

PUBLIC

Introduction to SAP HANA Cloud

DAT-161

Exercises / Solutions
Robert Waywell / SAP
Jason Hinsperger / SAP

TABLE OF CONTENTS

Pre-Requisites.....	3
Exercise 1. Walk Through the Provisioning Process	5
Exercise 2. Update a provisioned instance (Not Applicable for Trial Instances).....	7
Exercise 3: Introduction to HANA Cloud, Data Lake	8
Exercise 4. Setting Up a Remote Source To Amazon Athena.....	14
Exercise 5: Using NSE in HANA Cloud	22

www.sap.com/contactsap

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See www.sap.com/copyinginfo for additional trademark information and notices.

HANA Cloud is a service designed deliver a complete set of data management capabilities in the cloud and act as the gateway to all of your enterprise data. Along with the HANA database, HANA Cloud provides unique features like the ability to access remote data sources, enabling an easy transition between federated and replicated access to that data. HANA Cloud also enables you to control TCO by enabling storage of data of various temperatures using cost effective storage, including HANA Native Storage Extensions and the integrated HANA data lake.

Rather than focus on the capabilities of the HANA database in HANA Cloud, for which there is already an abundance of learning materials, this set of hand-on exercises enables you to explore some of the above-mentioned features that make HANA Cloud unique.

- You will go through the provisioning process (if you haven't already),
- You will work with the HANA data lake to create tables and add data to cold storage,
- You will connect to a virtual cloud data sources (Amazon Athena), and query and replicate data
- You will explore the use of warm storage using the HANA native storage extension (NSE) feature of HANA Cloud.

Pre-Requisites

The following exercise uses HANA Cloud Trial. If you have a trial account, go to Subaccount Entitlements and add SAP HANA Cloud if you haven't already. If you do not have an active trial account, sign-up through our HANA Cloud Trial registration form – available here: <https://www.sap.com/cmp/td/sap-hana-cloud-trial.html>





The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar contains a navigation menu with options: Overview, Spaces, Subscriptions, Connectivity, Destinations, Security, Quota Plans, and Entitlements. The main content area displays details for the subaccount 'trial'. It shows the Subdomain as 'ce3244c0trial' and the ID as '6fce28d4-86da-4793-be05-e08e7a563255'. Below this, there are tabs for 'Cloud Foundry Environment' and 'Entitlements'. The 'Org. Name' is listed as 'ce3244c0trial'.

The screenshot shows the 'Subaccount: trial - Entitlements' page in the SAP Cloud Platform Cockpit. It features a table with columns: Service, Plan, Subaccount Quota, Subaccount Assignment, Remaining Global Quota, and Actions. The table lists two entitlements: 'ABAP Trial' with a 'shared' plan and 'Application Runtime' with a 'MEMORY' plan. A 'Configure Entitlements' button is visible in the top right corner of the table area.

Service	Plan	Subaccount Quota	Subaccount Assignment	Remaining Global Quota	Actions
ABAP Trial	shared	1 Units		0	
Application Runtime	MEMORY	4 Units		0	

DAT-161

Subaccount: trial - Entitlements

<div><div> trial</div><div><div>SAP HANA Cloud</div><div><div>×</div><div>🔍</div></div></div><div><div>Add Service Plans</div><div>Save</div><div>Cancel</div></div></div>					
Service	Plan	Subaccount Quota	Subaccount Assignment	Remaining Global Quota	Actions
SAP HANA Cloud	hana-cloud-connection	limited ⓘ	limited	limited ⓘ	
	hana	limited ⓘ	limited	limited ⓘ	
	relational-data-lake	limited ⓘ	limited	limited ⓘ	

Once you log on to SAP Cloud Platform Cockpit, ensure that you have navigated to the correct global account, sub account, space, and you have selected SAP HANA Cloud from the left sidebar.

SAP Cloud Platform Cockpit

Applications

Services

Service Marketplace

Service Instances

User-Provided Services

SAP HANA Cloud

Trial Home / ce3244c0trial / trial / dev

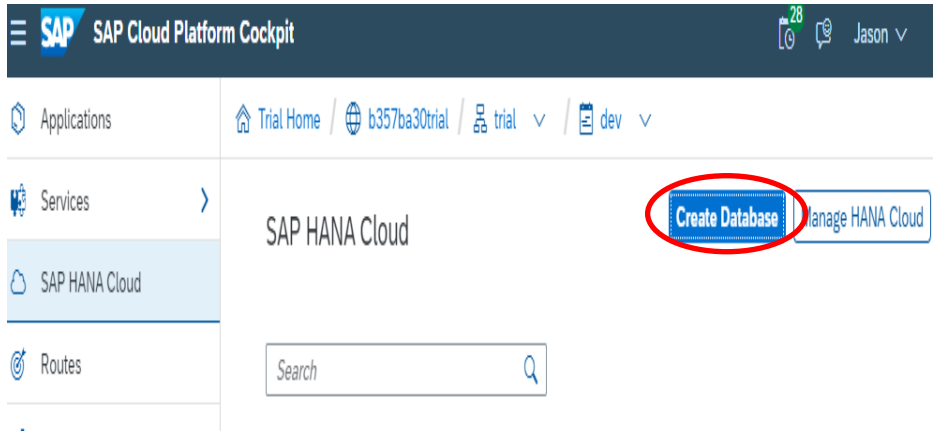
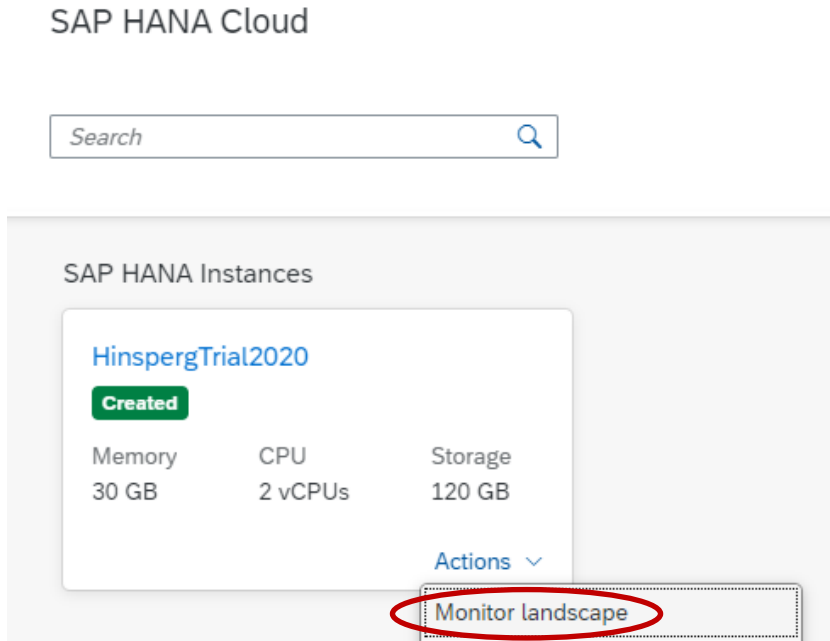
Space: dev - SAP HANA Cloud

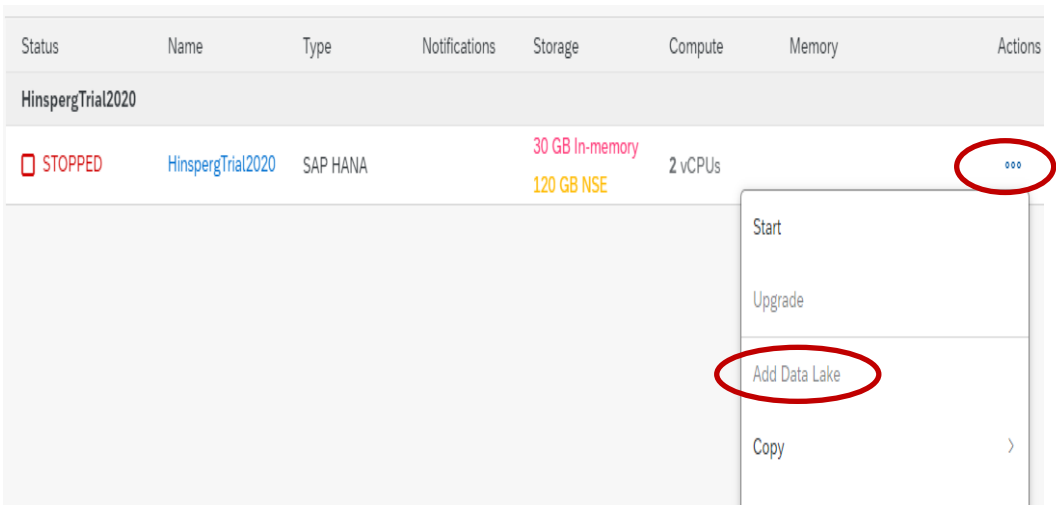
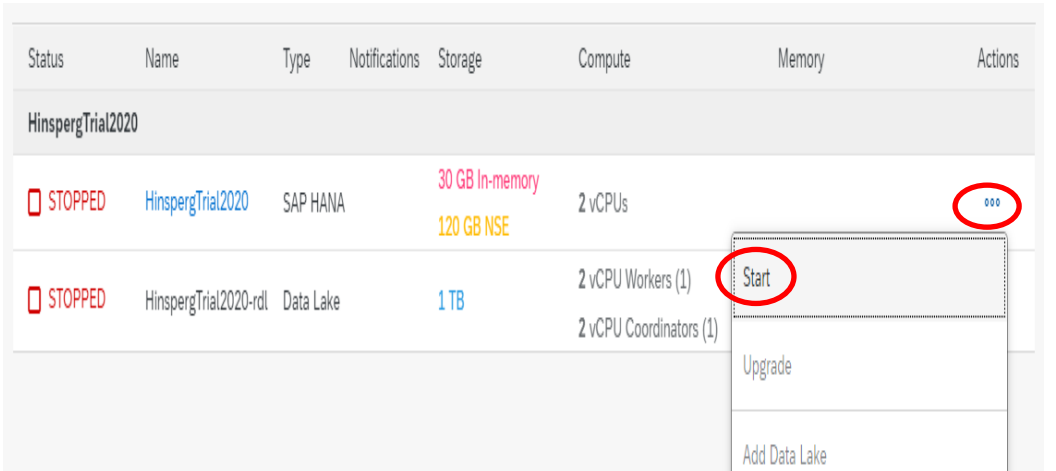
All Categories Search

SAP HANA Instances

Exercise 1. Walk Through the Provisioning Process

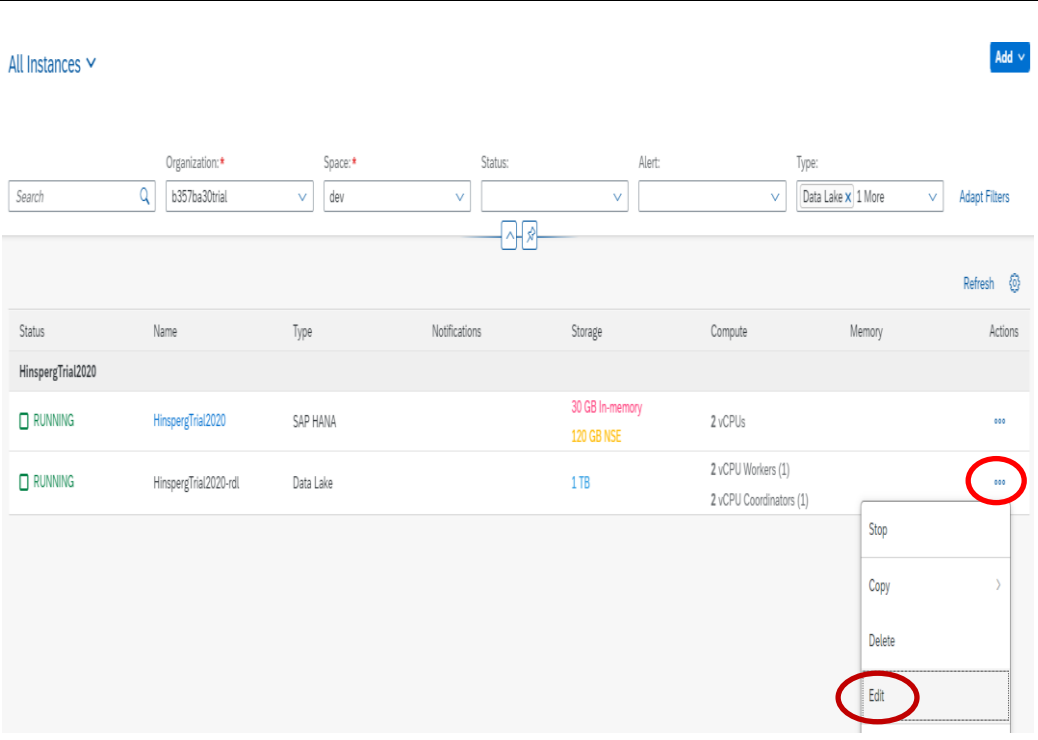
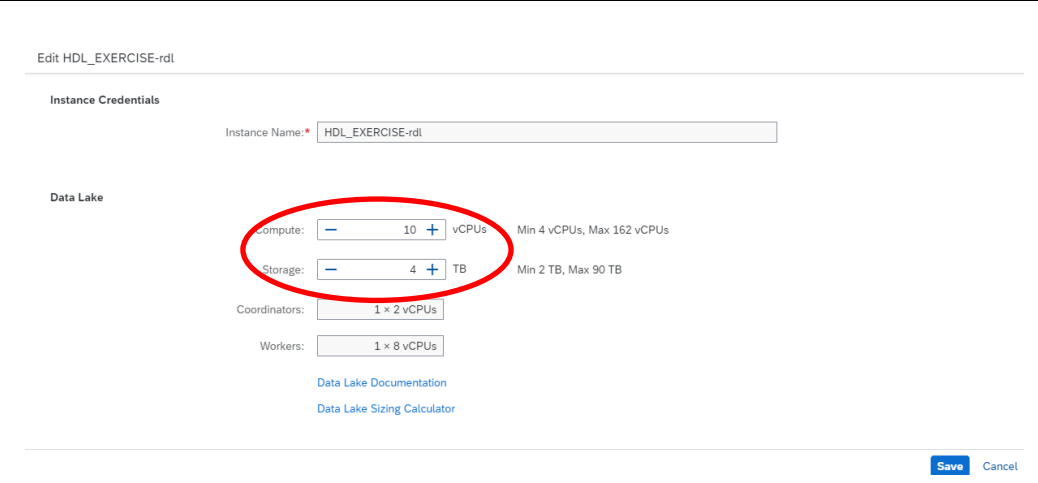
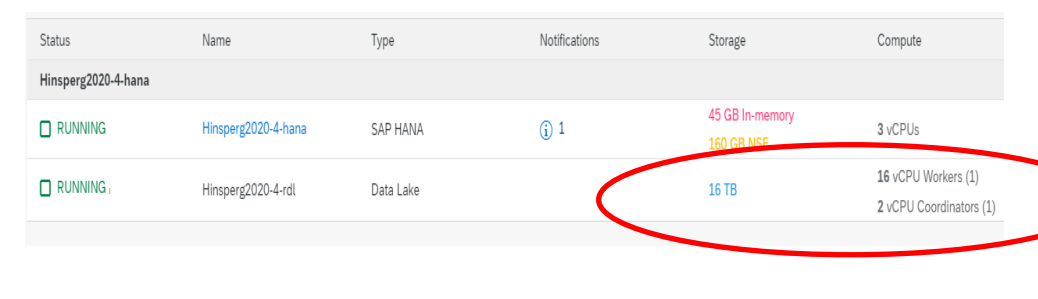
The goal of this exercise is to create a HANA Cloud instance and a Data Lake instance connected to it. Note, trial users may only create one instance per subaccount per geographic location. If you have already created an instance with your trial account, but have not created a data lake instance, complete steps 2 and 3. If you already have a data lake instance, skip to step 4 to ensure it is running.

Explanation	Screenshot
<p>1. From your SAP Cloud Platform trial account, open the “SAP HANA Cloud” item and click “Create Database”</p> <p>Follow through the wizard to create a HANA Cloud instance. Be sure to create a HANA data lake instance when given the option.</p>	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. The left sidebar contains a menu with 'Applications', 'Services', 'SAP HANA Cloud', and 'Routes'. The main area displays the 'SAP HANA Cloud' service tile. A red circle highlights the 'Create Database' button on the right side of the tile. The top navigation bar shows 'SAP Cloud Platform Cockpit' and the user 'Jason'.</p>
<p>2. Complete step 2 and 3 if you already have a HANA Cloud instance, but have not created a HANA data lake instance. From the HANA Cloud service tile inside of SCP, click 'Actions' and choose “Monitor landscape”</p>	 <p>The screenshot shows the 'SAP HANA Cloud' service tile. A search bar is at the top. Below it, the 'SAP HANA Instances' section displays a card for 'HinspergTrial2020'. The card indicates the instance is 'Created' and lists specifications: Memory (30 GB), CPU (2 vCPUs), and Storage (120 GB). An 'Actions' dropdown menu is visible at the bottom right of the card, with the 'Monitor landscape' option circled in red.</p>

Explanation	Screenshot																																
<p>3.</p> <p>When the HANA Cloud manager opens, choose “Add Data Lake” from the “Actions” menu in the landscape monitor.</p>	 <table><tr><th>Status</th><th>Name</th><th>Type</th><th>Notifications</th><th>Storage</th><th>Compute</th><th>Memory</th><th>Actions</th></tr><tr><td colspan="8">HinspergTrial2020</td></tr><tr><td>STOPPED</td><td>HinspergTrial2020</td><td>SAP HANA</td><td></td><td>30 GB In-memory 120 GB NSE</td><td>2 vCPUs</td><td></td><td>...</td></tr></table>	Status	Name	Type	Notifications	Storage	Compute	Memory	Actions	HinspergTrial2020								STOPPED	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...								
Status	Name	Type	Notifications	Storage	Compute	Memory	Actions																										
HinspergTrial2020																																	
STOPPED	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...																										
<p>4.</p> <p>When the HANA Cloud instance is ready the status will change to <i>Running</i>. If the instance was previously created and is in the “Stopped” state, you can start it from the “Actions” menu.</p>	 <table><tr><th>Status</th><th>Name</th><th>Type</th><th>Notifications</th><th>Storage</th><th>Compute</th><th>Memory</th><th>Actions</th></tr><tr><td colspan="8">HinspergTrial2020</td></tr><tr><td>STOPPED</td><td>HinspergTrial2020</td><td>SAP HANA</td><td></td><td>30 GB In-memory 120 GB NSE</td><td>2 vCPUs</td><td></td><td>...</td></tr><tr><td>STOPPED</td><td>HinspergTrial2020-rtl</td><td>Data Lake</td><td></td><td>1 TB</td><td>2 vCPU Workers (1) 2 vCPU Coordinators (1)</td><td></td><td>Start Upgrade Add Data Lake</td></tr></table>	Status	Name	Type	Notifications	Storage	Compute	Memory	Actions	HinspergTrial2020								STOPPED	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...	STOPPED	HinspergTrial2020-rtl	Data Lake		1 TB	2 vCPU Workers (1) 2 vCPU Coordinators (1)		Start Upgrade Add Data Lake
Status	Name	Type	Notifications	Storage	Compute	Memory	Actions																										
HinspergTrial2020																																	
STOPPED	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...																										
STOPPED	HinspergTrial2020-rtl	Data Lake		1 TB	2 vCPU Workers (1) 2 vCPU Coordinators (1)		Start Upgrade Add Data Lake																										

Exercise 2. Update a provisioned instance (Not Applicable for Trial Instances)

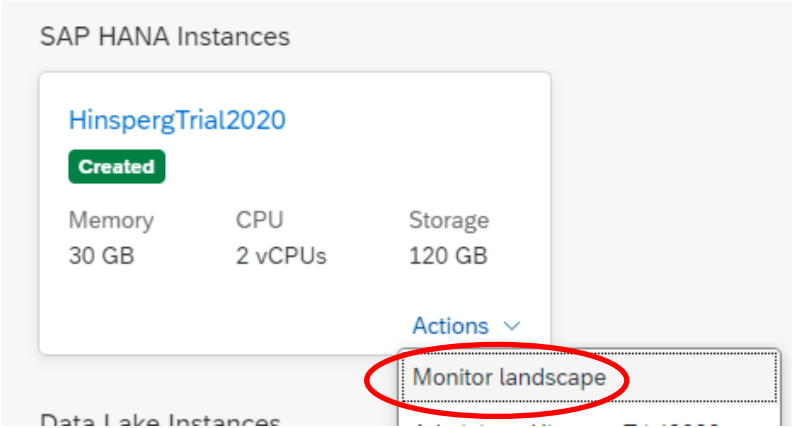
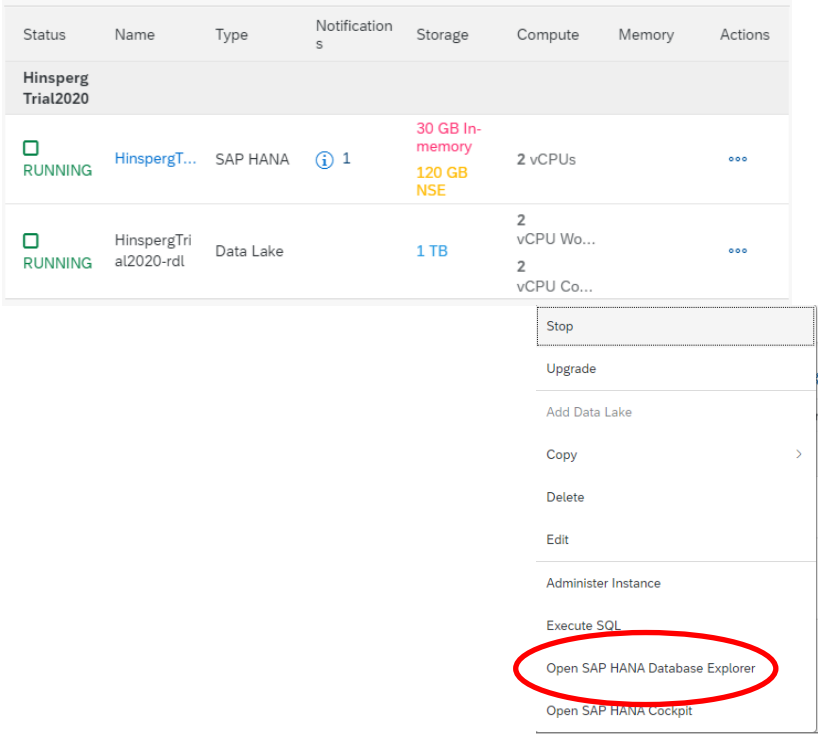
The goal of this exercise is to update the resources allocated to a provisioned data lake instance. HANA Cloud trial instances cannot be edited. This exercise only applies to provisioned instances of the full HANA Cloud service.

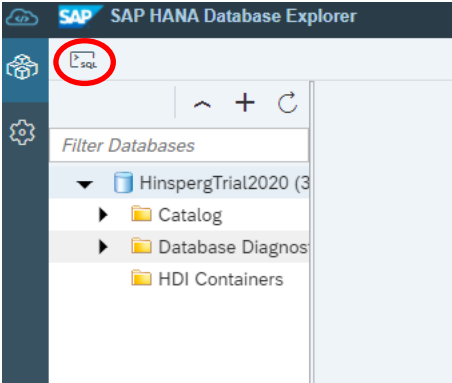
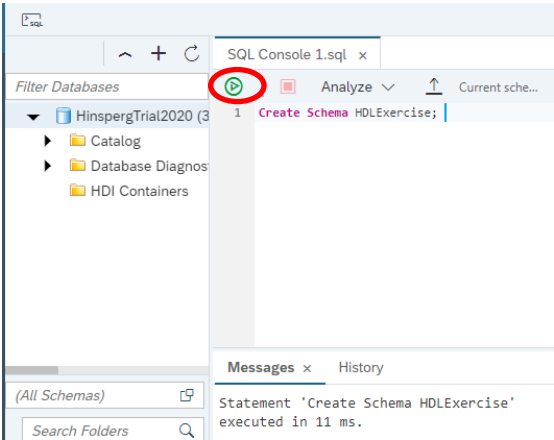
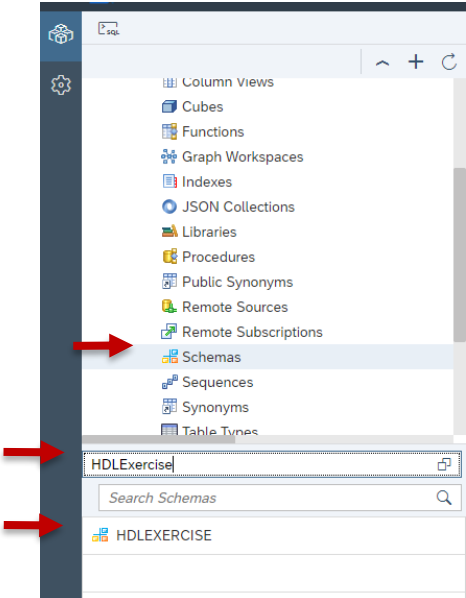
<p>1. Open the HANA Cloud management interface and click on the “Edit” option under the Actions extended menu.</p>	 <p>The screenshot shows the 'All Instances' page in the HANA Cloud management interface. It includes filters for Organization, Space, Status, Alert, and Type. A table lists instances with columns for Status, Name, Type, Notifications, Storage, Compute, Memory, and Actions. The instance 'HinspergTrial2020-rtl' is highlighted, and its 'Edit' option in the Actions menu is circled in red.</p>
<p>2. Increase Compute to 10 vCPU and increase storage to 4 TB. Note: we have now allocated 8 vCPUs to workers, and 2 vCPUs are coordinators. Click on Save.</p>	 <p>The screenshot shows the 'Edit HDL_EXERCISE-rtl' configuration page. It includes fields for Instance Name, Data Lake, Compute, Storage, Coordinators, and Workers. The 'Compute' field is set to 10 vCPUs and the 'Storage' field is set to 4 TB, both of which are circled in red.</p>
<p>3. Wait for the updated instance to start again. Once <i>Starting</i> turns to <i>Running</i>, the changes have been reflected in the Data Lake instance. This process could take a few minutes.</p>	 <p>The screenshot shows the 'All Instances' page with the updated instance 'Hinsperg2020-4-rtl'. The 'Compute' column for this instance shows '16 vCPU Workers (1)' and '2 vCPU Coordinators (1)', which are circled in red.</p>

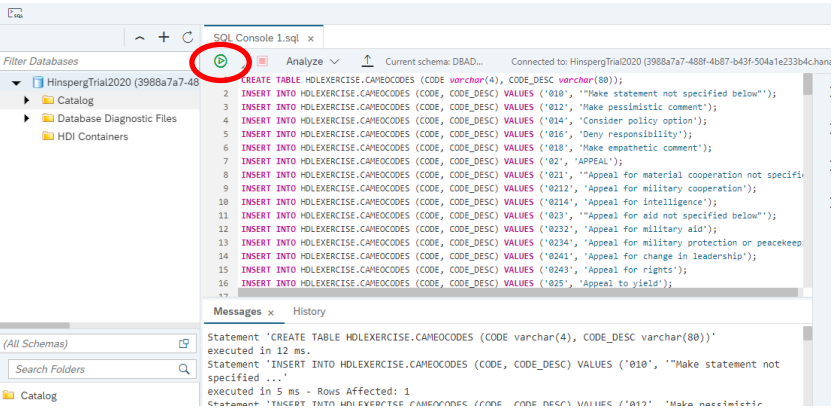
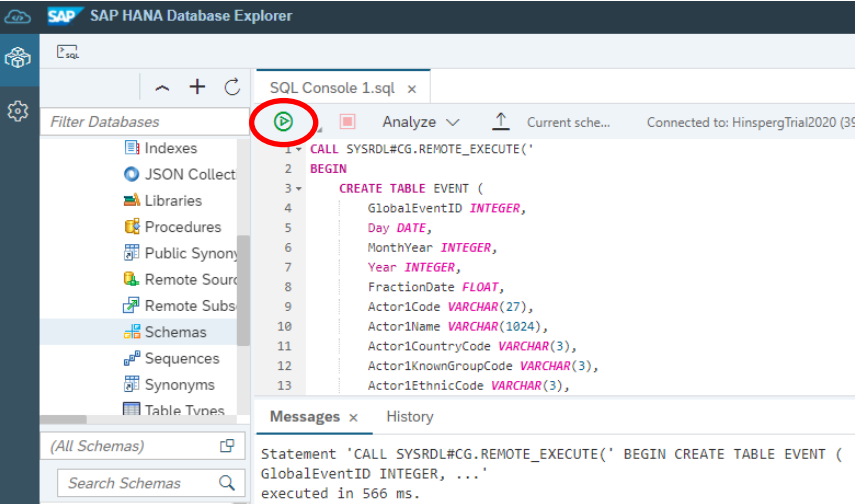
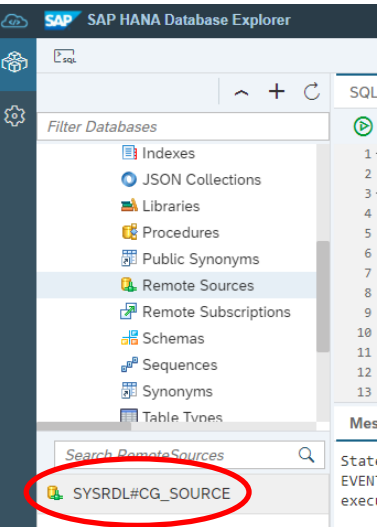
Exercise 3: Introduction to HANA Cloud, Data Lake

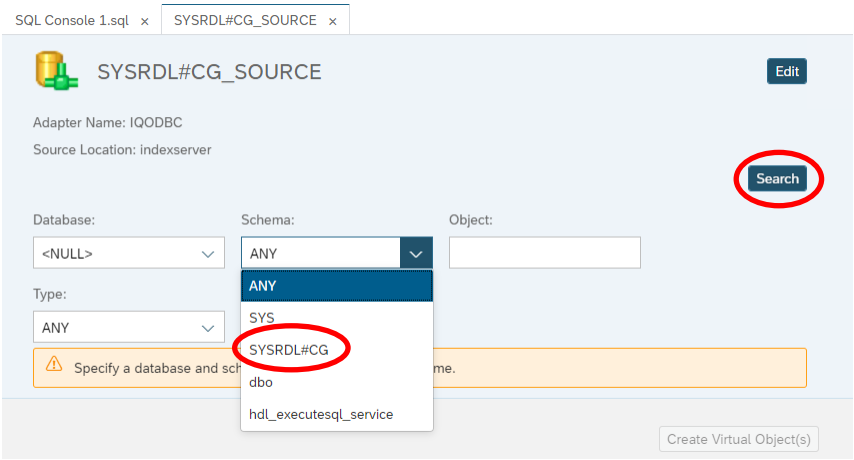
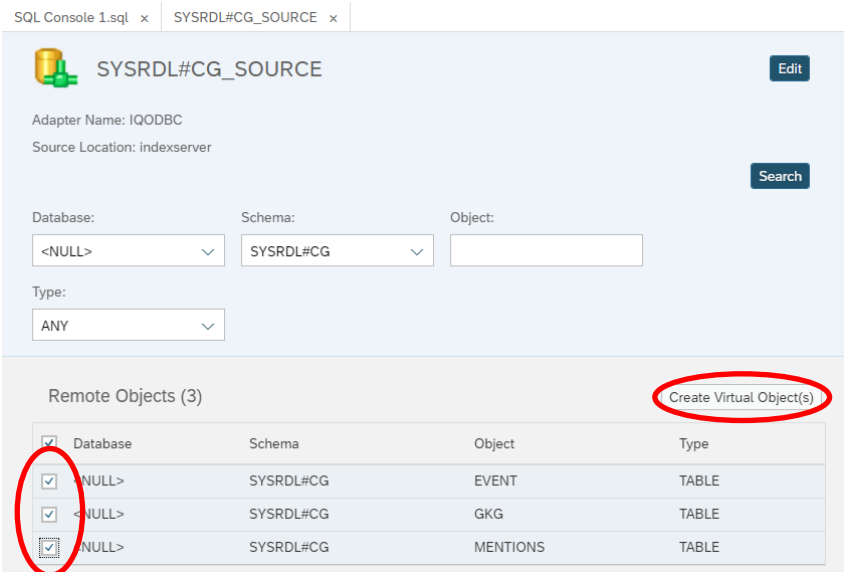
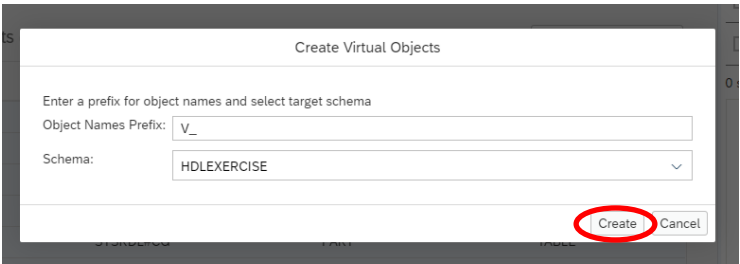
The goal of this exercise is to create some schema and add some data to a HANA cloud, data lake instance, then query that data through the standard HANA Cloud interfaces.

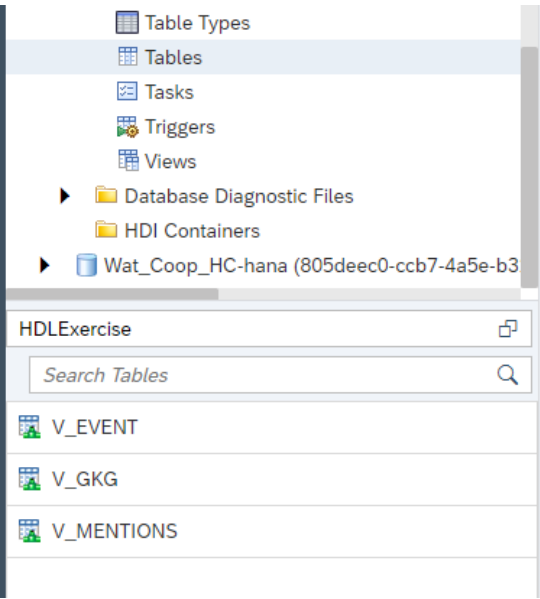
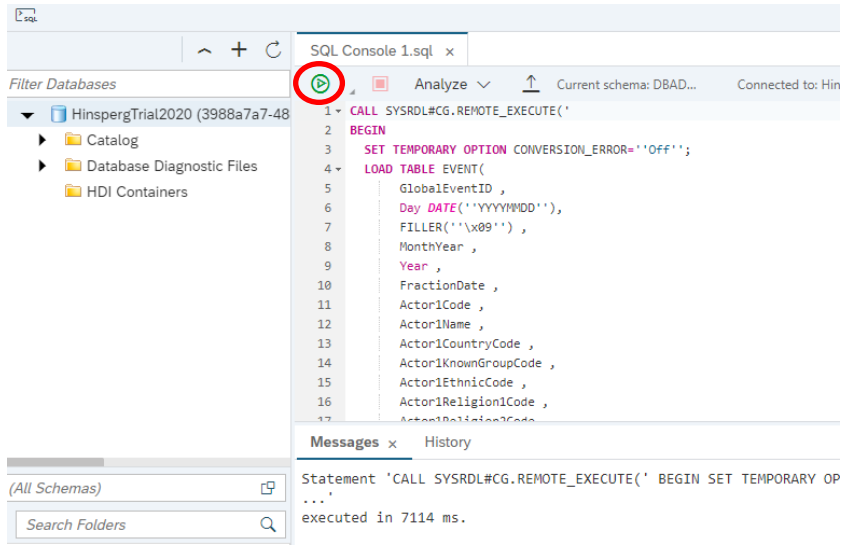
The sample data we are using comes from the open-source *Global Database of Events, Language, and Tone* (GDELT) project. If you're unfamiliar with the information inventoried by the GDELT project, go to www.gdeltproject.org for more information. We use a small subset of data from GDELT that lists events and event descriptions from various locations around the globe.

Explanation	Screenshot
1. Navigate to the HANA Cloud instance you created and open the landscape monitor.	
2. Under the action menu, open the SAP HANA Database Explorer. If prompted for login credentials, enter "DBADMIN" as the user id, and the password you specified when you created your HANA Cloud instance.	

Explanation	Screenshot
<p>3.</p> <p>Open a SQL console for your trial instance, If there is not one already open.</p>	 The screenshot shows the SAP HANA Database Explorer interface. On the left sidebar, there is a blue icon representing a SQL console, which is circled in red. The main pane shows a tree view of the database structure, including 'HinspergTrial2020 (3)' with sub-items like 'Catalog', 'Database Diagnostics', and 'HDI Containers'.
<p>4.</p> <p>In the SQL window, create a new schema for your data lake with the following command:</p> <p><i>Create Schema HDLEExercise;</i></p> <p>Click the green arrow to execute the command.</p>	 The screenshot shows the SQL Console window. The command 'Create Schema HDLEExercise;' is entered in the text area. A green circular button with a white play icon (execute) is circled in red. Below the command, the 'Messages' pane shows the execution result: 'Statement 'Create Schema HDLEExercise' executed in 11 ms.'
<p>5.</p> <p>Check that the Schema "HDLEExercise" has been created by navigating to Catalog, Schemas, and then type "HDLEExercise" in the Choose schema input form.</p> <p>Ensure that HDLEExercise exists.</p>	 The screenshot shows the SAP HANA Database Explorer interface. The 'Schemas' folder is selected in the left sidebar. The main pane displays a list of database objects, including 'Schemas'. Below the list, there is a search bar where 'HDLEExercise' has been typed. The search results show 'HDLEEXERCISE' as a valid schema.

Explanation	Screenshot
<p>6.</p> <p>First, we will create a HANA table which we will use later to demonstrate federated query access between the HANA Cloud, HANA database and the HANA data lake.</p> <p>Copy and paste the content of DAT161Exercise_CameoCodes.sql into the SQL console.</p> <p>Execute the command by pressing the run button.</p> <p>This command creates a table called CAMEOCODES and inserts some data into it.</p>	
<p>7.</p> <p>Copy and paste the content of DAT161Exercise_CreateTable.sql into SQL console. Execute the command by pressing the run button.</p> <p>This command creates the following tables in data lake: EVENT, GKG, MENTIONS.</p> <p>Note the create table statements are wrapped inside the <code>SYSRDL#CG.remote_execute</code> function. <code>remote_execute()</code> is a stored procedure which allows you to execute SQL natively against HANA data lake.</p>	
<p>8.</p> <p>Verify the tables have been created in the data lake.</p> <p>First, navigate to <i>Catalog</i>, then <i>Remote Sources</i> from the left side bar. Click on <i>Remote Sources</i>. You should see one remote source called <code>SYSRDL#CG_SOURCE</code>.</p> <p>Double Click on <code>SYSRDL#CG_SOURCE</code>.</p>	

Explanation	Screenshot
<p>9.</p> <p>A new remote source tab is open next to the „SQL Console 1.sql“, called „SYSRDL#CG_SOURCE“.</p> <p>On the remote source tab, open the schema menu and choose SYSRDL#CG from the drop down as the remote schema.</p> <p>Click on the Search button.</p>	
<p>10.</p> <p>You should see the list of 3 tables that you created in step 6.</p> <p>Click the checkbox beside each table and then click the “Create Virtual Objects” button.</p>	
<p>11.</p> <p>Set the „Object Names Prefix“ to „V_“</p> <p>Then set the „Schema“ name to the name of the schema created in step 4 (HDL_EXERCISE), and click on create.</p> <p>This creates a HANA virtual table for each data lake table, and names the virtual table V_<name of data lake table>. For instance, the virtual table referring to the data lake EVENT table will be named V_EVENT.</p>	

Explanation	Screenshot
<p>The virtual tables are create in the HANA schema HDLEXERCISE.</p>	
<p>12. To verify the virtual tables have been created successfully, navigate to the <i>Catalog</i> in the left hand nav., then choose <i>Tables</i>.</p> <p>Find the <i>HDLEExercise</i> in the list and you should see the virtual tables that you have just created.</p>	
<p>13. Next we Will load some data into the data lake tables. Copy and paste the command from <code>DAT161Exercise_LoadTable.sql</code> into the SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.</p> <p>This command loads data from a public AWS s3 bucket into the following tables in data lake: EVENT, GKG, MENTIONS.</p> <p>Note that we supply the location of the s3 bucket, its region, its access key id, as well as its secret access key directly in the LOAD statement so that the data lake can read the data directly from the S3 bucket to maximize load performance.</p>	

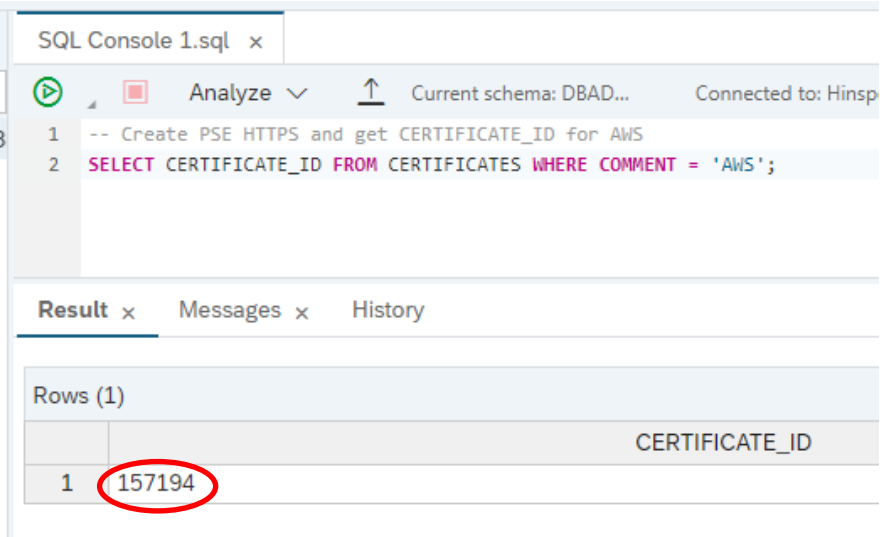
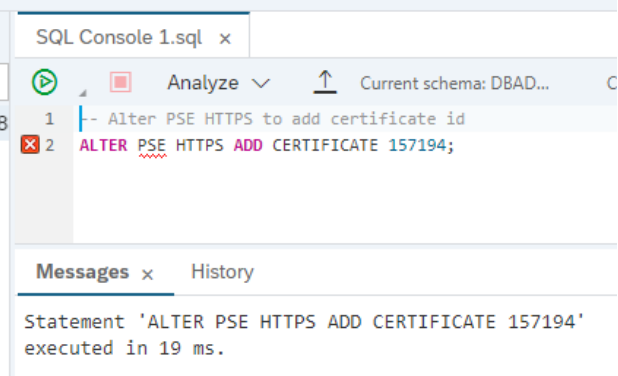
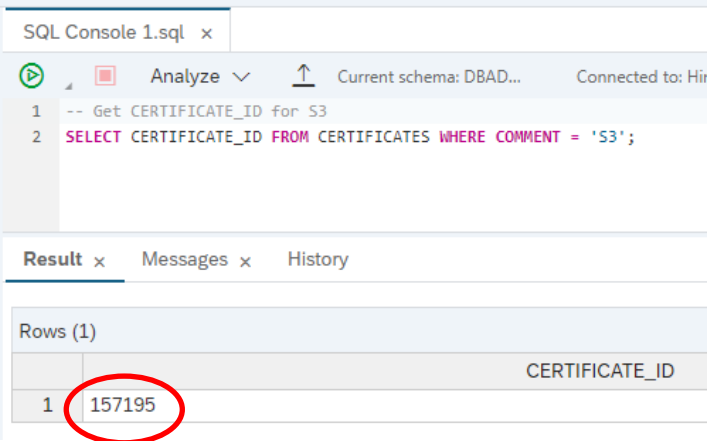
Explanation	Screenshot									
<p>14.</p> <p>Now we can query the data from the data lake using the previously created virtual tables.</p> <p>Copy and paste the command from <i>DAT161DataLake_Query1.sql</i> into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.</p>	<p>The screenshot displays the SAP HANA SQL Console interface. On the left, a sidebar shows the database structure with 'HinspergTrial2020' selected. The main area shows a SQL query being executed. The query is as follows:</p> <pre>1 SELECT COUNT(CODE_DESC), CODE_DESC 2 FROM (3 SELECT c.CODE_DESC, c.CODE, e.* 4 FROM HDLEXERCISE_V_EVENT e 5 INNER JOIN 6 HDLEXERCISE_V_MENTIONS m 7 ON c.CODE = e.EventCode 8) event 9 INNER JOIN 10 HDLEXERCISE_V_MENTIONS m 11 ON m.GlobalEventID = event.GlobalEventID 12 WHERE event."Day" > '2018-11-19' 13 GROUP BY CODE_DESC 14 ORDER BY COUNT(CODE_DESC) DESC;</pre> <p>Below the query, the 'Result' tab shows 45 rows. The first two rows are visible:</p> <table><tr><th>Row</th><th>COUNT(CODE_DESC)</th><th>CODE_DESC</th></tr><tr><td>1</td><td>113</td><td>"Consult not specified below"</td></tr><tr><td>2</td><td>103</td><td>"Make statement not specified below"</td></tr></table>	Row	COUNT(CODE_DESC)	CODE_DESC	1	113	"Consult not specified below"	2	103	"Make statement not specified below"
Row	COUNT(CODE_DESC)	CODE_DESC								
1	113	"Consult not specified below"								
2	103	"Make statement not specified below"								

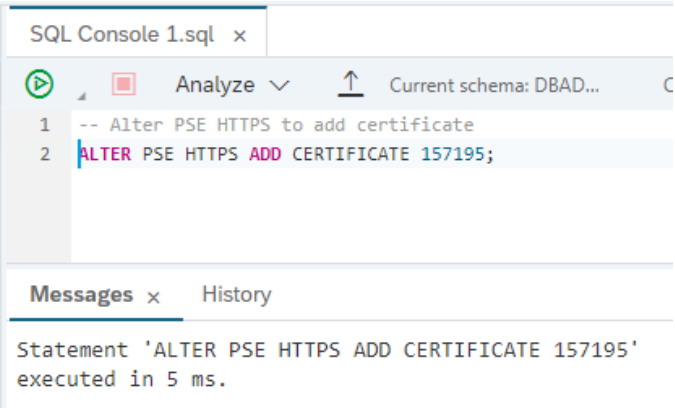
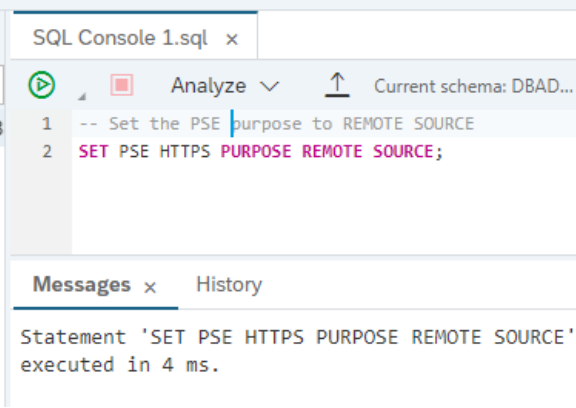
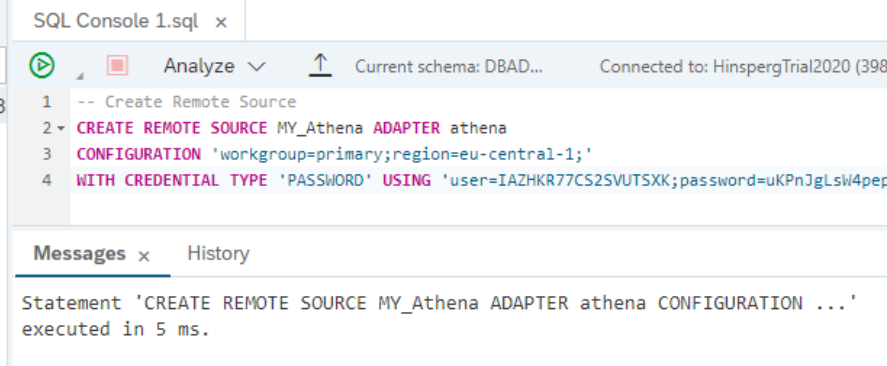
That concludes this exercise. You have successfully configured and created schema in your HANA Cloud data lake instance and loaded data into it. You have also successfully combined HANA and HANA data lake data together in a single query.

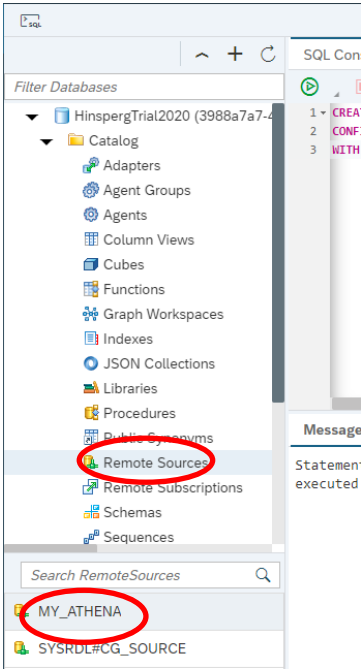
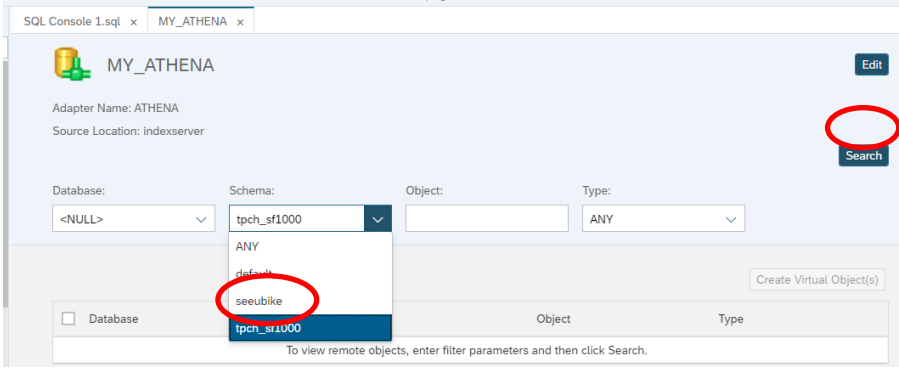
Exercise 4. Setting Up a Remote Source To Amazon Athena

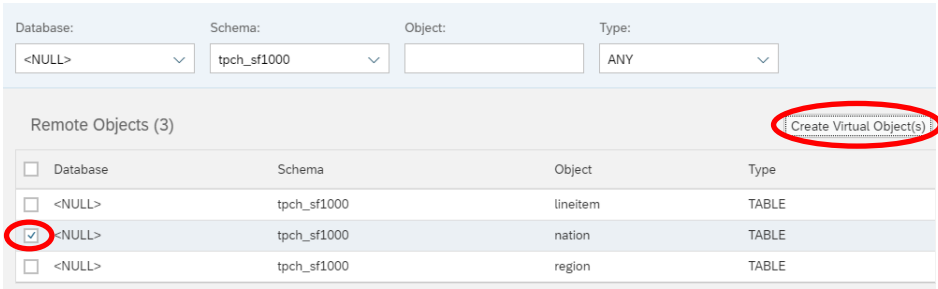
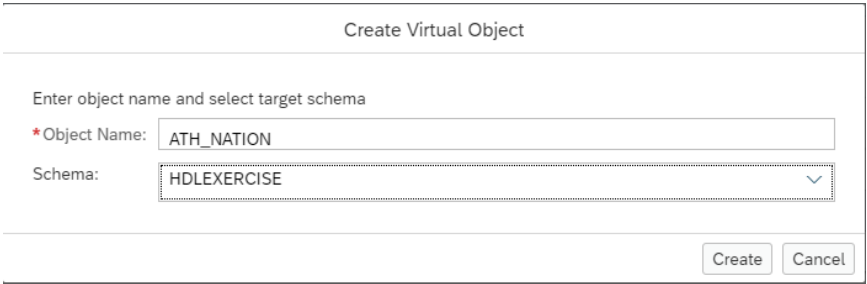
Amazon Athena is a serverless SQL engine which allows you to directly query files that are stored in S3 object storage. The goal of this exercise is to walk through the setup of a remote data source in HANA Cloud to Amazon Athena. We will be using the native SDA (Smart Data Access) support to create a connection to Amazon Athena and then query data in an Athena table from the SAP HANA Cloud Database Explorer. You will also replicate data from Athena into HANA Cloud in order to improve the performance of the Athena queries. A small Athena instance has been created for this hands on session which contains some schema and data from a well known dataset called TPCH.

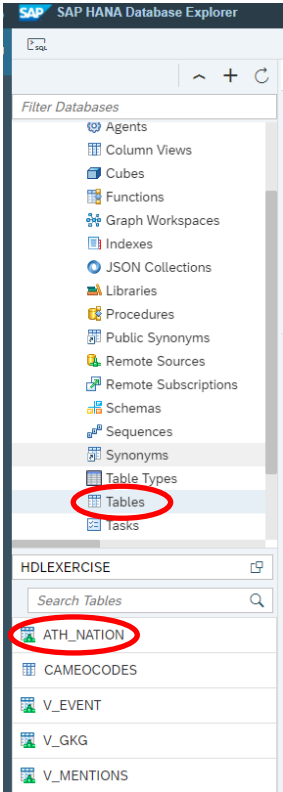
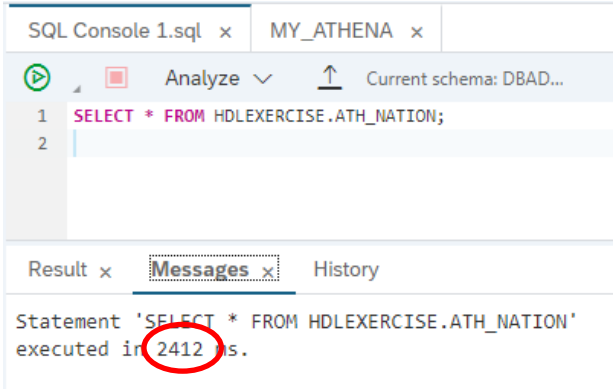
Explanation	Screenshot
<p>1.</p> <p>In order to create a remote source to Athena, we must first create 2 certificates in HANA Cloud to enable the connection. One certificate for connections to AWS, and one certificate for connections to S3, where the data for the Athena instance is stored. Open the file DAT161_AthenaCertsPSE.sql and copy the first 2 statements in the file to add the certificates to your HANA Cloud instance.</p>	 <pre> SQL Console 1.sql x Analyze Current schema: DBAD... Connected to: Hinsperg1 1 -- -- -- -- -- 2 CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE----- 3 MIIEEDZCAvegAwIBAgIBADANBgkqhkiG9w0BAQFADBoMQswCQYDVQGEwJVUzE1 4 MCMGA1UEChMcU3RhcncpZlVxckFRlY2hub2xvZ2llcywgSW5jLjEjYMDAGA1UECmMp 5 U3RhcncpZlVxckIENsYXNzIDIGQ2VydG1maWwHdG1vb1B8dXR0b3JpdHkwHhcNMjQw 6 NjI5MTczOTE2WmMzQWwNjI5MTczOTE2WjBoMQswCQYDVQGEwJVUzE1MCMGA1UE 7 ChMcU3RhcncpZlVxckFRlY2hub2xvZ2llcywgSW5jLjEjYMDAGA1UECmMpU3Rhcncp 8 ZlVxckIENsYXNzIDIGQ2VydG1maWwHdG1vb1B8dXR0b3JpdHkwHhcNMjQwNjI5 9 DQEBAAQAA4IBDQAwggEIAoIBAQC3Msj+6XGmBIWtDBFk385N78gDGIC/oav7PKaf 10 8M0h2tTYbitTkPskpD6E837oX+z1J0T1KKY/e97gKvDir1MvnsoFAZMej2YcOadN 11 +1q2cwQ1Zut3f+dZxkqZJRRU6ybH838Z1T8wJ6+wRir/respdefqgSho9T5iaU0 12 X9tdKyi22WY8sbi5gv2C0j4QyDvvBmVmpsZGD3/cVE8MC5fvj13c7Jd8mZDI1aa 13 K4UmkhynArPkPw2vChmCuDY96pzTNb08acr1zJ3o/W5NF4Azb15KXZnJHoe0nRrA 14 1W4TNSNe35tPe/W93bC6j67eA0cQmdrBNj41tpvi/JEoAgraE0d4HFMHCMB0G 15 A1UdDgQWBB5/X7fRzt0fhvRbVazc1xDCDqmISzCBkgYDVR0jBjIGKMGIGBBS/X7fR 16 zt0fhvRbVazc1xDCDqmISzCBkgYDVR0jBjIGKMGIGBBS/X7fRzt0fhvRbVazc1x 17 YXJmaWVsZC0BUZWNobm9sb2dpZXMsIE1uYy4xMjAwBGNVBA5TKVN0YXJmaWVsZC0B 18 bGFzc2cyAIE1cnRpb2ZlYXRpb24gQXV0aG9yaXR5gGgEAMAwGA1UdEQQFMAMBAF8w 19 DQYJKoZIhvcNAQEFBQADggEBAAdP4id0ckaVaGsaFPzWdqAYcaT1epoXkJKtv3 20 L7IezMdeatiDh6GX70k1PncGQVhiv45YuApnP+yz3SfMh81U+nLMPUx2A2IGud56D 21 eruix/U0F47ZEU0/CwqTRV/p2JdLiXTAAsGh1o+Re49L2L7ShZ3U0WixeDyLJ1 22 xy16paq8U4Zt3VekyvggQto8PT7dL5WXXp59fkdhMt1b71cZBDzI0fmgAKhynp 23 VSJYACp4xJDKVtHCN2MQWp18qj1IapBtJuh1b190TSrE9atvNziPTnNvT51cKEY 24 WQPJrSPnNvKtelttQKbfi3QBFgmh95DmK/D5fs4C8fF5Q=-----END CERTIFICATE-----' 25 COMMENT 'AWS'; 26 27 CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE----- 28 MIIEYzCCA0ugAwIBAgIQAYL4CY6i51a5GjsnhB+5rANBgkqhkiG9w0BAQsFAADBa 29 MQswCQYDVQGEwJ3RTE5MBAGA1UEChMjQmFsdG1tb3JlMRMwEQYDVQQL EwpDeWJ1 30 c1RydXN0MSIwIAYDVQDEQ1CYWw0aW1vcmUgQ31lZXJUCnVzdCB5S290MB4XDTE1 </pre> <p>Messages x History</p> <p>Statement 'CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE----- ...' executed in 4 ms.</p> <p>Statement 'CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE----- ...' executed in 4 ms.</p>
<p>2.</p> <p>Now we need to create a certificate collection (also known as a PSE – personal security environment) for these certificates that our remote source will refer to when we create the connection to Athena. Execute the following statement to create a new PSE called HTTPS:</p> <p>CREATE PSE HTTPS;</p>	 <pre> SQL Console 1.sql x Analyze Current schema: DBAD... Connected to: Hinsperg1 1 -- Create PSE HTTPS and get CERTIF 2 CREATE PSE HTTPS; 3 </pre> <p>Messages x History</p> <p>Statement 'CREATE PSE HTTPS' executed in 4 ms.</p>

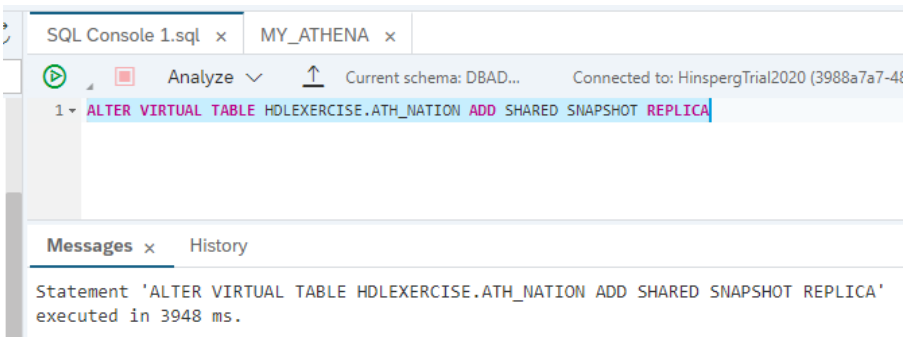
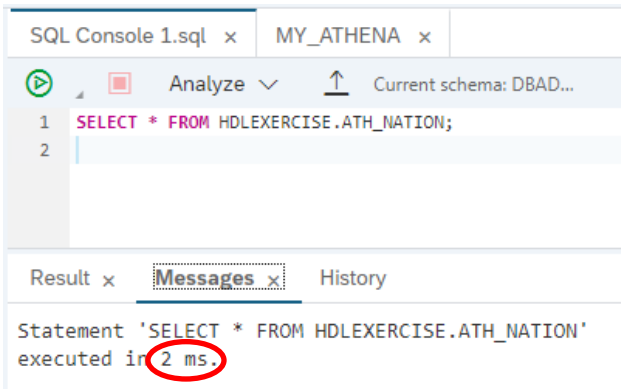
Explanation	Screenshot				
<p>3. Now we must add the certificates we created in step 1 to the PSE we created in step 2. To do that we must first lookup the id assigned to the AWS certificate when it was added to HANA. Execute the following statement to get the certificate id for the certificate created with the comment 'AWS':</p> <pre>SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';</pre> <p>Note the value of CERTIFICATE_ID as it will be used in the next step. It may not be the same as you see in the picture to the right.</p>	 <p>SQL Console 1.sql x</p> <p>Analyze v ↑ Current schema: DBAD... Connected to: Hinsp</p> <pre>1 -- Create PSE HTTPS and get CERTIFICATE_ID for AWS 2 SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';</pre> <p>Result x Messages x History</p> <p>Rows (1)</p> <table border="1"> <thead> <tr> <th></th> <th>CERTIFICATE_ID</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>157194</td> </tr> </tbody> </table>		CERTIFICATE_ID	1	157194
	CERTIFICATE_ID				
1	157194				
<p>4. Add the certificate id recorded in step 3 to the PSE created in step 2. Execute the following statement:</p> <pre>ALTER PSE HTTPS ADD CERTIFICATE <AWS CERT ID>;</pre> <p>Where <AWS CERT ID> is the CERTIFICATE_ID you wrote down from step 3.</p>	 <p>SQL Console 1.sql x</p> <p>Analyze v ↑ Current schema: DBAD... C</p> <pre>1 -- Alter PSE HTTPS to add certificate id 2 ALTER PSE HTTPS ADD CERTIFICATE 157194;</pre> <p>Messages x History</p> <p>Statement 'ALTER PSE HTTPS ADD CERTIFICATE 157194' executed in 19 ms.</p>				
<p>5. Lookup the id assigned to the S3 certificate when it was added to HANA. Execute the following statement to get the certificate id for the certificate created with the comment 'S3':</p> <pre>SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'S3';</pre> <p>Note the value of CERTIFICATE_ID as it will be used in the next step. It may not be the same as you see in the picture to the right.</p>	 <p>SQL Console 1.sql x</p> <p>Analyze v ↑ Current schema: DBAD... Connected to: Hir</p> <pre>1 -- Get CERTIFICATE_ID for S3 2 SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'S3';</pre> <p>Result x Messages x History</p> <p>Rows (1)</p> <table border="1"> <thead> <tr> <th></th> <th>CERTIFICATE_ID</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>157195</td> </tr> </tbody> </table>		CERTIFICATE_ID	1	157195
	CERTIFICATE_ID				
1	157195				

Explanation	Screenshot
<p>6. Add the certificate id recorded in step 5 to the PSE created in step 2. Execute the following statement:</p> <pre>ALTER PSE HTTPS ADD CERTIFICATE <S3 CERT ID>;</pre> <p>Where <S3 CERT ID> is the CERTIFICATE_ID you wrote down from step 5.</p>	 <p>The screenshot shows the SQL Console interface. The top bar indicates the current schema is DBAD... The SQL editor contains two lines: a comment '-- Alter PSE HTTPS to add certificate' and the statement 'ALTER PSE HTTPS ADD CERTIFICATE 157195;'. The Messages pane at the bottom shows the statement was executed in 5 ms.</p>
<p>7. Now that we have added the certificates to our certificate collection, we need to set the purpose of the collection so that HANA knows these certificates can be used to validate remote source connections. Execute the following statement:</p> <pre>SET PSE HTTPS PURPOSE REMOTE SOURCE;</pre>	 <p>The screenshot shows the SQL Console interface. The top bar indicates the current schema is DBAD... The SQL editor contains two lines: a comment '-- Set the PSE purpose to REMOTE SOURCE' and the statement 'SET PSE HTTPS PURPOSE REMOTE SOURCE;'. The Messages pane at the bottom shows the statement was executed in 4 ms.</p>
<p>8. Now we are ready to create our connection to Amazon Athena. Copy the CREATE REMOTE SOURCE statement from DAT161Exercise_Athena.sql and execute it.</p>	 <p>The screenshot shows the SQL Console interface. The top bar indicates the current schema is DBAD... and the connection is to HinspergTrial2020 (398). The SQL editor contains four lines: a comment '-- Create Remote Source', the statement 'CREATE REMOTE SOURCE MY_Athena ADAPTER athena', the configuration 'CONFIGURATION 'workgroup=primary;region=eu-central-1;'', and the credential 'WITH CREDENTIAL TYPE 'PASSWORD' USING 'user=IAZHKR77CS2SVUTSXX;password=uKpN3gLsw4pep''. The Messages pane at the bottom shows the statement was executed in 5 ms.</p>

Explanation	Screenshot
<p>9.</p> <p>Navigate to the remote sources folder of your instance. You should see a new remote source called "MY_ATHENA". Double click to open the Athena remote connection.</p>	
<p>10.</p> <p>Similar to the HANA data lake exercise, open the "schema" drop down list, and select "tpch_sf1000" as your schema. Then click the "Search" button to get a list of Athena tables.</p>	

Explanation	Screenshot
11. Click the checkbox next to the “nation” table, and then click the “Create Virtual Objects” button.	
12. Set the object name to ATH_NATION and select HDLEXERCISE as the schema name. Then click the “Create” button.	

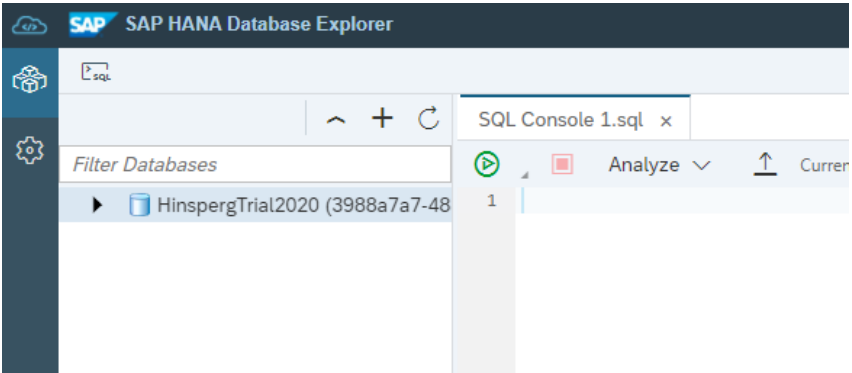
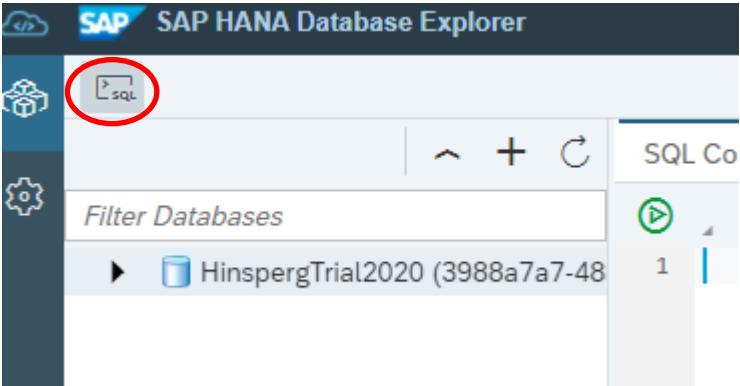
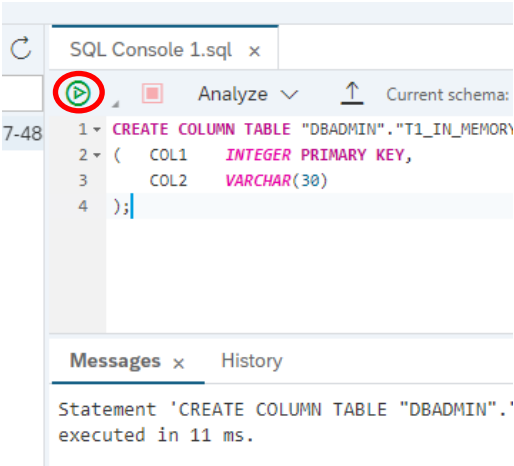
Explanation	Screenshot
<p>13.</p> <p>Navigate to the list of tables for your HDLEXERCISE schema in the left hand window. You should see the new ATH_NATION table.</p>	 <p>The screenshot shows the SAP HANA Database Explorer interface. On the left, a tree view lists various database objects. The 'Tables' item is highlighted with a red circle. Below this, the 'HDLEXERCISE' schema is selected, and a list of tables is displayed. The 'ATH_NATION' table is highlighted with a red circle. Other tables listed include CAMEOCODES, V_EVENT, V_GKG, and V_MENTIONS.</p>
<p>14.</p> <p>Move to the SQL window, and query the table from Athena by executing the following statement:</p> <pre>SELECT * FROM ATH_NATION;</pre> <p>When you execute the statement, HANA Cloud connects to the Athena instance, executes the statement and retrieves the results. Note the execution time for the statement.</p>	 <p>The screenshot shows the SQL Console interface. The top bar indicates the current schema is 'DBAD...'. The SQL statement 'SELECT * FROM HDLEXERCISE.ATH_NATION;' is entered in the console. Below the console, the 'Messages' tab is selected, showing the execution status: 'Statement 'SELECT * FROM HDLEXERCISE.ATH_NATION' executed in 2412 ms.' The execution time '2412' is highlighted with a red circle.</p>

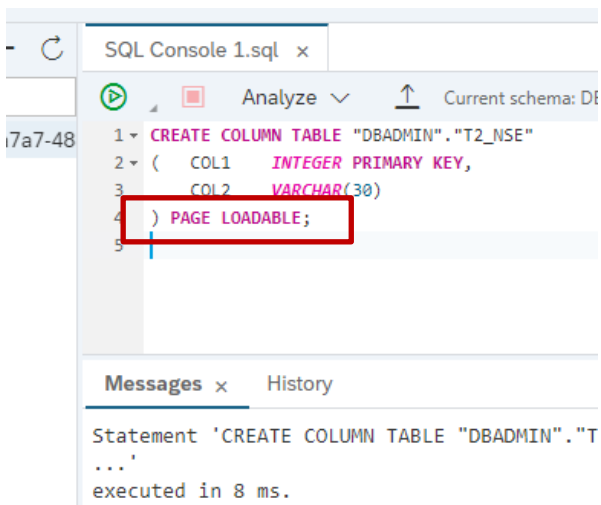
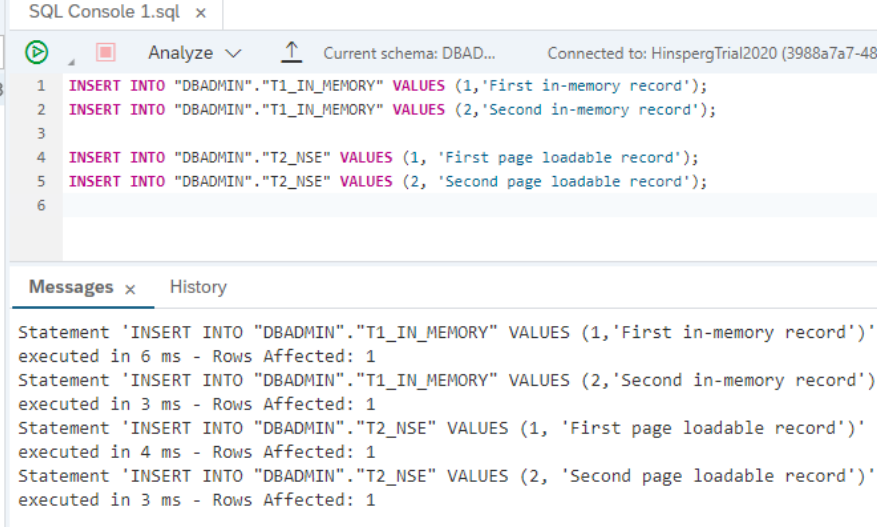
Explanation	Screenshot
<p>15. If we plan to use this table extensively within HANA Cloud, and it does change frequently, we can choose to replicate the data into HANA Cloud from Athena. To replicate the data for the Athena “nation” table into HANA Cloud, execute the following statement in the Database Explorer SQL console:</p> <pre>ALTER VIRTUAL TABLE HDLXERCISE.ATH_NATION ADD SHARED SNAPSHOT REPLICA;</pre>	 <p>SQL Console 1.sql x MY_ATHENA x</p> <p>Analyze v Current schema: DBAD... Connected to: HinspergTrial2020 (3988a7a7-48...</p> <p>1 ALTER VIRTUAL TABLE HDLXERCISE.ATH_NATION ADD SHARED SNAPSHOT REPLICA</p> <p>Messages x History</p> <p>Statement 'ALTER VIRTUAL TABLE HDLXERCISE.ATH_NATION ADD SHARED SNAPSHOT REPLICA' executed in 3948 ms.</p>
<p>16. Query the remote Athena table again by executing the same statement as in step 14:</p> <pre>SELECT * FROM ATH_NATION;</pre> <p>Note that the execution time has improved substantially for the statement.</p>	 <p>SQL Console 1.sql x MY_ATHENA x</p> <p>Analyze v Current schema: DBAD...</p> <p>1 SELECT * FROM HDLXERCISE.ATH_NATION;</p> <p>2</p> <p>Result x Messages x History</p> <p>Statement 'SELECT * FROM HDLXERCISE.ATH_NATION' executed in 2 ms.</p>

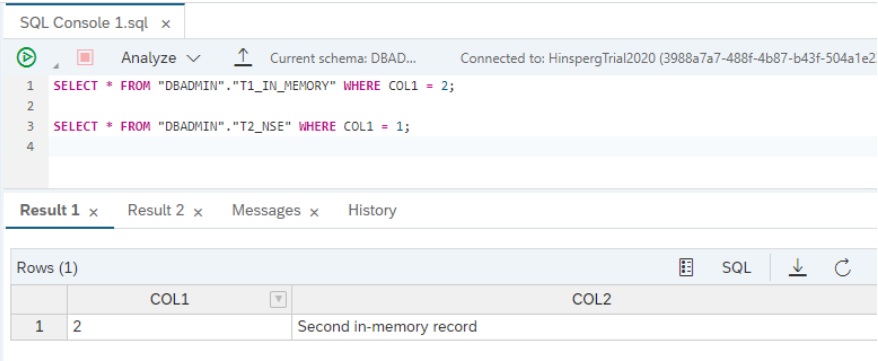
That concludes this exercise. You have successfully create a connection from HANA Cloud to Amazon Athena, create and queried a virtual table in Athena, and replicated data from Athena into HANA Cloud in order to improve query performance.

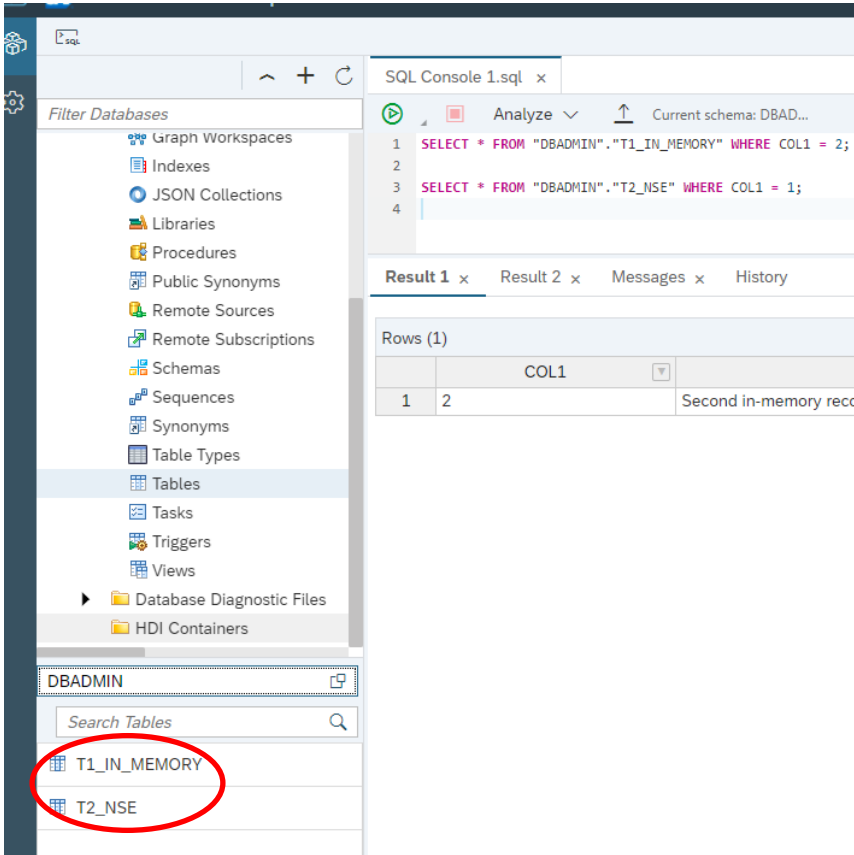
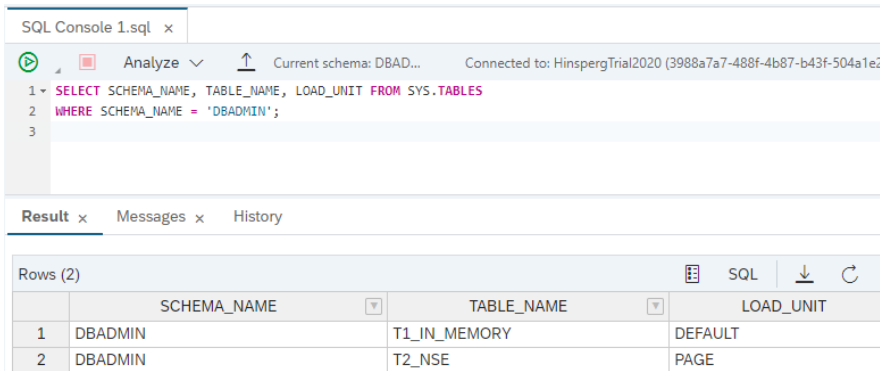
Exercise 5: Using NSE in HANA Cloud

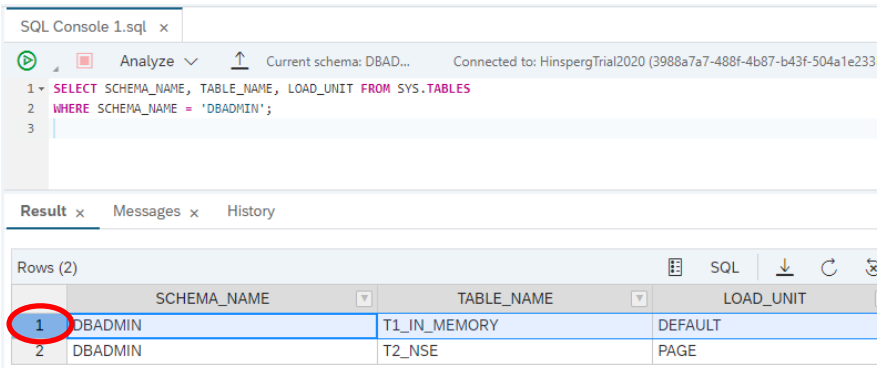
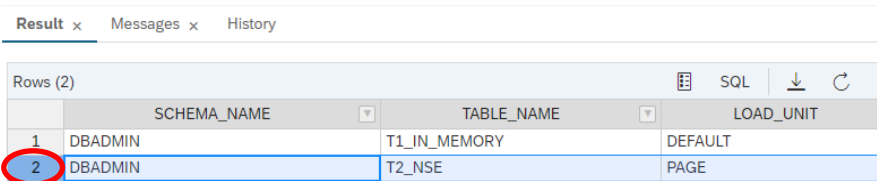
In this set of exercises you will explore the warm storage component of SAP HANA Cloud. This is called HANA native storage extension and you can find out more about it [here](#)

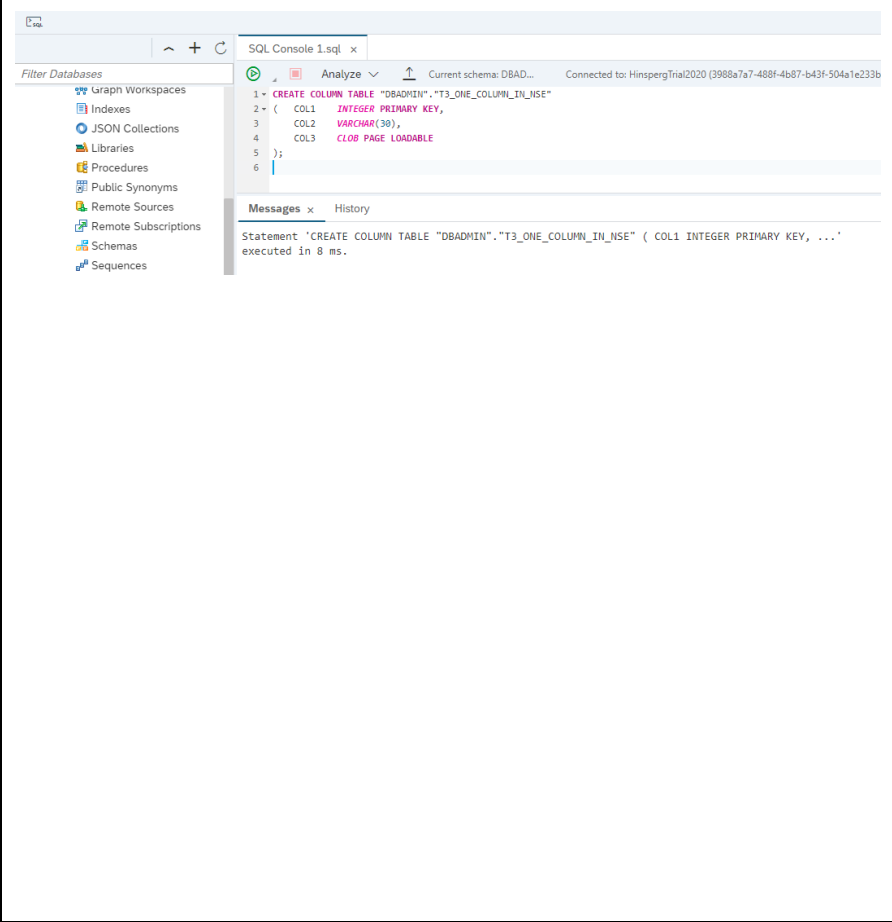
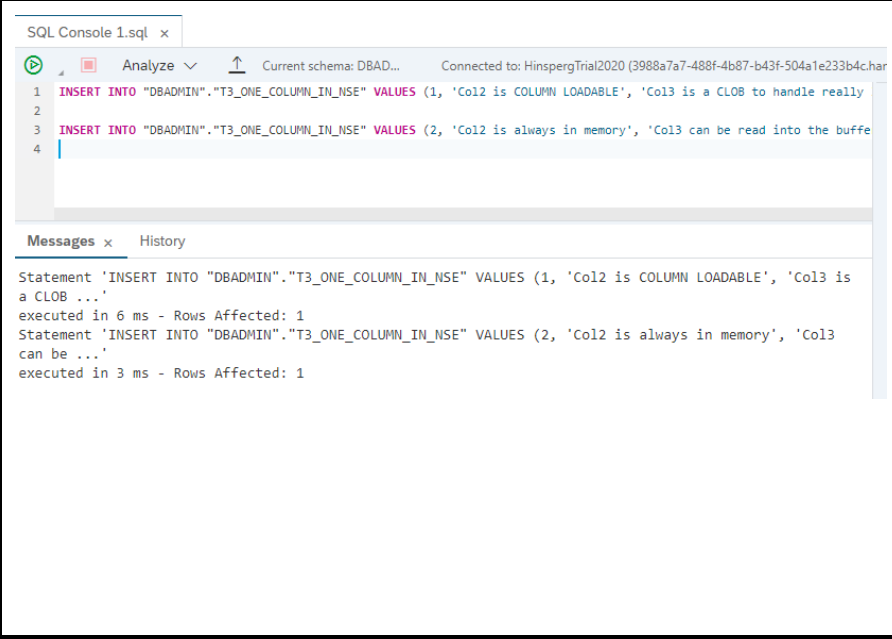
Explanation	Screenshot
<p>1.</p> <p>If it is not already open, open the Database Explorer for your HANA Cloud Instance. See Exercise 3, steps 1-3 if you are not familiar with how to do this.</p>	 <p>The screenshot shows the SAP HANA Database Explorer interface. The top bar displays the SAP logo and the title 'SAP HANA Database Explorer'. Below the top bar, there is a sidebar with icons for 'SQL', 'Filter Databases', and a database instance 'HinspergTrial2020 (3988a7a7-48)'. The main area shows the 'SQL Console 1.sql' tab with a green play button icon circled in red. The console is currently empty, showing only the number '1'.</p>
<p>2.</p> <p>With these first set of steps, you will create a table in NSE storage. Start by opening a SQL console by selecting the database and clicking on the SQL icon</p>	 <p>The screenshot shows the SAP HANA Database Explorer interface. The top bar displays the SAP logo and the title 'SAP HANA Database Explorer'. Below the top bar, there is a sidebar with icons for 'SQL', 'Filter Databases', and a database instance 'HinspergTrial2020 (3988a7a7-48)'. The main area shows the 'SQL Co' tab with a green play button icon circled in red. The console is currently empty, showing only the number '1'.</p>
<p>3.</p> <p>First create a table in HANA Cloud main memory by running the following statement:</p> <pre>CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30));</pre>	 <p>The screenshot shows the SAP HANA Database Explorer interface. The top bar displays the SAP logo and the title 'SAP HANA Database Explorer'. Below the top bar, there is a sidebar with icons for 'SQL', 'Filter Databases', and a database instance 'HinspergTrial2020 (3988a7a7-48)'. The main area shows the 'SQL Console 1.sql' tab with a green play button icon circled in red. The console contains the following SQL statement:</p> <pre>1 CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY" 2 (3 COL1 INTEGER 4 PRIMARY KEY, 5 COL2 VARCHAR(30) 6);</pre> <p>Below the console, there is a 'Messages' tab showing the execution result:</p> <pre>Statement 'CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY"' executed in 11 ms.</pre>

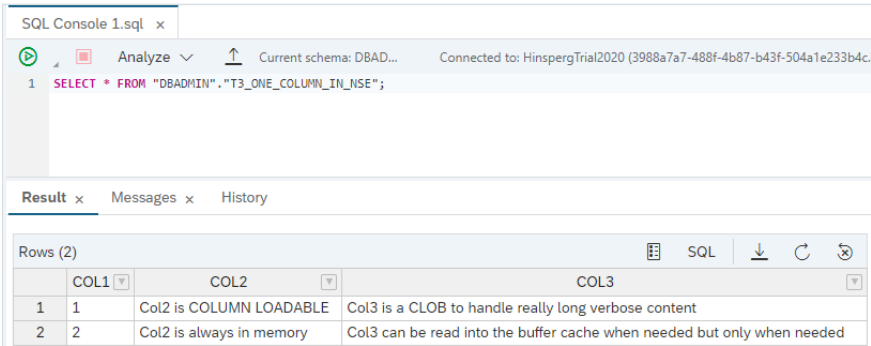
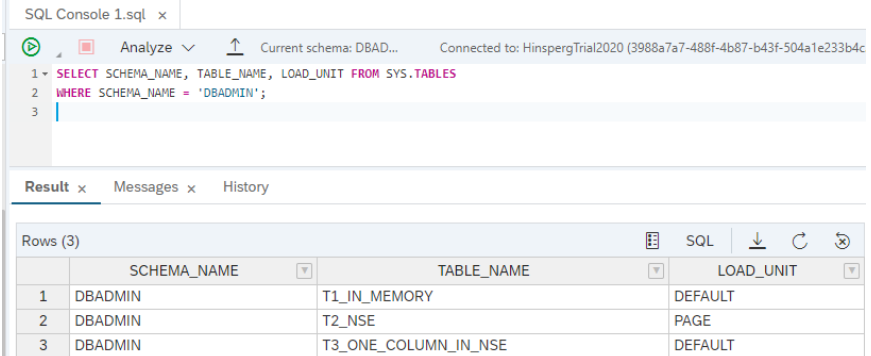
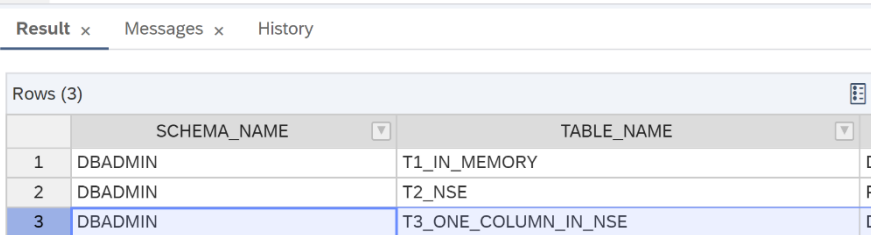
Explanation	Screenshot
<p>4.</p> <p>Now create an NSE table by first copying and pasting the following statement to the SQL console:</p> <pre>CREATE COLUMN TABLE "DBADMIN"."T2_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30)) PAGE LOADABLE;</pre> <p>Note: notice the only difference in syntax when creating an NSE table compared to an in-memory table is the PAGE LOADABLE clause</p>	 <p>The screenshot shows a SQL console window with the following SQL statement entered:</p> <pre>1 CREATE COLUMN TABLE "DBADMIN"."T2_NSE" 2 (COL1 INTEGER PRIMARY KEY, 3 COL2 VARCHAR(30) 4) PAGE LOADABLE; 5</pre> <p>The statement is highlighted with a red box. Below the console, the Messages pane shows the execution result:</p> <pre>Statement 'CREATE COLUMN TABLE "DBADMIN"."T ...' executed in 8 ms.</pre>
<p>5.</p> <p>Insert values into both tables by running the following statements:</p> <pre>INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (1,'First in-memory record'); INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (2,'Second in-memory record'); INSERT INTO "DBADMIN"."T2_NSE" VALUES (1, 'First page loadable record'); INSERT INTO "DBADMIN"."T2_NSE" VALUES (2, 'Second page loadable record');</pre> <p>Note: there is no syntax difference between the INSERT INTO statements. Similarly, no extra syntax is required for any DML operations (UPDATE and DELETE) on page loadable data.</p>	 <p>The screenshot shows a SQL console window with the following SQL statements entered:</p> <pre>1 INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (1,'First in-memory record'); 2 INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (2,'Second in-memory record'); 3 4 INSERT INTO "DBADMIN"."T2_NSE" VALUES (1, 'First page loadable record'); 5 INSERT INTO "DBADMIN"."T2_NSE" VALUES (2, 'Second page loadable record'); 6</pre> <p>Below the console, the Messages pane shows the execution results for each statement:</p> <pre>Statement 'INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (1,'First in-memory record')' executed in 6 ms - Rows Affected: 1 Statement 'INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (2,'Second in-memory record')' executed in 3 ms - Rows Affected: 1 Statement 'INSERT INTO "DBADMIN"."T2_NSE" VALUES (1, 'First page loadable record')' executed in 4 ms - Rows Affected: 1 Statement 'INSERT INTO "DBADMIN"."T2_NSE" VALUES (2, 'Second page loadable record')' executed in 3 ms - Rows Affected: 1</pre>

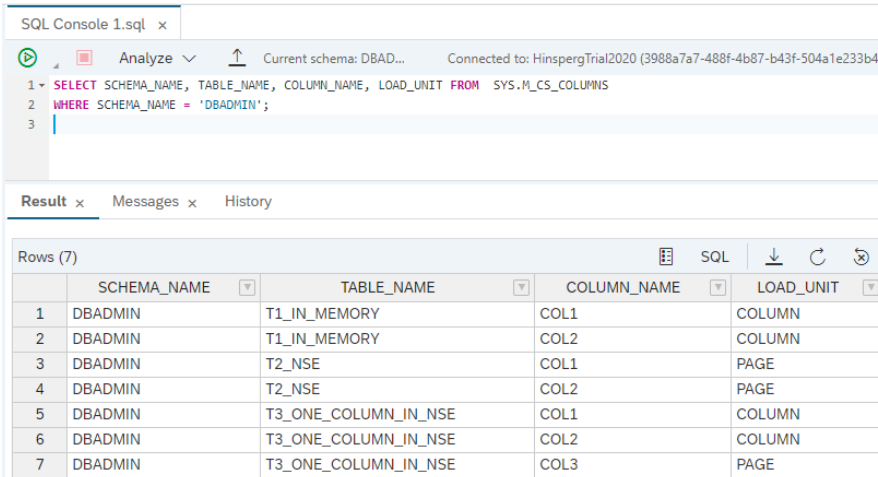
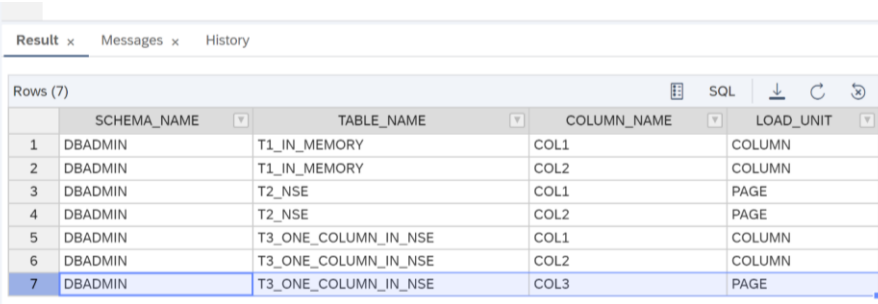
Explanation	Screenshot				
<p>6.</p> <p>Query the tables by running the following SELECT statements:</p> <p>SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;</p> <p>SELECT * FROM "DBADMIN"."T2_NSE" WHERE COL1 = 1;</p> <p>Note: The result from the first query is returned under the tab “Result 1” and the result from the second query is returned under the tab “Result 2”.</p>	 <p>The screenshot shows a SQL Console interface with two queries executed. The first query, 'SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;', is shown in the editor. The second query, 'SELECT * FROM "DBADMIN"."T2_NSE" WHERE COL1 = 1;', is also shown. Below the queries, the results are displayed in two tabs: 'Result 1' and 'Result 2'. 'Result 1' shows a single row with COL1 = 2 and COL2 = 'Second in-memory record'. 'Result 2' is currently empty.</p> <table><thead><tr><th>COL1</th><th>COL2</th></tr></thead><tbody><tr><td>2</td><td>Second in-memory record</td></tr></tbody></table>	COL1	COL2	2	Second in-memory record
COL1	COL2				
2	Second in-memory record				

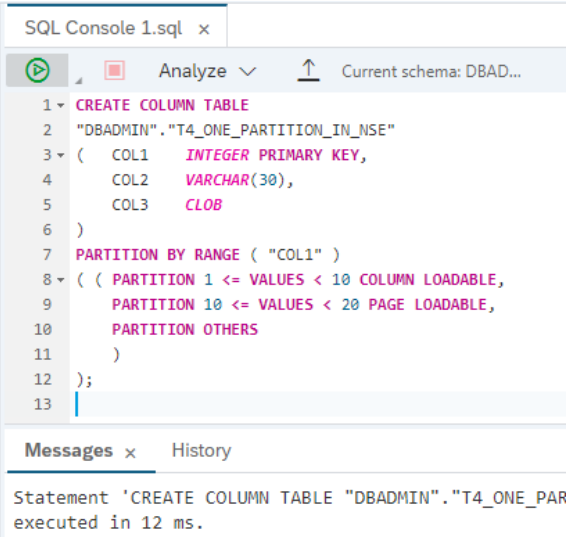
Explanation	Screenshot									
<p>7.</p> <p>The tables can also be viewed from the UI by navigating to Catalog and then Tables</p>	 <p>The screenshot shows the SQL console interface. On the left, the 'Filter Databases' pane is open, showing a list of database objects. The 'Tables' category is selected, and the 'DBADMIN' database is chosen. The 'Search Tables' field is empty. Below the search field, the tables 'T1_IN_MEMORY' and 'T2_NSE' are listed. A red circle highlights these two tables. On the right, the SQL console shows a query with two statements: 'SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;' and 'SELECT * FROM "DBADMIN"."T2_NSE" WHERE COL1 = 1;'. The results are displayed in a table with two columns: 'COL1' and 'Second in-memory reco'. The first row shows the value '2'.</p> <table><tr><th>COL1</th><th>Second in-memory reco</th></tr><tr><td>2</td><td></td></tr></table>	COL1	Second in-memory reco	2						
COL1	Second in-memory reco									
2										
<p>8.</p> <p>Lets look at the load configuration for the tables we just created.</p> <p>Clear the SQL console. Then copy and paste the following statement into the console. Run the statement.</p> <p>SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES WHERE SCHEMA_NAME = 'DBADMIN';</p> <p>Note: for this exercise, we are specifically looking at the SCHEMA_NAME, TABLE_NAME, and LOAD_UNIT columns in the SYS.TABLES view.</p>	 <p>The screenshot shows the SQL console interface. The SQL console shows a query with two statements: 'SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES' and 'WHERE SCHEMA_NAME = 'DBADMIN';'. The results are displayed in a table with three columns: 'SCHEMA_NAME', 'TABLE_NAME', and 'LOAD_UNIT'. The first row shows the values 'DBADMIN', 'T1_IN_MEMORY', and 'DEFAULT'. The second row shows the values 'DBADMIN', 'T2_NSE', and 'PAGE'.</p> <table><tr><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>DEFAULT</td></tr><tr><td>DBADMIN</td><td>T2_NSE</td><td>PAGE</td></tr></table>	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	DBADMIN	T1_IN_MEMORY	DEFAULT	DBADMIN	T2_NSE	PAGE
SCHEMA_NAME	TABLE_NAME	LOAD_UNIT								
DBADMIN	T1_IN_MEMORY	DEFAULT								
DBADMIN	T2_NSE	PAGE								

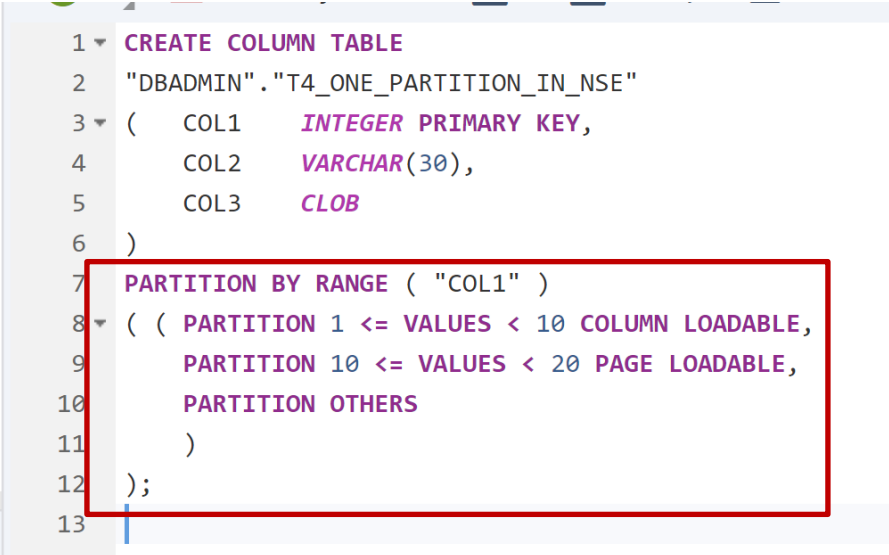
Explanation	Screenshot												
<p>9.</p> <p>Click on row “1”.</p> <p>Since we did not explicitly assign a LOAD UNIT in the CREATE TABLE statement for the T1_IN_MEMORY table, it shows a LOAD UNIT of DEFAULT. The default LOAD UNIT for a table is COLUMN. If we had explicitly specified COLUMN LOADABLE in the CREATE TABLE statement for the T1_IN_MEMORY table, then the LOAD UNIT would show as COLUMN.</p>	 <p>SQL Console 1.sql x</p> <p>Analyze Current schema: DBAD... Connected to: HinspergTrial2020 (3988a7a7-488f-4b87-b43f-504a1e233l)</p> <pre>1 SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES 2 WHERE SCHEMA_NAME = 'DBADMIN'; 3</pre> <p>Result x Messages x History</p> <p>Rows (2)</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>DEFAULT</td></tr><tr><td>2</td><td>DBADMIN</td><td>T2_NSE</td><td>PAGE</td></tr></table>		SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T1_IN_MEMORY	DEFAULT	2	DBADMIN	T2_NSE	PAGE
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT										
1	DBADMIN	T1_IN_MEMORY	DEFAULT										
2	DBADMIN	T2_NSE	PAGE										
<p>10.</p> <p>click on row “2”.</p> <p>The LOAD UNIT for the T2_NSE table is reported as PAGE indicating that this is a page loadable table.</p>	 <p>Result x Messages x History</p> <p>Rows (2)</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>DEFAULT</td></tr><tr><td>2</td><td>DBADMIN</td><td>T2_NSE</td><td>PAGE</td></tr></table>		SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T1_IN_MEMORY	DEFAULT	2	DBADMIN	T2_NSE	PAGE
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT										
1	DBADMIN	T1_IN_MEMORY	DEFAULT										
2	DBADMIN	T2_NSE	PAGE										

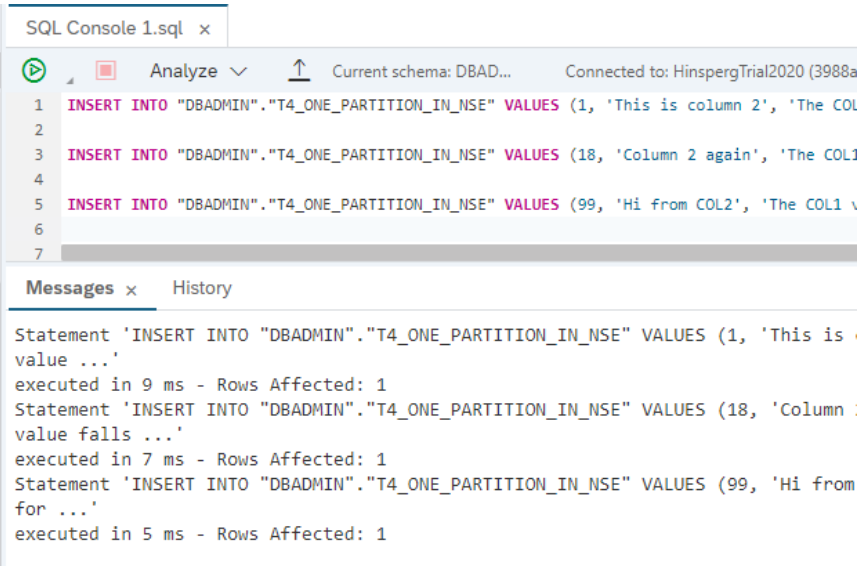
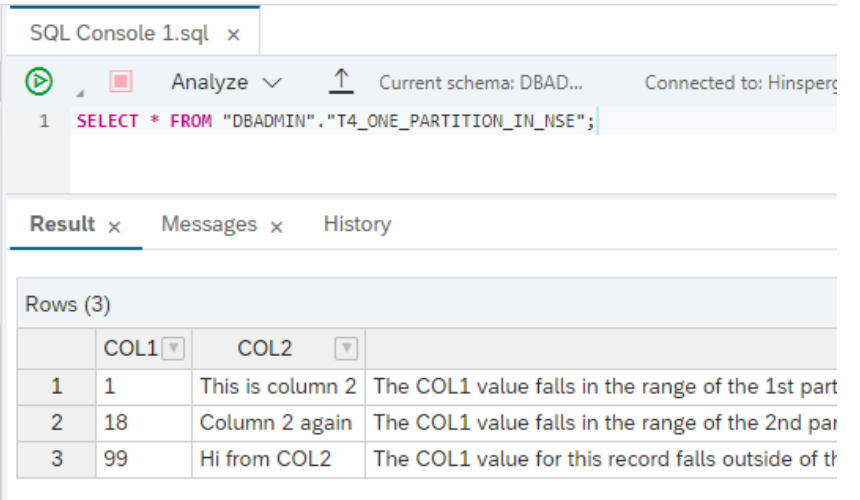
Explanation	Screenshot
<p>11.</p> <p>Now lets' create a table with a specific column that is page loadable. Clear the SQL console and run the following statement:</p> <pre>CREATE COLUMN TABLE "DBADMIN"."T3_ONE_COLUMN_ IN_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30), COL3 CLOB PAGE LOADABLE);</pre> <p>Note that the first 2 column definitions are the same as we used in the T1 and T2 tables. For our T3 table, a third column has been added to the table definition and specifically designated that column to be PAGE LOADABLE. Note the location of the PABE LOADABLE clause as part of the column definition. There is no PAGE LOADABLE clause after the column definitions. This means that the table will use the DEFAULT load unit (which is COLUMN LOADABLE) for any columns that are not explicitly set to be PAGE LOADABLE.</p>	
<p>12.</p> <p>Now run the following statement to insert values into T3_ONE_COLUMN_IN_NSE:</p> <pre>INSERT INTO "DBADMIN"."T3_ONE_COLUMN_ IN_NSE" VALUES (1, 'Col2 is COLUMN LOADABLE', 'Col3 is a CLOB to handle really long verbose content');</pre> <pre>INSERT INTO "DBADMIN"."T3_ONE_COLUMN_ IN_NSE" VALUES (2, 'Col2 is always in memory', 'Col3 can be read into the buffer cache when needed but only when needed');</pre>	

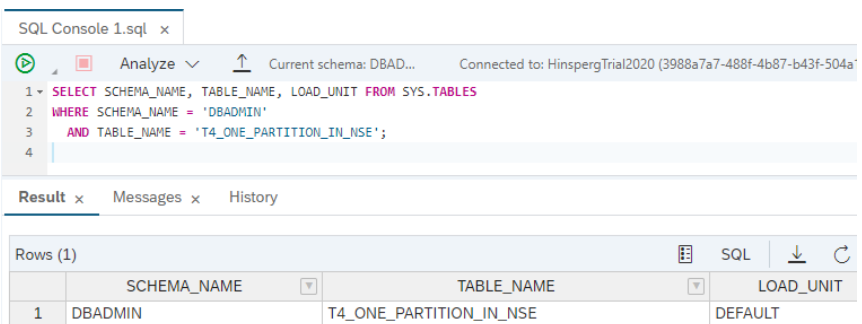
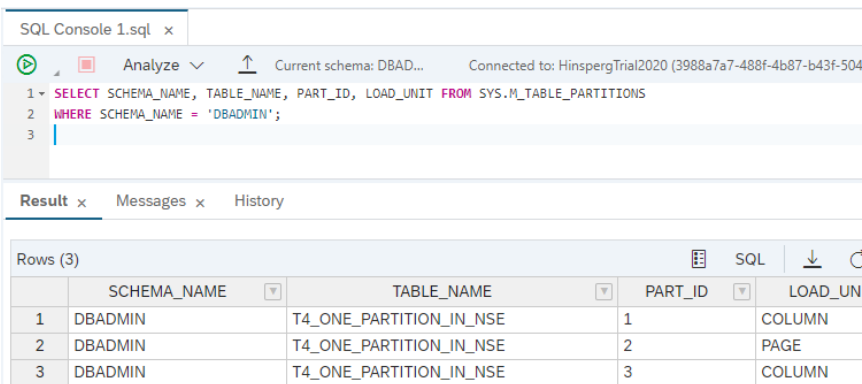
Explanation	Screenshot																
<p>13.</p> <p>Query the table by running the following statement:</p> <pre>SELECT * FROM "DBADMIN"."T3_ONE_COLUMN_IN_NSE";</pre>	 <p>SQL Console 1.sql x</p> <p>Analyze Current schema: DBAD... Connected to: HinspergTrial2020 (3988a7a7-488f-4b87-b43f-504a1e233b4c)</p> <pre>1 SELECT * FROM "DBADMIN"."T3_ONE_COLUMN_IN_NSE";</pre> <p>Result x Messages x History</p> <p>Rows (2)</p> <table><tr><th></th><th>COL1</th><th>COL2</th><th>COL3</th></tr><tr><td>1</td><td>1</td><td>Col2 is COLUMN LOADABLE</td><td>Col3 is a CLOB to handle really long verbose content</td></tr><tr><td>2</td><td>2</td><td>Col2 is always in memory</td><td>Col3 can be read into the buffer cache when needed but only when needed</td></tr></table>		COL1	COL2	COL3	1	1	Col2 is COLUMN LOADABLE	Col3 is a CLOB to handle really long verbose content	2	2	Col2 is always in memory	Col3 can be read into the buffer cache when needed but only when needed				
	COL1	COL2	COL3														
1	1	Col2 is COLUMN LOADABLE	Col3 is a CLOB to handle really long verbose content														
2	2	Col2 is always in memory	Col3 can be read into the buffer cache when needed but only when needed														
<p>14.</p> <p>Now lets' look at the load configuration for the individual columns.</p> <p>Query the SYS.TABLES view to see what LOAD UNIT is configured at the table level by running the following statement:</p> <pre>SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES WHERE SCHEMA_NAME = 'DBADMIN';</pre>	 <p>SQL Console 1.sql x</p> <p>Analyze Current schema: DBAD... Connected to: HinspergTrial2020 (3988a7a7-488f-4b87-b43f-504a1e233b4c)</p> <pre>1 SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES 2 WHERE SCHEMA_NAME = 'DBADMIN'; 3</pre> <p>Result x Messages x History</p> <p>Rows (3)</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>DEFAULT</td></tr><tr><td>2</td><td>DBADMIN</td><td>T2_NSE</td><td>PAGE</td></tr><tr><td>3</td><td>DBADMIN</td><td>T3_ONE_COLUMN_IN_NSE</td><td>DEFAULT</td></tr></table>		SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T1_IN_MEMORY	DEFAULT	2	DBADMIN	T2_NSE	PAGE	3	DBADMIN	T3_ONE_COLUMN_IN_NSE	DEFAULT
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT														
1	DBADMIN	T1_IN_MEMORY	DEFAULT														
2	DBADMIN	T2_NSE	PAGE														
3	DBADMIN	T3_ONE_COLUMN_IN_NSE	DEFAULT														
<p>15.</p> <p>Notice the LOAD UNIT for the T3_ONE_COLUMN_IN_NSE table is shown as "DEFAULT" (which is equivalent to COLUMN). If any 1 column in a table is COLUMN LOADABLE, then the LOAD UNIT for the table will be reported as DEFAULT or COLUMN</p>	 <p>Result x Messages x History</p> <p>Rows (3)</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>DEFAULT</td></tr><tr><td>2</td><td>DBADMIN</td><td>T2_NSE</td><td>PAGE</td></tr><tr><td>3</td><td>DBADMIN</td><td>T3_ONE_COLUMN_IN_NSE</td><td>DEFAULT</td></tr></table>		SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T1_IN_MEMORY	DEFAULT	2	DBADMIN	T2_NSE	PAGE	3	DBADMIN	T3_ONE_COLUMN_IN_NSE	DEFAULT
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT														
1	DBADMIN	T1_IN_MEMORY	DEFAULT														
2	DBADMIN	T2_NSE	PAGE														
3	DBADMIN	T3_ONE_COLUMN_IN_NSE	DEFAULT														

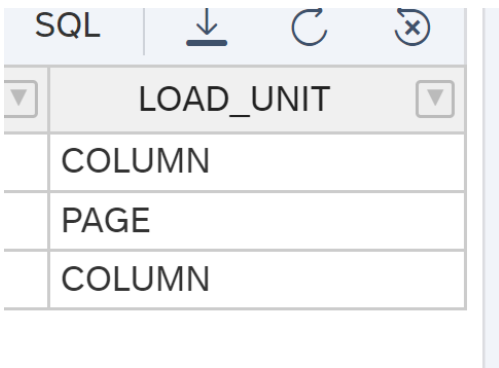
Explanation	Screenshot																																								
<p>16.</p> <p>To see the LOAD UNIT configuration at a column level, run the following statement:</p> <p>SELECT SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, LOAD_UNIT FROM SYS.M_CS_COLUMNS WHERE SCHEMA_NAME = 'DBADMIN';</p>	 <p>The screenshot shows a SQL console window with the following SQL query: <code>SELECT SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, LOAD_UNIT FROM SYS.M_CS_COLUMNS WHERE SCHEMA_NAME = 'DBADMIN';</code> The results are displayed in a table with 7 rows. The columns are SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, and LOAD_UNIT. The results are as follows:</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>COLUMN_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>COL1</td><td>COLUMN</td></tr><tr><td>2</td><td>DBADMIN</td><td>T1_IN_MEMORY</td><td>COL2</td><td>COLUMN</td></tr><tr><td>3</td><td>DBADMIN</td><td>T2_NSE</td><td>COL1</td><td>PAGE</td></tr><tr><td>4</td><td>DBADMIN</td><td>T2_NSE</td><td>COL2</td><td>PAGE</td></tr><tr><td>5</td><td>DBADMIN</td><td>T3_ONE_COLUMN_IN_NSE</td><td>COL1</td><td>COLUMN</td></tr><tr><td>6</td><td>DBADMIN</td><td>T3_ONE_COLUMN_IN_NSE</td><td>COL2</td><td>COLUMN</td></tr><tr><td>7</td><td>DBADMIN</td><td>T3_ONE_COLUMN_IN_NSE</td><td>COL3</td><td>PAGE</td></tr></table>		SCHEMA_NAME	TABLE_NAME	COLUMN_NAME	LOAD_UNIT	1	DBADMIN	T1_IN_MEMORY	COL1	COLUMN	2	DBADMIN	T1_IN_MEMORY	COL2	COLUMN	3	DBADMIN	T2_NSE	COL1	PAGE	4	DBADMIN	T2_NSE	COL2	PAGE	5	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL1	COLUMN	6	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL2	COLUMN	7	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL3	PAGE
	SCHEMA_NAME	TABLE_NAME	COLUMN_NAME	LOAD_UNIT																																					
1	DBADMIN	T1_IN_MEMORY	COL1	COLUMN																																					
2	DBADMIN	T1_IN_MEMORY	COL2	COLUMN																																					
3	DBADMIN	T2_NSE	COL1	PAGE																																					
4	DBADMIN	T2_NSE	COL2	PAGE																																					
5	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL1	COLUMN																																					
6	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL2	COLUMN																																					
7	DBADMIN	T3_ONE_COLUMN_IN_NSE	COL3	PAGE																																					
<p>17.</p> <p>Note the LOAD UNIT for the third column is shown as PAGE because that column was specifically designated as PAGE LOADABLE in the CREATE TABLE statement in Exercise 2 Step 1.</p> <p>Also note that the LOAD UNIT for the first 2 columns of T3_ONE_COLUMN_IN_NSE (row 6 and 7) are shown as COLUMN because that is the default LOAD UNIT and no other LOAD UNIT was specified for these columns in the CREATE TABLE statement</p>	 <p>The screenshot shows the same SQL console window as above, but with the 7th row of the results table highlighted. The 7th row is: DBADMIN, T3_ONE_COLUMN_IN_NSE, COL3, PAGE.</p>																																								

Explanation	Screenshot
<p>18. Now lets create a table which has a specific partition stored in NSE.</p> <p>Run the following statement to create a table:</p> <pre>CREATE COLUMN TABLE "DBADMIN"."T4_ONE_PARTITIO N_IN_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30), COL3 CLOB) PARTITION BY RANGE ("COL1") ((PARTITION 1 <= VALUES < 10 COLUMN LOADABLE, PARTITION 10 <= VALUES < 20 PAGE LOADABLE, PARTITION OTHERS));</pre> <p>Note: The column definitions are the same as the columns created in T3, but since no LOAD UNIT is specified, all the columns will be assigned the default load unit of COLUMN.</p> <p>The new CREATE TABLE clause that is being used for the T4 table is the PARTITION BY RANGE clause. This defines a single level range partitioning based on the values in COL1</p>	 <p>The screenshot shows a SQL console window titled 'SQL Console 1.sql'. The code being executed is a 'CREATE COLUMN TABLE' statement for 'DBADMIN"."T4_ONE_PARTITION_IN_NSE'. The table has three columns: COL1 (INTEGER PRIMARY KEY), COL2 (VARCHAR(30)), and COL3 (CLOB). It is partitioned by RANGE on COL1 with three partitions: PARTITION 1 (VALUES < 10, COLUMN LOADABLE), PARTITION 10 (VALUES < 20, PAGE LOADABLE), and PARTITION OTHERS. The console shows the statement executed successfully in 12 ms.</p> <pre>1 CREATE COLUMN TABLE 2 "DBADMIN"."T4_ONE_PARTITION_IN_NSE" 3 (4 COL1 INTEGER PRIMARY KEY, 5 COL2 VARCHAR(30), 6 COL3 CLOB 7) 8 PARTITION BY RANGE ("COL1") 9 ((PARTITION 1 <= VALUES < 10 COLUMN LOADABLE, 10 PARTITION 10 <= VALUES < 20 PAGE LOADABLE, 11 PARTITION OTHERS 12) 13);</pre> <p>Messages x History</p> <p>Statement 'CREATE COLUMN TABLE "DBADMIN"."T4_ONE_PAR' executed in 12 ms.</p>

Explanation	Screenshot
<p>19.</p> <p>Note: In the previous SQL statement, the first partition will hold records where the VALUES of COL1 range from 1 to 9 (COL1 is an INTEGER column). This partition is explicitly assigned to be stored in-memory as COLUMN LOADABLE.</p> <p>The second partition will hold records where the VALUES of COL1 range from 10 to 19. This partition is explicitly assigned to be stored on disk as PAGE LOADABLE.</p> <p>The 3rd partition is the 'catch-all' partition OTHERS. The OTHERS partition is not assigned records in a specific range. Instead the OTHERS partition will hold any records where the value of COL1 falls outside of any of the other partitions. And since no LOAD UNIT is specified for the OTHERS partition, it will be assigned the DEFAULT load unit of COLUMN.</p>	 <pre>1 CREATE COLUMN TABLE 2 "DBADMIN"."T4_ONE_PARTITION_IN_NSE" 3 (COL1 INTEGER PRIMARY KEY, 4 COL2 VARCHAR(30), 5 COL3 CLOB 6) 7 PARTITION BY RANGE ("COL1") 8 ((PARTITION 1 <= VALUES < 10 COLUMN LOADABLE, 9 PARTITION 10 <= VALUES < 20 PAGE LOADABLE, 10 PARTITION OTHERS 11) 12); 13</pre>

Explanation	Screenshot
<p>20.</p> <p>Insert a record into each of the 3 partitions by running the following statement:</p> <p>INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (1, 'This is column 2', 'The COL1 value falls in the range of the 1st partition so this record will be in a COLUMN LOADABLE partition. ');</p> <p>INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (18, 'Column 2 again', 'The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE partition');</p> <p>INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (99, 'Hi from COL2', 'The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this record will be in the catch-all OTHERS partition');</p>	 <p>The screenshot shows a SQL console window with three INSERT statements and their execution results. The statements are: 1. INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (1, 'This is column 2', 'The COL1 value falls in the range of the 1st partition so this record will be in a COLUMN LOADABLE partition. '); 2. INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (18, 'Column 2 again', 'The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE partition'); 3. INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (99, 'Hi from COL2', 'The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this record will be in the catch-all OTHERS partition');</p> <p>The execution results show that each statement was executed successfully, with 1 row affected in each case. The execution times were 9 ms, 7 ms, and 5 ms respectively.</p>
<p>21.</p> <p>Select from the table by running the following statement:</p> <p>SELECT * FROM "DBADMIN"."T4_ONE_PARTITION_IN_NSE";</p>	 <p>The screenshot shows a SQL console window with a SELECT statement and its result set. The statement is: SELECT * FROM "DBADMIN"."T4_ONE_PARTITION_IN_NSE";</p> <p>The result set shows 3 rows, with columns COL1, COL2, and a description of the partition the record falls into. The rows are: 1. COL1: 1, COL2: This is column 2, Description: The COL1 value falls in the range of the 1st partition; 2. COL1: 18, COL2: Column 2 again, Description: The COL1 value falls in the range of the 2nd partition; 3. COL1: 99, COL2: Hi from COL2, Description: The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this record will be in the catch-all OTHERS partition;</p>

Explanation	Screenshot																				
<p>22.</p> <p>Review the table level LOAD UNIT for the T4 table by running the following statement:</p> <pre>SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES WHERE SCHEMA_NAME = 'DBADMIN' AND TABLE_NAME = 'T4_ONE_PARTITION_IN_NSE';</pre> <p>Note: Just as with the T1 and T3 tables, the LOAD UNIT for the T4_ONE_PARTITION_IN_NSE table is reported as DEFAULT because no LOAD UNIT was explicitly set for overall table. Remember that DEFAULT is equivalent to COLUMN.</p>	 <p>The screenshot shows the SQL Console interface with the following query and results:</p> <pre>1 SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES 2 WHERE SCHEMA_NAME = 'DBADMIN' 3 AND TABLE_NAME = 'T4_ONE_PARTITION_IN_NSE'; 4</pre> <p>The results table shows one row:</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>LOAD_UNIT</th></tr><tr><td>1</td><td>DBADMIN</td><td>T4_ONE_PARTITION_IN_NSE</td><td>DEFAULT</td></tr></table>		SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T4_ONE_PARTITION_IN_NSE	DEFAULT												
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT																		
1	DBADMIN	T4_ONE_PARTITION_IN_NSE	DEFAULT																		
<p>23.</p> <p>Review the details of individual table partitions, including the LOAD UNIT configuration by running the following statement:</p> <pre>SELECT SCHEMA_NAME, TABLE_NAME, PART_ID, LOAD_UNIT FROM SYS.M_TABLE_PARTITIONS WHERE SCHEMA_NAME = 'DBADMIN';</pre> <p>Note: Since the only partitioned table in your schema is the T4_ONE_PARTITION_IN_NSE table, you will only see results for that table. Partition IDs are assigned in the order that the PARTITION definitions appeared in the CREATE TABLE statement.</p>	 <p>The screenshot shows the SQL Console interface with the following query and results:</p> <pre>1 SELECT SCHEMA_NAME, TABLE_NAME, PART_ID, LOAD_UNIT FROM SYS.M_TABLE_PARTITIONS 2 WHERE SCHEMA_NAME = 'DBADMIN'; 3</pre> <p>The results table shows three rows:</p> <table><tr><th></th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>PART_ID</th><th>LOAD_UN</th></tr><tr><td>1</td><td>DBADMIN</td><td>T4_ONE_PARTITION_IN_NSE</td><td>1</td><td>COLUMN</td></tr><tr><td>2</td><td>DBADMIN</td><td>T4_ONE_PARTITION_IN_NSE</td><td>2</td><td>PAGE</td></tr><tr><td>3</td><td>DBADMIN</td><td>T4_ONE_PARTITION_IN_NSE</td><td>3</td><td>COLUMN</td></tr></table>		SCHEMA_NAME	TABLE_NAME	PART_ID	LOAD_UN	1	DBADMIN	T4_ONE_PARTITION_IN_NSE	1	COLUMN	2	DBADMIN	T4_ONE_PARTITION_IN_NSE	2	PAGE	3	DBADMIN	T4_ONE_PARTITION_IN_NSE	3	COLUMN
	SCHEMA_NAME	TABLE_NAME	PART_ID	LOAD_UN																	
1	DBADMIN	T4_ONE_PARTITION_IN_NSE	1	COLUMN																	
2	DBADMIN	T4_ONE_PARTITION_IN_NSE	2	PAGE																	
3	DBADMIN	T4_ONE_PARTITION_IN_NSE	3	COLUMN																	

Explanation	Screenshot				
<p>24.</p> <p>Note: In the previous result: The LOAD UNIT of the first partition was specifically configured as COLUMN LOADABLE in the CREATE TABLE statement, so it is shown as COLUMN .</p> <p>The LOAD UNIT of the second partition was specifically configured as PAGE LOADABLE in the CREATE TABLE statement, so it is shown as PAGE .</p> <p>The LOAD UNIT of the OTHERS partition, which was defined as the 3rd partition, was not specifically configured in the CREATE TABLE statement, so it defaults to COLUMN . (Note that the OTHERS partition is required to always be COLUMN LOADABLE).</p>	 <p>The screenshot shows a user interface for configuring a table. At the top, there is a tab labeled 'SQL' and three icons: a download arrow, a refresh circular arrow, and a close 'X' icon. Below these is a table with a header row labeled 'LOAD_UNIT' and three data rows. The first data row contains the text 'COLUMN', the second row contains 'PAGE', and the third row contains 'COLUMN'. Each row has a small dropdown arrow on its left and right sides.</p> <table><tr><th>LOAD_UNIT</th></tr><tr><td>COLUMN</td></tr><tr><td>PAGE</td></tr><tr><td>COLUMN</td></tr></table>	LOAD_UNIT	COLUMN	PAGE	COLUMN
LOAD_UNIT					
COLUMN					
PAGE					
COLUMN					

That concludes this exercise. You have created tables and added data to warm storage in HANA Cloud using the Native Storage Extensions (NSE), and examined the meta-data for those tables that show which parts of the table are stored in hot storage (in-memory) and which tables are stored in warm storage (NSE).