssets\Scripts** and open **4_vs_query.hql** file.

3. Select **Execute Via HDILABDB** to submit the hive job to the HDInsight cluster.

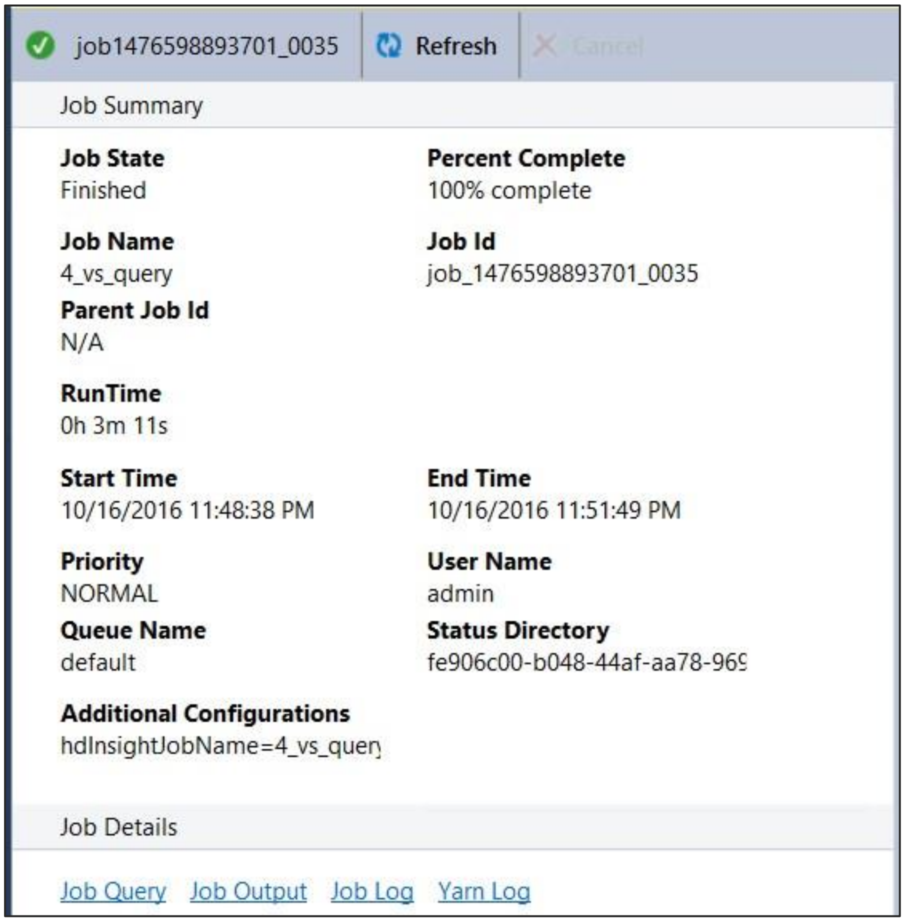
```
     Select categoryname,Sum(Quantity) As quantitysold FROM HDILABDB.weblogs
WHERE PurchaseType="Purchased" GROUP BY categoryname ORDER BY quantitysold Desc;
```

4. Click on dropdown on the left of **Submit** and select your HDInsight cluster.


Job View : job_1505245093962_0015
4_vs_query.hql
Hive Job Output (...05245093962_0013)
Job View : job_1505245093962_0013
Batch
Open Session
Submit
[HDInsight]: MTLHDiLab

```
Select categoryname,Sum(Quantity) As quantitysold FROM HDILABDB.weblogs
WHERE PurchaseType="Purchased" GROUP BY categoryname ORDER BY quantitysold Desc;
```

5. A hive job is submitted to HDInsight cluster and a Job View is displayed. You may have to refresh the job view to get the current job status.



Job State
Finished

Percent Complete
100% complete

Job Name
4_vs_query

Job Id
job_1476598893701_0035

Parent Job Id
N/A

RunTime
0h 3m 11s

Start Time
10/16/2016 11:48:38 PM

End Time
10/16/2016 11:51:49 PM

Priority
NORMAL

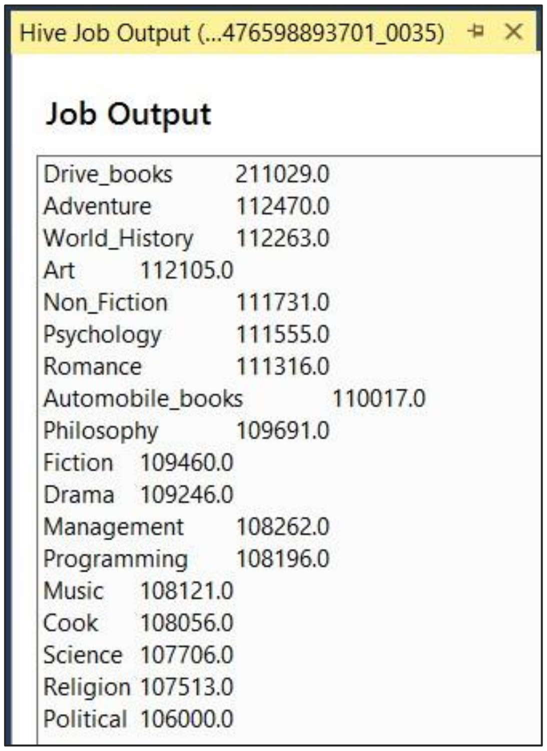
User Name
admin

Queue Name
default

Status Directory
fe906c00-b048-44af-aa78-969

Additional Configurations
hdInsightJobName=4_vs_query

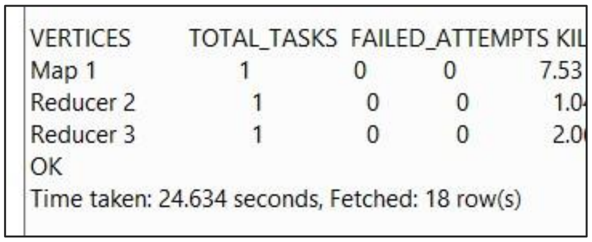
Once the job completes, click the **job output** link at the bottom of the job view. The query result is displayed in a new tab.



```
Drive_books	211029.0
Adventure	112470.0
World_History	112263.0
Art	112105.0
Non_Fiction	111731.0
Psychology	111555.0
Romance	111316.0
Automobile_books	110017.0
Philosophy	109691.0
Fiction	109460.0
Drama	109246.0
Management	108262.0
Programming	108196.0
Music	108121.0
Cook	108056.0
Science	107706.0
Religion	107513.0
Political	106000.0
```

Close the **job output** tab.

6. On the **Job View** tab, click on **Job Log** link. Observe that the query completed in **24.6 seconds**.



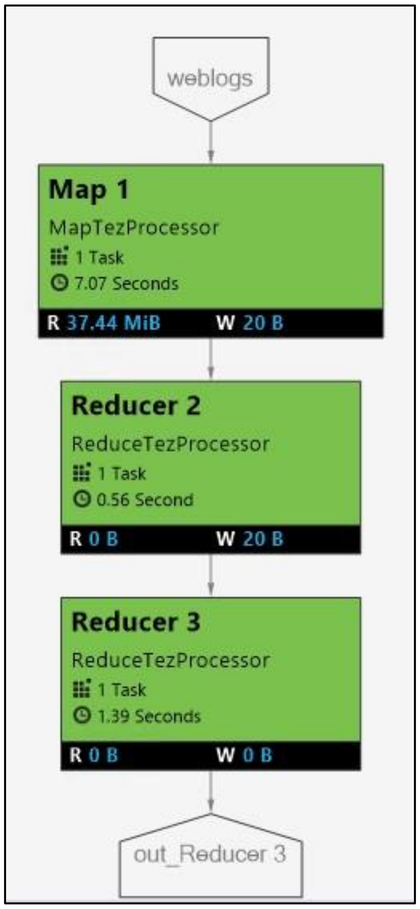
```
VERTICES       TOTAL_TASKS  FAILED_ATTEMPTS KIL
Map 1               1            0        0       7.53
Reducer 2           1            0        0       1.0
Reducer 3           1            0        0       2.0
OK
Time taken: 24.634 seconds, Fetched: 18 row(s)
```

Tez Performance Graph

Apache Tez is a distributed execution framework for data processing applications. Built on top of Yarn, it is based on expressing computation as a directed acyclic graph. Vertices in graph represent data transformation and edges represent data movement from producers to consumers.

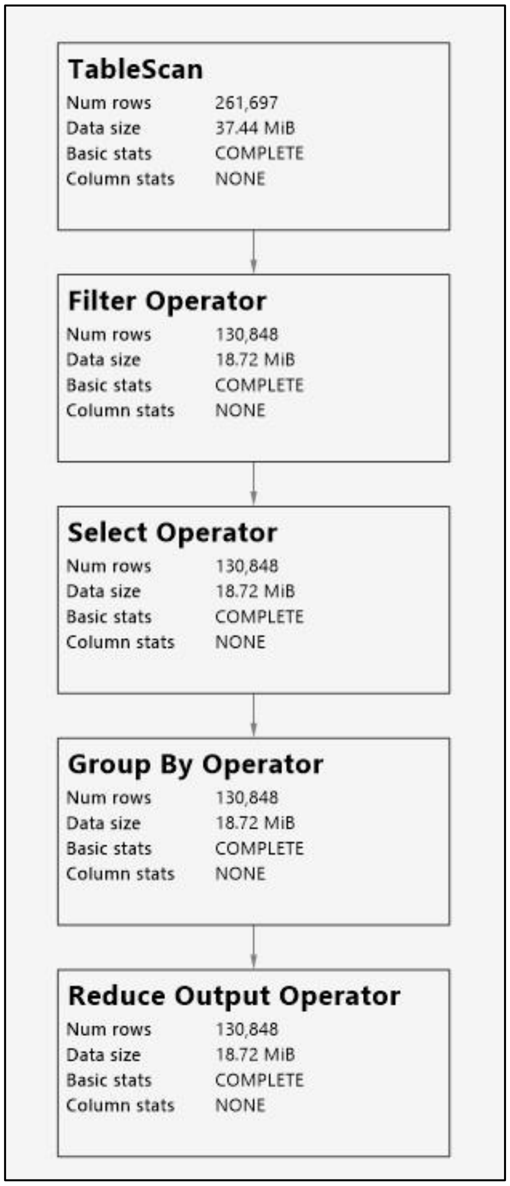
Visual Studio allows for submitting hive jobs through HDInsight tools for visual studio. You can install it using Web Platform Installer (http://go.microsoft.com/fwlink/?linkid=255386&clcid=0x409). Hindsight tools for visual studio are packaged with the Azure SDKK for .Net. Install the one that matches your visual studio version. *Note: This feature is only available for HDInsight cluster version above 3.2.4.593, and can only work for completed jobs. This works for both Windows and Linux based clusters.*

In the last step, you executed hive query using visual studio tools for HDInsight. Navigate to the **Job View** tab as mentioned in the last step. Observe the Tez performance graph on the right hand side of the **Job View** tab. The performance graph consists of Edges and Vertices. Vertices represent data transformation and edges represents data movement between vertices. The Map 1 vertex took 7.07 seconds to read 37.04 MB of data and write 20 B of data.

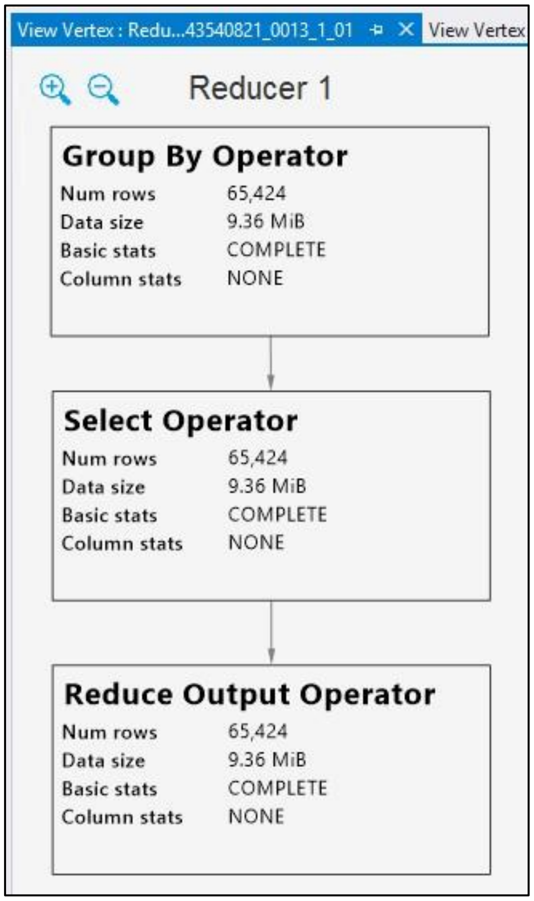


Note: The time taken by each vertices may be different in your case.

The job was executed with 1 mapper and 2 reducers. Double click on "Map 1" vertex. A detailed vertex view is displayed. The graph shows the operators used in a vertex. It also shows the number of rows and size of the data processed by each operator.



Double click on “Reducer 2” and “Reducer 3”. A detailed vertex view is displayed, showing the execution plan of the reducers.



Group By Operator

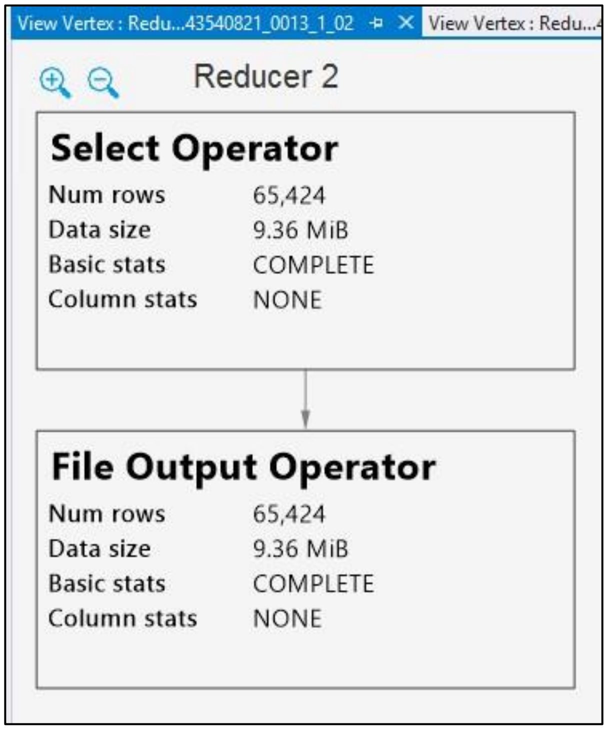
Num rows	65,424
Data size	9.36 MiB
Basic stats	COMPLETE
Column stats	NONE

Select Operator

Num rows	65,424
Data size	9.36 MiB
Basic stats	COMPLETE
Column stats	NONE

Reduce Output Operator

Num rows	65,424
Data size	9.36 MiB
Basic stats	COMPLETE
Column stats	NONE



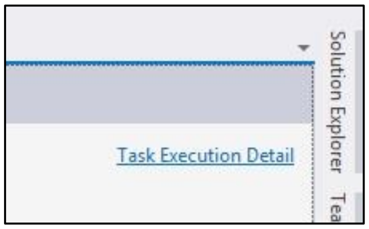
Select Operator

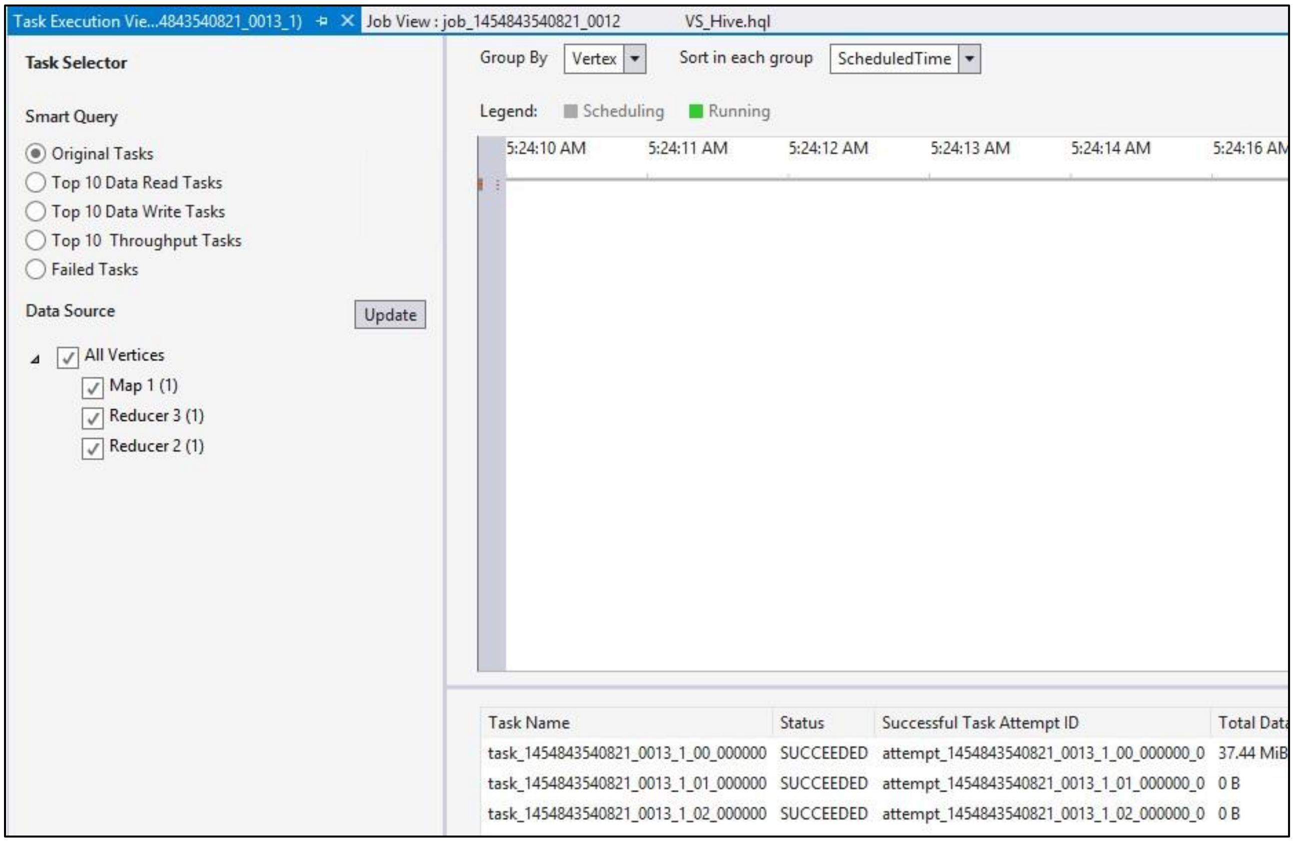
Num rows	65,424
Data size	9.36 MiB
Basic stats	COMPLETE
Column stats	NONE

File Output Operator

Num rows	65,424
Data size	9.36 MiB
Basic stats	COMPLETE
Column stats	NONE

7. On the top right corner of the performance graph view, click "Task Execution Detail" link, to get more detailed job information.





Task Name	Status	Successful Task Attempt ID	Total Data
task_1454843540821_0013_1_00_000000	SUCCEEDED	attempt_1454843540821_0013_1_00_000000_0	37.44 MiB
task_1454843540821_0013_1_01_000000	SUCCEEDED	attempt_1454843540821_0013_1_01_000000_0	0 B
task_1454843540821_0013_1_02_000000	SUCCEEDED	attempt_1454843540821_0013_1_02_000000_0	0 B

Hive on Tez vs Hive on MapReduce

The task view displays how each task operator and detailed information about each task such as data read/write, spill written to disk. This information is helpful in tuning jobs.

Note: The default execution engine for Linux based HDInsight cluster is Tez.

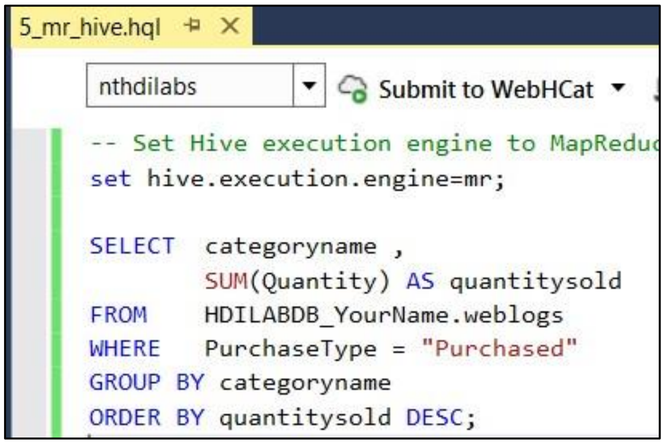
To execute the query with MapReduce execution engine, follow the below steps,

1. Navigate to **File -> Open ->File**. In the file open dialog box, browse to **\LabAssets\Scripts** and open **5_mr_hive.hql** file.

```
set hive.execution.engine=mr;
SELECT  categoryname ,
        SUM(Quantity) AS quantitysold
FROM    HDILABDB.weblogs
WHERE   PurchaseType = "Purchased"
GROUP BY categoryname
ORDER BY quantitysold DESC;
```

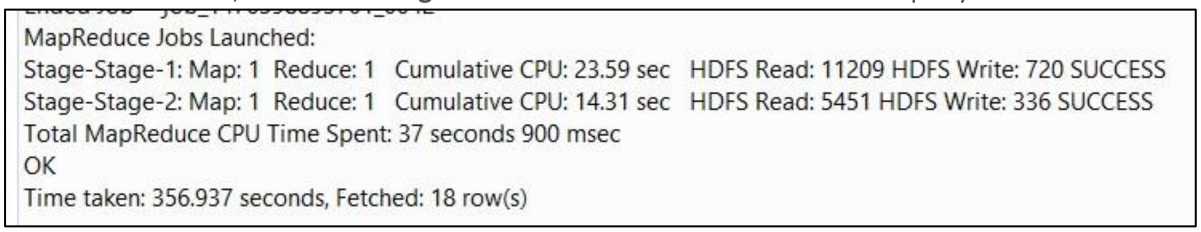
2. Observe that the query is same as 4_vs_query.hql, however, the hive configuration parameter **hive.execute.engine** is set to **mr**. This will force cluster to run the query with MapReduce execution engine.

3. Make sure that the cluster name is set to your HDInsight cluster. Select **Execute Via HiveServer2** to execute the query.



```
-- Set Hive execution engine to MapReduc
set hive.execution.engine=mr;

SELECT   categoryname ,
         SUM(Quantity) AS quantitysold
FROM     HDILABDB_YourName.weblogs
WHERE    PurchaseType = "Purchased"
GROUP BY categoryname
ORDER BY quantitysold DESC;
```

4. In the **Job View** tab, click on **Job Log** and observe the execution time of the query.



```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 23.59 sec   HDFS Read: 11209 HDFS Write: 720 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 14.31 sec   HDFS Read: 5451 HDFS Write: 336 SUCCESS
Total MapReduce CPU Time Spent: 37 seconds 900 msec
OK
Time taken: 356.937 seconds, Fetched: 18 row(s)
```

The query took **357 seconds** to execute. Also, observer that the Tez performance graph isn't generated because the query is executed with MapReduce execution engine.

In the section "**Execute hive query with visual studio tools for HDInsight**", you executed a hive query. The query was executed on Tez execution engine. The query took **24.6 seconds** to execute. The Tez execution is much faster than the MapReduce execution engine.

Note, delete the Azure resource group **MTLLAB**

Hive on Tez vs Hive on Tez+LLAP

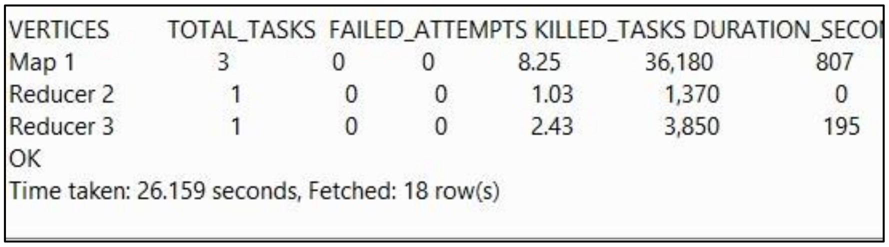
Live Long and Process (LLAP) added to Hive 2.0, is a hybrid execution model, consists of long-lived daemon. The daemon process functionalities such as caching, pre-fetching and access control. Small or short queries are processed by the daemon whereas heavy queries are processed by the YARN containers.

LLAP is not an execution engine, rather it's an enhancement over the existing execution engine. This is available as Tech Preview in HDInsight 3.5.

Please deploy a new HDInsight but select Hive Interactive as the cluster type.

1. Once deployed, copy the lab data to the storage account as per the previous step
2. open the Azure Management portal and select the HDInsight cluster. On the cluster pane, select **dashboard** from the menu. This will open the Ambari in a new tab.
3. Switch to Hive view
4. Create the same weblogs table.
5. Navigate to the **4_vs_query.hql** query window in visual studio, confirm that the correct HDInsight cluster is selected, select the **interactive** method and **execute** the query.

   Note: The query will now run on Tez + LLAP. There are no other configuration settings required at the query level.
6. Once the query completes, click the **Job Log** link or **HiveServer2 Output Tab** at the bottom of the **Job View**. Observe the execute time. The query takes **26 seconds** to execute.



```
VERTICES      TOTAL_TASKS  FAILED_ATTEMPTS KILLED_TASKS DURATION_SECO
Map 1              3            0        0       8.25         36,180          807
Reducer 2            1            0        0       1.03          1,370            0
Reducer 3            1            0        0       2.43          3,850          195
OK
Time taken: 26.159 seconds, Fetched: 18 row(s)
```

Perform book store sales analysis

In this section, you'll run hive queries to analyse the data in the weblogs table. The weblogs table contains transactional data of an imaginary online bookstore. You'll have to analyse the sales data and prepare a sales report.

All analysis is based on the weblogs table, created earlier in the lab. The table description is given below

Column	Description
TransactionDate	The date of the transaction
CustomerId	Unique Id assigned to the customer
BookId	Unique id assigned to a book in the book store

PurchaseType	1. **Purchased:** Customer bought the book 2. **Browsed:** Customer browsed but not purchased the book. 3. **Added to Cart:** Customer added the book to the shopping cart
TransactionId	Unique Id assigned to a transaction
OrderId	Unique order id
BookName	The name of the book accessed by the customer
CategoryName	The category of the book accessed by the customer
Quantity	Quantity of the book purchased. Valid only for PurchaseType = Purchased
ShippingAmount	Shipping cost
InvoiceNumber	Invoice number if a customer purchased the book
InvoiceStatus	The status of the invoice
PaymentAmount	Total amount paid by the customer. Valid only for PurchaseType = Purchased

Problem Statement

Write a query to return the total payment amount for each category per month. The output should look like this.

CategoryName	QuantitySold	TotalAmount
Drive_books	211029	2064435
Adventure	112470	1022195
World_History	112263	1048990
Art	112105	1043190
Non_Fiction	111731	1046410
Psychology	111555	1024255
Romance	111316	1038265
Automobile_books	110017	1030720
Philosophy	109691	1042410
Fiction	109460	1032795
Drama	109246	1038565
Management	108262	1030805
Programming	108196	1013210
Music	108121	998930
Cook	108056	1051710
Science	107706	1063445
Religion	107513	999780
Political	106000	1034820

Save the result in a table, **SalesbyCategory**. The table should be created in a new folder **SalesbyCategory** in the container created in step "Create a new Azure Storage Container".

Solution

1. Double click visual studio icon on desktop to open the Visual Studio.

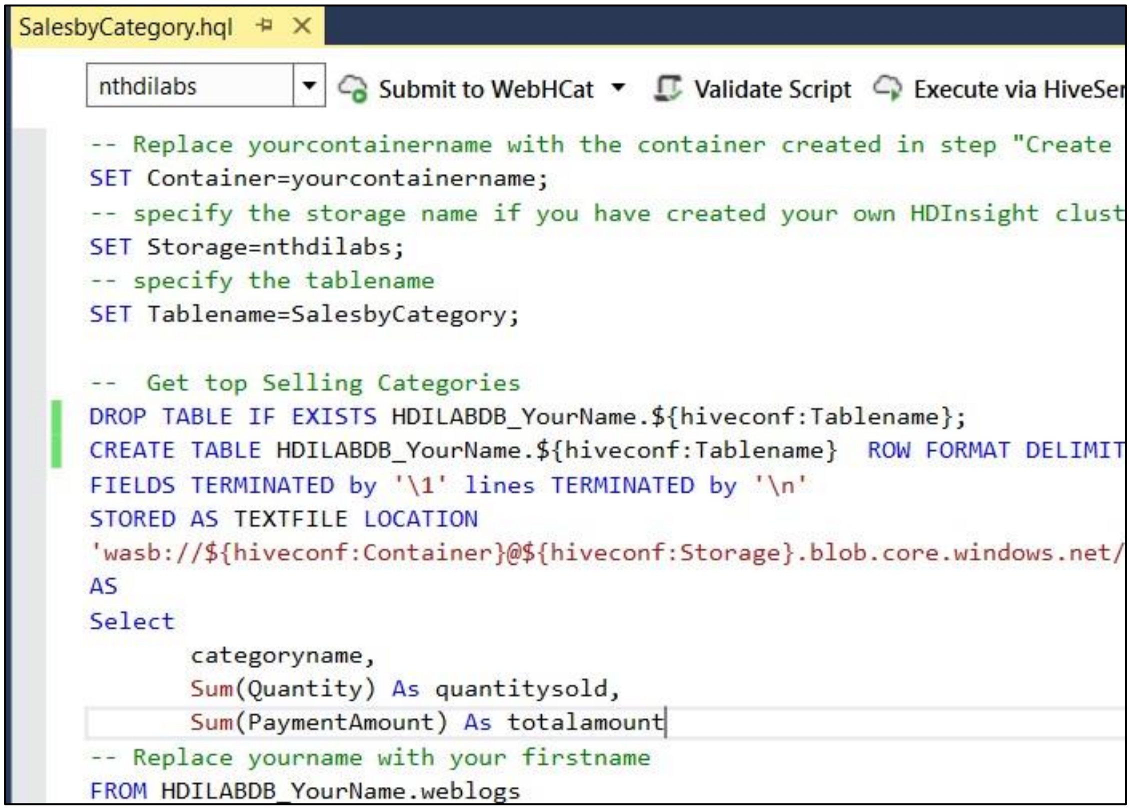
2. Navigate to **File->Open->Project/Solution**. In the file open dialog box, select **\LabAssets\Solutions\bookstoresalesanalaysis\bookstoresalesanalaysis.sln**. This will load the **bookstoresalesanalysis** solution into the visual studio.

3. From the solution explorer, open **SalesbyCategory.hql**.

```
-- Replace yourcontainername with the container created in step n
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>
-- specify the tablename
SET Tablename=SalesbyCategory;

--  Get top Selling Categories
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename}  ROW FORMAT DELIMITED
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstor
e/HDILABDB.${hiveconf:Tablename}/'
AS
Select
       categoryname,
       Sum(Quantity) As quantitysold,
       Sum(PaymentAmount) As totalamount
FROM HDILABDB.weblogs
WHERE PurchaseType="Purchased"
GROUP BY CategoryName
ORDER BY QuantitySold Desc;
```

Note: The above query sums up the quantity and the payment amount for each available category. The result is stored in SalesbyCategory table.

4. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the query.



```
-- Replace yourcontainername with the container created in step "Create
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsight clust
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=SalesbyCategory;

--  Get top Selling Categories
DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename}  ROW FORMAT DELIMIT
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/
AS
Select
       categoryname,
       Sum(Quantity) As quantitysold,
       Sum(PaymentAmount) As totalamount
-- Replace yourname with your firstname
FROM HDILABDB_YourName.weblogs
```

Problem Statement

Write a query to return the total payment amount and the total quantity sold per book. The output should look like this.

BookName	QuantitySold	TotalAmount
The voyages of Captain Cook	232414	2194890
Advances in school psychology	231410	2193740
Science in Dispute	231408	2168425
History of political economy	231255	2190040
THE BOOK OF WITNESSES	230872	2145540
The adventures of Arthur Conan Doyle	230023	2191910
Space fact and fiction	229908	2171820
New Christian poetry	228849	2185845
Understanding American politics	228598	2182720

Save the result in a table, **SalesbyBooks**. The table should be created in a new folder **SalesbyBooks** in the container created in step “Create a new Azure Storage Container”.

Solution

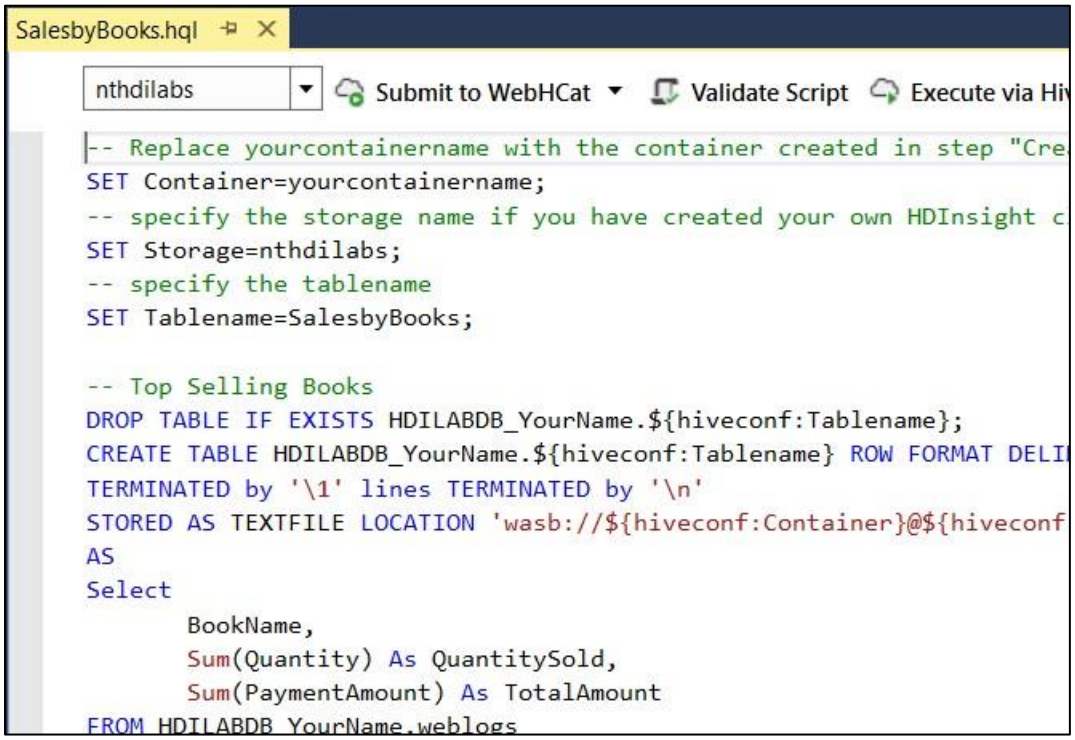
1. From the solution explorer, open **SalesbyBooks.hql**.

```
-- Replace yourcontainername with the container created in step "Create a new Azure
Storage Container"
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
```

```
-- specify the tablename
SET Tablename=SalesbyBooks;

-- Top Selling Books
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/HDILABDB.${h
iveconf:Tablename}/'
AS
Select
       BookName,
       Sum(Quantity) As QuantitySold,
       Sum(PaymentAmount) As TotalAmount
FROM HDIDBLAB.weblogs
WHERE PurchaseType='Purchased'
GROUP BY BookName
ORDER BY QuantitySold Desc;
```

Note: The above query returns the total quantity sold for each book available in the bookstore. The result is saved in SalesByBooks table.

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **SalesByBooks.hql**.



```
-- Replace yourcontainername with the container created in step "Cre
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsight c
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=SalesbyBooks;

-- Top Selling Books
DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT DELI
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hiveconf
AS
Select
       BookName,
       Sum(Quantity) As QuantitySold,
       Sum(PaymentAmount) As TotalAmount
FROM HDILABDB_YourName.weblogs
```

Problem Statement

Write a query to return the top 3 books browsed by the customers who also browsed the book, **THE BOOK OF WITNESSES**. Your output should look like this

BookName	cnt
New Christian poetry	9445

History of political economy	9384
Science in Dispute	9367

Save the result in a table, **BookSuggestions**. The table should be created in a new folder **BookSuggestions** in the container created in step "Create a new Azure Storage Container".

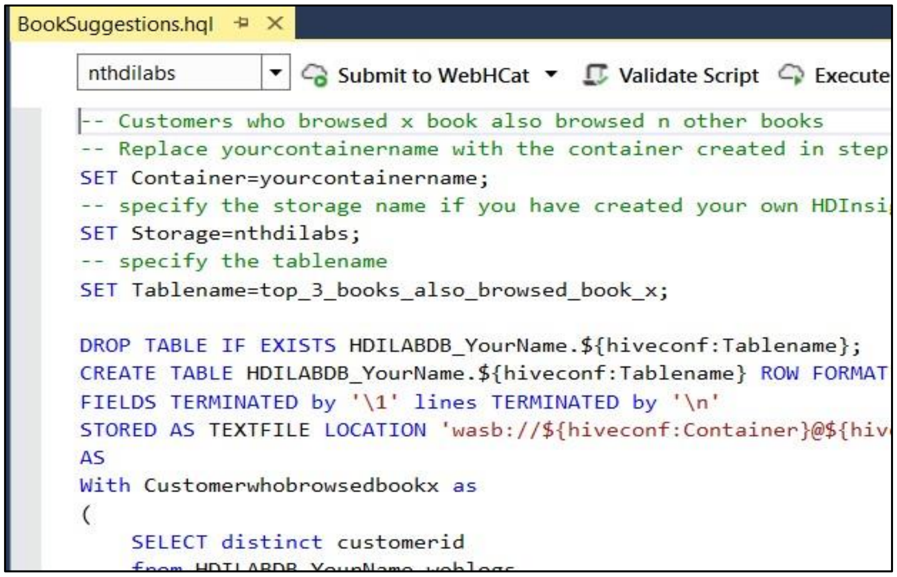
Solution

1. From the solution explorer, open **BookSuggestions.hql**

```
-- Customers who browsed x book also browsed n other books
-- Replace yourcontainername with the container created in step n
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename
SET Tablename= BookSuggestions;

DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HDILABDB
AS
With Customerwhobrowsedbookx as
(
      SELECT distinct customerid    from weblogs
      WHERE PurchaseType="Browsed" and BookName="THE BOOK OF WITNESSES"
)
SELECT w.BookName,count(*) as cnt from HDILABDB.weblogs w
JOIN Customerwhobrowsedbookx cte on w.CustomerId=cte.CustomerId
WHERE w.PurchaseType="Browsed"
AND w.BookName Not in ("THE BOOK OF WITNESSES") group by w.bookname having
count(*)>10 order by cnt desc
LIMIT 3;
```

*Note: The above query uses common table expression (CTE) to find all the distinct customers who browsed the book **"The Book Of Witnesses"**. The outer query then joins weblogs with the CTE on **customerid** column to find top 3 books browsed by the customers who browsed, **"The Book Of Witnesses".***

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **BookSuggestions.hql**.



```
-- Customers who browsed x book also browsed n other books
-- Replace yourcontainername with the container created in step
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsi
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=top_3_books_also_browsed_book_x;

DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hiv
AS
With Customerwhobrowsedbookx as
(
    SELECT distinct customerid
```

Problem Statement

Use Rank, ranking function to write a query to assign a rank to each customer based on the quantity of product purchased across all transactions. Use the Rank function to assign the rank. The output should look like this

CustomerId	totalquantity	RankCustomerbyQuantity
HNTG8YGROW	210806	1
OCLA2XGTB8	210383	2
1AYIINRJT0	209640	3
9HEX4GFUH3	209171	4
JRK07IRCIJ	208835	5
T7DVIB2J8L	206842	6
Y9SPOA6IW7	205878	7
3CO5UI3EXX	205409	8
ZJX5W46O91	205013	9
VS753ZSZJA	202760	10

Save the result in a table, **CustomerRank**. The table should be created in a new folder **CustomerRank** in the container created in step "Create a new Azure Storage Container".

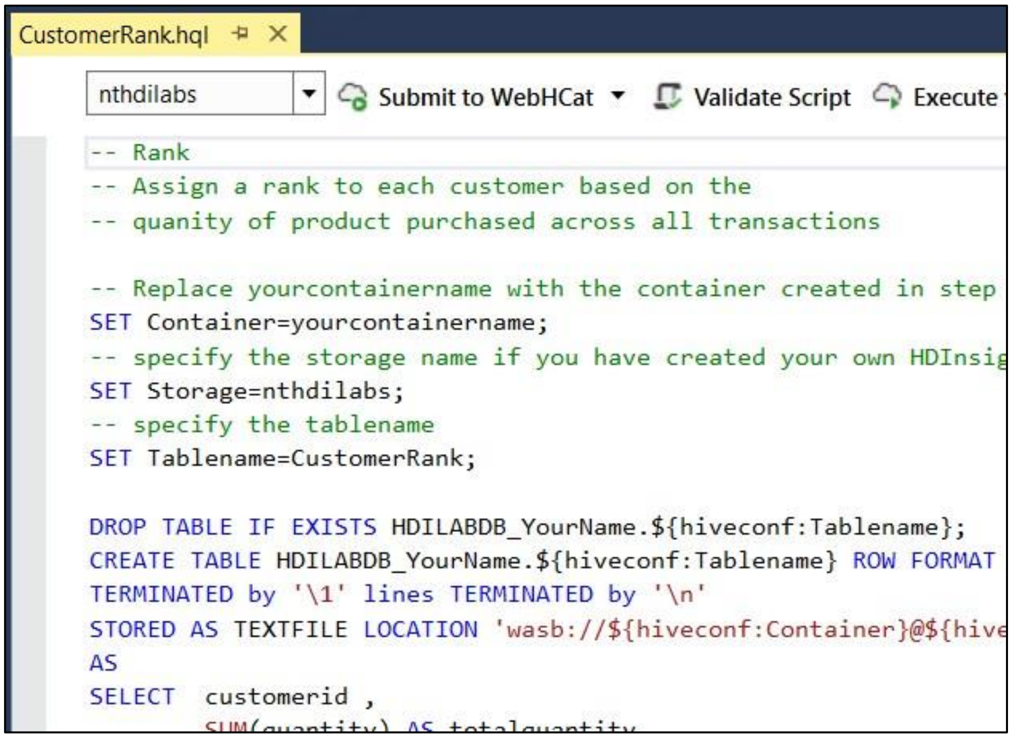
Solution

1. From the solution explorer, open **CustomerRank.hql**

```
-- Rank
-- Assign a rank to each customer based on the
-- quanity of product purchased across all transactions
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename
SET Tablename=CustomerRank;
```

```
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/
bookstore/HDILABDB.${hiveconf:Tablename}/'
AS
SELECT  customerid ,
        SUM(quantity) AS totalquantity ,
        RANK() OVER ( ORDER BY SUM(quantity) DESC ) AS RankCustomerbyQuantity
FROM    HDILABDB.weblogs
WHERE   purchasetype = "Purchased"
GROUP BY customerid;
```

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **CustomerRank.hql**



```
-- Rank
-- Assign a rank to each customer based on the
-- quanity of product purchased across all transactions

-- Replace yourcontainername with the container created in step
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsig
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=CustomerRank;

DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hive
AS
SELECT  customerid ,
```

Problem Statement

Use RowNumber ranking function to write a query to return total sales made in each month per category. Assign a row number (1 - n) to each row in result set, with 1 being the month with highest total sales and n being the month with the lowest total sales. Your output should look like this

rn	monthnumber	CategoryName	TotalSales
1	9	Adventure	202935
2	2	Adventure	199670
3	3	Adventure	116560
4	5	Adventure	112230
5	1	Adventure	104365

6	12	Adventure	98100
7	7	Adventure	95880
8	4	Adventure	92455
1	2	Art	215285
2	9	Art	207675
3	5	Art	119435
4	7	Art	105985
5	4	Art	101775
6	12	Art	100225
7	1	Art	97005
8	3	Art	95805
1	2	Automobile_books	217140
2	9	Automobile_books	203750
3	4	Automobile_books	108090
4	3	Automobile_books	107365
5	12	Automobile_books	99790
6	7	Automobile_books	98895
7	1	Automobile_books	98505
8	5	Automobile_books	97185

Note: The above output is trimmed for brevity. The total number of rows returned are 144.

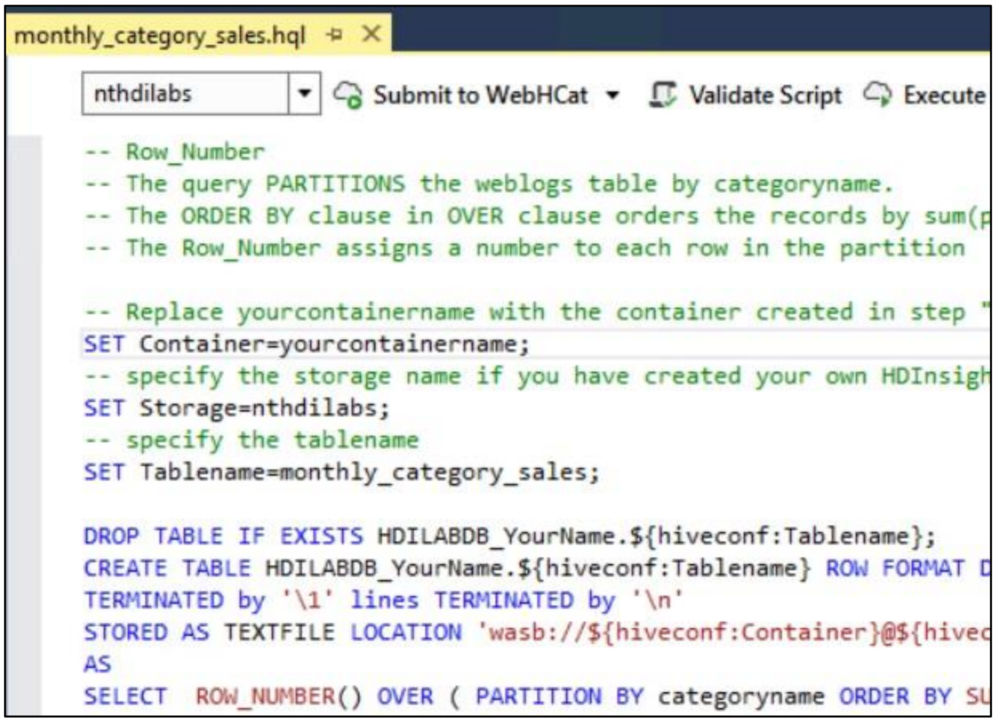
Save the result in a table, **monthly_category_sales**. The table should be created in a new folder **monthly_category_sales** in the container created in step “Create a new Azure Storage Container”.

Solution

1. From the solution explorer, open **monthly_category_sales.hql**

```
-- Row_Number
-- The query PARTITIONS the weblogs table by categoryname.
-- The ORDER BY clause in OVER clause orders the records by sum(paymentamount) in
each partition
-- The Row_Number assigns a number to each row in the partition
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>
-- specify the tablename
SET Tablename=monthly_category_sales;

DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS  TERMINATED
by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/
HDILABDB.${hiveconf:Tablename}/' AS
SELECT  ROW_NUMBER() OVER ( PARTITION BY categoryname ORDER BY SUM(paymentamount) )
AS rn ,
        MONTH(transactiondate) AS month ,

        categoryname,
        SUM(paymentamount) AS totalsales
FROM    HDILABDB.weblogs
WHERE   purchasetype = "Purchased"
GROUP BY MONTH(transactiondate) ,        categoryname;
```

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **monthly_category_sales.hql**



```
-- Row_Number
-- The query PARTITIONS the weblogs table by categoryname.
-- The ORDER BY clause in OVER clause orders the records by sum(p
-- The Row_Number assigns a number to each row in the partition

-- Replace yourcontainername with the container created in step "
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsigh
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=monthly_category_sales;

DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT D
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hivec
AS
SELECT  ROW_NUMBER() OVER ( PARTITION BY categoryname ORDER BY SU
```

Problem Statement

Use **ntile** ranking function to write a query to divide categories into four different groups numbered 1-4 based on the total sales made under each category. Group 1 consist of categories with highest sales, group 2 consists of categories with next highest sales and so on. Your output should look like this

Quartile	CategoryName	TotalSales
1	Drive_books	2064435
1	Science	1063445
1	Cook	1051710
1	World_History	1048990
1	Non_Fiction	1046410
2	Art	1043190
2	Philosophy	1042410
2	Drama	1038565
2	Romance	1038265
2	Political	1034820
3	Fiction	1032795
3	Management	1030805
3	Automobile_books	1030720
3	Psychology	1024255
4	Adventure	1022195
4	Programming	1013210
4	Religion	999780
4	Music	998930

Save the result in a table, **ntile**. The table should be created in a new folder **ntile** in the container created in step “Create a new Azure Storage Container”.

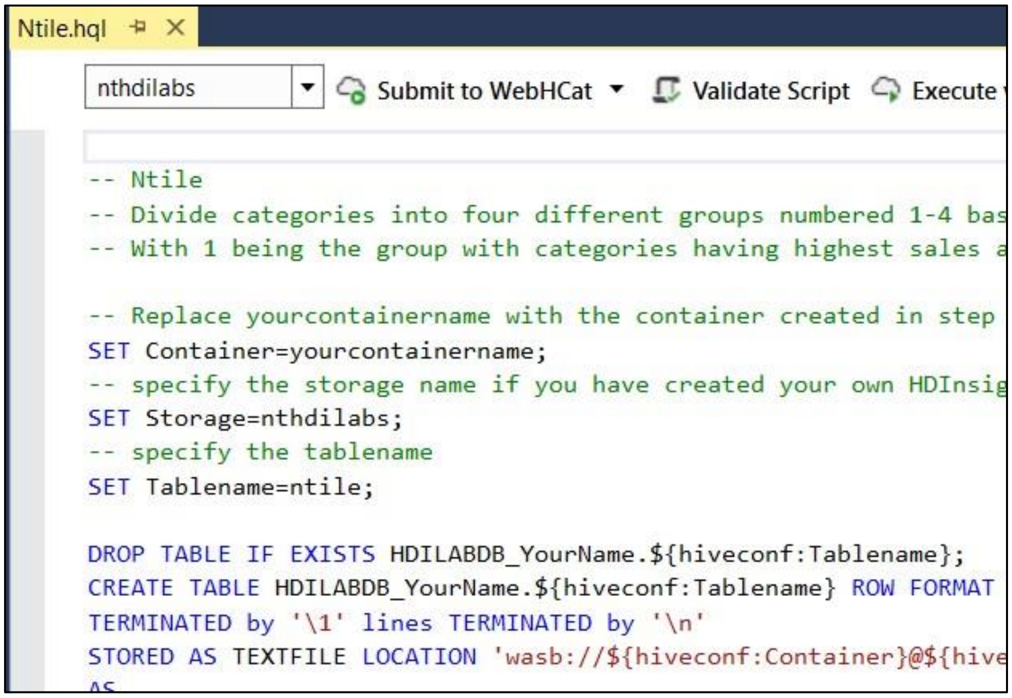
Solution

1. From the solution explorer, open **ntile.hql**

```
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
SET Tablename=ntile;

-- Ntile
-- Divide categories into four different groups numbered 1-4 based on total sales
made under each category.
-- With 1 being the group with categories having highest sales and 4 being the group
with categories having lowest sales
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS
```

```
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HD
ILABDB.${hiveconf:Tablename}/'
AS
SELECT  NTILE(4) OVER (ORDER BY SUM(PaymentAmount) DESC) AS Quartile ,
        CategoryName,
        SUM(PaymentAmount) AS TotalSales
FROM    HDILABDB.weblogs
WHERE   PurchaseType = "Purchased"
GROUP BY CategoryName ORDER BY Quartile;
```

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **Ntile.hql**



```
-- Ntile
-- Divide categories into four different groups numbered 1-4 bas
-- With 1 being the group with categories having highest sales a

-- Replace yourcontainername with the container created in step
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsig
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=ntile;

DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hive
AS
```

HiveQL Challenge

Write a query to return total sales for each month. The output should look like this,

month	sales
1	1966225
2	3937040
3	1979190
4	1964065
5	1927185

7	1921600
9	3960655
12	1968970

Save the result in a table, **MonthlySales**. The table should be created in a new folder **MonthlySales** in the container created in step “Create a new Azure Storage Container”.

Solution

1. From the solution explorer, open **MonthlySales.hql**

```
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename
SET Tablename=MonthlySales;

DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};

CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS  TERMINATED
by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HD
ILABDB.${hiveconf:Tablename}/' AS
SELECT  MONTH(transactiondate) AS month ,
        SUM(paymentamount) AS totalsales
FROM    HDILABDB.weblogs
WHERE   purchasetype = "Purchased"
GROUP BY MONTH(transactiondate)
ORDER BY month;
```

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **MonthlySales.hql**

Problem Statement

Write a C# user defined function to convert month number to their corresponding month names. Use this function to return, monthnumber, monthname and totalsales from the monthlysales table created earlier in HiveQL challenge. The output should look like this.

month	MonthName	sales
1	January	1966225
2	February	3937040
3	March	1979190
4	April	1964065
5	May	1927185

7	July	1921600
9	September	3960655
12	December	1968970

Save the result in a table, **MonthNameMonthlySales**. The table should be created in a new folder **MonthNameMonthlySales** in the container created in step "Create a new Azure Storage Container".

Solution

1. From the solution explorer, open **MonthNameMonthlySales.hql**.

```
SET hive.llap.execution.mode=all;
-- Replace yourcontainername with the container created in step n
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename
SET Tablename=MonthNameMonthlySales;

-- Using C# UDF
add file wasb:///Hiveapp.exe;

DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HD
ILABDB.${hiveconf:Tablename}/'
AS
SELECT TRANSFORM (month, totalsales)
USING 'Hiveapp.exe' AS
(month int, monthname string, sales string)
FROM HDILABDB.MonthlySales
ORDER BY monthnumber;
```

Note: The query uses Hiveapp.exe C# application to convert month numbers to month name and store the result in MonthNameMonthlySales *table. The Hiveapp.exe was uploaded to bookstore container in the section* **"Upload files to Azure Storage".** *The file is added to the hive query. The **TRANSFORM** statement specifies the column names (monthnumber and totalsales) or the parameters and output the monthnumber, monthname and the totalsales columns.*

Hiveapp.exe

The hiveapp.exe is a C# console application. Navigate to \LabAssets\Solutions\bookstoresalesanalaysis\bookstoresalesanalaysis\ and open the program.cs file in notepad.

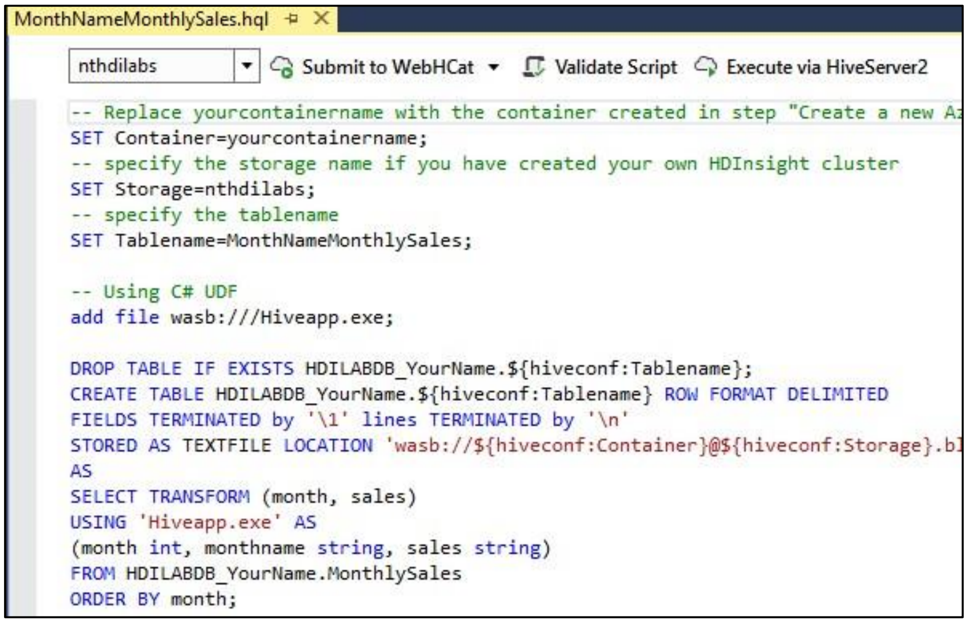
```
/// <summary>
        /// returns monthname for the specified monthnumber         /// </summary>
        /// <param name="monthnumber"></param>
        /// <returns></returns>
        static string monthnumbertoname(Int16 monthnumber)
        {
            System.Globalization.DateTimeFormatInfo mfi = new
System.Globalization.DateTimeFormatInfo();             return
mfi.GetMonthName(monthnumber).ToString();         }
```

Note: The function monthnumbertoname takes a monthnumber as parameter and returns the corresponding month name. For example, if the value of month number parameter is 1, January will be returned as the month name, if the month number value is 2, February will be returned as the month name and so on.

```
static void Main(string[] args)
        {
            System.Globalization.DateTimeFormatInfo mfi = new
System.Globalization.DateTimeFormatInfo();

            string line;
            // Read input in a loop
            while ((line = Console.ReadLine()) != null)
            {
                // Parse the string, trimming line feeds
                // and splitting fields at tabs
line = line.TrimEnd('\n');
                string[] column = line.Split('\t');
                Int16 monthnumber = Convert.ToInt16(column[0]);
string totalsales = column[1];
                Console.WriteLine("{0}\t{1}\t{2}", monthnumber,
monthnumbertoname(monthnumber), totalsales);
            }
        }
```

*Note: The Main function is the entry point of a console application. The **TRANSFORM** function discussed above, passes the value to the hiveapp.exe row by row with column delimited by tab. The console application, reads a line, splits the tab delimited column into a string array, gets the monthname from monthnumbertoname function and outputs a tab delimited column list. The result is consumed by hive.*

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **MonthNameMonthlySales.hql**



```
-- Replace yourcontainername with the container created in step "Create a new Az
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=MonthNameMonthlySales;

-- Using C# UDF
add file wasb:///Hiveapp.exe;

DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT DELIMITED
FIELDS TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hiveconf:Storage}.bl
AS
SELECT TRANSFORM (month, sales)
USING 'Hiveapp.exe' AS
(month int, monthname string, sales string)
FROM HDILABDB_YourName.MonthlySales
ORDER BY month;
```

Problem Statement

Write a query to return the previous and next month sales from the MonthNameMonthlySales (created earlier in the lab) table, using LAG and LEAD function. The output should look like this

MonthNumber	MonthName	TotalSales	PreviousSales	NextSales
1	January		NULL	3937674.82
2	February		1966523.37	1979494.31
3	March		3937674.82	1964377.65
4	April		1979494.31	1927477.37
5	May	1966523.37	1964377.65	1921908.06
7	July	3937674.82 1979494.31 1964377.65 1927477.37 1921908.06 3961270.23 1969280.85	1927477.37	3961270.23
9	September		1921908.06	1969280.85
12	December		3961270.23	NULL

Note: Observe, that the first value of PreviousSales column is null. This is because there isn't any previous value as it's the first record of the result set. Similarly, the last value of the NextSales column is null as it's the last record and there isn't any record after it

Save the result in a table, **LagLead**. The table should be created in a new folder **LagLead** in the container created in step "Create a new Azure Storage Container".

Solution

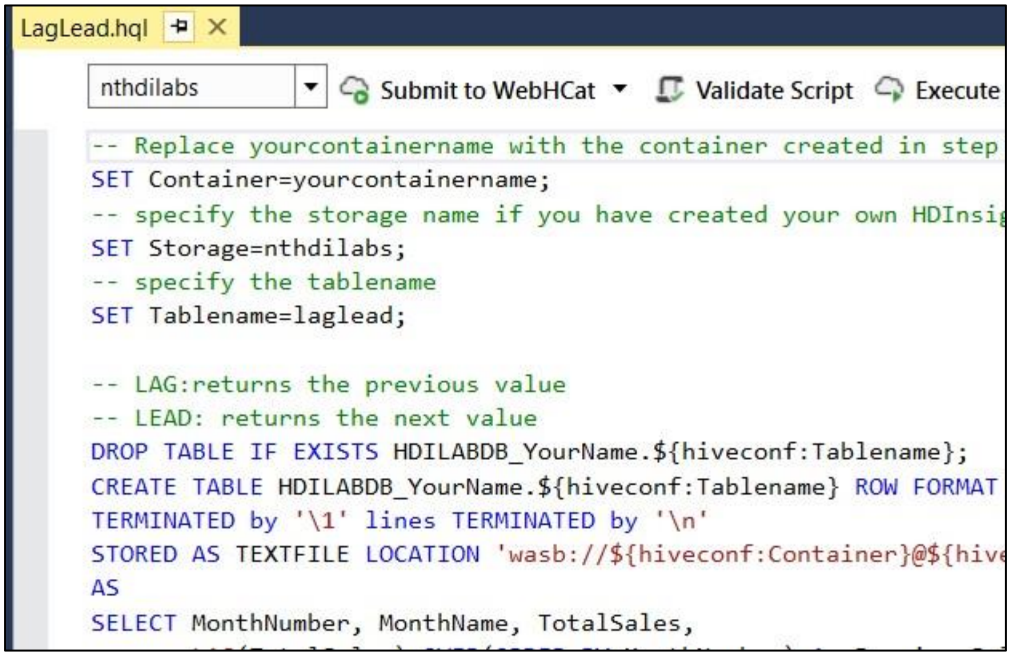
1. From the solution explorer, open **LagLead.hql**.

```
-- Replace yourcontainername with the container created in step n
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename
SET Tablename=laglead;

-- LAG:returns the previous value
-- LEAD: returns the next value
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HD
ILABDB.${hiveconf:Tablename}/'
AS
SELECT Month, MonthName, Sales,
       LAG(Sales) OVER(ORDER BY Month) As PreviousSales,
       lead(Sales) OVER(ORDER BY Month) As NextSale
FROM HDILABDB.MonthNameMonthlySales ORDER BY Month;
```

*Note: The Query 1 uses **LAG** and **LEAD** function to return the previous and next total sales value from the*

MonthNameMonthlySales *table respectively. The result from the query 1 is saved in laglead table. The PreviousSales value for the month of february is TotalSales value for the month of January (LAG). The NextSales value for the month of February is TotalSales value for the month of March (LEAD)..*

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **LagLead.hql**



```
-- Replace yourcontainername with the container created in step
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsi
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=laglead;

-- LAG:returns the previous value
-- LEAD: returns the next value
DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hive
AS
SELECT MonthNumber, MonthName, TotalSales,
```

Problem Statement

Use LAG and LEAD function to calculate the difference between the previous and next month sales for each month, in the table MonthNameMonthlySales. The output should look like this

monthnumber	monthname	totalsales	PreviousSales	salesdifference
1	January	1966523.37	NULL	NULL
2	February		1966523.37	1971151.45
3	March		3937674.82	-1958180.51
4	April	3937674.82	1979494.31	-15116.66
5	May	1979494.31	1964377.65	-36900.28
		1964377.65		
		1927477.37		
		1921908.06		
		3961270.23		
7	July		1927477.37	-5569.31
9	September		1921908.06	2039362.17
12	December	1969280.85	3961270.23	-1991989.38

Save the result in a table, **DiffMonthlySales**. The table should be created in a new folder **DiffMonthlySales** in the container created in step “Create a new Azure Storage Container”.

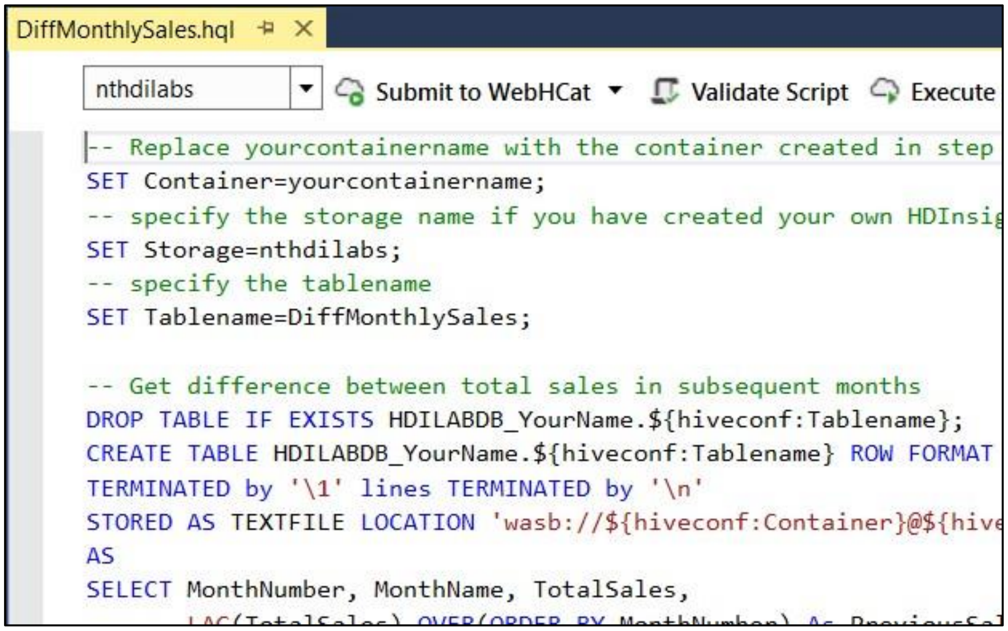
Solution

1. From the solution explorer, open **DiffMonthlySales.hql**.

```
-- Replace yourcontainername with the container created in step n
SET Container=hdi;
-- specify the storage name if you have created your own HDInsight cluster
SET Storage=mtllab<1-12>;
-- specify the tablename SET Tablename=DiffMonthlySales;

-- Get difference between total sales in subsequent months
DROP TABLE IF EXISTS HDILABDB.${hiveconf:Tablename};
CREATE TABLE HDILABDB.${hiveconf:Tablename} ROW FORMAT DELIMITED FIELDS  TERMINATED by '\1
'\n'
STORED AS TEXTFILE LOCATION
'wasb://${hiveconf:Container}@${hiveconf:Storage}.blob.core.windows.net/bookstore/HDILABDB
AS
SELECT Month, MonthName, Sales,
         LAG(Sales) OVER(ORDER BY Month) As PreviousSales,
       Sales - LAG(Sales) OVER(ORDER BY Month) As SalesDifference
FROM HDILABDB.MonthNameMonthlySales
ORDER BY Month;
```

Note: The PreviousSales value for the month of February is TotalSales value for the month of January (LAG). The NextSales value for the month of February is TotalSales value for the month of March (LEAD). The sales difference for the month of February is the ***(totalsales – PreviousSales (January's totalsales))*** *and so on for the rest of the months.*

2. Select **MTLHDillap<1-12>** in the server drop down and select **Interactive** mode to execute the **DiffMonthlySales.hql**



```
-- Replace yourcontainername with the container created in step
SET Container=yourcontainername;
-- specify the storage name if you have created your own HDInsig
SET Storage=nthdilabs;
-- specify the tablename
SET Tablename=DiffMonthlySales;

-- Get difference between total sales in subsequent months
DROP TABLE IF EXISTS HDILABDB_YourName.${hiveconf:Tablename};
CREATE TABLE HDILABDB_YourName.${hiveconf:Tablename} ROW FORMAT
TERMINATED by '\1' lines TERMINATED by '\n'
STORED AS TEXTFILE LOCATION 'wasb://${hiveconf:Container}@${hive
AS
SELECT MonthNumber, MonthName, TotalSales,
```

Reviewing the Hive Table Data

In this exercise, you will review the Hive table data. This will involve creating an ODBC data source to connect to the Hadoop cluster, and then using Power BI Desktop to retrieve the table data.

Creating an ODBC Data Source

In this task, you will create an ODBC data source to connect to the Hadoop cluster.

*You must create a data source that matches the architecture (x86 or x64) of your Windows operating system. If you are working with the pre-configured lab virtual machine, you should use the 64-bit application. If you are working on your own machine, take care to open the appropriate **ODBC Data Sources** application.*

1. Open **ODBC Data Sources (64-bit)**.

2. In the **ODBC Data Source Administrator** window, click **Add**.

3. In the **Create New Data Source** window, select **Microsoft Hive ODBC Driver**, then click **Finish**.

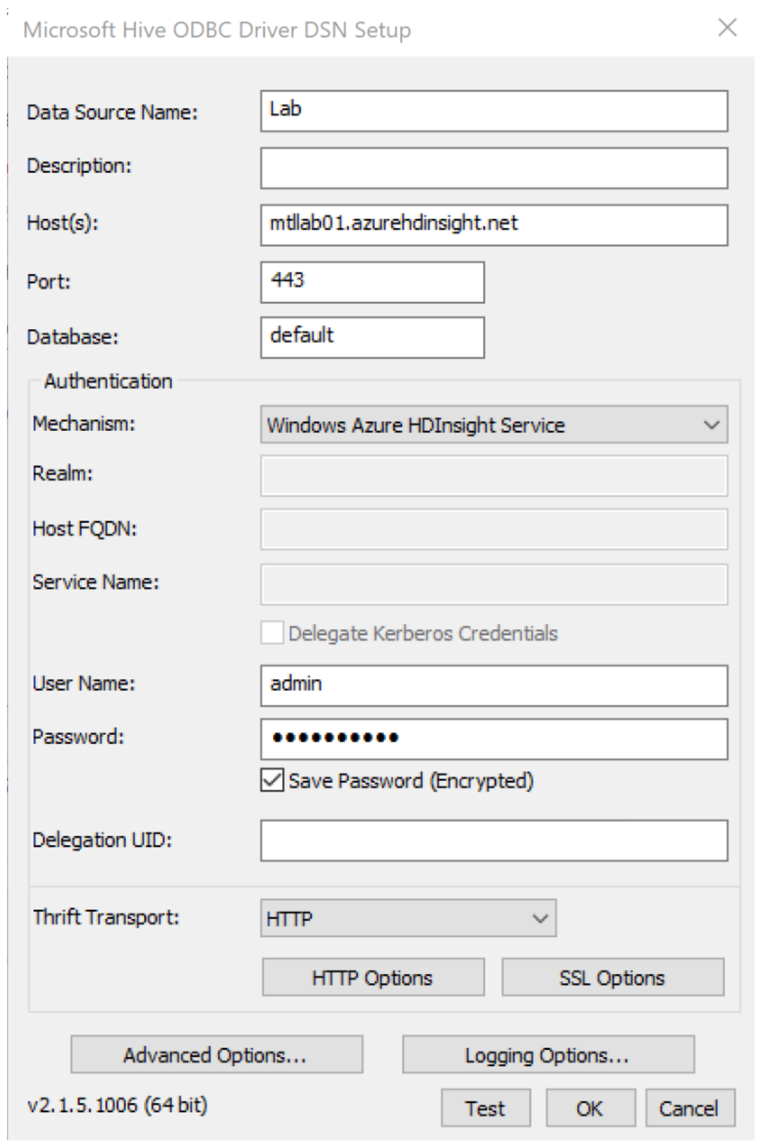
   The Hive ODBC Driver was installed with the Azure SDK or can be downloaded and installed separately from https://www.microsoft.com/en-in/download/details.aspx?id=40886&751be11f-ede8-5a0c-058c-2ee190a24fa6=True&e6b34bbe-475b-1abd-2c51-b5034bcdd6d2=True

4. In the **Microsoft Hive ODBC Driver DSN Setup** window, inside the **Data Source Name** box, enter **Lab**.

5. Inside the **Host** box, enter your server name, followed by **azurehdinsight.net** (e.g. **yourservername.azurehdinsight.net**).

6. Enter the cluster credentials:

   - User Name: **admin**
   - Password: ***HDItut@123***


Microsoft Hive ODBC Driver DSN Setup

Data Source Name:	Lab
Description:	
Host(s):	mtllab01.azurehdinsight.net
Port:	443
Database:	default

Authentication

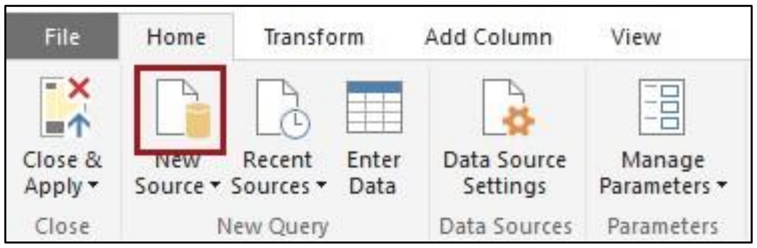
Mechanism:	Windows Azure HDInsight Service
Realm:	
Host FQDN:	
Service Name:	
	[ ] Delegate Kerberos Credentials
User Name:	admin
Password:	●●●●●●●●●●●
	[✓] Save Password (Encrypted)
Delegation UID:	
Thrift Transport:	HTTP

1. Verify that you have entered the following.
2. Click **Test**.
3. When the test succeeds, click **OK**.
4. In the **Microsoft Hive ODBC Driver DSN Setup** window, click **OK**.
5. In the **ODBC Data Source Administrator** window, click **OK**.

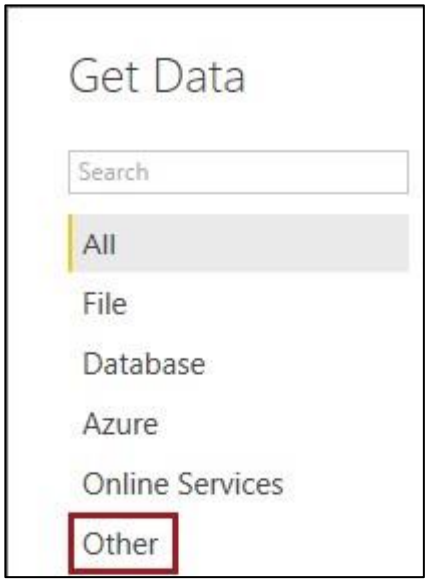
Reviewing the Hive Table Data

In this task, you will use Power BI Desktop to review the Hive table data.

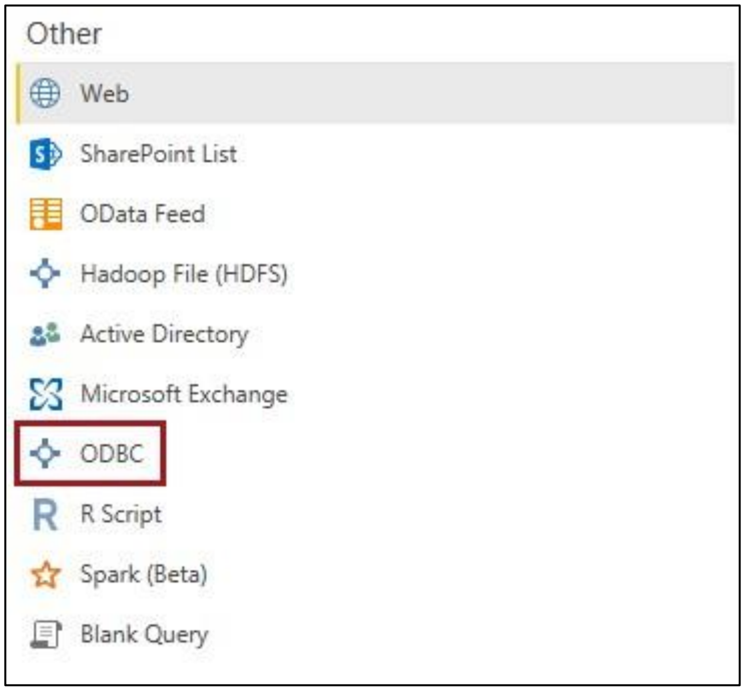
1. In Power BI Desktop, in the **Query Editor** window, on the **Home** ribbon, from inside the **New Query** group, click the **New Source** icon.



2. In the **Get Data** dialog window, on the left side, select **Other**.



3. In the data source list, select **ODBC**.



4. Click **Connect**.

5. In the **From ODBC** window, in the **Data Source Name (DSN)** dropdown list, ensure that **Lab** is selected.

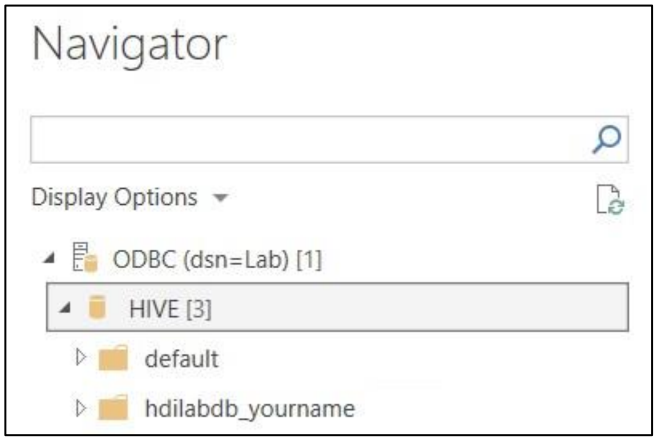
6. Click **OK**.

7. Re-enter the cluster credentials:

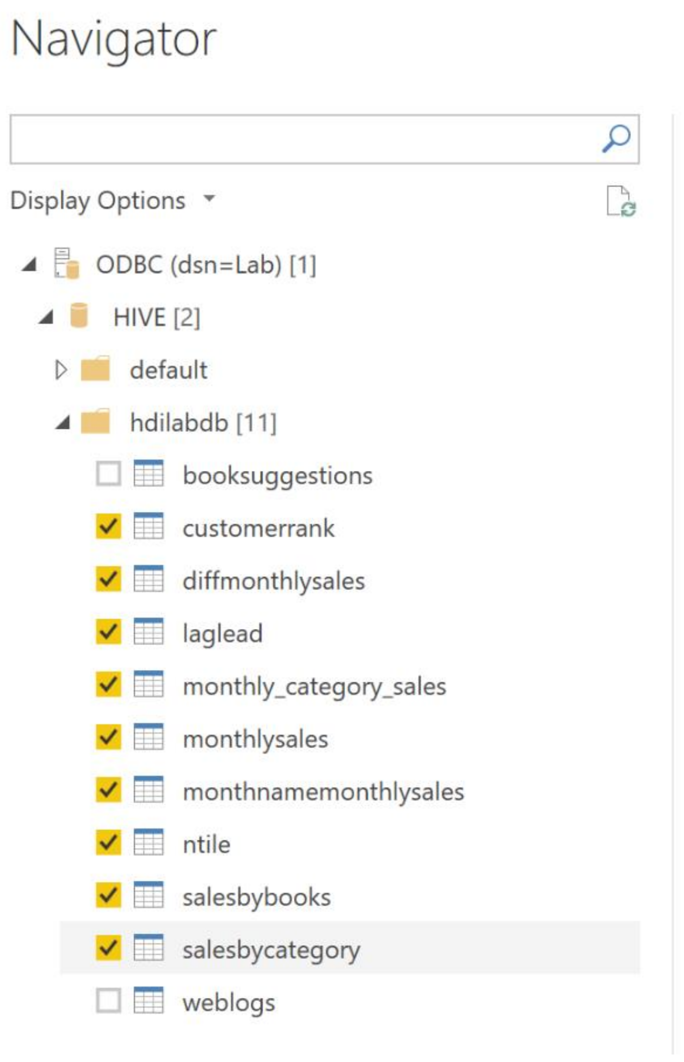
   - User Name: **admin**
   - Password: **HDItut@123**

8. Click **Connect**.

9. In the **Navigator** dialog window, expand the **HIVE** database, and then expand **hdilabdb**.



10. Check the following ten tables.



Navigator

Display Options

- ODBC (dsn=Lab) [1]
  - HIVE [2]
    - default
    - hdilabdb [11]
      - ☐ booksuggestions
      - ✓ customerrank
      - ✓ diffmonthlysales
      - ✓ laglead
      - ✓ monthly_category_sales
      - ✓ monthlysales
      - ✓ monthnamemonthlysales
      - ✓ ntile
      - ✓ salesbybooks
      - ✓ salesbycategory
      - ☐ weblogs

11. Click **Load**.

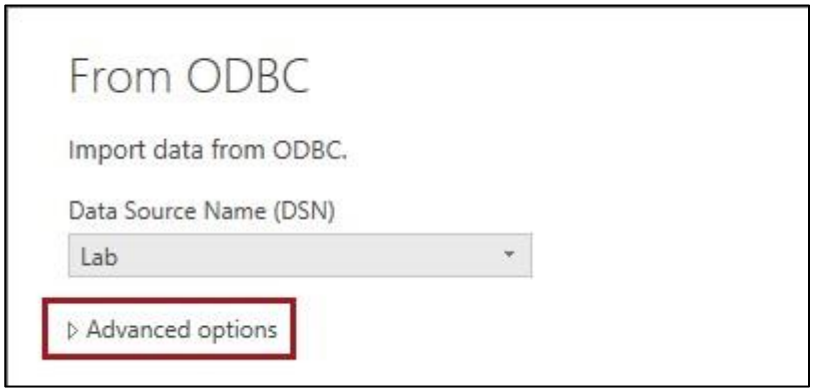
    The data will be loaded into the Power BI Desktop file.

12. Once loaded, in the **Queries** pane (located at the left), select each of the queries to review the data from each Hive table.

Creating a New Query

In this task, you will create a new Power BI Desktop query based on a Hive query.

1. On the **Home** ribbon, create another ODBC source.
2. In the **From ODBC** window, in the **Data Source Name (DSN)** dropdown list, ensure that **Lab** is selected.
3. Expand **Advanced Options**.


From ODBC

Import data from ODBC.

Data Source Name (DSN)

Lab

4. In the **SQL Statement** box, enter the following query.

   *For convenience, the query can be copied from the **\LabAssets\Scripts\6_PowerBIQuery.txt** file.*

   HiveQL

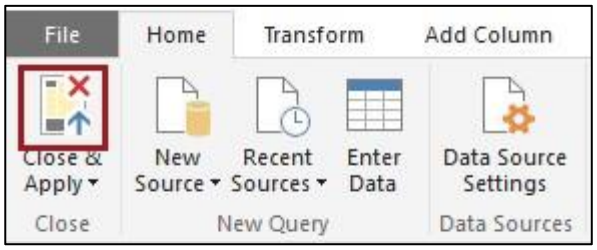
```
SELECT
      BookName AS Book,
      SUM(PaymentAmount) AS Sales
FROM
      HDILABDB.weblogs
WHERE
      PurchaseType = "Purchased"
GROUP BY
      BookName
ORDER BY
      Sales DESC
LIMIT 10;
```

5. Click **OK**.

6. When the query result has been imported, click **EDIT**.

7. In the **Query Settings** pane (located at the right), in the **Name** box, replace the text with **top_10_books**.

8. On **Home** ribbon, click the **Close & Apply** icon.



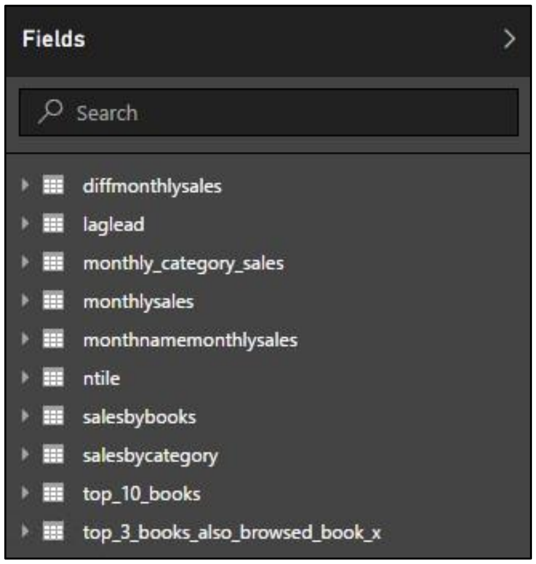
Creating a Power BI Report

In this exercise, you will design an interactive report based on the queries.

Designing the Report

In this task, you will design a report layout.

1. In the **Fields** pane (located at the right), notice the eight tables sourced from Hive queries.



Fields

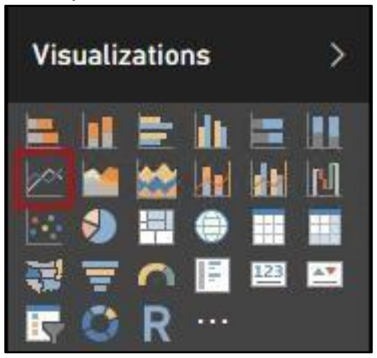
Search

- diffmonthlysales
- laglead
- monthly_category_sales
- monthlysales
- monthnamemonthlysales
- ntile
- salesbybooks
- salesbycategory
- top_10_books
- top_3_books_also_browsed_book_x

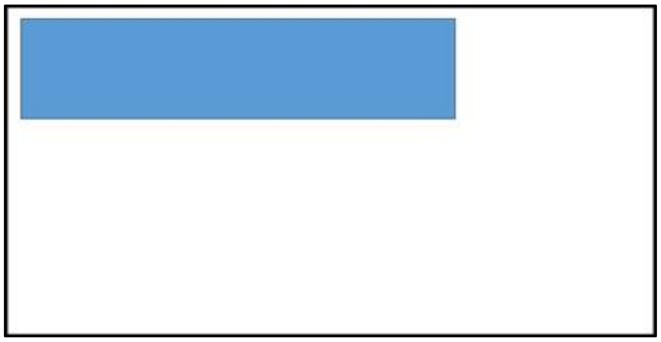
2. Expand the **monthnamemonthlysales** table.

3. To add a line chart, from inside the **Visualizations** pane, click the line chart icon.

   Tip: You can hover the cursor over each icon to reveal a tooltip describing the type of visualization.



4. Reposition and resize the visualization based on the following diagram.

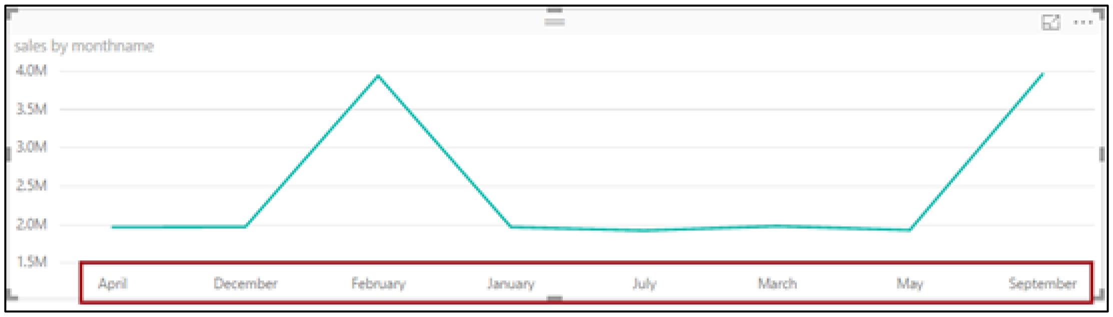


5. From the **Fields** pane, from inside the expanded table, drag the **monthname** field and drop it inside the line chart visualization.

   Recall that this table was sourced from the Hive query that used the UDF to retrieve the full month name.

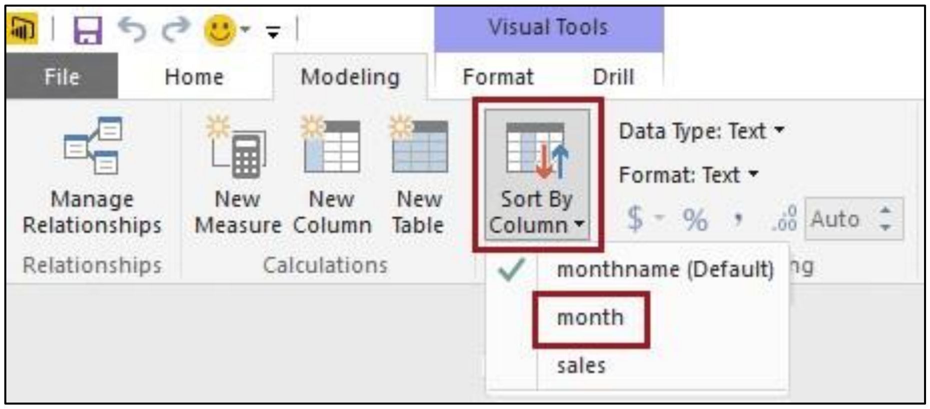
6. Drag also the **sales** field into the visualization.

7. Notice that the chart axis displays months sorted alphabetically.



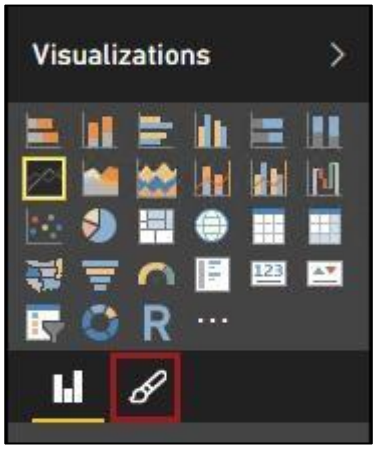
8. To sort the month chronologically, in the field list, select the **monthname** field.

9. On the **Modeling** ribbon, click **Sort by Column**, and then select **month**.



10. Notice that the line chart now presents months sorted in the correct order.

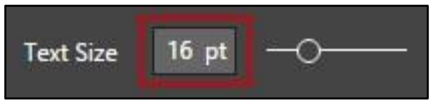
11. To update the visualization title, in the **Visualizations** pane, click the **Format** icon.



12. Expand the **Title** group.

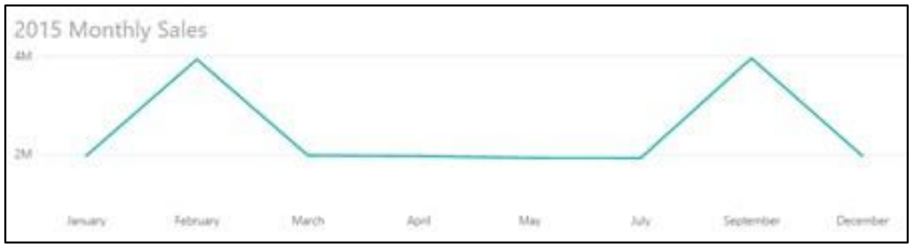
13. In the **Title Text** box, replace the text with **2015 Monthly Sales**.

14. Increase the **Text Size** to **16pt**.

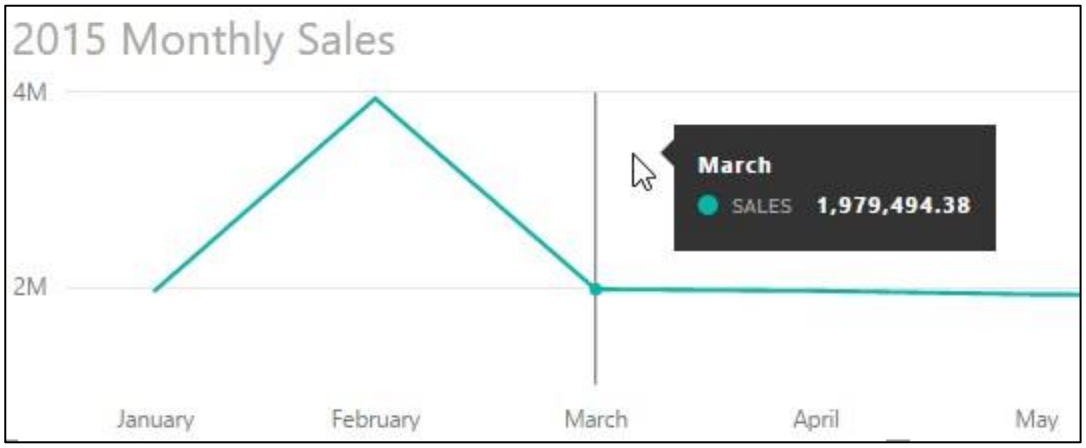


Text Size 16 pt

15. Verify that the visualization looks like the following.



16. Hover the cursor over the line to reveal data point values.

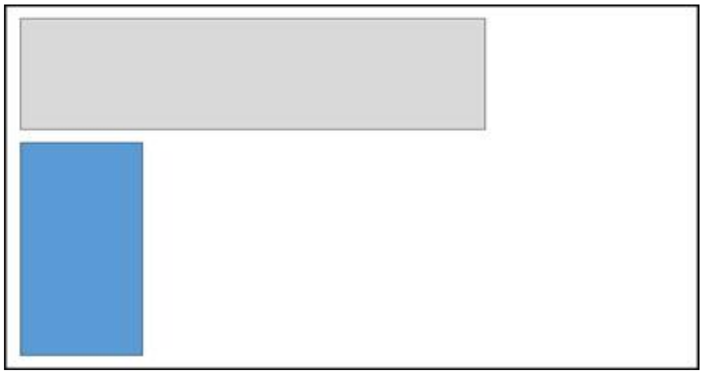


17. To create a new visualization, in the **Fields** pane, expand the **monthly_category_sales** table.

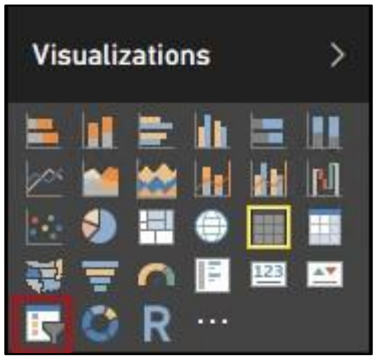
18. From inside the **monthly_category_sales** table, drag the **category** field, and drop it directly beneath the line chart visualization.

    By default, dragging a text field to the report canvas will create a table visualization.

19. Reposition and resize the visualization based on the following diagram.

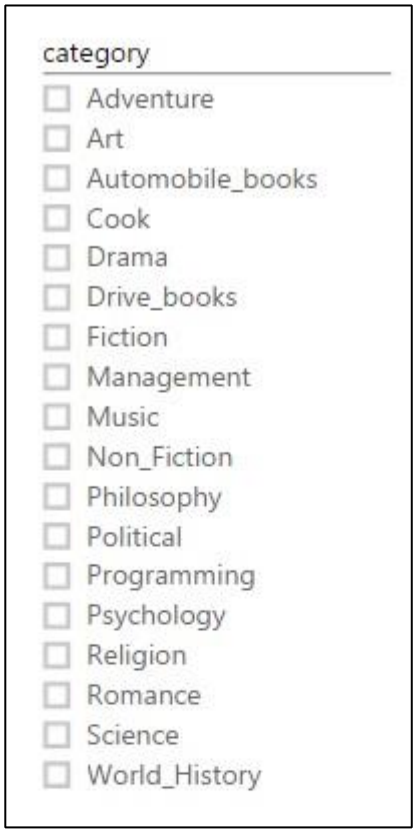


20. To switch the table visualization to a slicer, in the **Visualizations** pane, click the slicer icon.



Slicers enable interactive filtering of the report.

21. Verify that the visualization looks like the following.



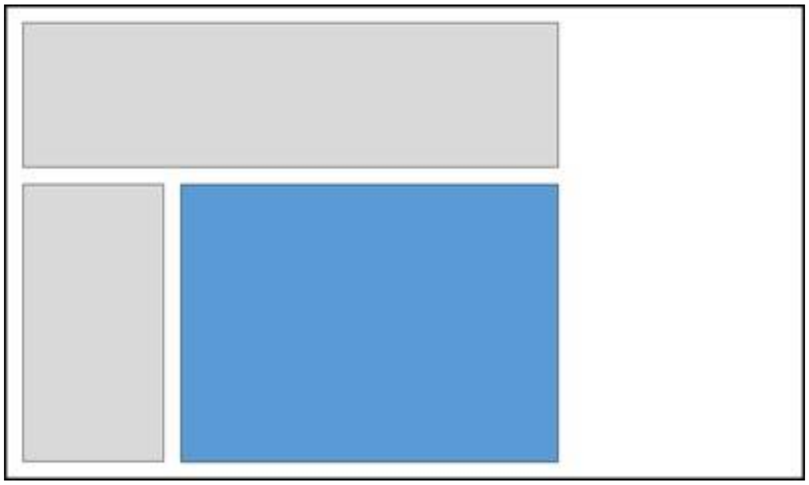
category

- Adventure
- Art
- Automobile_books
- Cook
- Drama
- Drive_books
- Fiction
- Management
- Music
- Non_Fiction
- Philosophy
- Political
- Programming
- Psychology
- Religion
- Romance
- Science
- World_History

22. To create a new visualization, in the **Fields** list, from inside the **monthly_category_sales** table, drag the **sales** field, and drop it directly to the left of the slicer.

By default, dragging a numeric field to the report canvas will create a column chart visualization.

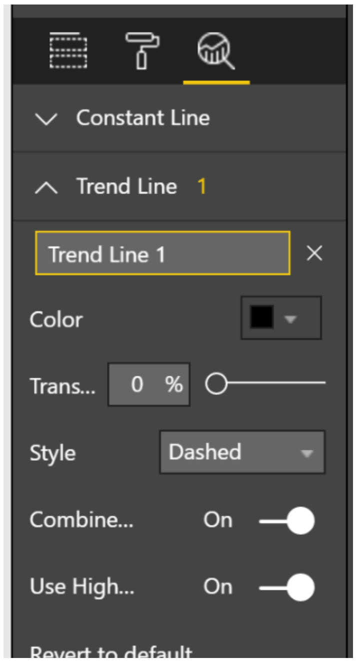
23. Reposition and resize the visualization based on the following diagram.



24. From the **Fields** list, from inside the **monthly_category_sales** table, drag the **monthnumber** field, and drop it directly into the column chart visualization.

25. Modify the visualization title to **Monthly Category Sales**, and then set the text size to **16pt**.

26. Add a **Trend Line**.



Constant Line

Trend Line 1

Trend Line 1

Color

Trans... 0 %

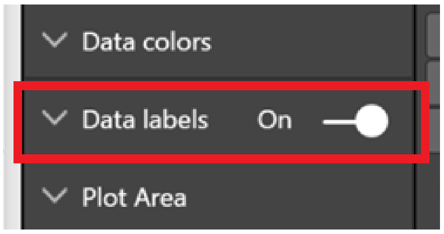
Style Dashed

Combine... On

Use High... On

Revert to default

27. Set **Data Labels** to **On.**

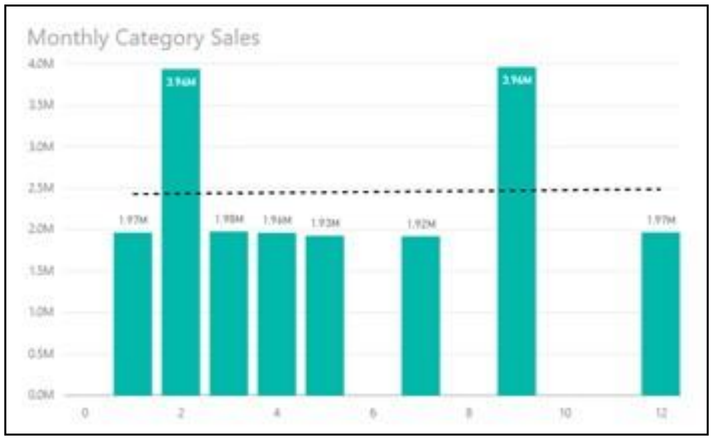


Data colors

Data labels On

Plot Area

27. Verify that the visualization looks like the following.



28. In the slicer, select any category, and notice that the column chart filters by the selection.

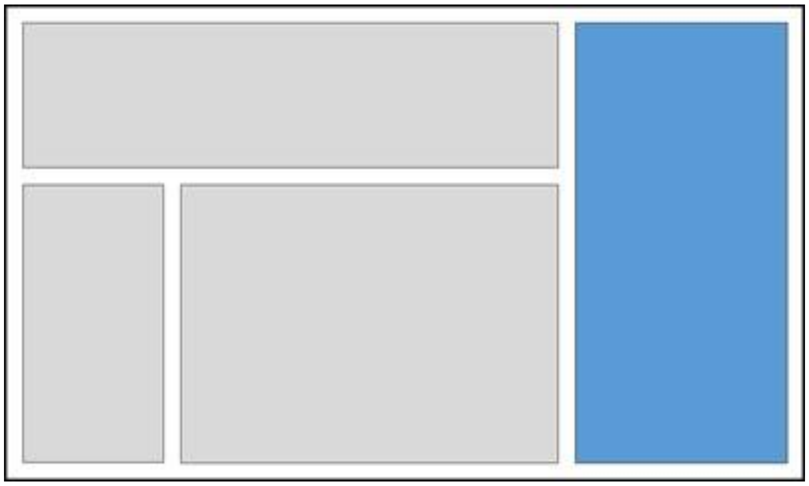
29. To create a new visualization, in the **Fields** pane, expand the **top_10_books** table.

    Recall that this table was sourced from the Hive query that you created in Power BI Desktop.

30. From inside the **top_10_books** table, drag the **book** field, and drop it in the right area of the report.

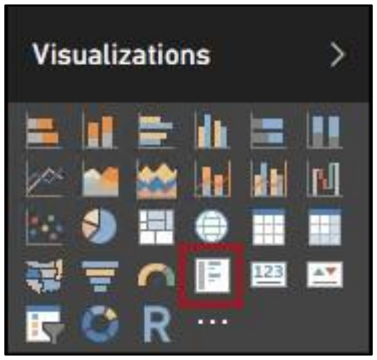
    By default, dragging a text field to the report canvas will create a table visualization.

31. Reposition and resize the visualization based on the following diagram.

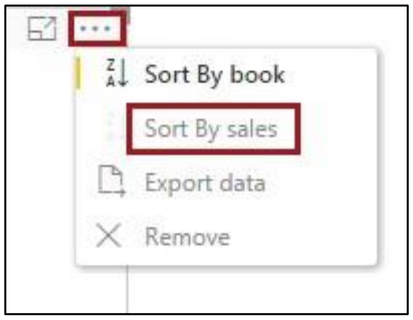


32. Drag also the **sales** field into the visualization.

33. Switch the table visualization to a mulit-row card.



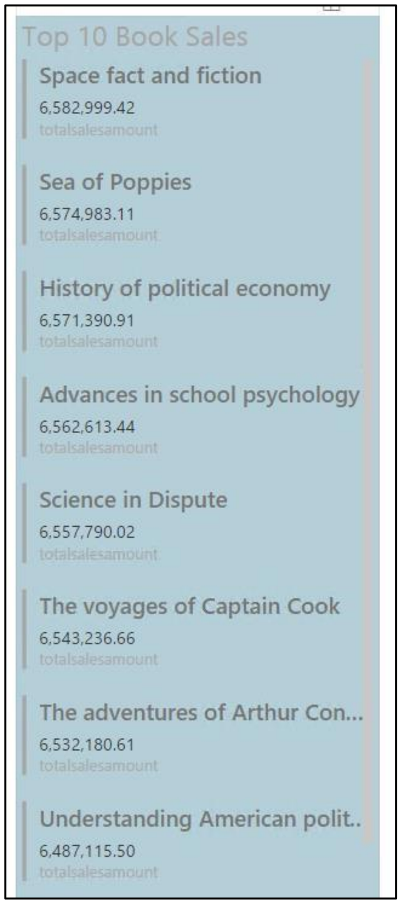
34. To sort the list by sales, at the top right corner, click the ellipsis, and then select **Sort by Sales**.



35. Set the visualization title to **Top 10 Book Sales** (you will need to turn titles on), and then set the text size to **16pt**.

36. Set the visualization background color to any color.

37. Verify that the visualization looks like the following.



Top 10 Book Sales

Space fact and fiction
6,582,999.42
totalsalesamount

Sea of Poppies
6,574,983.11
totalsalesamount

History of political economy
6,571,390.91
totalsalesamount

Advances in school psychology
6,562,613.44
totalsalesamount

Science in Dispute
6,557,790.02
totalsalesamount

The voyages of Captain Cook
6,543,236.66
totalsalesamount

The adventures of Arthur Con...
6,532,180.61
totalsalesamount

Understanding American polit..
6,487,115.50
totalsalesamount

Disclaimer: Once you have completed the lab, to reduce costs associated with your Azure subscription, you may want to delete your clusters.

Note, delete the Azure resource group **MTLLAB**