Exercise - implement workload management

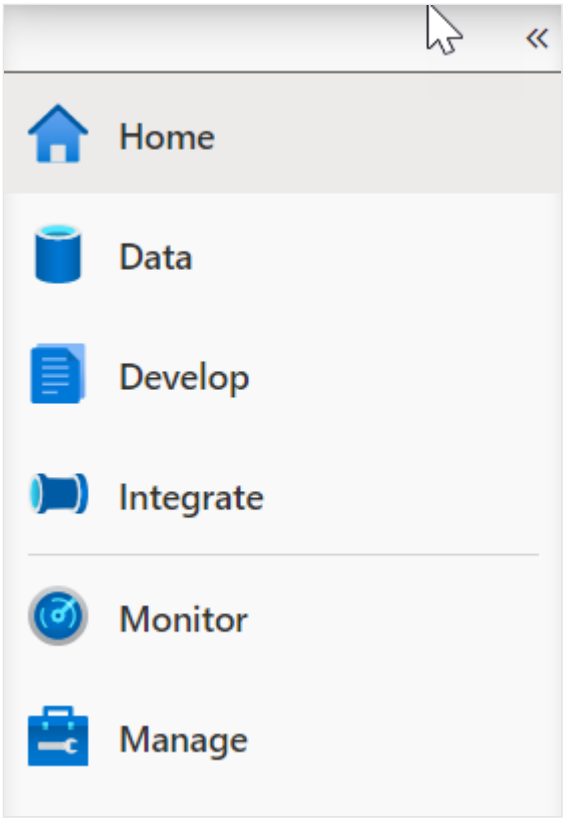
10 minutes

Create a workload classifier to add importance to certain queries

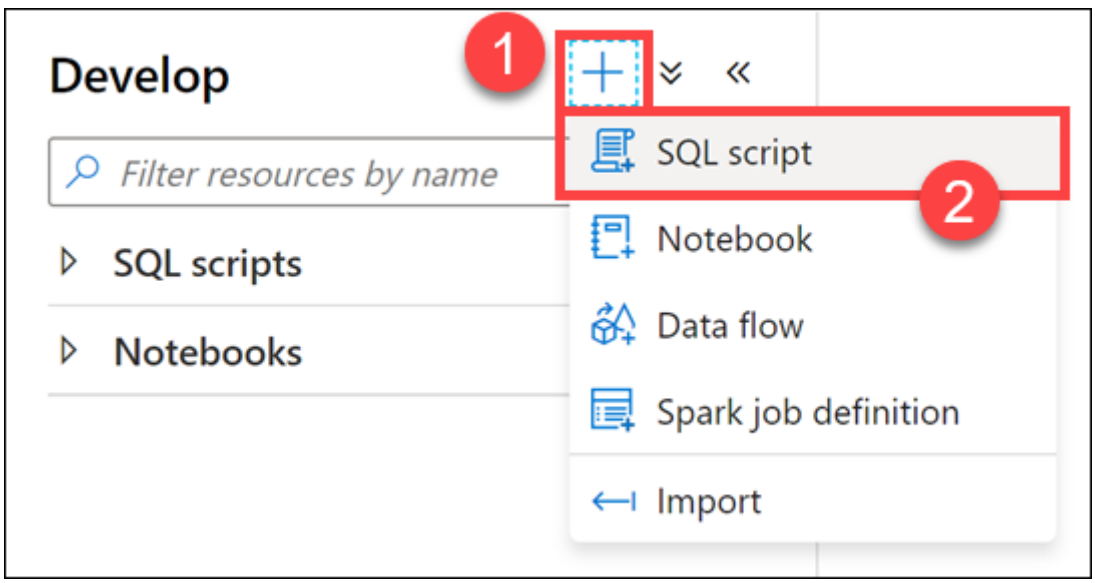
Your organization has asked you if there is a way to mark queries executed by the CEO as more important than others, so they don't appear slow due to heavy data loading or other workloads in the queue. You decide to create a workload classifier and add importance to prioritize the CEO's queries. Keep in mind that workload groups correspond to resources classes which are created automatically for backward compatibility and cannot be dropped.

> **Note**
>
> Eight (8) additional user-defined workload groups can be created in addition to the system defined workload groups and should be used sparingly and not take excessive amount of resources.

1. Select the **Develop** hub.



2. From the **Develop** menu, select the + button **(1)** and choose **SQL Script (2)** from the context menu.



3. In the toolbar menu, connect to the **SQL Pool** database to execute the query.

4. In the query window, replace the script with the following to confirm that there are no queries currently being run by users logged in as `asa.sql.workload01`, representing the CEO of the organization or `asa.sql.workload02` representing the data analyst working on the project:

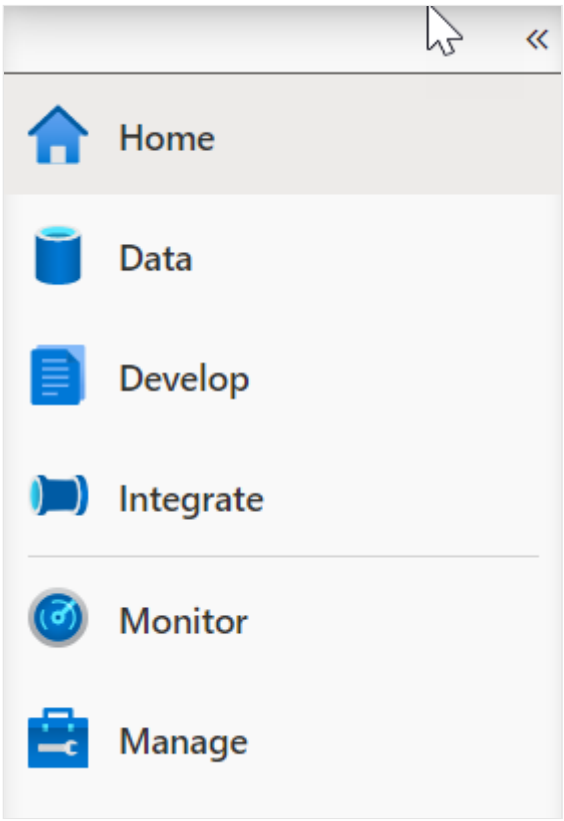
```
SQL
```

```
--First, let's confirm that there are no queries currently being run
by users logged in workload01 or workload02

SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and
Importance
is not NULL AND r.[status] in ('Running','Suspended')
--and submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,s.login_name
```

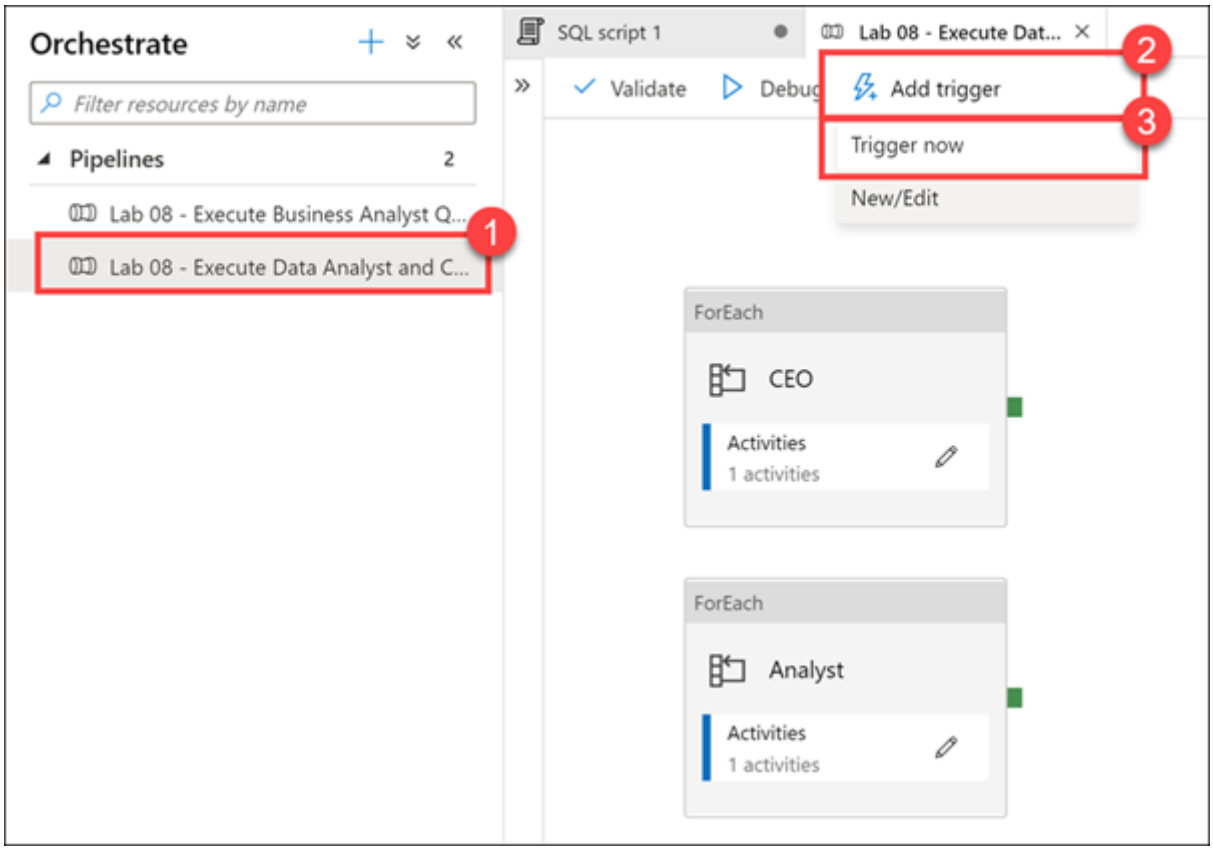
5. Select **Run** from the toolbar menu to execute the SQL command.

Now that we have confirmed that there are no running queries, we need to flood the system with queries and see what happens for `asa.sql.workload01` and `asa.sql.workload02`. To do this, we'll run an Azure Synapse Pipeline which triggers queries.

6. Select the **Integrate** hub.



7. Select the **Lab 08 - Execute Data Analyst and CEO Queries** Pipeline **(1)**, which will run / trigger the `asa.sql.workload01` and `asa.sql.workload02` queries. Select **Add trigger (2)**, then **Trigger now (3)**. In the dialog that appears, select **OK**.

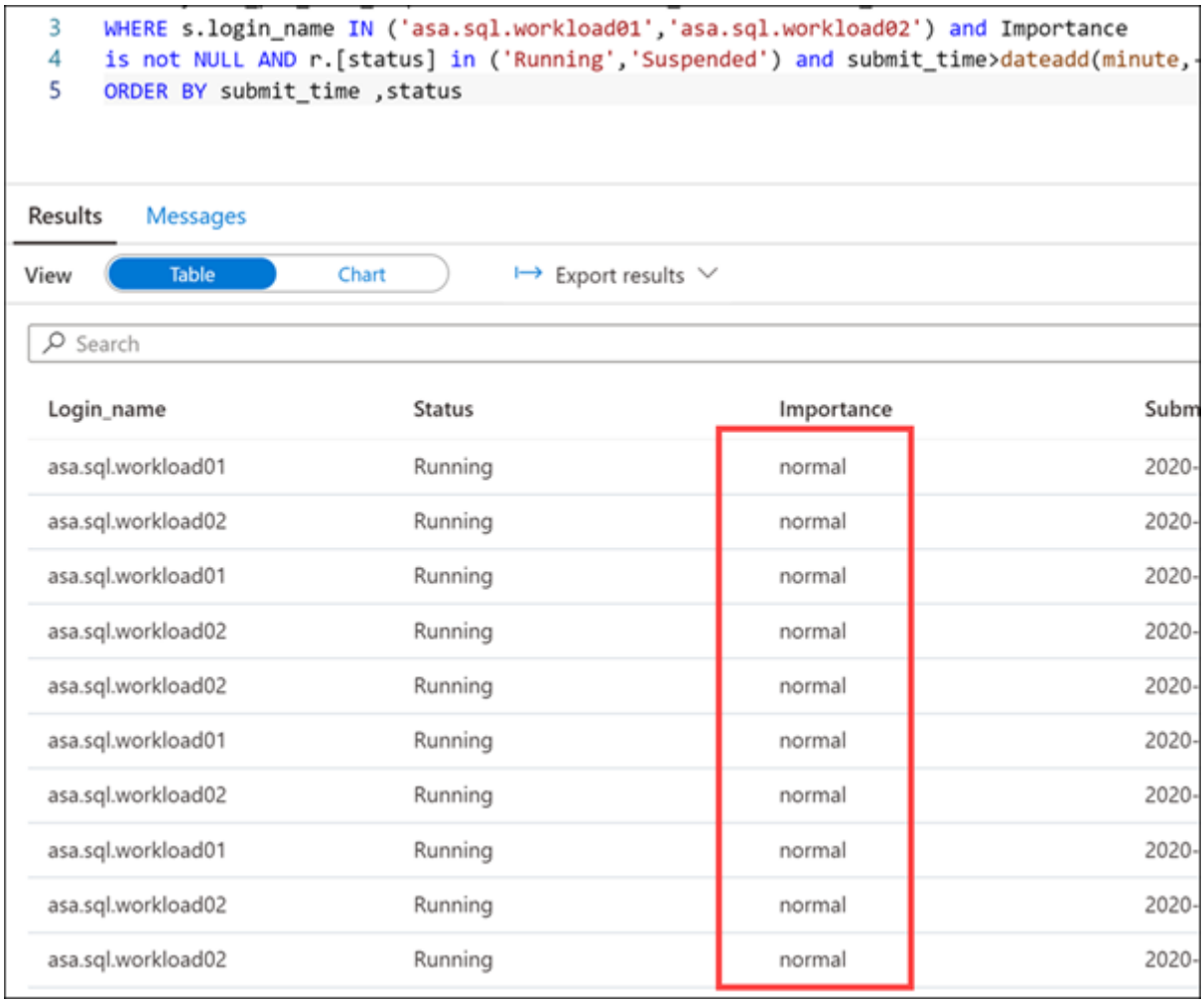


8. Let's see what happened to all the queries we just triggered as they flood the system. In the query window, replace the script with the following:

```
SQL

SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time
,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and
Importance
is not NULL AND r.[status] in ('Running','Suspended') and
submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status
```

9. Select **Run** from the toolbar menu to execute the SQL command.

You should see an output similar to the following:



Notice that the **Importance** level for all queries is set to **normal**.

10. We will give our `asa.sql.workload01` user queries priority by implementing the **Workload Importance** feature. In the query window, replace the script with the following:

```
SQL

IF EXISTS (SELECT * FROM sys.workload_management_workload_classifiers
WHERE name = 'CEO')
BEGIN
    DROP WORKLOAD CLASSIFIER CEO;
END
CREATE WORKLOAD CLASSIFIER CEO
  WITH (WORKLOAD_GROUP = 'largerc'
  ,MEMBERNAME = 'asa.sql.workload01',IMPORTANCE = High);
```

We are executing this script to create a new **Workload Classifier** named `CEO` that uses the `largerc` Workload Group and sets the **Importance** level of the queries to **High**.

11. Select **Run** from the toolbar menu to execute the SQL command.

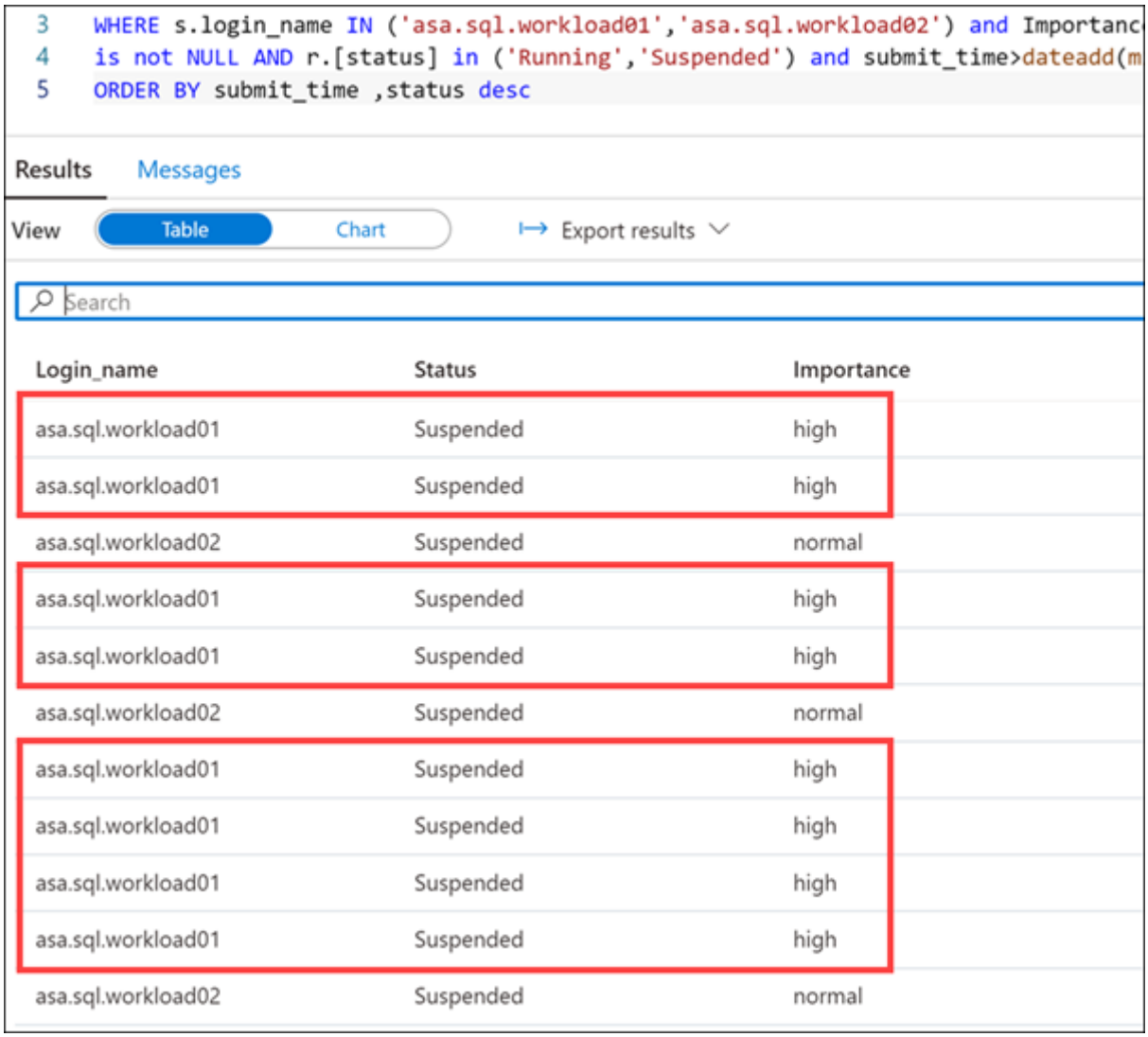
12. Let's flood the system again with queries and see what happens this time for `asa.sql.workload01` and `asa.sql.workload02` queries. To do this, we'll run an Azure Synapse Pipeline which triggers queries. **Select** the `Integrate` Tab, **run** the **Lab 08 - Execute Data Analyst and CEO Queries** Pipeline, which will run / trigger the `asa.sql.workload01` and `asa.sql.workload02` queries.

13. In the query window, replace the script with the following to see what happens to the `asa.sql.workload01` queries this time:

```
SQL

SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time
,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and
Importance
is not NULL AND r.[status] in ('Running','Suspended') and
submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status desc
```

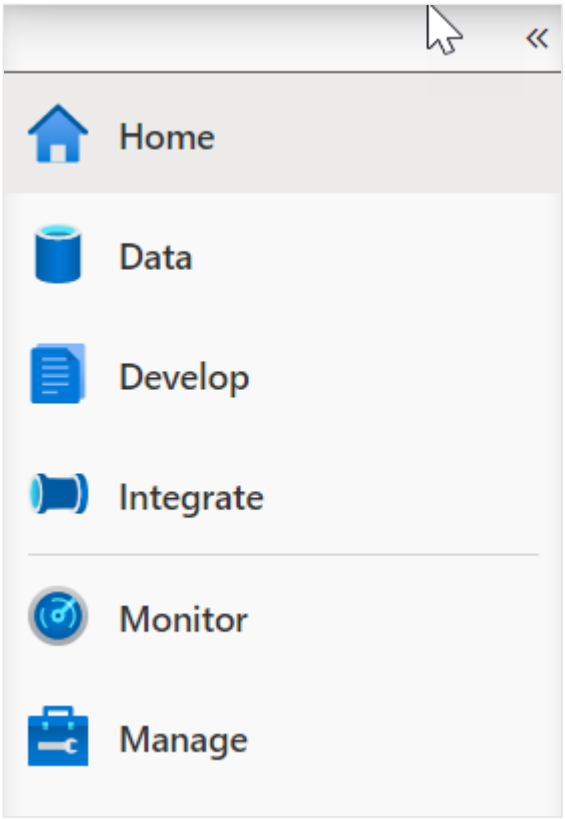
14. Select **Run** from the toolbar menu to execute the SQL command.

You should see an output similar to the following:

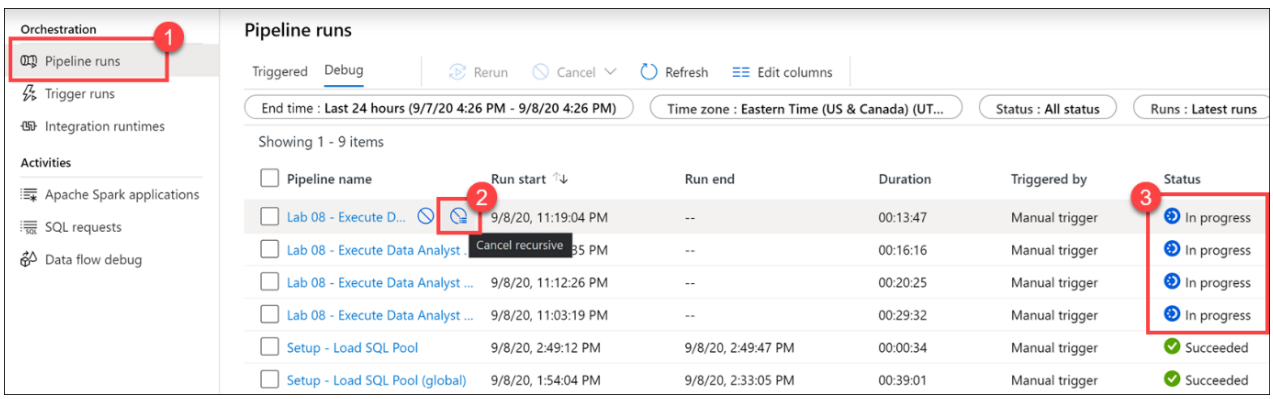


Notice that the queries executed by the `asa.sql.workload01` user have a **high** importance.

15. Select the **Monitor** hub.



16. Select **Pipeline runs (1)**, and then select **Cancel recursive (2)** for each running Lab 08 pipelines, marked **In progress (3)**. This will help speed up the remaining tasks.



Reserve resources for specific workloads through workload isolation

Workload isolation means resources are reserved, exclusively, for a workload group. Workload groups are containers for a set of requests and are the basis for how workload management, including workload isolation, is configured on a system. A simple workload management configuration can manage data loads and user queries.

In the absence of workload isolation, requests operate in the shared pool of resources. Access to resources in the shared pool is not guaranteed and is assigned on an importance basis.

Given the workload requirements provided by Tailwind Traders, you decide to create a new workload group called `CEODemo` to reserve resources for queries executed by the CEO.

Let's start by experimenting with different parameters.

1. In the query window, replace the script with the following:

```
SQL

IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_groups
where name = 'CEODemo')
BEGIN
    Create WORKLOAD GROUP CEODemo WITH
    ( MIN_PERCENTAGE_RESOURCE = 10        -- integer value
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 5 -- integer value
    ,CAP_PERCENTAGE_RESOURCE = 100
    )
END
```

   The script creates a workload group called `CEODemo` to reserve resources exclusively for the workload group. In this example, a workload group with a `MIN_PERCENTAGE_RESOURCE` set to 10% and `REQUEST_MIN_RESOURCE_GRANT_PERCENT` set to 5% is guaranteed 2 concurrency.

2. Select **Run** from the toolbar menu to execute the SQL command.

3. In the query window, replace the script with the following to create a Workload Classifier called `CEODreamDemo` that assigns a workload group and importance to incoming requests:

```
SQL

IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_classi-
fiers where  name = 'CEODreamDemo')
BEGIN
    Create Workload Classifier CEODreamDemo with
    ( Workload_Group ='CEODemo',MemberName='asa.sql.workload02',IMPOR-
TANCE = BELOW_NORMAL);
END
```

   This script sets the Importance to **BELOW_NORMAL** for the `asa.sql.workload02` user, through the new `CEODreamDemo` Workload Classifier.

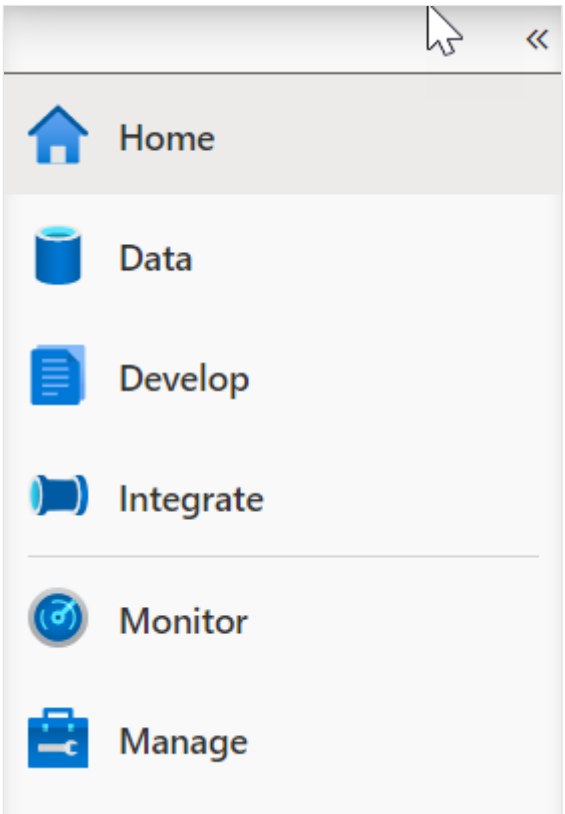
4. Select **Run** from the toolbar menu to execute the SQL command.

5. In the query window, replace the script with the following to confirm that there are no active queries being run by `asa.sql.workload02` (suspended queries are OK):

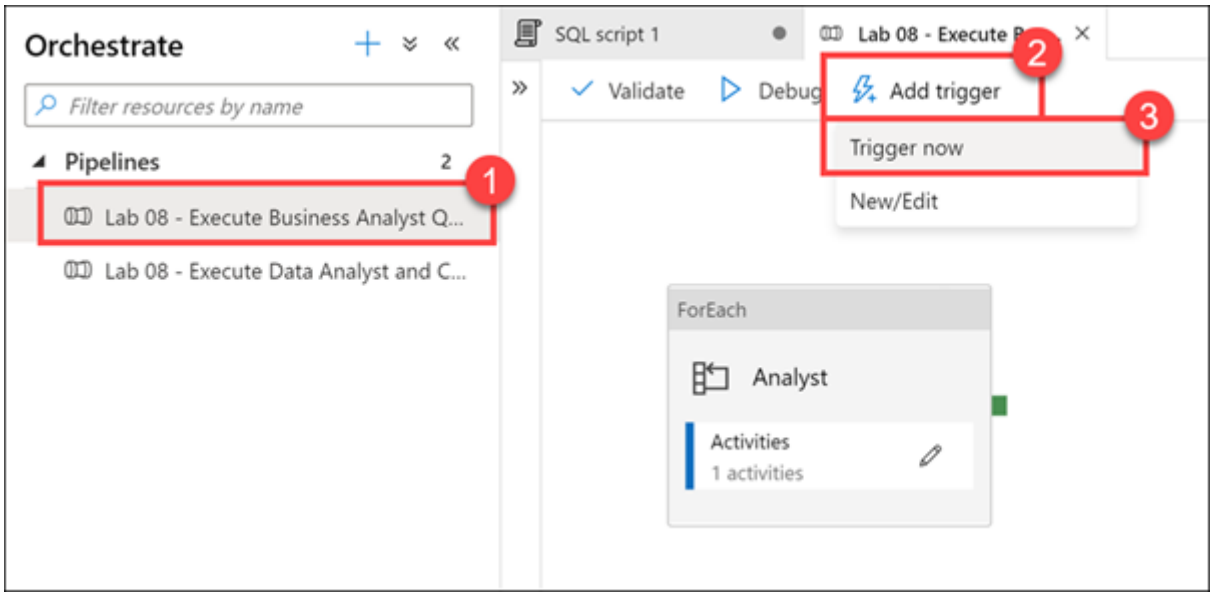
```
SQL

SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

6. Select **Run** from the toolbar menu to execute the SQL command.

7. Select the **Integrate** hub.



8. Select the **Lab 08 - Execute Business Analyst Queries** Pipeline **(1)**, which will run / trigger `asa.sql.workload02` queries. Select **Add trigger (2)**, then **Trigger now (3)**. In the dialog that appears, select **OK**.

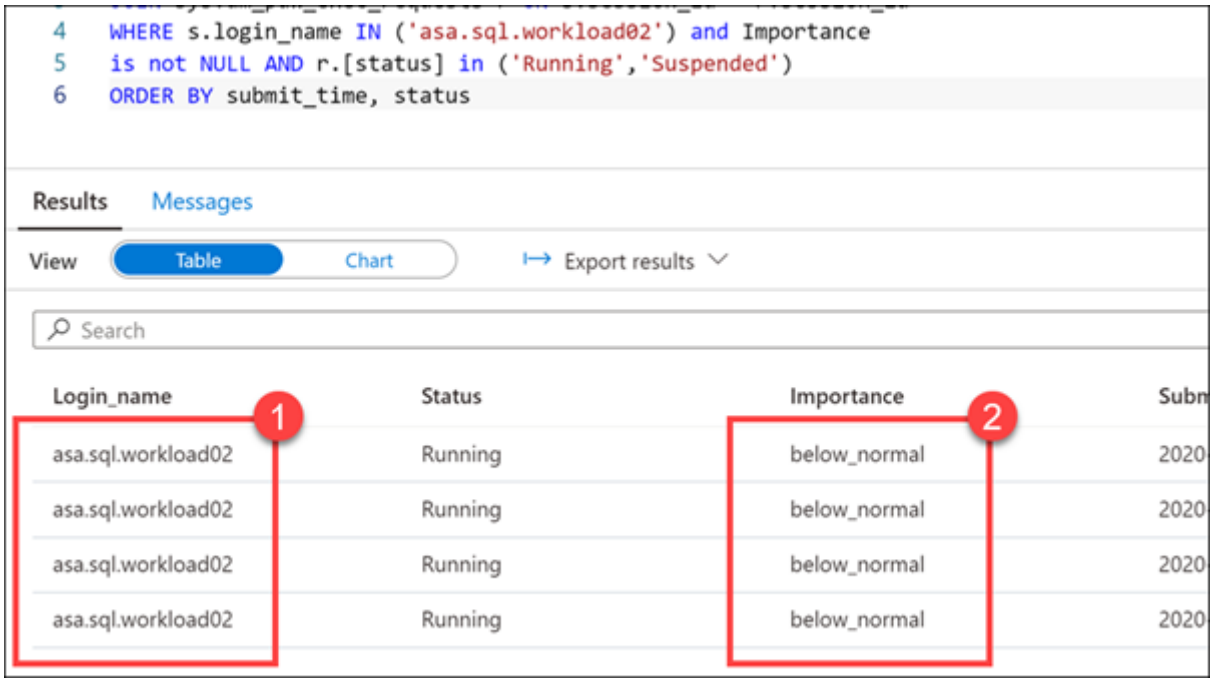


9. In the query window, replace the script with the following to see what happened to all the `asa.sql.workload02` queries we just triggered as they flood the system:

```
SQL

SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

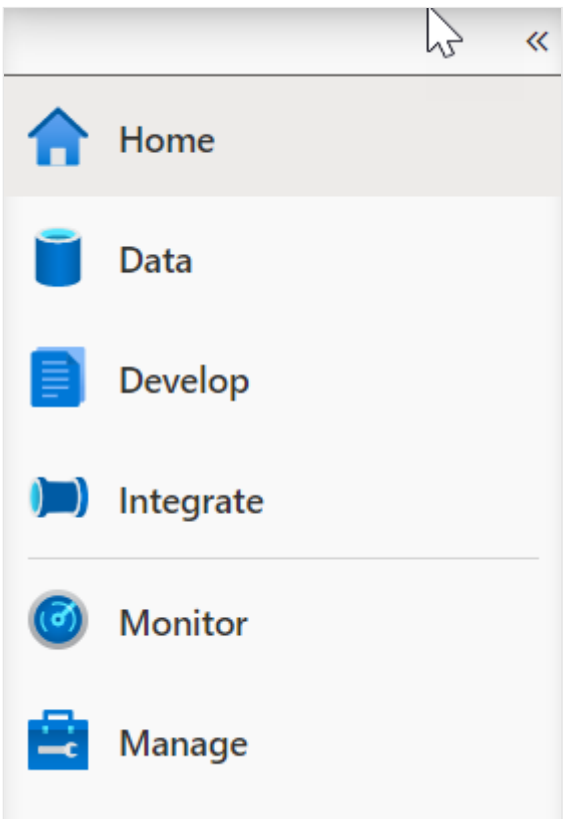
10. Select **Run** from the toolbar menu to execute the SQL command.

You should see an output similar to the following that shows the importance for each session set to `below_normal`:

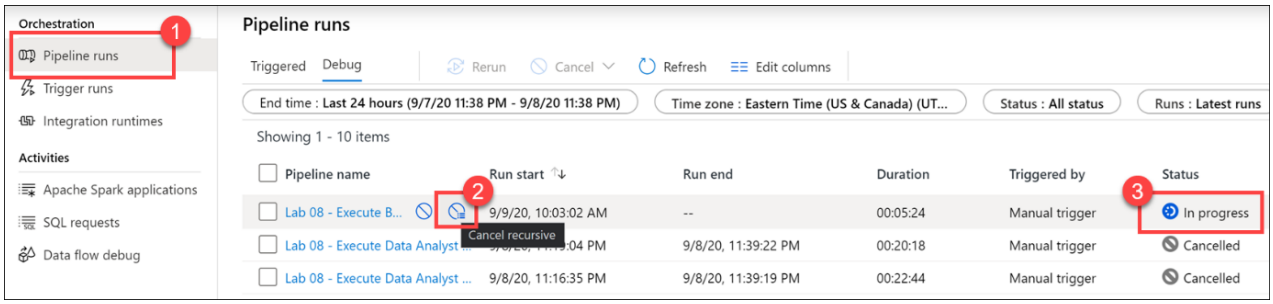


Notice that the running scripts are executed by the `asa.sql.workload02` user **(1)** with an Importance level of **below_normal (2)**. We have successfully configured the business analyst queries to execute at a lower importance than the CEO queries. We can also see that the `CEODreamDemo` Workload Classifier works as expected.

11. Select the **Monitor** hub.



12. Select **Pipeline runs (1)**, and then select **Cancel recursive (2)** for each running Lab 08 pipelines, marked **In progress (3)**. This will help speed up the remaining tasks.



13. Return to the query window under the **Develop** hub. In the query window, replace the script with the following to set 3.25% minimum resources per request:

SQL

```
IF  EXISTS (SELECT * FROM sys.workload_management_workload_classifiers
where group_name = 'CEODemo')
BEGIN
    Drop Workload Classifier CEODreamDemo
    DROP WORKLOAD GROUP CEODemo
    --- Creates a workload group 'CEODemo'.
        Create  WORKLOAD GROUP CEODemo WITH
    (MIN_PERCENTAGE_RESOURCE = 26 -- integer value
        ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 3 -- factor of 26 (guar-
anteed more than 8 concurrencies)
    ,CAP_PERCENTAGE_RESOURCE = 100
    )
    --- Creates a workload Classifier 'CEODreamDemo'.
    Create Workload Classifier CEODreamDemo with
    (Workload_Group ='CEODemo',MemberName='asa.sql.workload02',IMPOR-
TANCE = BELOW_NORMAL);
END
```

> **Note**
>
> Configuring workload containment implicitly defines a maximum level of concurrency. With a CAP_PERCENTAGE_RESOURCE set to 60% and a REQUEST_MIN_RESOURCE_GRANT_PERCENT set to 1%, up to a 60-concurrency level is allowed for the workload group. Consider the method included below for determining the maximum concurrency:
>
> [Max Concurrency] = [CAP_PERCENTAGE_RESOURCE] / [REQUEST_MIN_RESOURCE_GRANT_PERCENT]

14. Select **Run** from the toolbar menu to execute the SQL command.

15. Let's flood the system again and see what happens for `asa.sql.workload02`. To do this, we will run an Azure Synapse Pipeline which triggers queries. Select the `Integrate` Tab. **Run** the **Lab 08 - Execute Business Analyst Queries** Pipeline, which will run / trigger `asa.sql.workload02` queries.

16. In the query window, replace the script with the following to see what happened to all of the `asa.sql.workload02` queries we just triggered as they flood the system:

```
SQL

SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is  not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

17. Select **Run** from the toolbar menu to execute the SQL command.

After several moments (up to a minute), we should see several concurrent executions by the `asa.sql.workload02` user running at **below_normal** importance. We have validated that the modified Workload Group and Workload Classifier works as expected.

```
4   WHERE s.login_name IN ('asa.sql.workload02') and Importance
5   is  not NULL AND r.[status] in ('Running','Suspended')
6   ORDER BY submit_time, status
```

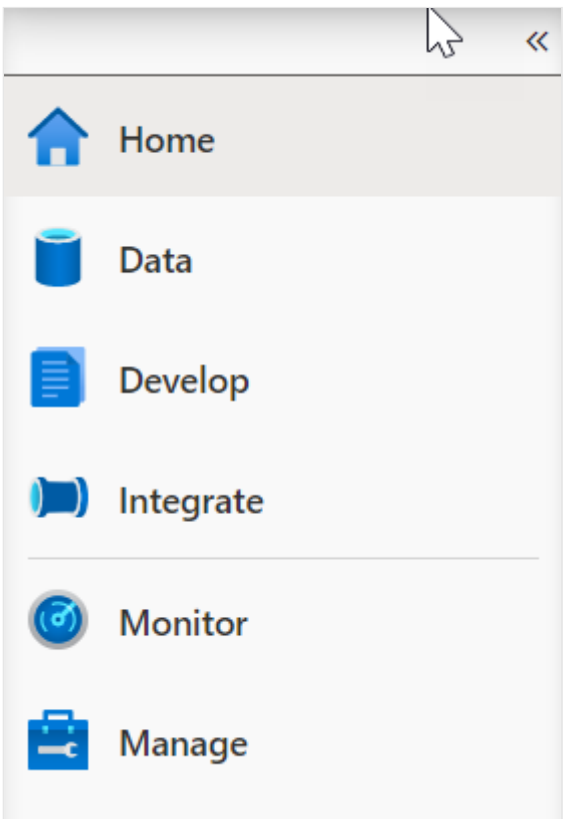
Results | Messages

View: Table | Chart — Export results

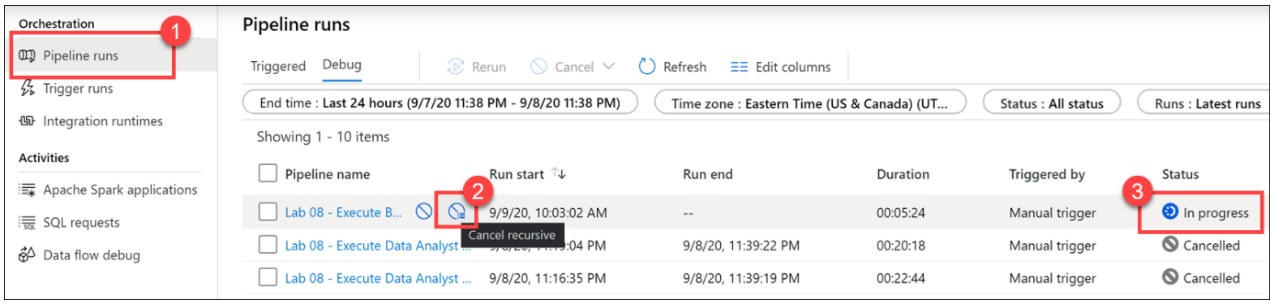
Search

Login_name	Status	Importance
asa.sql.workload02	Running	below_normal
asa.sql.workload02	Running	below_normal
asa.sql.workload02	Running	below_normal
asa.sql.workload02	Running	below_normal
asa.sql.workload02	Running	below_normal

18. Select the **Monitor** hub.



19. Select **Pipeline runs (1)**, and then select **Cancel recursive (2)** for each running Lab 08 pipelines, marked **In progress (3)**. This will help speed up the remaining tasks.


Orchestration
Pipeline runs
Trigger runs
Integration runtimes
Activities
Apache Spark applications
SQL requests
Data flow debug
1
Pipeline runs
Triggered
Debug
Rerun
Cancel
Refresh
Edit columns
End time : Last 24 hours (9/7/20 11:38 PM - 9/8/20 11:38 PM)
Time zone : Eastern Time (US & Canada) (UT...
Status : All status
Runs : Latest runs
Showing 1 - 10 items
Pipeline name
Run start
Run end
Duration
Triggered by
Status
2
Lab 08 - Execute B...
9/9/20, 10:03:02 AM
--
00:05:24
Manual trigger
In progress
Cancel recursive
3
Lab 08 - Execute Data Analyst ...
9/8/20, 11:39:22 PM
00:20:18
Manual trigger
Cancelled
Lab 08 - Execute Data Analyst ...
9/8/20, 11:16:35 PM
9/8/20, 11:39:19 PM
00:22:44
Manual trigger
Cancelled