

DAT 161 – Introduction to SAP HANA Cloud

Contents

HANA Cloud Hands on exercise	3
Pre-requisite.....	3
HANA Cloud, Provisioning.....	5
Exercise 1. Walk Through the Provisioning Process	5
Exercise 2. Update a provisioned instance (Not Applicable for Trial Instances)	8
Introduction to HANA Cloud, Data Lake	10
Exercise 1. Create Schema	10
Exercise 2. Create physical tables in Data Lake	13
Exercise 3. Create Virtual Tables.....	15
Exercise 4. Load data into physical tables.....	17
Exercise 5. Query data from Virtual Table	18
Setting up a Remote Source to Athena.....	19
Exercise 1. Query the remote source and create a virtual table	19
Exercise 2. Create virtual tables.....	27
Exercise 3. Query Virtual Tables	29
Exercise 4. Turn on Table Replication	30
Exercise 5. Re-Run Query and Note Performance Difference	31
Operating HANA with Native Storage Extension	32
Prerequisite. Open WebIDE and connect to HANA system	32
Exercise 1. Execute SQL to create a table in NSE storage	38

Exercise 2. View the Load Unit configuration 43

..... 43

..... 44

..... 44

Exercise 3. Create and use a table with a specific column configured as page loadable 45

Exercise 4. View the load unit configuration of individual columns..... 47

Exercise 5. Create and use a table with a specific partition configured as page loadable 50

Exercise 6. View the load unit configuration of individual partitions..... 54

HANA Cloud Hands on exercise

Pre-requisite

- 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 menu with the following items: Overview, Spaces, Subscriptions, Connectivity, Destinations, Security, Quota Plans, and Entitlements. The 'Entitlements' item is circled in red. The main content area displays the subaccount 'trial' with details: Subdomain: ce3244c0trial, ID: 6fce28d4-86da-4793-be05-e08e7a563255, and Org. Name: ce3244c0trial. The 'Cloud Foundry Environment' and 'Entitlements' tabs are visible at the bottom of the main content area.

The screenshot shows the SAP Cloud Platform Cockpit interface for the 'trial' subaccount. The left sidebar contains the same menu as the previous screenshot, with 'Entitlements' selected. The main content area displays the 'trial' subaccount details. A table lists the entitlements for the 'trial' subaccount. The 'Configure Entitlements' button is circled in red.

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

Subaccount: trial - Entitlements

trial

SAP HANA Cloud



Add Service Plans

Save

Cancel

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	

- b. 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

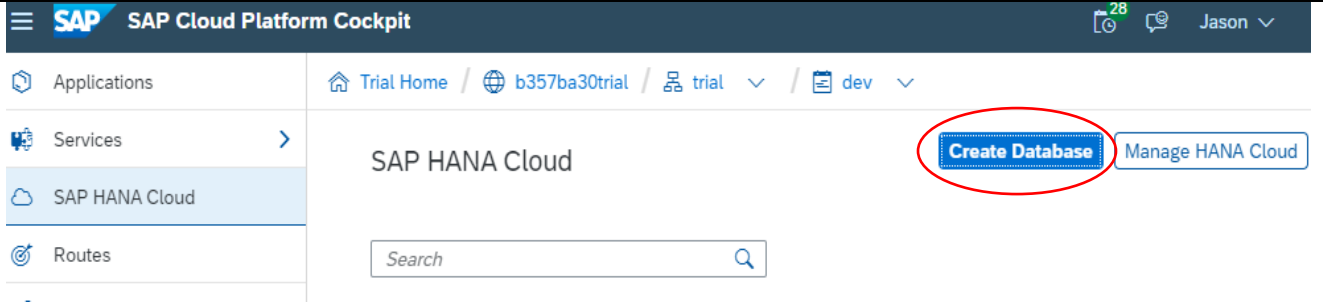
Search

SAP HANA Instances

HANA Cloud, Provisioning

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.

Step Explanation	Screenshot
1. From your SAP Cloud Platform trial account, open the “SAP HANA Cloud” item and click “Create Database” Follow through the wizard to create a HANA Cloud instance. Be sure to create a HANA data lake instance when given the option.	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. The top navigation bar includes the SAP logo, the title 'SAP Cloud Platform Cockpit', and a user profile 'Jason'. Below the navigation bar, there is a breadcrumb trail: 'Trial Home / b357ba30trial / trial / dev'. The left sidebar contains a menu with 'Applications', 'Services', 'SAP HANA Cloud', and 'Routes'. The 'SAP HANA Cloud' item is selected. The main content area displays 'SAP HANA Cloud' with a search bar. Two buttons are visible: 'Create Database' (highlighted with a red circle) and 'Manage HANA Cloud'.</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"

SAP HANA Cloud

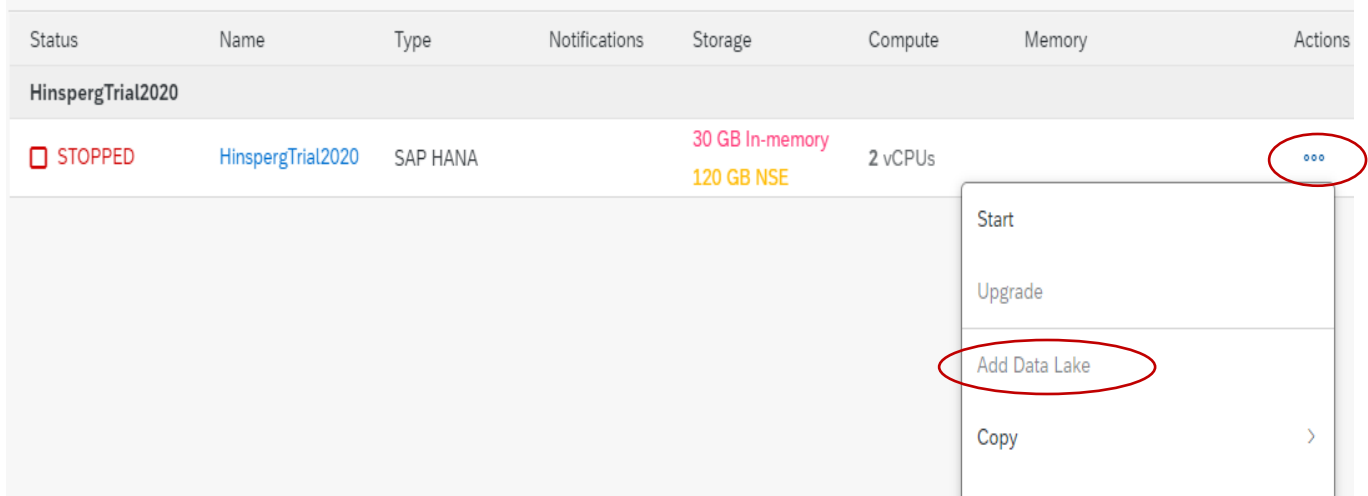
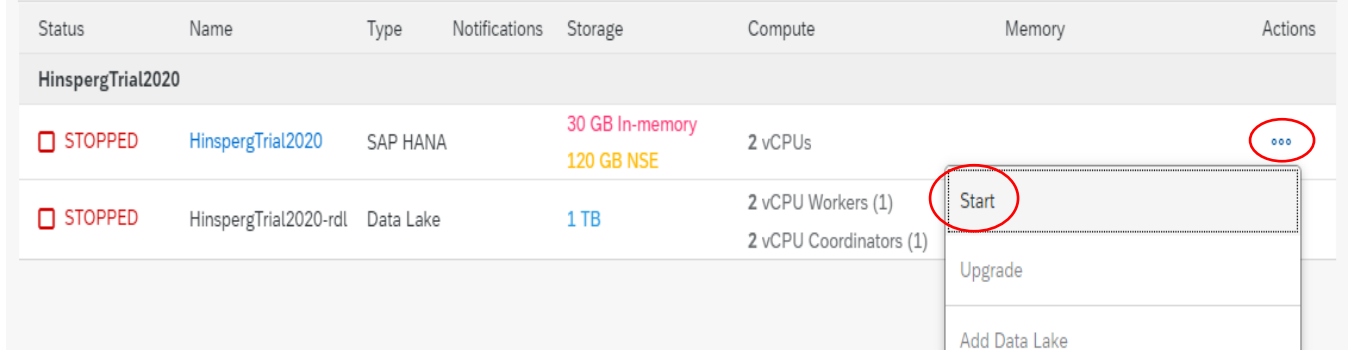
SAP HANA Instances

HinspergTrial2020
Created

Memory	CPU	Storage
30 GB	2 vCPUs	120 GB

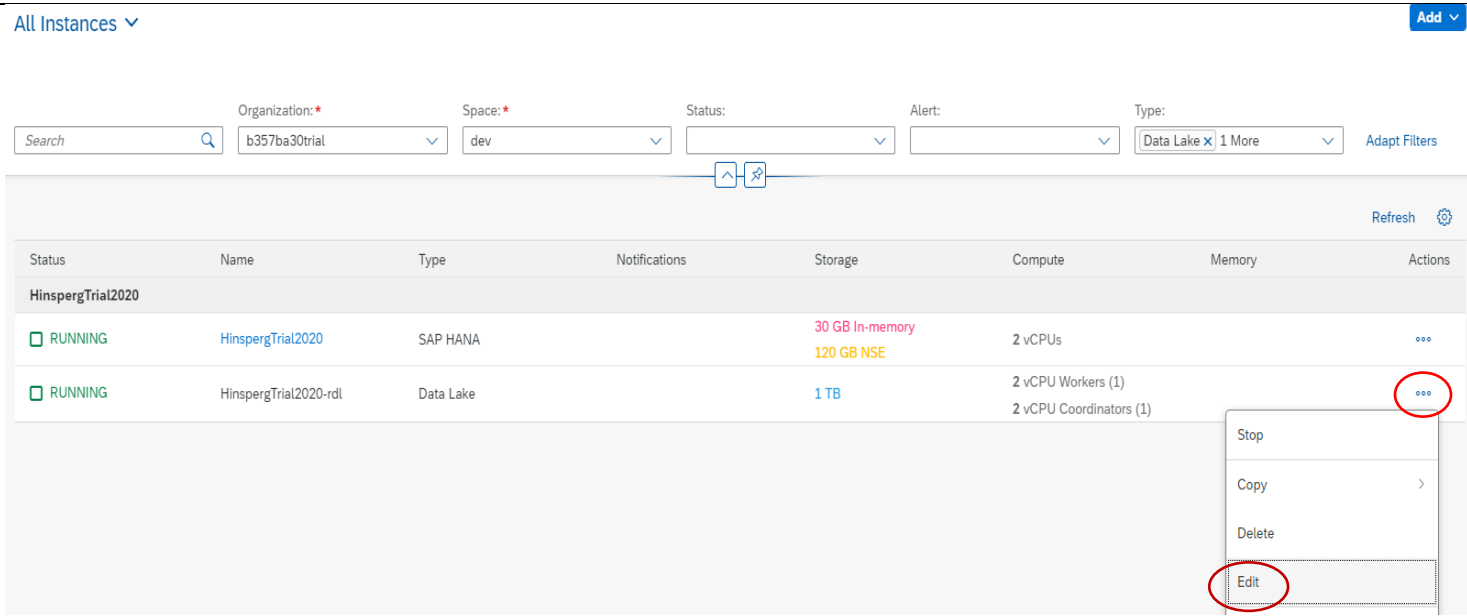
Actions ▾

Monitor landscape

Step 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>	 <p>The screenshot shows a table with columns: Status, Name, Type, Notifications, Storage, Compute, Memory, and Actions. The first row is for 'HinspergTrial2020' with status 'STOPPED', type 'SAP HANA', and storage '30 GB In-memory' and '120 GB NSE'. The 'Actions' menu is open, showing options: Start, Upgrade, Add Data Lake (highlighted), and Copy.</p>
<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>	 <p>The screenshot shows a table with columns: Status, Name, Type, Notifications, Storage, Compute, Memory, and Actions. The first row is for 'HinspergTrial2020' with status 'STOPPED', type 'SAP HANA', and storage '30 GB In-memory' and '120 GB NSE'. The second row is for 'HinspergTrial2020-rdl' with status 'STOPPED', type 'Data Lake', and storage '1 TB'. The 'Actions' menu is open, showing options: Start (highlighted), Upgrade, and Add Data Lake.</p>

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.

Step Explanation	Screenshot																								
1. Open the HANA Cloud management interface and click on the “Edit” option under the Actions extended menu.	 <p>The screenshot displays the HANA Cloud management interface. At the top, there is a header with 'All Instances' and an 'Add' button. Below the header, there are filters for Organization, Space, Status, Alert, and Type. The main table lists instances with columns for Status, Name, Type, Notifications, Storage, Compute, Memory, and Actions. Two instances are listed: 'HinspergTrial2020' (SAP HANA) and 'HinspergTrial2020-rdl' (Data Lake). The 'Edit' option in the Actions menu for the 'HinspergTrial2020-rdl' instance is highlighted with a red circle.</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>🟢 RUNNING</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>🟢 RUNNING</td><td>HinspergTrial2020-rdl</td><td>Data Lake</td><td></td><td>1 TB</td><td>2 vCPU Workers (1) 2 vCPU Coordinators (1)</td><td></td><td>... Stop Copy Delete Edit</td></tr></table>	Status	Name	Type	Notifications	Storage	Compute	Memory	Actions	🟢 RUNNING	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...	🟢 RUNNING	HinspergTrial2020-rdl	Data Lake		1 TB	2 vCPU Workers (1) 2 vCPU Coordinators (1)		... Stop Copy Delete Edit
Status	Name	Type	Notifications	Storage	Compute	Memory	Actions																		
🟢 RUNNING	HinspergTrial2020	SAP HANA		30 GB In-memory 120 GB NSE	2 vCPUs		...																		
🟢 RUNNING	HinspergTrial2020-rdl	Data Lake		1 TB	2 vCPU Workers (1) 2 vCPU Coordinators (1)		... Stop Copy Delete Edit																		

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*.

Edit HDL_EXERCISE-rdl

Instance Credentials

Instance Name: * HDL_EXERCISE-rdl

Data Lake

Compute: vCPUs Min 4 vCPUs, Max 162 vCPUs

Storage: TB Min 2 TB, Max 90 TB

Coordinators:

Workers:

[Data Lake Documentation](#)

[Data Lake Sizing Calculator](#)

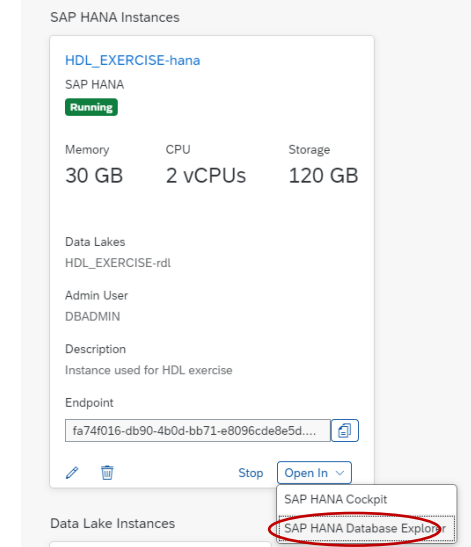
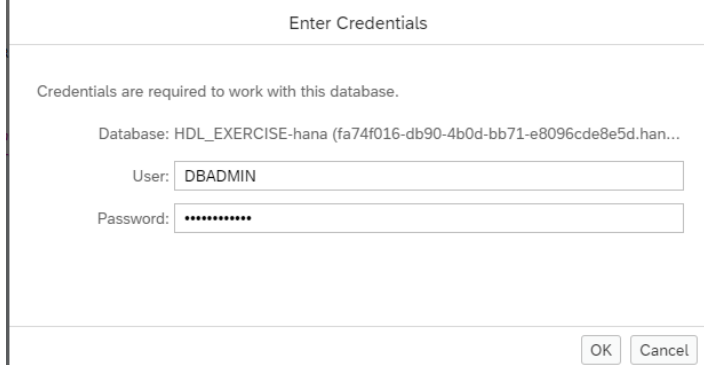
3.
Wait for the updated instance to start again. Once *Starting* turns to *Running*, the changes have been reflected in the Data Lake instance. This process could take a few minutes.

Status	Name	Type	Notifications	Storage	Compute
Hinsperg2020-4-hana					
<input checked="" type="checkbox"/> RUNNING	Hinsperg2020-4-hana	SAP HANA	i 1	45 GB In-memory 160 GB NFS	3 vCPUs
<input checked="" type="checkbox"/> RUNNING	Hinsperg2020-4-rdl	Data Lake		16 TB	16 vCPU Workers (1) 2 vCPU Coordinators (1)

Introduction to HANA Cloud, Data Lake

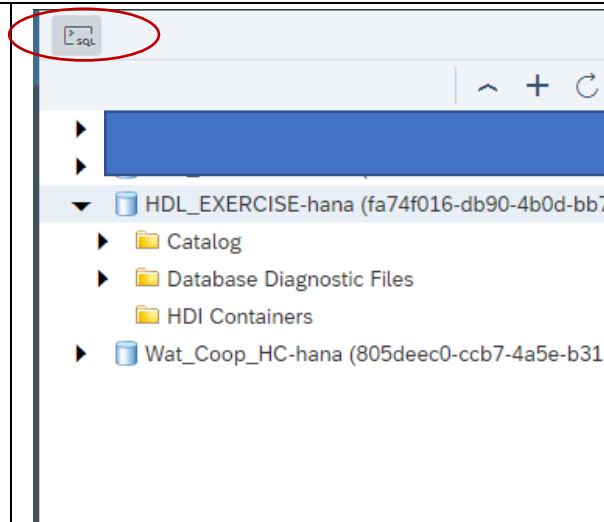
Exercise 1. Create Schema

The goal of this exercise is to create a schema in the HANA Cloud instance.

Step Explanation	Screenshot
<p>1.</p> <p>Navigate to the HANA Cloud instance we just created: <i>HDL_EXERCISE-hana</i>. Click on open in and select SAP HANA database explorer</p>	
<p>2.</p> <p>Enter Credential if prompted: User: "DBADMIN" Password: "HDLexercise1"</p>	

3.

Click on HDL_EXERCISE-hana to ensure it is selected. Then click on the “SQL” icon on the top left corner to launch a SQL console.



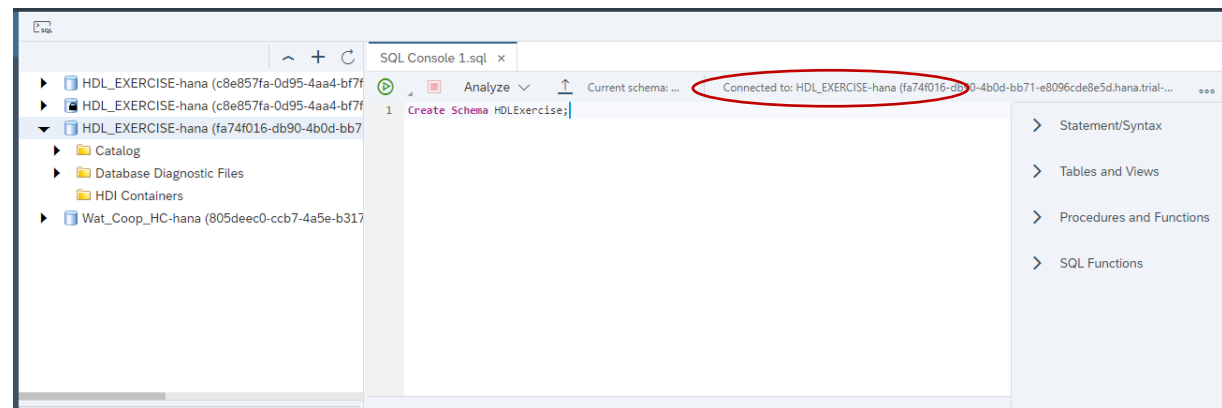
4.

Once the SQL Console is launched,

type

Create Schema HDLExercise;

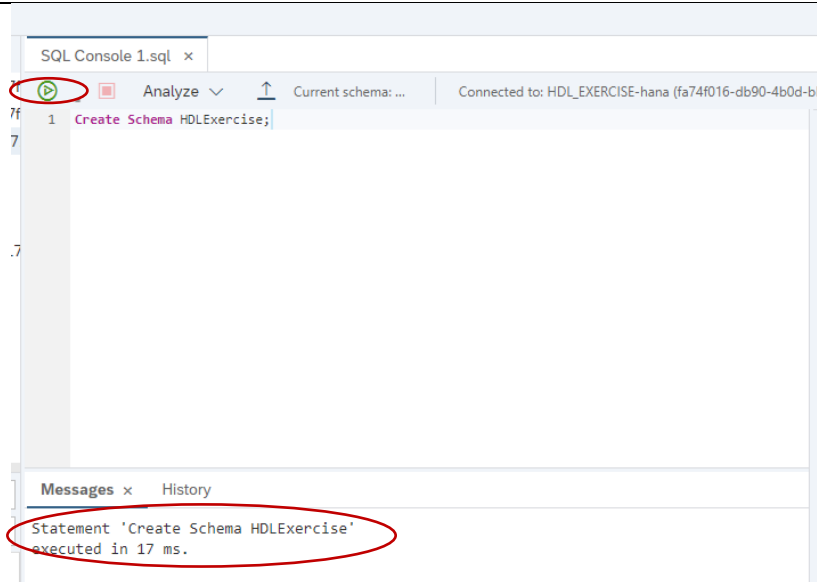
Note: *SQL Console 1.sql* is connected to HDL_EXERCISE-hana instance.



5.

Click on the green run button to execute the query.

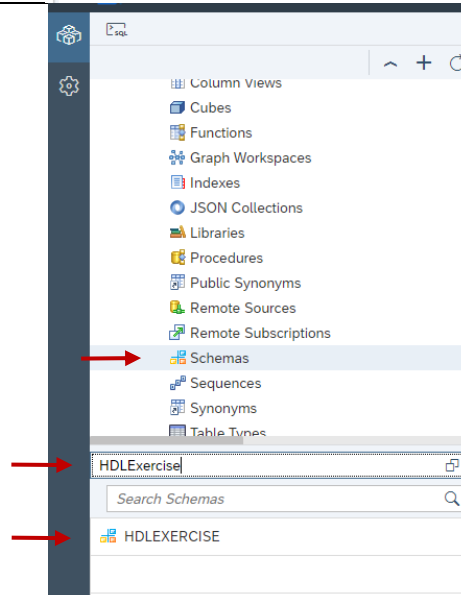
Make sure the query executes successfully by verifying in the *Messages* tab.



6.

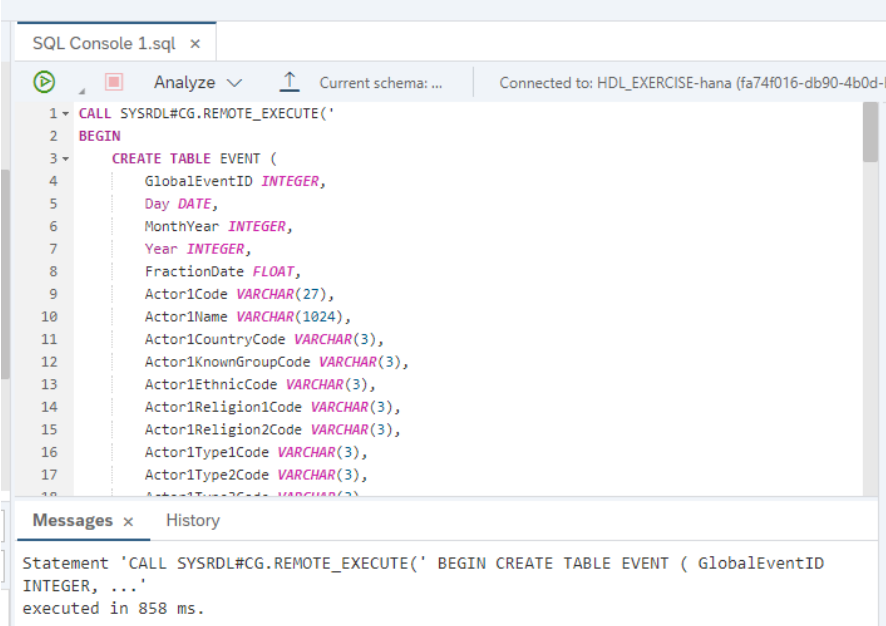
Check the Schema “HDLEExercise” has been created by navigating to Catalog, Schemas, and then type “HDLEExercise” in the Choose schema input form.

Ensure that HDLEExercise exists.



Exercise 2. Create physical tables in Data Lake

The goal of this exercise is to create physical tables in the data lake.

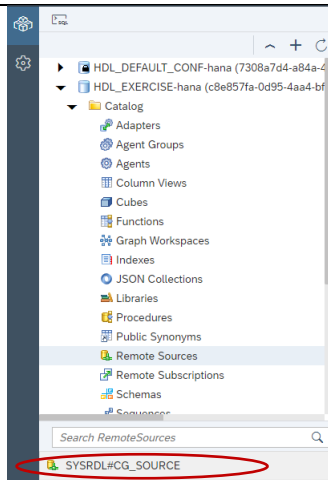
Step Explanation	Screenshot
<p>1.</p> <p>Copy and paste the command from <code>DAT161Exercise_CreateTable.sql</code> into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.</p> <p>This command creates the following tables in data lake: EVENT, GKG, MENTIONS.</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 in data lake.</p>	 <p>The screenshot displays the SQL Console interface. The top bar shows 'SQL Console 1.sql' and 'Analyze' button. The main area contains the following SQL code:</p> <pre>1 CALL SYSRDL#CG.REMOTE_EXECUTE(' 2 BEGIN 3 CREATE TABLE EVENT (4 GlobalEventID INTEGER, 5 Day DATE, 6 MonthYear INTEGER, 7 Year INTEGER, 8 FractionDate FLOAT, 9 Actor1Code VARCHAR(27), 10 Actor1Name VARCHAR(1024), 11 Actor1CountryCode VARCHAR(3), 12 Actor1KnownGroupCode VARCHAR(3), 13 Actor1EthnicCode VARCHAR(3), 14 Actor1Religion1Code VARCHAR(3), 15 Actor1Religion2Code VARCHAR(3), 16 Actor1Type1Code VARCHAR(3), 17 Actor1Type2Code VARCHAR(3), 18 Actor1Type3Code VARCHAR(3), 19 Actor1Type4Code VARCHAR(3), 20 Actor1Type5Code VARCHAR(3), 21 Actor1Type6Code VARCHAR(3), 22 Actor1Type7Code VARCHAR(3), 23 Actor1Type8Code VARCHAR(3), 24 Actor1Type9Code VARCHAR(3), 25 Actor1Type10Code VARCHAR(3), 26 Actor1Type11Code VARCHAR(3), 27 Actor1Type12Code VARCHAR(3), 28 Actor1Type13Code VARCHAR(3), 29 Actor1Type14Code VARCHAR(3), 30 Actor1Type15Code VARCHAR(3), 31 Actor1Type16Code VARCHAR(3), 32 Actor1Type17Code VARCHAR(3), 33 Actor1Type18Code VARCHAR(3), 34 Actor1Type19Code VARCHAR(3), 35 Actor1Type20Code VARCHAR(3), 36 Actor1Type21Code VARCHAR(3), 37 Actor1Type22Code VARCHAR(3), 38 Actor1Type23Code VARCHAR(3), 39 Actor1Type24Code VARCHAR(3), 40 Actor1Type25Code VARCHAR(3), 41 Actor1Type26Code VARCHAR(3), 42 Actor1Type27Code VARCHAR(3), 43 Actor1Type28Code VARCHAR(3), 44 Actor1Type29Code VARCHAR(3), 45 Actor1Type30Code VARCHAR(3), 46 Actor1Type31Code VARCHAR(3), 47 Actor1Type32Code VARCHAR(3), 48 Actor1Type33Code VARCHAR(3), 49 Actor1Type34Code VARCHAR(3), 50 Actor1Type35Code VARCHAR(3), 51 Actor1Type36Code VARCHAR(3), 52 Actor1Type37Code VARCHAR(3), 53 Actor1Type38Code VARCHAR(3), 54 Actor1Type39Code VARCHAR(3), 55 Actor1Type40Code VARCHAR(3), 56 Actor1Type41Code VARCHAR(3), 57 Actor1Type42Code VARCHAR(3), 58 Actor1Type43Code VARCHAR(3), 59 Actor1Type44Code VARCHAR(3), 60 Actor1Type45Code VARCHAR(3), 61 Actor1Type46Code VARCHAR(3), 62 Actor1Type47Code VARCHAR(3), 63 Actor1Type48Code VARCHAR(3), 64 Actor1Type49Code VARCHAR(3), 65 Actor1Type50Code VARCHAR(3), 66 Actor1Type51Code VARCHAR(3), 67 Actor1Type52Code VARCHAR(3), 68 Actor1Type53Code VARCHAR(3), 69 Actor1Type54Code VARCHAR(3), 70 Actor1Type55Code VARCHAR(3), 71 Actor1Type56Code VARCHAR(3), 72 Actor1Type57Code VARCHAR(3), 73 Actor1Type58Code VARCHAR(3), 74 Actor1Type59Code VARCHAR(3), 75 Actor1Type60Code VARCHAR(3), 76 Actor1Type61Code VARCHAR(3), 77 Actor1Type62Code VARCHAR(3), 78 Actor1Type63Code VARCHAR(3), 79 Actor1Type64Code VARCHAR(3), 80 Actor1Type65Code VARCHAR(3), 81 Actor1Type66Code VARCHAR(3), 82 Actor1Type67Code VARCHAR(3), 83 Actor1Type68Code VARCHAR(3), 84 Actor1Type69Code VARCHAR(3), 85 Actor1Type70Code VARCHAR(3), 86 Actor1Type71Code VARCHAR(3), 87 Actor1Type72Code VARCHAR(3), 88 Actor1Type73Code VARCHAR(3), 89 Actor1Type74Code VARCHAR(3), 90 Actor1Type75Code VARCHAR(3), 91 Actor1Type76Code VARCHAR(3), 92 Actor1Type77Code VARCHAR(3), 93 Actor1Type78Code VARCHAR(3), 94 Actor1Type79Code VARCHAR(3), 95 Actor1Type80Code VARCHAR(3), 96 Actor1Type81Code VARCHAR(3), 97 Actor1Type82Code VARCHAR(3), 98 Actor1Type83Code VARCHAR(3), 99 Actor1Type84Code VARCHAR(3), 100 Actor1Type85Code VARCHAR(3), 101 Actor1Type86Code VARCHAR(3), 102 Actor1Type87Code VARCHAR(3), 103 Actor1Type88Code VARCHAR(3), 104 Actor1Type89Code VARCHAR(3), 105 Actor1Type90Code VARCHAR(3), 106 Actor1Type91Code VARCHAR(3), 107 Actor1Type92Code VARCHAR(3), 108 Actor1Type93Code VARCHAR(3), 109 Actor1Type94Code VARCHAR(3), 110 Actor1Type95Code VARCHAR(3), 111 Actor1Type96Code VARCHAR(3), 112 Actor1Type97Code VARCHAR(3), 113 Actor1Type98Code VARCHAR(3), 114 Actor1Type99Code VARCHAR(3), 115 Actor1Type100Code VARCHAR(3), 116 Actor1Type101Code VARCHAR(3), 117 Actor1Type102Code VARCHAR(3), 118 Actor1Type103Code VARCHAR(3), 119 Actor1Type104Code VARCHAR(3), 120 Actor1Type105Code VARCHAR(3), 121 Actor1Type106Code VARCHAR(3), 122 Actor1Type107Code VARCHAR(3), 123 Actor1Type108Code VARCHAR(3), 124 Actor1Type109Code VARCHAR(3), 125 Actor1Type110Code VARCHAR(3), 126 Actor1Type111Code VARCHAR(3), 127 Actor1Type112Code VARCHAR(3), 128 Actor1Type113Code VARCHAR(3), 129 Actor1Type114Code VARCHAR(3), 130 Actor1Type115Code VARCHAR(3), 131 Actor1Type116Code VARCHAR(3), 132 Actor1Type117Code VARCHAR(3), 133 Actor1Type118Code VARCHAR(3), 134 Actor1Type119Code VARCHAR(3), 135 Actor1Type120Code VARCHAR(3), 136 Actor1Type121Code VARCHAR(3), 137 Actor1Type122Code VARCHAR(3), 138 Actor1Type123Code VARCHAR(3), 139 Actor1Type124Code VARCHAR(3), 140 Actor1Type125Code VARCHAR(3), 141 Actor1Type126Code VARCHAR(3), 142 Actor1Type127Code VARCHAR(3), 143 Actor1Type128Code VARCHAR(3), 144 Actor1Type129Code VARCHAR(3), 145 Actor1Type130Code VARCHAR(3), 146 Actor1Type131Code VARCHAR(3), 147 Actor1Type132Code VARCHAR(3), 148 Actor1Type133Code VARCHAR(3), 149 Actor1Type134Code VARCHAR(3), 150 Actor1Type135Code VARCHAR(3), 151 Actor1Type136Code VARCHAR(3), 152 Actor1Type137Code VARCHAR(3), 153 Actor1Type138Code VARCHAR(3), 154 Actor1Type139Code VARCHAR(3), 155 Actor1Type140Code VARCHAR(3), 156 Actor1Type141Code VARCHAR(3), 157 Actor1Type142Code VARCHAR(3), 158 Actor1Type143Code VARCHAR(3), 159 Actor1Type144Code VARCHAR(3), 160 Actor1Type145Code VARCHAR(3), 161 Actor1Type146Code VARCHAR(3), 162 Actor1Type147Code VARCHAR(3), 163 Actor1Type148Code VARCHAR(3), 164 Actor1Type149Code VARCHAR(3), 165 Actor1Type150Code VARCHAR(3), 166 Actor1Type151Code VARCHAR(3), 167 Actor1Type152Code VARCHAR(3), 168 Actor1Type153Code VARCHAR(3), 169 Actor1Type154Code VARCHAR(3), 170 Actor1Type155Code VARCHAR(3), 171 Actor1Type156Code VARCHAR(3), 172 Actor1Type157Code VARCHAR(3), 173 Actor1Type158Code VARCHAR(3), 174 Actor1Type159Code VARCHAR(3), 175 Actor1Type160Code VARCHAR(3), 176 Actor1Type161Code VARCHAR(3), 177 Actor1Type162Code VARCHAR(3), 178 Actor1Type163Code VARCHAR(3), 179 Actor1Type164Code VARCHAR(3), 180 Actor1Type165Code VARCHAR(3), 181 Actor1Type166Code VARCHAR(3), 182 Actor1Type167Code VARCHAR(3), 183 Actor1Type168Code VARCHAR(3), 184 Actor1Type169Code VARCHAR(3), 185 Actor1Type170Code VARCHAR(3), 186 Actor1Type171Code VARCHAR(3), 187 Actor1Type172Code VARCHAR(3), 188 Actor1Type173Code VARCHAR(3), 189 Actor1Type174Code VARCHAR(3), 190 Actor1Type175Code VARCHAR(3), 191 Actor1Type176Code VARCHAR(3), 192 Actor1Type177Code VARCHAR(3), 193 Actor1Type178Code VARCHAR(3), 194 Actor1Type179Code VARCHAR(3), 195 Actor1Type180Code VARCHAR(3), 196 Actor1Type181Code VARCHAR(3), 197 Actor1Type182Code VARCHAR(3), 198 Actor1Type183Code VARCHAR(3), 199 Actor1Type184Code VARCHAR(3), 200 Actor1Type185Code VARCHAR(3), 201 Actor1Type186Code VARCHAR(3), 202 Actor1Type187Code VARCHAR(3), 203 Actor1Type188Code VARCHAR(3), 204 Actor1Type189Code VARCHAR(3), 205 Actor1Type190Code VARCHAR(3), 206 Actor1Type191Code VARCHAR(3), 207 Actor1Type192Code VARCHAR(3), 208 Actor1Type193Code VARCHAR(3), 209 Actor1Type194Code VARCHAR(3), 210 Actor1Type195Code VARCHAR(3), 211 Actor1Type196Code VARCHAR(3), 212 Actor1Type197Code VARCHAR(3), 213 Actor1Type198Code VARCHAR(3), 214 Actor1Type199Code VARCHAR(3), 215 Actor1Type200Code VARCHAR(3), 216 Actor1Type201Code VARCHAR(3), 217 Actor1Type202Code VARCHAR(3), 218 Actor1Type203Code VARCHAR(3), 219 Actor1Type204Code VARCHAR(3), 220 Actor1Type205Code VARCHAR(3), 221 Actor1Type206Code VARCHAR(3), 222 Actor1Type207Code VARCHAR(3), 223 Actor1Type208Code VARCHAR(3), 224 Actor1Type209Code VARCHAR(3), 225 Actor1Type210Code VARCHAR(3), 226 Actor1Type211Code VARCHAR(3), 227 Actor1Type212Code VARCHAR(3), 228 Actor1Type213Code VARCHAR(3), 229 Actor1Type214Code VARCHAR(3), 230 Actor1Type215Code VARCHAR(3), 231 Actor1Type216Code VARCHAR(3), 232 Actor1Type217Code VARCHAR(3), 233 Actor1Type218Code VARCHAR(3), 234 Actor1Type219Code VARCHAR(3), 235 Actor1Type220Code VARCHAR(3), 236 Actor1Type221Code VARCHAR(3), 237 Actor1Type222Code VARCHAR(3), 238 Actor1Type223Code VARCHAR(3), 239 Actor1Type224Code VARCHAR(3), 240 Actor1Type225Code VARCHAR(3), 241 Actor1Type226Code VARCHAR(3), 242 Actor1Type227Code VARCHAR(3), 243 Actor1Type228Code VARCHAR(3), 244 Actor1Type229Code VARCHAR(3), 245 Actor1Type230Code VARCHAR(3), 246 Actor1Type231Code VARCHAR(3), 247 Actor1Type232Code VARCHAR(3), 248 Actor1Type233Code VARCHAR(3), 249 Actor1Type234Code VARCHAR(3), 250 Actor1Type235Code VARCHAR(3), 251 Actor1Type236Code VARCHAR(3), 252 Actor1Type237Code VARCHAR(3), 253 Actor1Type238Code VARCHAR(3), 254 Actor1Type239Code VARCHAR(3), 255 Actor1Type240Code VARCHAR(3), 256</pre>

2.

Verify the tables have been created in Data lake.

First, navigate to *Catalog*, then *Remote Sources* from the left side bar. Click on *Remote Sources*. You should see one remote source called *SYSRDL#CG_SOURCE*.

Double Click on *SYSRDL#CG_SOURCE*.



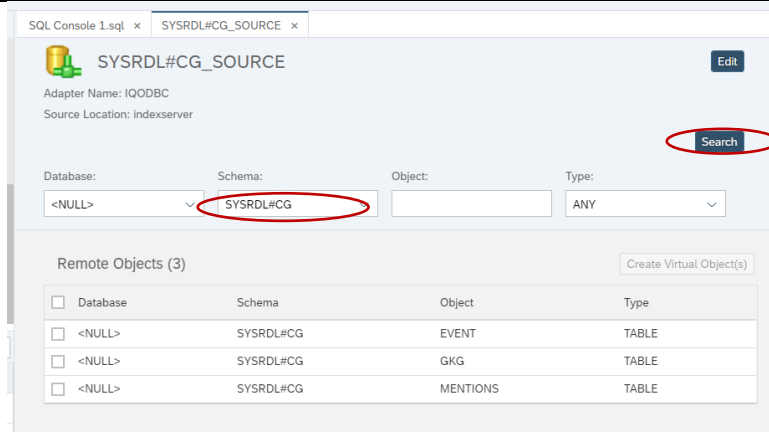
3.

This opens a tab next to *SQL Console 1.sql*.

Choose *SYSRDL#CG* from drop down menu for Schema.

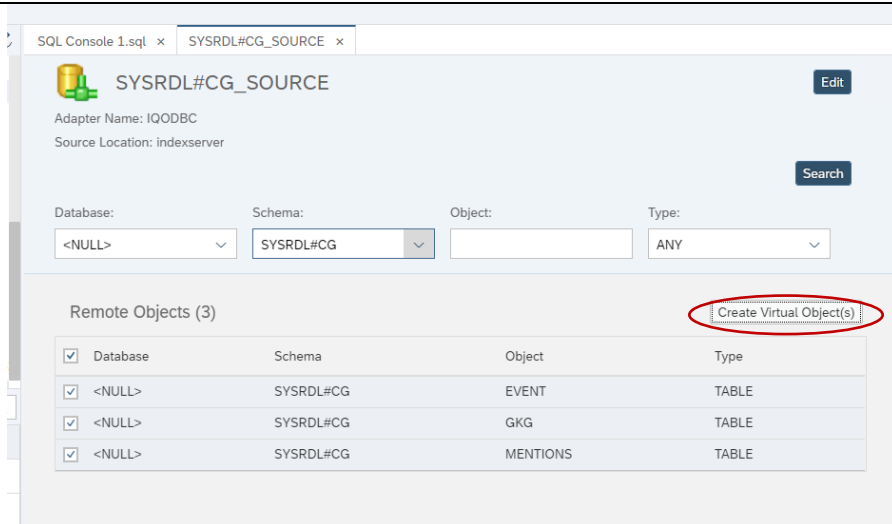
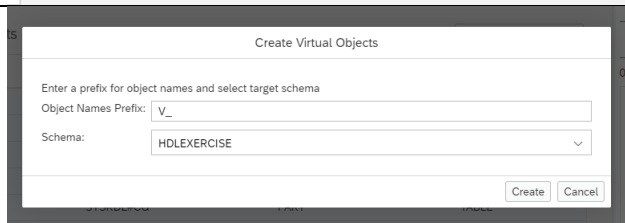
Click on the Search button.

The tables that we just created should appear in the *Remote Objects* section.



Exercise 3. Create Virtual Tables

The goal of this exercise is to create virtual objects referring to the physical tables in the data lake.

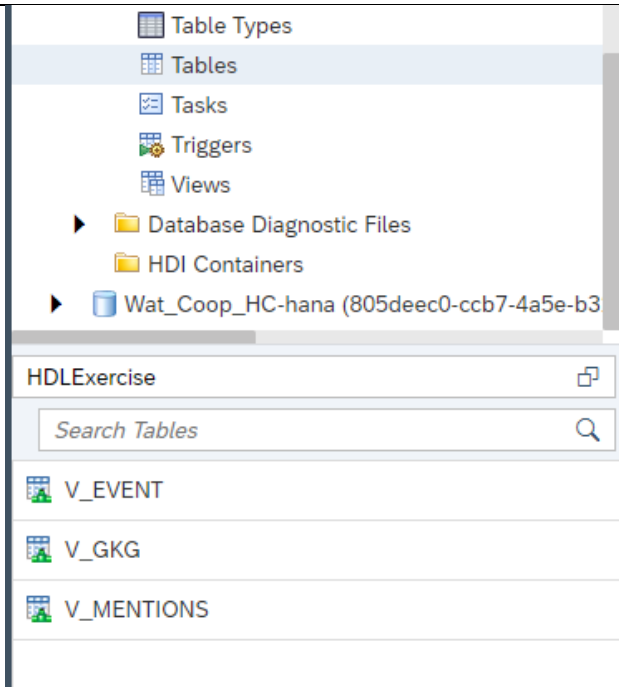
Step Explanation	Screenshot																
<p>1.</p> <p>From the same interface as the previous exercise, select all the tables, and click on <i>Create Virtual Objects</i>.</p>	 <p>The screenshot shows the 'SYSRDL#CG_SOURCE' interface. At the top, it displays 'Adapter Name: IQODBC' and 'Source Location: indexserver'. Below this, there are dropdown menus for 'Database' (set to '<NULL>'), 'Schema' (set to 'SYSRDL#CG'), 'Object' (empty), and 'Type' (set to 'ANY'). A 'Search' button is located to the right of these dropdowns. Below the search section, there is a table titled 'Remote Objects (3)'. The table has columns for 'Database', 'Schema', 'Object', and 'Type'. All three rows are selected with checkboxes. The 'Object' column contains 'EVENT', 'GKG', and 'MENTIONS', all of which are 'TABLE' type. A button labeled 'Create Virtual Object(s)' is circled in red at the top right of the table.</p> <table><tr><th>Database</th><th>Schema</th><th>Object</th><th>Type</th></tr><tr><td><NULL></td><td>SYSRDL#CG</td><td>EVENT</td><td>TABLE</td></tr><tr><td><NULL></td><td>SYSRDL#CG</td><td>GKG</td><td>TABLE</td></tr><tr><td><NULL></td><td>SYSRDL#CG</td><td>MENTIONS</td><td>TABLE</td></tr></table>	Database	Schema	Object	Type	<NULL>	SYSRDL#CG	EVENT	TABLE	<NULL>	SYSRDL#CG	GKG	TABLE	<NULL>	SYSRDL#CG	MENTIONS	TABLE
Database	Schema	Object	Type														
<NULL>	SYSRDL#CG	EVENT	TABLE														
<NULL>	SYSRDL#CG	GKG	TABLE														
<NULL>	SYSRDL#CG	MENTIONS	TABLE														
<p>2.</p> <p>Set <i>Object Names Prefix</i> to V_</p> <p>Then select the schema created from the previous exercise <i>HDLEXERCISE</i>, and click on create.</p> <p>This creates a virtual table for each physical table, and names the virtual table V_<name of physical table>. For instance, the virtual table referring to EVENT will be named V_EVENT.</p> <p>The virtual tables are associated with the schema HDLEXERCISE.</p>	 <p>The screenshot shows a 'Create Virtual Objects' dialog box. It has a title bar 'Create Virtual Objects'. Inside, there is a text input field for 'Object Names Prefix' with the value 'V_'. Below it, there is a dropdown menu for 'Schema' with the value 'HDLEXERCISE'. At the bottom right, there are 'Create' and 'Cancel' buttons.</p>																

3.

To verify the virtual tables have been created successfully, navigate to *Catalog*, then *Tables*.

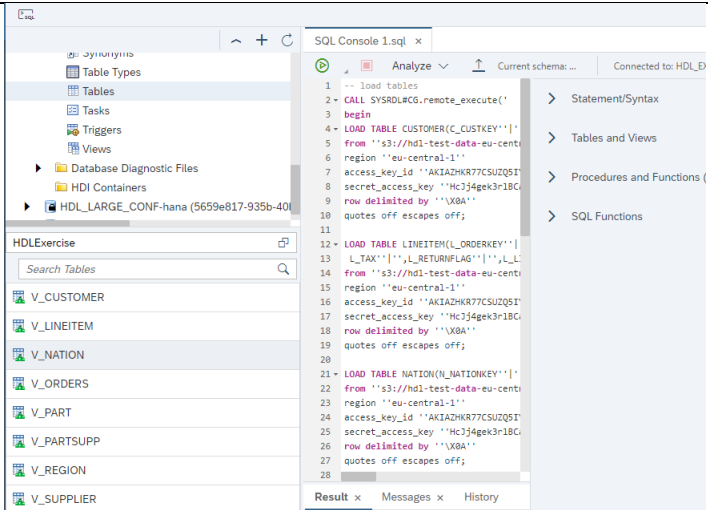
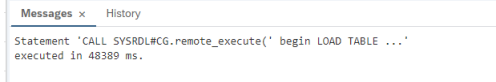
Type *HDLExercise* in the choose schema input form.

You should see the virtual tables that you have just created.



Exercise 4. Load data into physical tables

The goal of this exercise is to load data from an external source (an AWS s3 bucket), into the physical tables we have previously created in data lake.

Step Explanation	Screenshot
<p>1.</p> <p>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 so that data lake can read the data directly from the S3 bucket, and the data is not going through the Hana Cloud instance.</p>	 <p>The screenshot displays the SAP HANA SQL Console interface. On the left, a tree view shows the database structure, including 'HDL_EXERCISE' and various tables like 'V_CUSTOMER', 'V_LINEITEM', 'V_NATION', 'V_ORDERS', 'V_PART', 'V_PARTSUPP', 'V_REGION', and 'V_SUPPLIER'. The main area shows a SQL script titled 'SQL Console 1.sql' with the following content:</p> <pre>1 -- load tables 2 - CALL SYSDDL#CG.remote_execute(' 3 begin 4 - LOAD TABLE CUSTOMER(C_CUSTKEY') 5 - from 's3://hdl-test-data-eu-centr 6 region 'eu-central-1' 7 access_key_id 'AKIAZHKR77CSUQ5T 8 secret_access_key 'Hc3j4gek3r1BG 9 row delimited by ''XBA'' 10 quotes off escapes off; 11 12 - LOAD TABLE LINEITEM(L_ORDERKEY') 13 L_TAX''',L_RETURNFLAG''',L_L 14 from 's3://hdl-test-data-eu-centr 15 region 'eu-central-1' 16 access_key_id 'AKIAZHKR77CSUQ5T 17 secret_access_key 'Hc3j4gek3r1BG 18 row delimited by ''XBA'' 19 quotes off escapes off; 20 21 - LOAD TABLE NATION(N_NATIONKEY') 22 from 's3://hdl-test-data-eu-centr 23 region 'eu-central-1' 24 access_key_id 'AKIAZHKR77CSUQ5T 25 secret_access_key 'Hc3j4gek3r1BG 26 row delimited by ''XBA'' 27 quotes off escapes off; 28</pre> <p>On the right, a sidebar shows navigation options: 'Statement/Syntax', 'Tables and Views', 'Procedures and Functions', and 'SQL Functions'. Below the script, tabs for 'Result', 'Messages', and 'History' are visible.</p>
<p>2.</p> <p>This process could take a few minutes. Verify that the statements execute successfully.</p>	 <p>The screenshot shows the 'Messages' tab in the SQL Console. It displays the following message:</p> <pre>Statement 'CALL SYSDDL#CG.remote_execute(' begin LOAD TABLE ...' executed in 48389 ms.</pre>

Exercise 5. Query data from Virtual Table

The goal of this exercise is to query data from virtual tables.

<div>Step Explanation</div> <div>1. Copy and paste the command from <i>Query1.sql</i> into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.</div>	<div>Screenshot</div> <div><div>SQL Console 1.sql x V_NATION x</div><div><div>Analyze</div><div>Current schema: ...</div><div>Connected to: HDL_EXERCISE-hana (c8e857fa-0d95-4aa4-b77f-f10afb0)</div></div><div><pre>1- SELECT sum("l_extendedprice" * "l_discount") AS revenue 2- FROM HDLExercise.V_LINETYPE 3- WHERE "l_shipdate" >= '1994-01-01' 4- AND "l_shipdate" < add_years(TO_DATE('1994-01-01', 'YYYY-MM-DD'), 1) 5- AND --ASA date '1994-01-01' + interval '1' year 6- "l_discount" BETWEEN .06 - 0.01 AND .06 + 0.01 7- AND "l_quantity" < 24;</pre></div><div><div>Result x Messages x History</div><div>Rows (1)</div><div>SQL</div><div>REVENUE</div><div>1 123141078.22829963</div></div></div>																									
<div>2. Copy and paste the command from <i>Query2.sql</i> into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.</div>	<div><div>SQL Console 1.sql x</div><div><div>Analyze</div><div>Current schema: ...</div><div>Connected to: HDL_EXERCISE-hana (c8e857fa-0d95-4aa4-b77f-f10afb0)</div></div><div><pre>1- SELECT 2- supp_nation, 3- cust_nation, 4- l_year, 5- sum(volume) AS revenue 6- FROM 7- (8- SELECT 9- n1."n_name" AS supp_nation, 10- n2."n_name" AS cust_nation, 11- year("l_shipdate") AS l_year, 12- "l_extendedprice" * (1 - "l_discount") AS volume 13- FROM 14- HDLExercise.V_SUPPLIER, 15- HDLExercise.V_LINETYPE, 16- HDLExercise.V_ORDERS, 17- HDLExercise.V_CUSTOMER, 18- HDLExercise.V_NATION AS n1, 19- HDLExercise.V_NATION AS n2 20- WHERE "s_suppkey" = "l_suppkey" 21- AND "o_orderkey" = "l_orderkey" 22- AND "c_custkey" = "o_custkey" 23- AND "s_nationkey" = n1."n_nationkey" 24- AND "c_nationkey" = n2."n_nationkey" 25- AND ((n1."n_name" = 'FRANCE' 26- AND n2."n_name" = 'GERMANY') 27- OR (n1."n_name" = 'GERMANY' 28- AND n2."n_name" = 'FRANCE'))</pre></div><div><div>Result x Messages x History</div><div>Rows (4)</div><div>SQL</div><div><table><tr><th></th><th>SUPP_NATION</th><th>CUST_NATION</th><th>L_YEAR</th><th>REVENUE</th></tr><tr><td>1</td><td>FRANCE</td><td>GERMANY</td><td>1995</td><td>54639732.733599976</td></tr><tr><td>2</td><td>FRANCE</td><td>GERMANY</td><td>1996</td><td>54633083.30760002</td></tr><tr><td>3</td><td>GERMANY</td><td>FRANCE</td><td>1995</td><td>52531746.669700034</td></tr><tr><td>4</td><td>GERMANY</td><td>FRANCE</td><td>1996</td><td>52520549.02239998</td></tr></table></div></div></div>		SUPP_NATION	CUST_NATION	L_YEAR	REVENUE	1	FRANCE	GERMANY	1995	54639732.733599976	2	FRANCE	GERMANY	1996	54633083.30760002	3	GERMANY	FRANCE	1995	52531746.669700034	4	GERMANY	FRANCE	1996	52520549.02239998
	SUPP_NATION	CUST_NATION	L_YEAR	REVENUE																						
1	FRANCE	GERMANY	1995	54639732.733599976																						
2	FRANCE	GERMANY	1996	54633083.30760002																						
3	GERMANY	FRANCE	1995	52531746.669700034																						
4	GERMANY	FRANCE	1996	52520549.02239998																						




Setting up a Remote Source to Athena

Exercise 1. Query the remote source and create a virtual table

The goal of this exercise is to set up a remote source to Athena and create virtual tables using the SAP HANA database explorer.

Step Explanation	Screenshot
1. Create a remote source called "ATHENASOURCE" by running the first statement in <i>DAT161Exercise_Athena.sql</i> under "-- Create Remote Source" in a SQL console	

SQL Console 1.sql x

9.   Analyze  Disconnected from: HDL_EXERCISE-hana (25227f88-2923-4f56-bc52-170759e33f31.hana.prod-us10)

1 CREATE REMOTE SOURCE "ATHENASOURCE" ADAPTER "athena" CONFIGURATION '<?xml version="1.0" encoding="UTF-8"?><Co
2 WITH CREDENTIAL TYPE 'PASSWORD' using 'user=AKIA3E3KHFB LD64DBVXC;password=Y2U8eLUaDI tFag+6aHyu/NaRCwGrO H FH2or
b

Messages x History

Statement 'CREATE REMOTE SOURCE "ATHENASOURCE" ADAPTER "athena"
CONFIGURATION '<?xml version="1.0" ...'
executed in 4 ms.

2. Add AWS and S3 certificates by copying the `CREATE CERTIFICATE FROM ...` statement under “-- Add AWS certificate and S3 certificate” from DAT161Exercise_Athena.sql

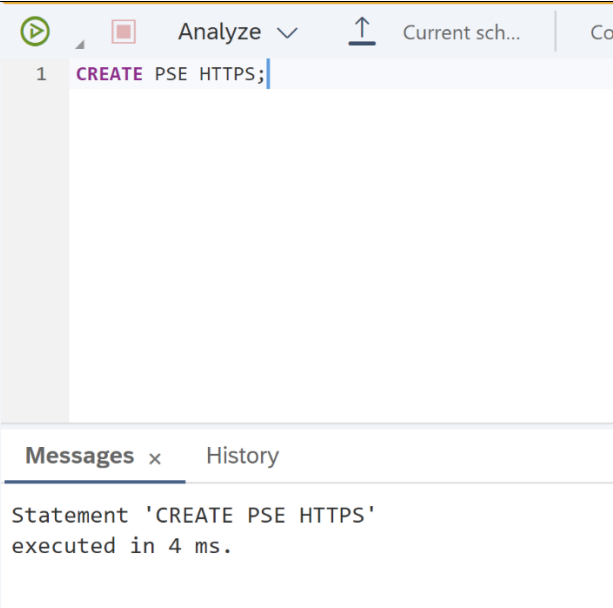
```
SQL Console 1.sql x
Analyze Disconnected from: HDL_EXERCISE-hana (25227f88-2923-4f56-bc52-170759e33f31.hana.p)

1 -- Add AWS certificate and S3 certificate
2 CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE-----
3 MIIEDzCCAvEgAwIBAgIBADANBgkqhkiG9w0BAQUFADBoMQswCQYDVQQGEwJVUzE1
4 MCMGA1UEChMcU3RhcmZpZWxkIFRlY2hub2xvZ2llcywgSW5jLjEYMDAGA1UECXMp
5 U3RhcmZpZWxkIENsYXNzIDIGQ2VydGlmYWVhdG1vbiBBdXR0b3JpdHkwHhcnMDQw
6 NjI5MTczOTE2WhcnMzQWwNjI5MTczOTE2WjBoMQswCQYDVQQGEwJVUzE1MCMGA1UE
7 ChMcU3RhcmZpZWxkIFRlY2hub2xvZ2llcywgSW5jLjEYMDAGA1UECXMpU3RhcmZp
8 ZWxkIENsYXNzIDIGQ2VydGlmYWVhdG1vbiBBdXR0b3JpdHkwGgEgMA0GCSqGSIb3
9 DQEBAQUAA4IBDQAwggEIAoIBAQC3Msj+6XGmBIWtDBFk385N78gDGic/oav7PKaf
10 8Moh2tTYbitTkPskpD6E8J7oX+z1J0T1KKY/e97gKvDIr1MvnsoFAZMej2YcOadN
11 +1q2cwQLZut3f+dZxkqZJRRU6ybh838Z1TBwj6+wRir/resp7defqg5Ho9T51aU0
12 X9tDkYI22WY8sb15gv2c0j4QyDvvBmVmepsZGD3/cVE8MC5fvj13c7JdBmzDI1aa
13 K4UmkhynArPkPw2vCHmCuDY96pzTNbO8acr1zJ3o/WSNF4Azbl5KXZnJHoe0nRrA
14 1W4TNSNe35tfPe/w93bC6j67eA0cQmdrBNj41tpv1/JEoAGrAgEdo4HFMIHCB0G
15 A1UdDgQWBBS/X7fRzt0fhvRbVazc1xDCDqmISzCBkgYDVR0jBIGKIMIGHGBS/X7fR
16 zt0fhvRbVazc1xDCDqmIS6FspGowaDELMakGA1UEBhMCVVMxJTAjBgNVBAoTHFN0
17 YXJmaWVsZCZBUZWNobm9sb2dpZXMsIEluYy4xMjAwBgNVBAsTKVNOYXJmaWVsZCZB
18 bGZcyAyIENlcnRpZmljYXRpb24gQXV0aG9yaXR5gEAMAwGA1UdEwQFMAMBAf8w
19 DQYJKoZIhvcNAQEFBQADggEBAAdP4id0ckaVaGsaffPzWdqbAYcaT1epoXkJKtv3
20 L7IezMdeatiDh6GX70k1PncGQVh1v45YuApnP+yz3SFmH81U+nLMPUxA2IGvd56D
21 eruix/U0F47ZEUD0/CwqTRV/p2JdLiXTAAsGh1o+Re49L2L7ShZ3U0WixeDyLJl
22 xy16paqU4Zt3VekyvvggQQt08PT7dL5WXXp59fkdhMt1b71cZBDzI0fmgAKhynp
```

```
Messages x History
Statement 'CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE----- ...'
executed in 5 ms.
```

3. Create PSE HTTPS by running the following statement in the SQL console:

CREATE PSE HTTPS;

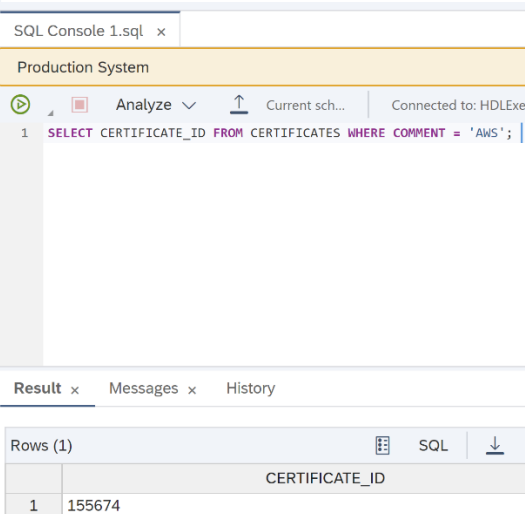


The screenshot shows a SQL console interface. At the top, there's a toolbar with a play button, a red square, a dropdown menu labeled 'Analyze', an upward arrow, and text 'Current sch...' and 'Co'. Below the toolbar, a text area contains the SQL statement: `1 CREATE PSE HTTPS;`. At the bottom, there's a 'Messages' tab selected, showing the message: 'Statement 'CREATE PSE HTTPS' executed in 4 ms.'

4. Get the CERTIFICATE_ID value for AWS by running the following statement in the SQL console:

SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';

Note: keep track of the CERTIFICATE_ID as it will be used in the next step



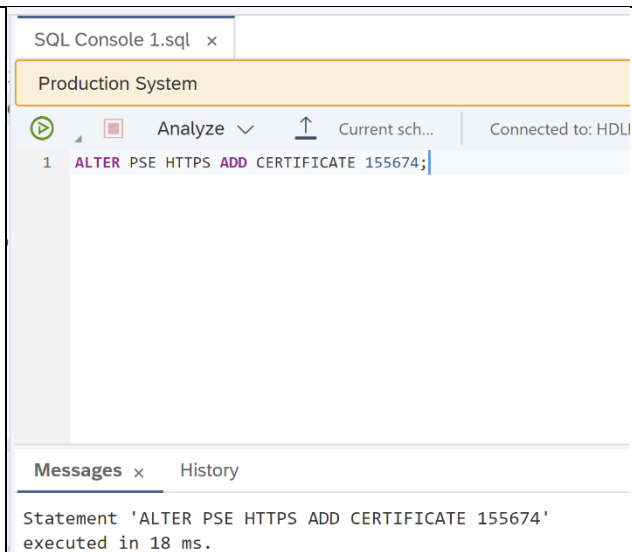
The screenshot shows a SQL console interface. At the top, there's a toolbar with a play button, a red square, a dropdown menu labeled 'Analyze', an upward arrow, and text 'Current sch...' and 'Connected to: HDLExe'. Below the toolbar, a text area contains the SQL statement: `1 SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';`. At the bottom, there's a 'Result' tab selected, showing the output of the query. The output is a table with one row and one column, 'CERTIFICATE_ID', with the value '155674'.

Rows (1)	CERTIFICATE_ID
1	155674

5. Alter PSE HTTPS to add the AWS CERTIFICATE_ID by running the following statement in the SQL console:

```
ALTER PSE HTTPS ADD CERTIFICATE  
[CERTIFICATE_ID]
```

Note: replace [CERTIFICATE_ID] with the AWS CERTIFICATE_ID from previous step



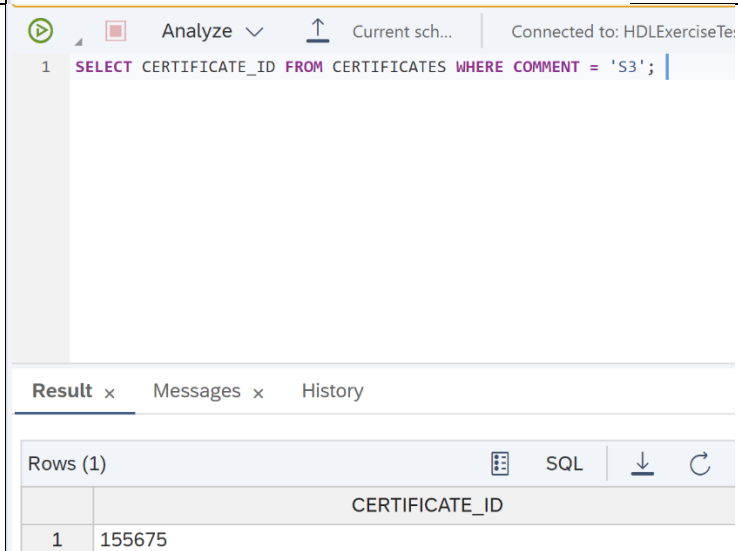
The screenshot shows the SQL console interface. At the top, there's a tab labeled 'SQL Console 1.sql'. Below it, a yellow banner indicates 'Production System'. The main area shows a SQL statement: `1 ALTER PSE HTTPS ADD CERTIFICATE 155674;`. Below the statement, a 'Messages' tab is active, displaying the message: 'Statement 'ALTER PSE HTTPS ADD CERTIFICATE 155674' executed in 18 ms.'

6. Now repeat step 4-5 for the S3 certificate.

First, get the CERTIFICATE_ID value for S3 by running the following statement in the SQL console:

```
SELECT CERTIFICATE_ID FROM  
CERTIFICATES WHERE COMMENT = 'S3';
```

Note: keep track of the CERTIFICATE_ID as it will be used in the next step



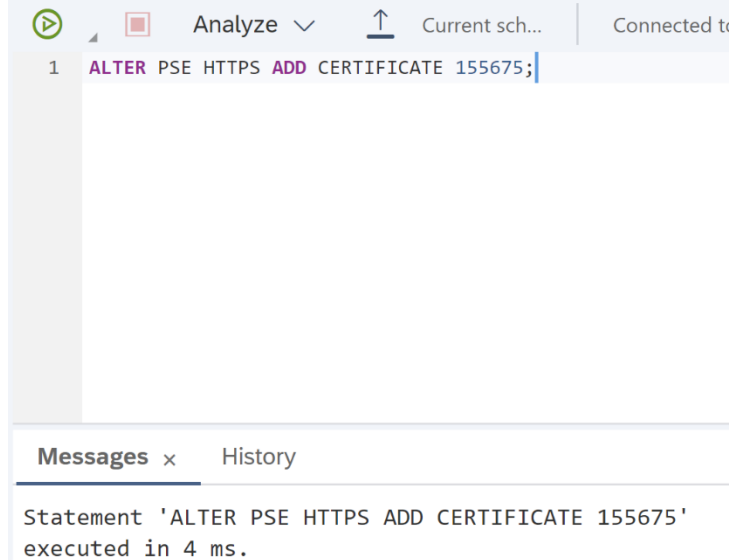
The screenshot shows the SQL console interface. At the top, there's a tab labeled 'SQL Console 1.sql'. Below it, a yellow banner indicates 'Production System'. The main area shows a SQL statement: `1 SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'S3';`. Below the statement, a 'Result' tab is active, displaying the result of the query. The result is a table with one row and one column, showing the CERTIFICATE_ID value 155675.

CERTIFICATE_ID
155675

7. Alter PSE HTTPS to add the S3 CERTIFICATE_ID by running the following statement in the SQL console:

```
ALTER PSE HTTPS ADD CERTIFICATE  
[CERTIFICATE_ID]
```

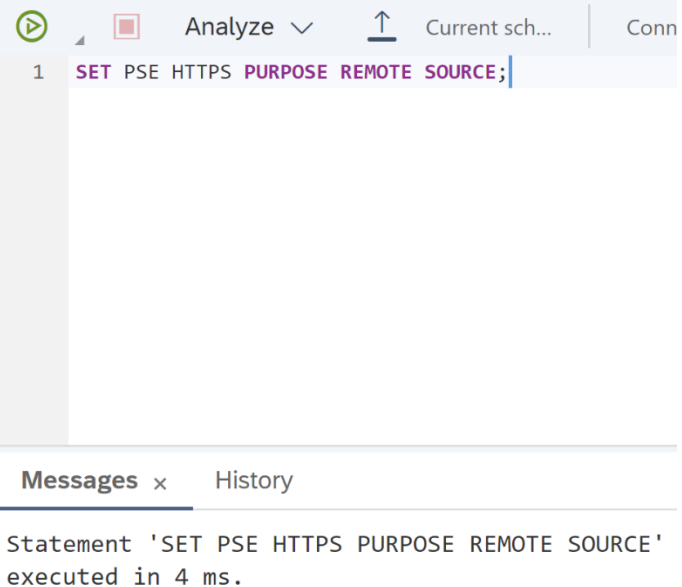
Note: replace [CERTIFICATE_ID] with the S3 CERTIFICATE_ID from previous step



The screenshot shows a SQL console interface. At the top, there is a toolbar with a green play button, a red square icon, a dropdown menu labeled 'Analyze', an upward arrow icon, and text labels 'Current sch...' and 'Connected to'. Below the toolbar, a text area contains the SQL statement: `1 ALTER PSE HTTPS ADD CERTIFICATE 155675;`. At the bottom of the console, there is a 'Messages' tab (active) and a 'History' tab. The 'Messages' tab displays the text: 'Statement 'ALTER PSE HTTPS ADD CERTIFICATE 155675' executed in 4 ms.'

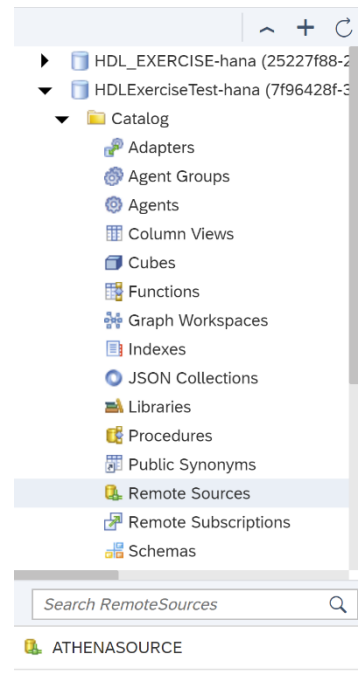
8. Set the purpose of PSE HTTPS to remote source by running the following statement:

```
SET PSE HTTPS PURPOSE REMOTE SOURCE;
```




The screenshot shows a SQL console interface. At the top, there is a toolbar with a green play button, a red square icon, a dropdown menu labeled 'Analyze', an upward arrow icon, and text labels 'Current sch...' and 'Conn'. Below the toolbar, a text area contains the SQL statement: `1 SET PSE HTTPS PURPOSE REMOTE SOURCE;`. At the bottom of the console, there is a 'Messages' tab (active) and a 'History' tab. The 'Messages' tab displays the text: 'Statement 'SET PSE HTTPS PURPOSE REMOTE SOURCE' executed in 4 ms.'

9. Now, verify that AthenaSource has been added as a remote source to the HANA Cloud instance by navigating to Catalog, and then Remote Sources. Double click on ATHENASOURCE.



10. Select the Schema *tpchsf01* from the drop down. Click on Search to reveal a list of remote objects.

SQL Console 1.sql x ATHENASOURCE x

 ATHENASOURCE Edit

Adapter Name: athena
Source Location: indexserver

Search

Database: <NULL> Schema: **tpchsf01** Object:

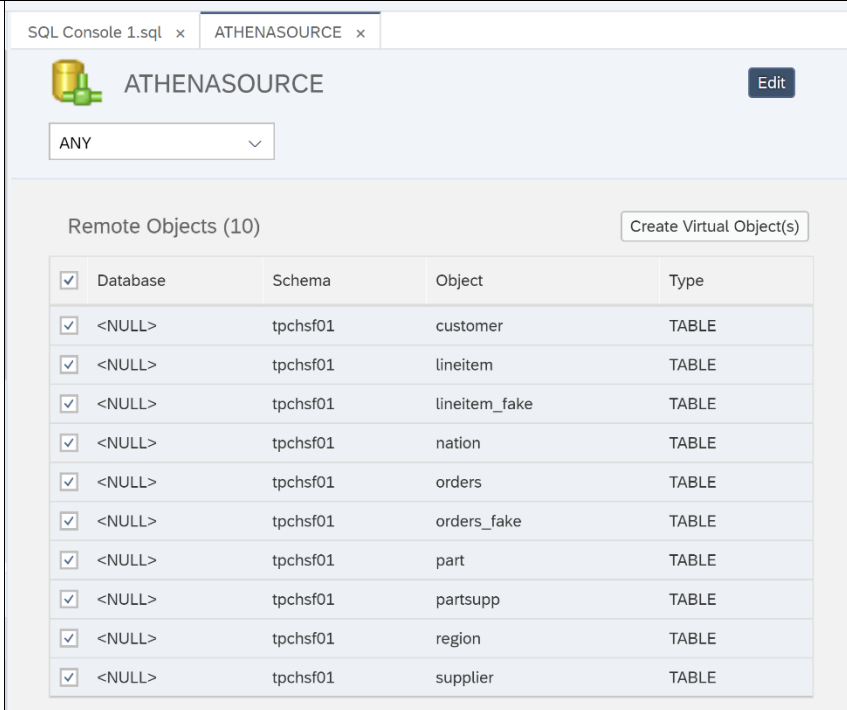
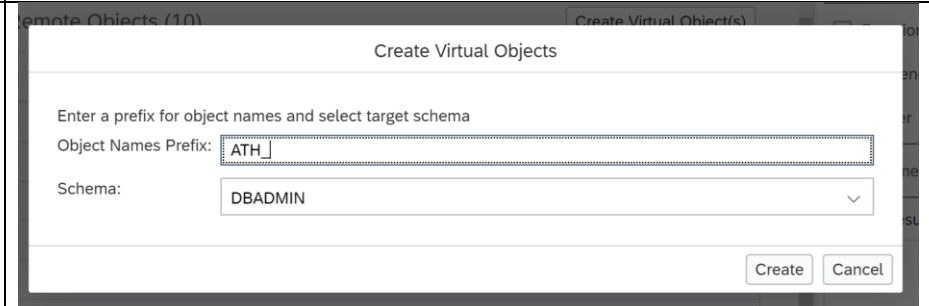
Type: ANY

Remote Objects (10) Create Virtual Object(s)

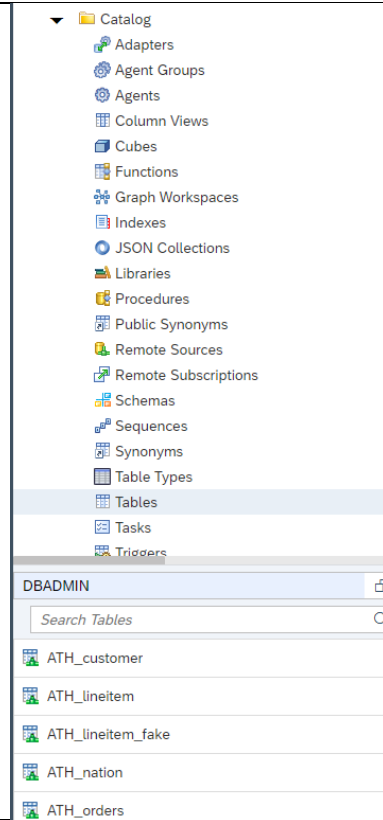
<input type="checkbox"/>	Database	Schema	Object	Type
<input type="checkbox"/>	<NULL>	tpchsf01	customer	TABLE
<input type="checkbox"/>	<NULL>	tpchsf01	lineitem	TABLE
<input type="checkbox"/>	<NULL>	tpchsf01	lineitem_fake	TABLE
<input type="checkbox"/>	<NULL>	tpchsf01	nation	TABLE
<input type="checkbox"/>	<NULL>	tpchsf01	orders	TABLE
<input type="checkbox"/>	<NULL>	tpchsf01	orders_fake	TABLE

Exercise 2. Create virtual tables

The goal of this exercise is to create virtual tables pointing to remote objects in ATHENASOURCE.

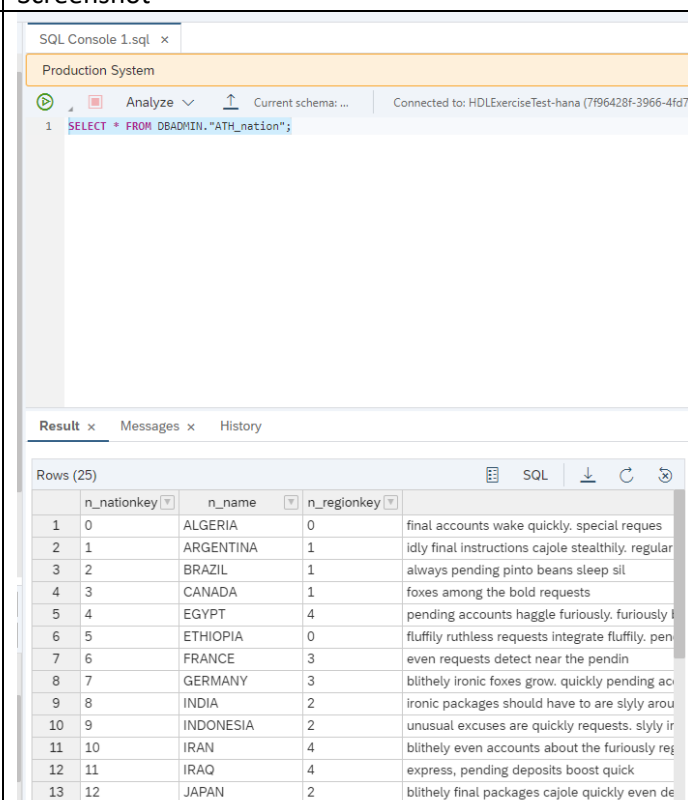
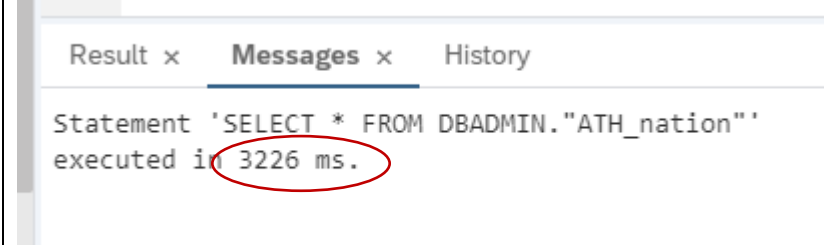
Step Explanation	Screenshot																																																						
1. Select all object in the remote source and click on the `Create Virtual Object(s)` button	 <p>The screenshot shows the ATHENASOURCE interface. At the top, there's a tab for 'ATHENASOURCE' and a dropdown menu set to 'ANY'. Below this, a section titled 'Remote Objects (10)' contains a table of objects. All objects in the table have their checkboxes selected. A button labeled 'Create Virtual Object(s)' is located in the top right corner of the table area.</p> <table><tr><th>Database</th><th>Schema</th><th>Object</th><th>Type</th></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>customer</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>lineitem</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>lineitem_fake</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>nation</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>orders</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>orders_fake</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>part</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>partsupp</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>region</td><td>TABLE</td></tr><tr><td><input checked="" type="checkbox"/></td><td><NULL></td><td>tpchsf01</td><td>supplier</td><td>TABLE</td></tr></table>	Database	Schema	Object	Type	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	customer	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	lineitem	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	lineitem_fake	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	nation	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	orders	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	orders_fake	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	part	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	partsupp	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	region	TABLE	<input checked="" type="checkbox"/>	<NULL>	tpchsf01	supplier	TABLE
Database	Schema	Object	Type																																																				
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	customer	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	lineitem	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	lineitem_fake	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	nation	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	orders	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	orders_fake	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	part	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	partsupp	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	region	TABLE																																																			
<input checked="" type="checkbox"/>	<NULL>	tpchsf01	supplier	TABLE																																																			
2. Set prefix as ATH_ and leave Schema as DBADMIN. Click on Create.	 <p>The screenshot shows a dialog box titled 'Create Virtual Objects'. It has two input fields: 'Object Names Prefix' with the value 'ATH_' and 'Schema' with the value 'DBADMIN'. At the bottom right, there are 'Create' and 'Cancel' buttons.</p>																																																						

3. Once the virtual objects are created, navigate to Catalog, Tables to verify the tables are created successfully.



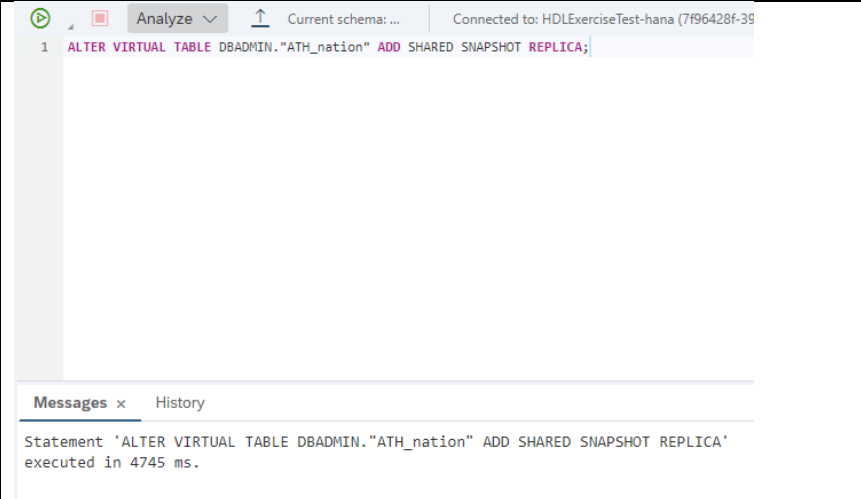
Exercise 3. Query Virtual Tables

The goal of this exercise is to query virtual tables for results and evaluate its performance

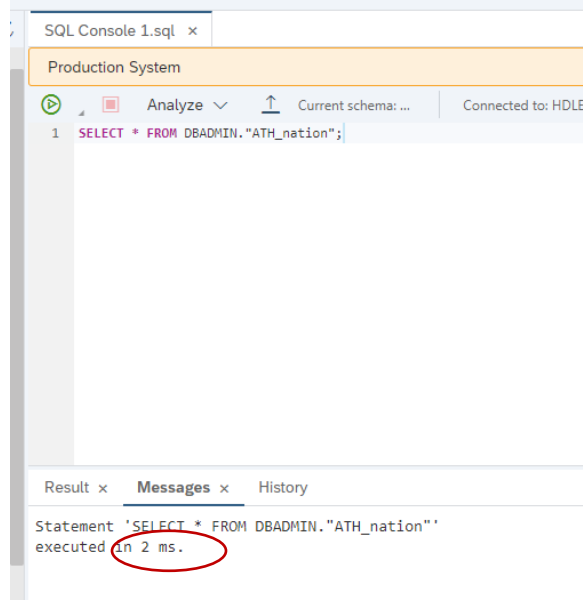
Step Explanation	Screenshot																																																								
<p>1. Select all the rows in the ATH_nation virtual table that we have just created by running the following statement:</p> <p>SELECT * FROM DBADMIN."ATH_nation";</p>	 <p>The screenshot shows the SQL Developer interface. The top pane displays the query: <code>SELECT * FROM DBADMIN."ATH_nation";</code>. The bottom pane shows the results of the query, which are 25 rows of data. The columns are <code>n_nationkey</code>, <code>n_name</code>, and <code>n_regionkey</code>. The data is as follows:</p> <table><tr><th></th><th>n_nationkey</th><th>n_name</th><th>n_regionkey</th></tr><tr><td>1</td><td>0</td><td>ALGERIA</td><td>0</td></tr><tr><td>2</td><td>1</td><td>ARGENTINA</td><td>1</td></tr><tr><td>3</td><td>2</td><td>BRAZIL</td><td>1</td></tr><tr><td>4</td><td>3</td><td>CANADA</td><td>1</td></tr><tr><td>5</td><td>4</td><td>EGYPT</td><td>4</td></tr><tr><td>6</td><td>5</td><td>ETHIOPIA</td><td>0</td></tr><tr><td>7</td><td>6</td><td>FRANCE</td><td>3</td></tr><tr><td>8</td><td>7</td><td>GERMANY</td><td>3</td></tr><tr><td>9</td><td>8</td><td>INDIA</td><td>2</td></tr><tr><td>10</td><td>9</td><td>INDONESIA</td><td>2</td></tr><tr><td>11</td><td>10</td><td>IRAN</td><td>4</td></tr><tr><td>12</td><td>11</td><td>IRAQ</td><td>4</td></tr><tr><td>13</td><td>12</td><td>JAPAN</td><td>2</td></tr></table>		n_nationkey	n_name	n_regionkey	1	0	ALGERIA	0	2	1	ARGENTINA	1	3	2	BRAZIL	1	4	3	CANADA	1	5	4	EGYPT	4	6	5	ETHIOPIA	0	7	6	FRANCE	3	8	7	GERMANY	3	9	8	INDIA	2	10	9	INDONESIA	2	11	10	IRAN	4	12	11	IRAQ	4	13	12	JAPAN	2
	n_nationkey	n_name	n_regionkey																																																						
1	0	ALGERIA	0																																																						
2	1	ARGENTINA	1																																																						
3	2	BRAZIL	1																																																						
4	3	CANADA	1																																																						
5	4	EGYPT	4																																																						
6	5	ETHIOPIA	0																																																						
7	6	FRANCE	3																																																						
8	7	GERMANY	3																																																						
9	8	INDIA	2																																																						
10	9	INDONESIA	2																																																						
11	10	IRAN	4																																																						
12	11	IRAQ	4																																																						
13	12	JAPAN	2																																																						
<p>2. Note the performance of the statement which queries a virtual table (3226 ms).</p>	 <p>The screenshot shows the Messages pane in SQL Developer. It displays the following message: <code>Statement 'SELECT * FROM DBADMIN."ATH_nation"' executed in 3226 ms.</code> The value <code>3226 ms.</code> is circled in red.</p>																																																								

Exercise 4. Turn on Table Replication

The goal of this exercise is to execute the statement which enables table replication – this will move data from a remote source into HANA Cloud's main memory.

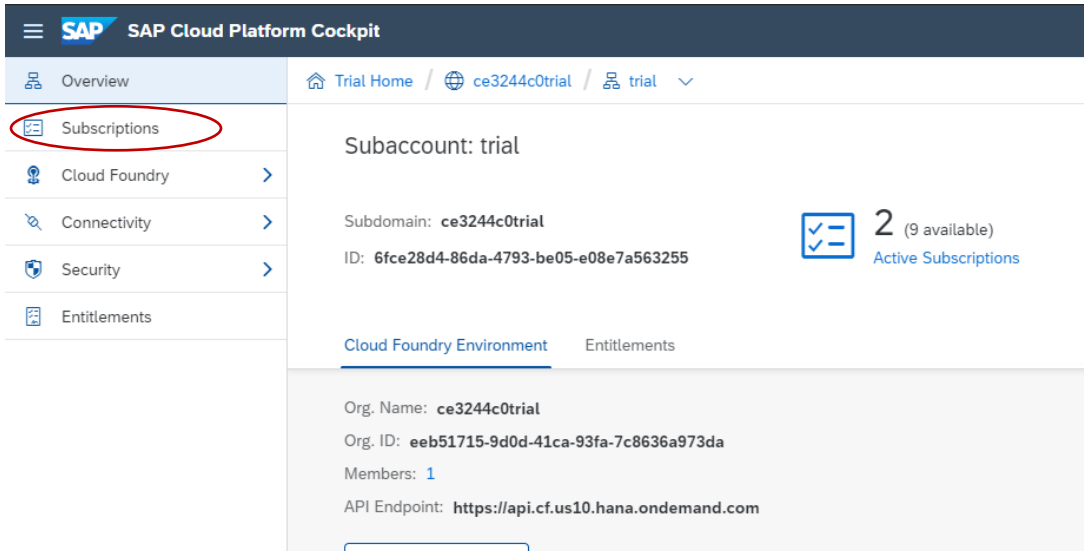
Step Explanation	Screenshot
<p>1. Replicate the table in HANA Cloud by running the following statement:</p> <p><code>ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA;</code></p> <p>Note: to avoid excessive memory usage this statement should ideally be used with smaller tables</p>	 The screenshot shows the SAP HANA Cloud SQL console interface. At the top, there's a toolbar with icons for play, stop, and analyze, along with a dropdown menu. The status bar indicates 'Current schema: ...' and 'Connected to: HDLExerciseTest-hana (7f96428f-39)'. The main area displays a single SQL statement: <code>1 ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA;</code> . Below the statement, there's a 'Messages' tab and a 'History' tab. The 'Messages' tab is active, showing a confirmation message: 'Statement 'ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA' executed in 4745 ms.'

Exercise 5. Re-Run Query and Note Performance Difference

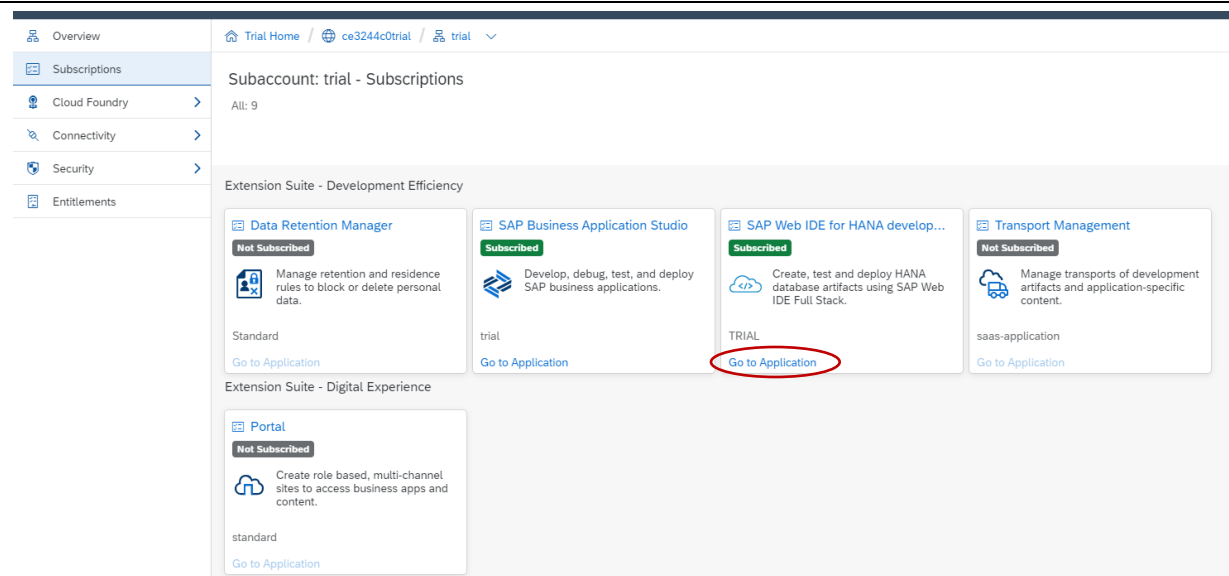
Step Explanation	Screenshot
<p>1. Rerun the statement <code>SELECT * FROM DBADMIN."ATH_nation";</code></p> <p>Note: since the table is now in HANA Cloud's main memory, the query should execute much faster. The performance difference between Exercise 3 and Exercise 5, while executing the same query, went from 3226 ms to 2 ms.</p>	 <p>The screenshot shows the SQL Console interface. At the top, there's a tab for 'SQL Console 1.sql'. Below it, a banner for 'Production System' is visible. The main area shows the SQL statement: <code>1 SELECT * FROM DBADMIN."ATH_nation";</code>. At the bottom, there's a 'Messages' tab that displays the execution result: 'Statement 'SELECT * FROM DBADMIN."ATH_nation"' executed in 2 ms.' The text 'in 2 ms.' is circled in red.</p>

Operating HANA with Native Storage Extension

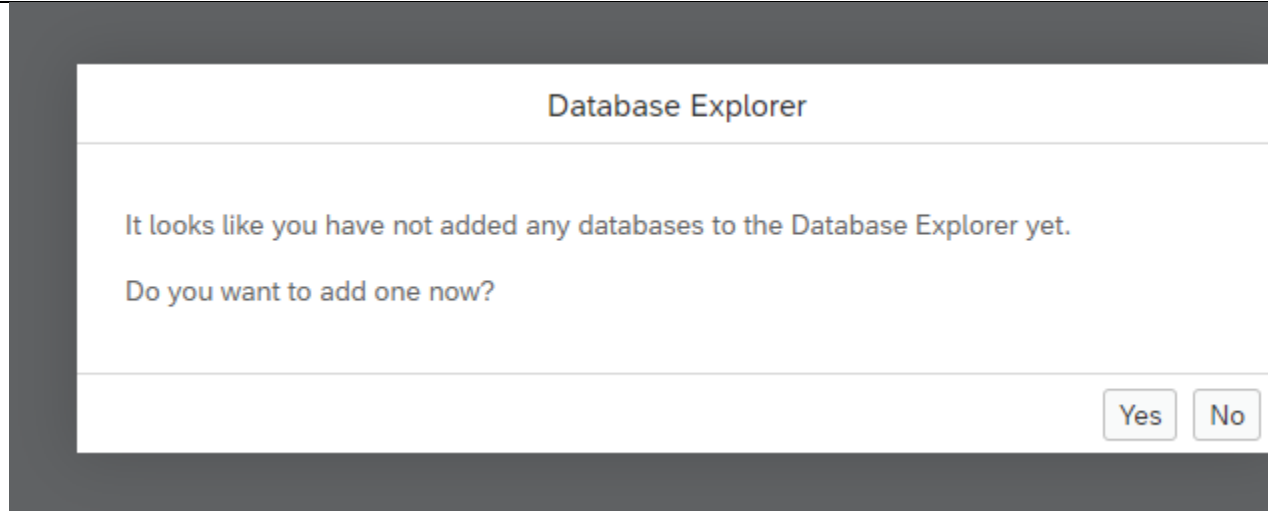
Prerequisite. Open WebIDE and connect to HANA system

Step Explanation	Screenshot
1. Navigate to the subaccount level, click on Subscriptions on the left	 <p>The screenshot displays the SAP Cloud Platform Cockpit interface. On the left sidebar, the 'Subscriptions' menu item is highlighted with a red circle. The main content area shows the 'Subaccount: trial' page. It includes a breadcrumb trail: Trial Home / ce3244c0trial / trial. The page displays the subdomain 'ce3244c0trial' and the ID '6fce28d4-86da-4793-be05-e08e7a563255'. A checkmark icon indicates that 2 subscriptions are active out of 9 available. Below this, there are tabs for 'Cloud Foundry Environment' and 'Entitlements'. The 'Cloud Foundry Environment' tab is selected, showing details for the organization: Org. Name: ce3244c0trial, Org. ID: eeb51715-9d0d-41ca-93fa-7c8636a973da, Members: 1, and API Endpoint: https://api.cf.us10.hana.ondemand.com.</p>

2. Find SAP Web IDE tile, and click on Go to Application



3. When prompted to add a database to the database explorer, click on yes



4. Navigate to your HANA Cloud instance and copy the end point

SAP HANA Instances

[HDL_Exercise-hana](#)

SAP HANA

Running

Memory

30 GB

CPU

2 vCPUs

Storage

120 GB

Data Lakes

HDL_Exercise-rdl

Admin User

DBADMIN

Description

None

Endpoint

1b132f97-cf25-45dd-a098-699a7b2418e9.h...



Stop

Open In ▾

5. Select database type as SAP HANA Database, and paste the end point into Host field. Remove the port number from host url, and input it as Port number.

Note: the port number is typically the last 3 digits of the host url. For example, for the endpoint:

1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.com:443

Input:

1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.com as the **Host** and 443 as the **Port Number**

Add Database

Database Type:

SAP HANA Database

*Host:

1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.com

*Identifier:

☐ Instance number

☒ Port number

443

*User:

*Password:

☐ Save password (stored in the SAP HANA secure store)

☐ Connect to the database securely using TLS/SSL (prevents data eavesdropping)

☐ Verify the server's certificate using the trusted certificate below

Advanced Options:

Name to Show in Display:

OK

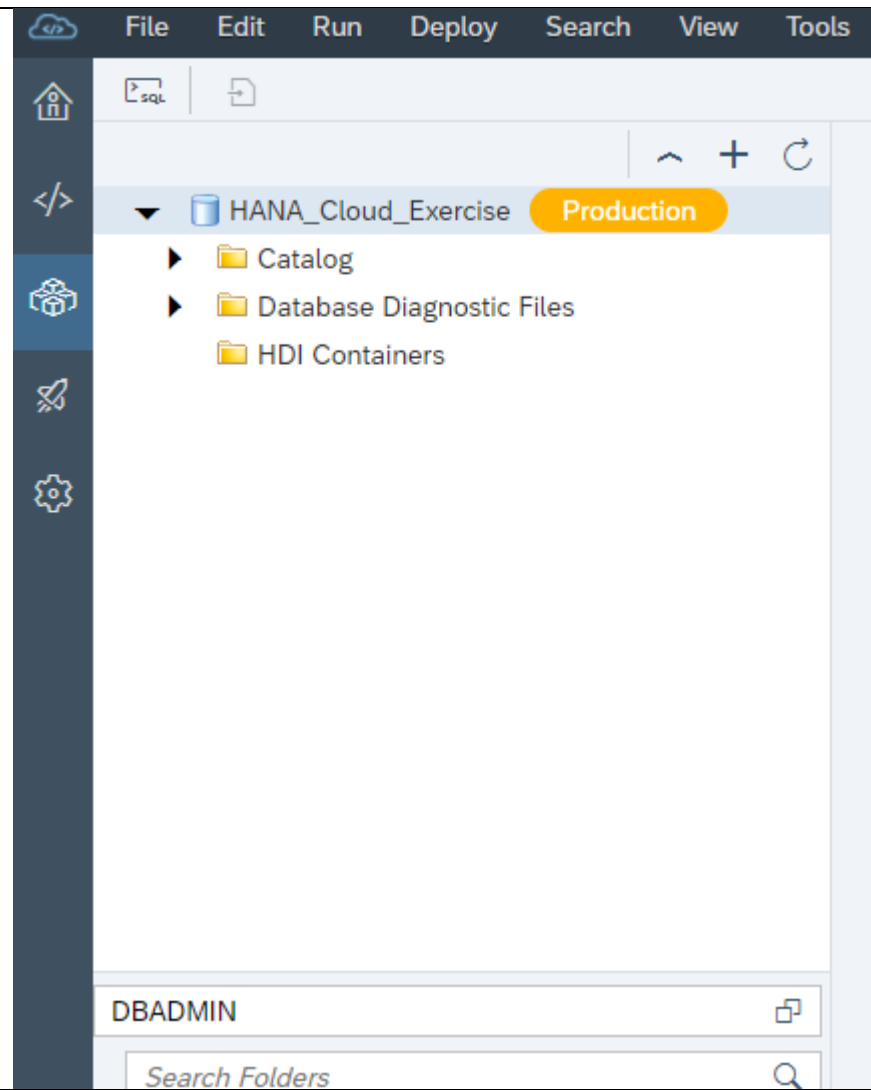
Cancel

6. Enter DBADMIN as user and the password. Select Save password and Connect to the database securely using TLS/SSL. Leave the third one unchecked. Optionally edit “Name to Show in Display” to something easily recognizable. Click on OK.

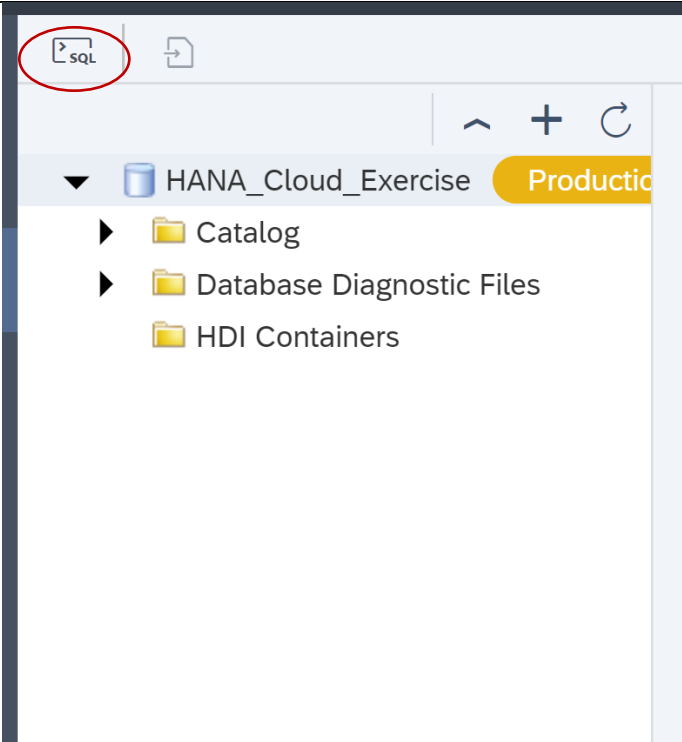
The screenshot shows the 'Add Database' dialog box with the following fields and options:

- Database Type:** SAP HANA Database
- *Host:** 1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.cor
- *Identifier:** Instance number (unchecked) and Port number (checked) 443
- *User:** DBADMIN
- *Password:** (masked with dots)
- Options:**
 - ☒ Save password (stored in the SAP HANA secure store)
 - ☒ Connect to the database securely using TLS/SSL (prevents data eavesdropping)
 - ☐ Verify the server's certificate using the trusted certificate below
- Advanced Options:** (empty field)
- Name to Show in Display:** HANA_Cloud_Exercise (circled in red)
- Buttons:** OK and Cancel

7. You should now see the database

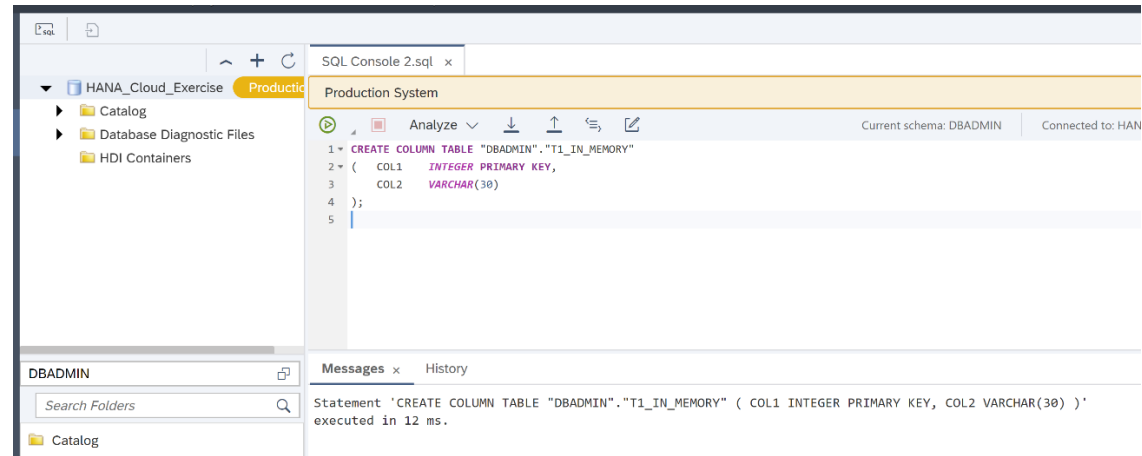


Exercise 1. Execute SQL to create a table in NSE storage

Step Explanation	Screenshot
<p>1. Open a SQL console by selecting the database and clicking on the SQL icon</p>	 <p>The screenshot shows the SAP HANA Cloud console interface. At the top, there is a navigation bar with a blue 'SQL' icon circled in red. Below the navigation bar, the 'HANA_Cloud_Exercise' database is selected, and a 'Production' status badge is visible. The main content area displays a list of folders: 'Catalog', 'Database Diagnostic Files', and 'HDI Containers'.</p>

2. First create a table in HANA Cloud main memory by running the following statement:

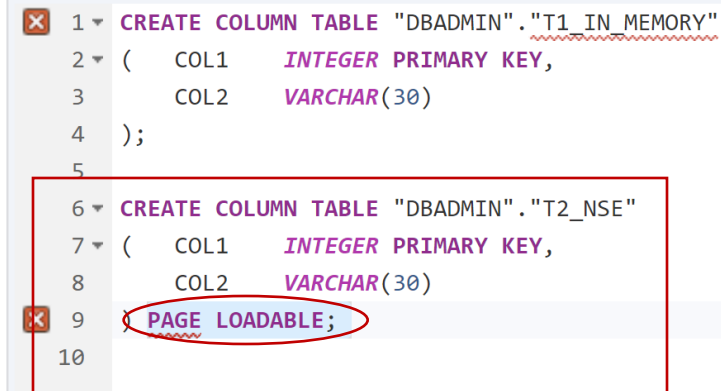
```
CREATE COLUMN TABLE
"DBADMIN"."T1_IN_MEMORY"
(
    COL1    INTEGER
    PRIMARY KEY,
    COL2    VARCHAR(30)
);
```



3. Now create an NSE table by first copying and pasting the following statement to the SQL console:






```
CREATE COLUMN TABLE
"DBADMIN"."T2_NSE"
(
    COL1    INTEGER
    PRIMARY KEY,
    COL2    VARCHAR(30)
) PAGE LOADABLE;
```

Note: notice the only difference in syntax when creating an NSE table compared to an in-memory table is the PAGE LOADABLE clause



4. Select and highlight the statement and press the run button to execute it

Production System

Analyze      Current schema: DBADMIN Connected to: H.

```
1 CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY"
2 ( COL1 INTEGER PRIMARY KEY,
3   COL2 VARCHAR(30)
4 );
5
6 CREATE COLUMN TABLE "DBADMIN"."T2_NSE"
7 ( COL1 INTEGER PRIMARY KEY,
8   COL2 VARCHAR(30)
9 ) PAGE LOADABLE;
10
```

Messages x History

Statement 'CREATE COLUMN TABLE "DBADMIN"."T2_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30)) PAGE ...' executed in 7 ms.

5. Insert values into both tables by running the following statements:

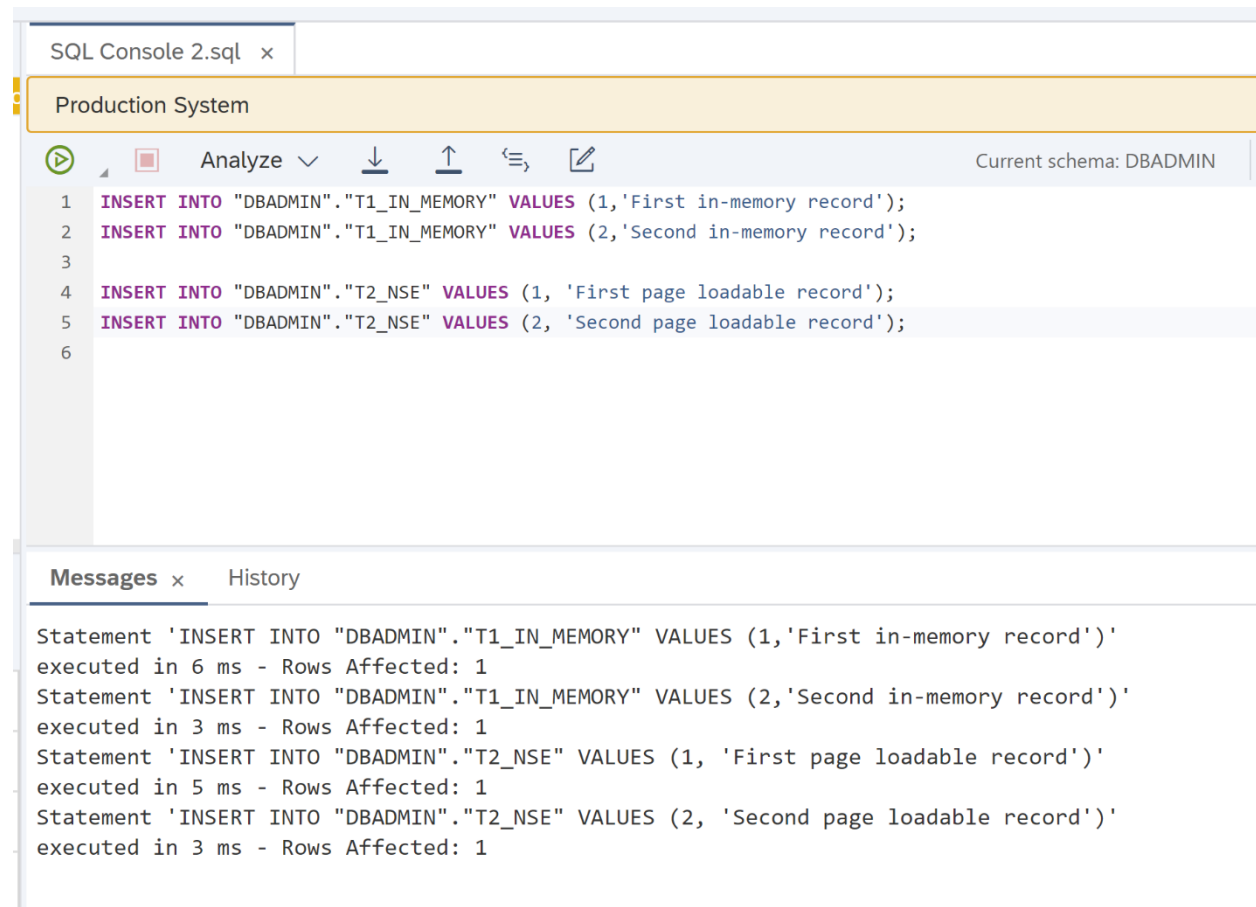
```
INSERT INTO
"DBADMIN"."T1_IN_MEMORY"
VALUES (1,'First in-memory
record');
INSERT INTO
"DBADMIN"."T1_IN_MEMORY"
VALUES (2,'Second in-memory
record');
```

```
INSERT INTO
"DBADMIN"."T2_NSE" VALUES
(1, 'First page loadable record');
INSERT INTO
"DBADMIN"."T2_NSE" VALUES
(2, 'Second page loadable
record');
```

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.

6. Query the tables by running the following SELECT statements:

```
SELECT * FROM
"DBADMIN"."T1_IN_MEMORY"
WHERE COL1 = 2;
```



The screenshot shows a SQL console window titled "SQL Console 2.sql". The main area displays a script for the "Production System" with the current schema set to "DBADMIN". The script contains four INSERT statements: two for "T1_IN_MEMORY" and two for "T2_NSE". Below the script, the "Messages" tab shows the execution results for each statement, including the SQL text, execution time, and the number of rows affected.

```
SQL Console 2.sql x
Production System
Current schema: DBADMIN
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

Messages x History
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 5 ms - Rows Affected: 1
Statement 'INSERT INTO "DBADMIN"."T2_NSE" VALUES (2, 'Second page loadable record')'
executed in 3 ms - Rows Affected: 1
```

SELECT * FROM
"DBADMIN"."T2_NSE" WHERE
COL1 = 1;

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".

The screenshot displays the SQL Console interface. At the top, a tab labeled "SQL Console 1.sql" is active. Below it, a yellow banner indicates the "Production System" environment. The current schema is set to "DBADMIN". The SQL editor contains two queries:

```
1 SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;  
2  
3 SELECT * FROM "DBADMIN"."T2_NSE" WHERE COL1 = 1;  
4
```

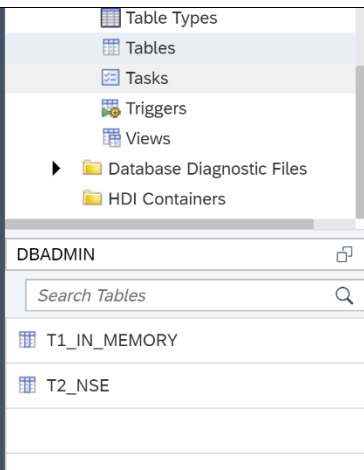
Below the editor, the "Result 1" tab is selected, showing a table with two columns, COL1 and COL2. The first row contains the value 2 in COL1 and "Second in-memory record" in COL2.

	COL1	COL2
1	2	Second in-memory record

The "Result 2" tab is also visible, showing a table with two columns, COL1 and COL2. The first row contains the value 1 in COL1 and "First page loadable record" in COL2.

	COL1	COL2
1	1	First page loadable record

7. The tables can also be viewed from the UI by navigating to Catalog and then Tables

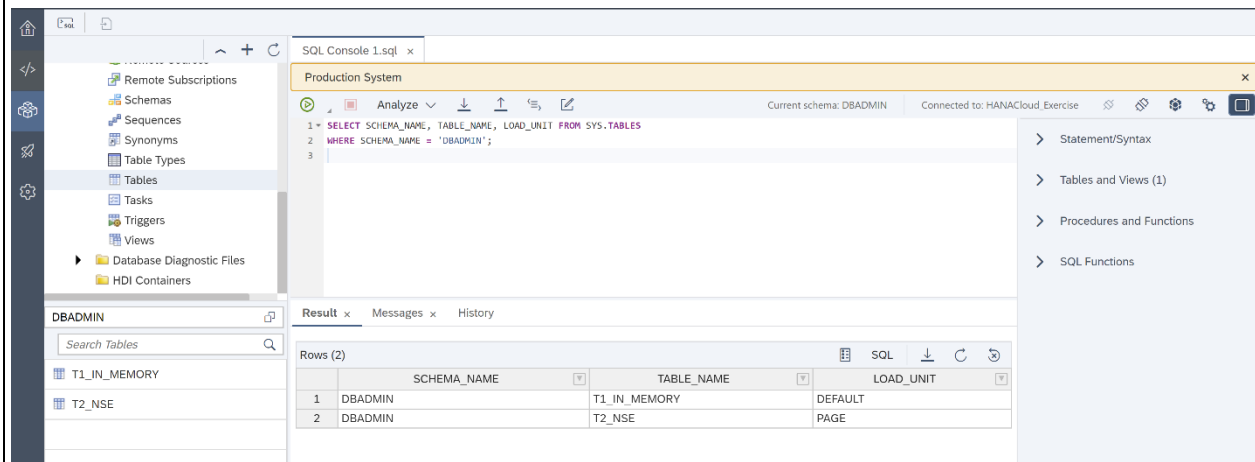


Exercise 2. View the Load Unit configuration

1. Clear the SQL console. Then copy and paste the following statement into the console. Run the statement.

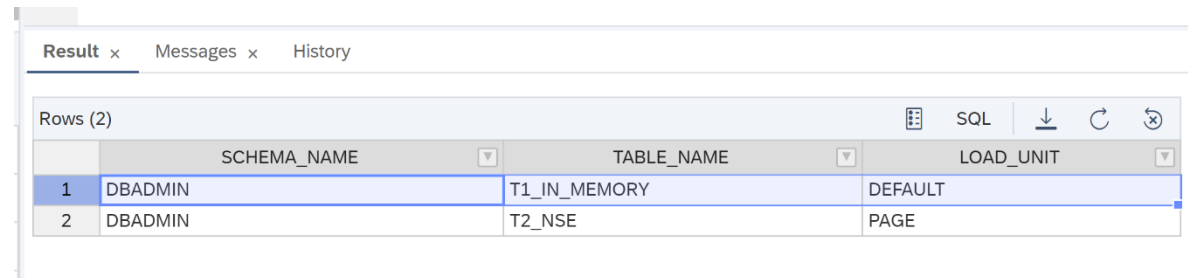
```
SELECT SCHEMA_NAME,  
TABLE_NAME, LOAD_UNIT FROM  
SYS.TABLES  
WHERE SCHEMA_NAME =  
'DBADMIN';
```

Note: for this exercise, we are specifically looking at the SCHEMA_NAME, TABLE_NAME, and LOAD_UNIT columns in the SYS.TABLES view.



2. Click on “1”.

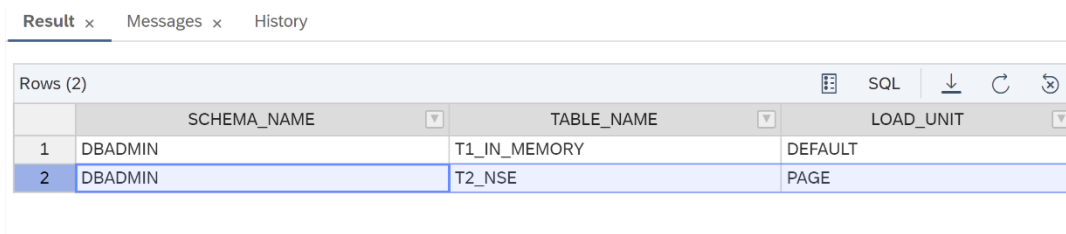
Since we did not explicitly assign a LOAD UNIT in the CREATE TABLE statement for the T1_IN_MEMORY table, it shows a LOAD UNIT of **DEFAULT**. The default LOAD UNIT for a table is **COLUMN**. If we had explicitly specified **COLUMN LOADABLE** in the CREATE TABLE statement for the T1_IN_MEMORY table, then the LOAD UNIT would show as **COLUMN**.



Result x Messages x History			
Rows (2)			
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT
1	DBADMIN	T1_IN_MEMORY	DEFAULT
2	DBADMIN	T2_NSE	PAGE

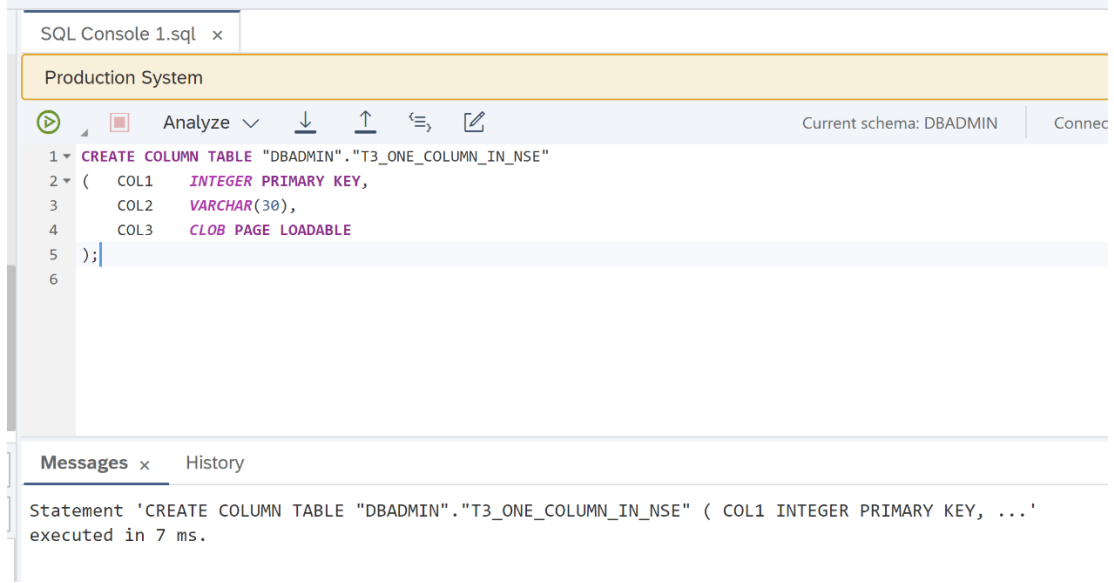
3. click on “2”.

The LOAD UNIT for the **T2_NSE** table is reported as **PAGE** indicating that this is a page loadable table.



Result x Messages x History			
Rows (2)			
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT
1	DBADMIN	T1_IN_MEMORY	DEFAULT
2	DBADMIN	T2_NSE	PAGE

Exercise 3. Create and use a table with a specific column configured as page loadable

Step Explanation	Screenshot
<p>1. 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>The screenshot displays the SQL Developer interface. The top pane shows the SQL Console with a tab for 'SQL Console 1.sql'. The 'Production System' schema is selected. The SQL statement being executed is: <code>CREATE COLUMN TABLE "DBADMIN"."T3_ONE_COLUMN_IN_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30), COL3 CLOB PAGE LOADABLE);</code>. The bottom pane shows the Messages tab, indicating that the statement was executed successfully in 7 ms.</p>

2. Now run the following statement to insert values into T3_ONE_COLUMN_IN_NSE:

```
INSERT INTO
"DBADMIN"."T3_ONE_COLUMN_IN_NSE"
VALUES (1, 'Col2 is COLUMN LOADABLE',
'Col3 is a CLOB to handle really long
verbose content');
```

```
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');
```

The screenshot shows the SQL Console interface with two statements executed. The first statement inserts a row with ID 1, and the second statement inserts a row with ID 2. The Messages pane shows the execution details for both statements.

```
SQL Console 1.sql x
Production System
Current schema: DBADMIN Connected to: HANAC
1 INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (1, 'Col2 is COLUMN LOADABLE', 'Col3 is a CLOB to handle really long verb
2
3 INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (2, 'Col2 is always in memory', 'Col3 can be read into the buffer cache w
4

Messages x History
Statement 'INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (1, 'Col2 is COLUMN LOADABLE', 'Col3 is a CLOB
...
'
executed in 6 ms - Rows Affected: 1
Statement 'INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (2, 'Col2 is always in memory', 'Col3 can be
...
'
executed in 2 ms - Rows Affected: 1
```

3. Query the table by running the following statement:

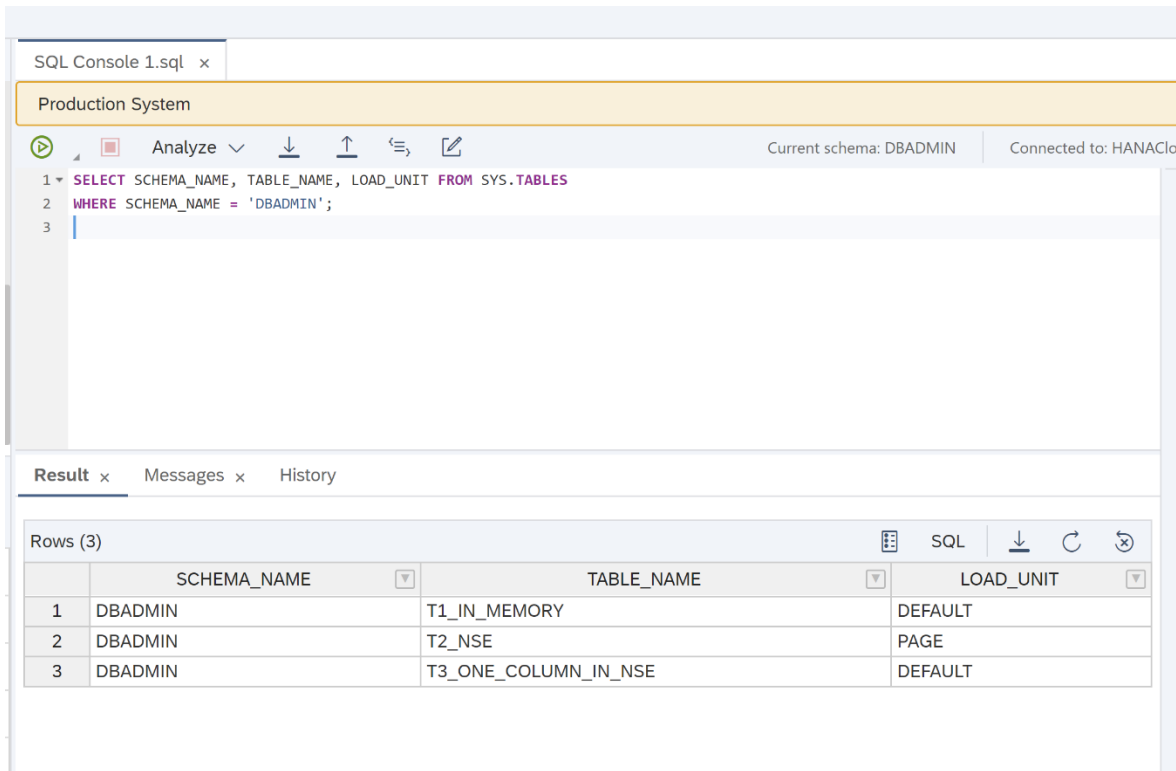
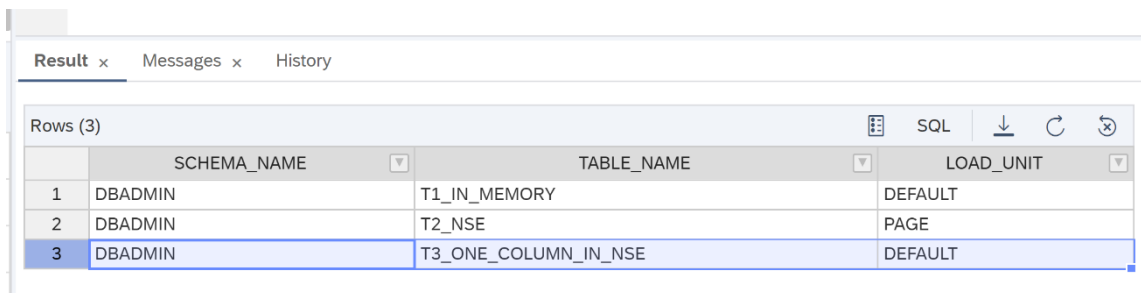
```
SELECT * FROM
"DBADMIN"."T3_ONE_COLUMN_IN_NSE";
```

The screenshot shows the SQL Console interface with a SELECT query executed. The Results pane displays two rows of data from the T3_ONE_COLUMN_IN_NSE table.

```
SQL Console 1.sql x
Production System
Current schema: DBADMIN Connected to: HANAC
1 SELECT * FROM "DBADMIN"."T3_ONE_COLUMN_IN_NSE";

Result x Messages x History
Rows (2)
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
```

Exercise 4. View the load unit configuration of individual columns

<div>Step Explanation</div> <div>1. Query the SYS.TABLES view to see what LOAD UNIT is configured at the table level by running the following statement:</div> <div>SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES WHERE SCHEMA_NAME = 'DBADMIN';</div>	<div>Screenshot</div> <div><p>The screenshot shows the SQL Console interface. The top bar indicates the current schema is DBADMIN and the connection is to HANAClo. The query editor contains the following SQL statement:</p><pre>1 SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES 2 WHERE SCHEMA_NAME = 'DBADMIN'; 3</pre><p>The result set is displayed below the query editor, showing 3 rows of data:</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></div>		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														
<div>2. 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.</div>	<div><p>The screenshot shows the same SQL Console interface as the previous one, but with the third row of the result set highlighted in blue. The result set is as follows:</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></div>		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														

3. To see the LOAD UNIT configuration at a **column level**, run the following statement:

```
SELECT SCHEMA_NAME,  
TABLE_NAME, COLUMN_NAME,  
LOAD_UNIT FROM  
SYS.M_CS_COLUMNS  
WHERE SCHEMA_NAME =  
'DBADMIN';
```

SQL Console 1.sql x

Production System

Analyze v | Download | Upload | SQL | Current schema: DBADMIN | Connected to: HANAC

```
1 SELECT SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, LOAD_UNIT FROM SYS.M_CS_COLUMNS  
2 WHERE SCHEMA_NAME = 'DBADMIN';  
3
```

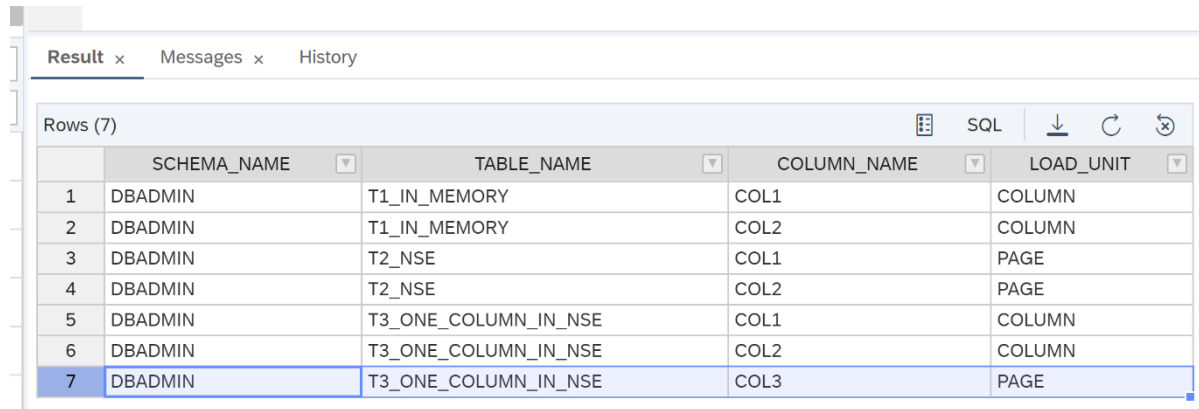
Result x | Messages x | History

Rows (7)

	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

4. 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.

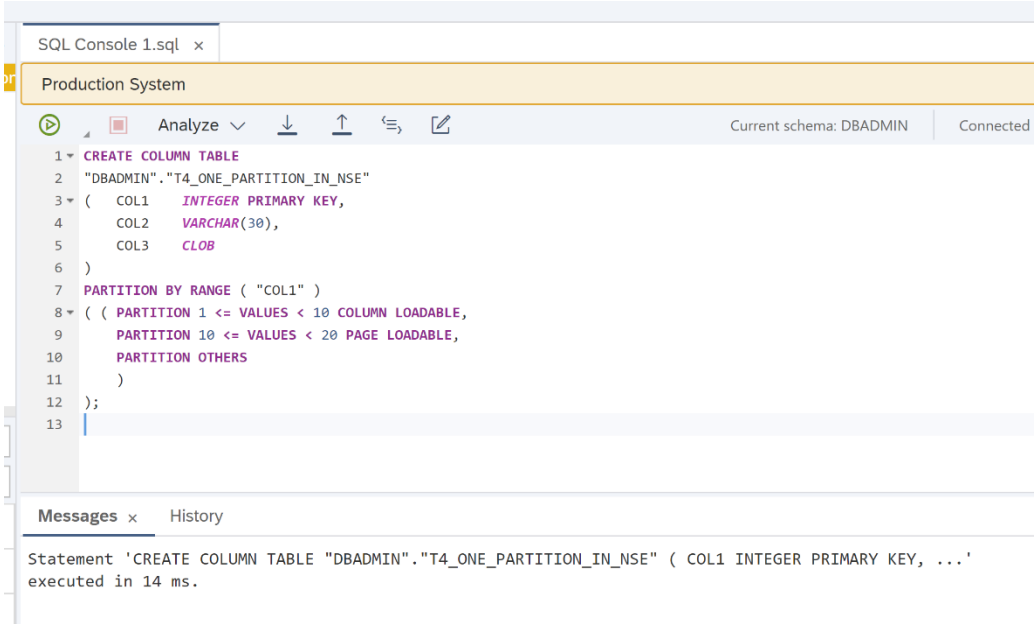
Also note that the **LOAD UNIT** for the first 2 columns of **T3_ONE_COLUMN_IN_NSE** (row 6 and 7) are shown as **COLUMN** because that is the default LOAD UNIT and no other LOAD UNIT was specified for these columns in the CREATE TABLE statement.



The screenshot shows a database query result window with the following data:

	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

Exercise 5. Create and use a table with a specific partition configured as page loadable

Step Explanation	Screenshot
<p>1. Run the following statement to create a table:</p> <pre>CREATE COLUMN TABLE "DBADMIN"."T4_ONE_PARTITION_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 displays the SQL Console interface. The top bar shows 'SQL Console 1.sql' and 'Production System'. The main area contains the following SQL code:</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> <p>The bottom panel shows the 'Messages' tab with the following message:</p> <pre>Statement 'CREATE COLUMN TABLE "DBADMIN"."T4_ONE_PARTITION_IN_NSE" (COL1 INTEGER PRIMARY KEY, ...' executed in 14 ms.</pre>

2. 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**.

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**.

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**.

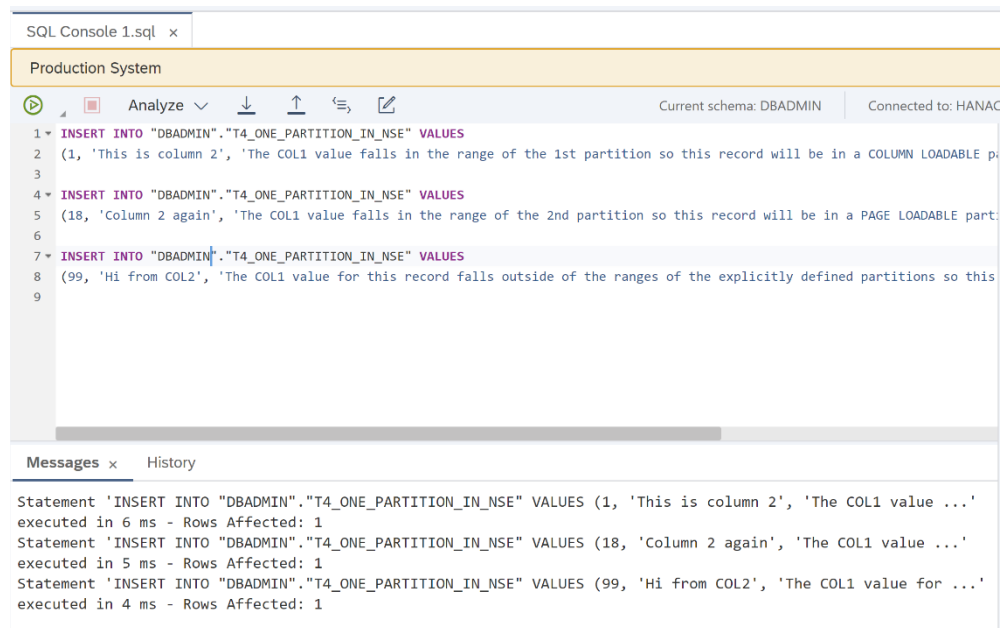
```
1 CREATE COLUMN TABLE
2 "DBADMIN"."T4_ONE_PARTITION_IN_NSE"
3 ( COL1    INTEGER PRIMARY KEY,
4   COL2    VARCHAR(30),
5   COL3    CLOB
6 )
7 PARTITION BY RANGE ( "COL1" )
8 ( ( PARTITION 1 <= VALUES < 10 COLUMN LOADABLE,
9   PARTITION 10 <= VALUES < 20 PAGE LOADABLE,
10  PARTITION OTHERS
11  )
12 );
13
```

3. Insert a record into each of the 3 partitions by running the following statement:

```
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. ');
```

```
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');
```

```
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');
```



The screenshot shows a SQL console window titled "SQL Console 1.sql x" with a tab for "Production System". The console is connected to a HANAC database with the current schema set to "DBADMIN". It displays three SQL statements and their execution results:

```
1 INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES
2 (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 p
3
4 INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES
5 (18, 'Column 2 again', 'The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE part
6
7 INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES
8 (99, 'Hi from COL2', 'The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this
9
```

The "Messages" pane at the bottom shows the execution details for each statement:

- Statement 'INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (1, 'This is column 2', 'The COL1 value ...' executed in 6 ms - Rows Affected: 1
- Statement 'INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (18, 'Column 2 again', 'The COL1 value ...' executed in 5 ms - Rows Affected: 1
- Statement 'INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES (99, 'Hi from COL2', 'The COL1 value for ...' executed in 4 ms - Rows Affected: 1

4. Select from the table by running the following statement:

```
SELECT * FROM  
"DBADMIN"."T4_ONE_PARTITION_IN_NSE";
```

SQL Console 1.sql x

Production System

Analyze v Download Up Arrow Left Arrow Right Arrow Edit

Current schema: DBADMIN Connected to: HANA

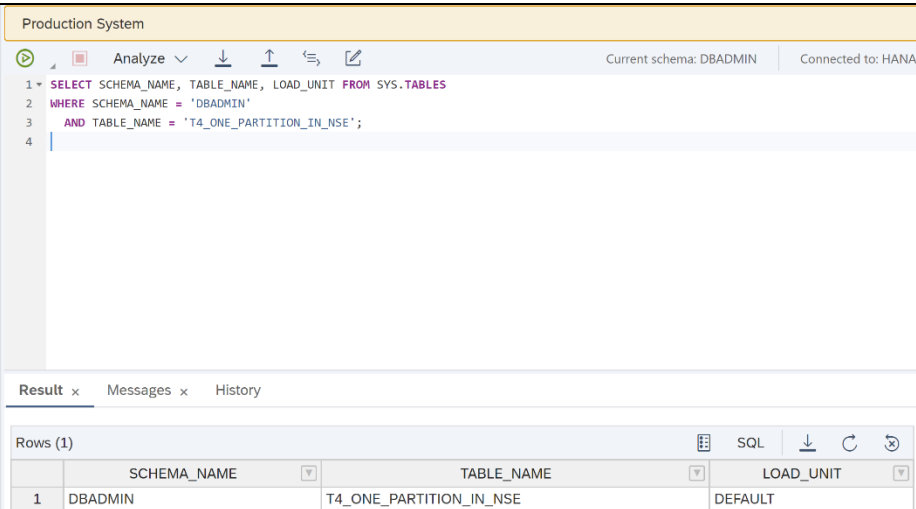
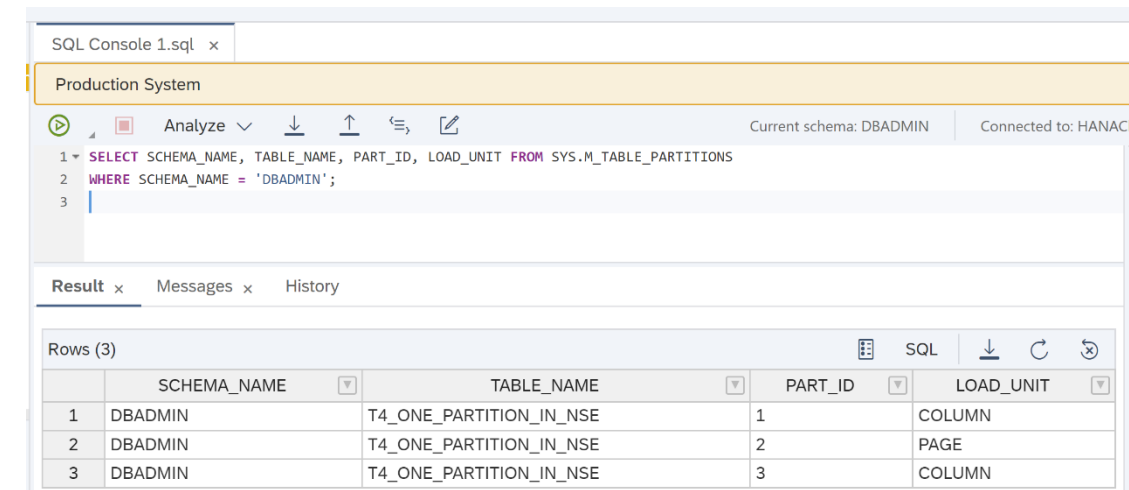
```
1 SELECT * FROM "DBADMIN"."T4_ONE_PARTITION_IN_NSE";
```

Result x Messages x History

Rows (3)

	COL1	COL2	COL3
1	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 pa
2	18	Column 2 again	The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE parti
3	99	Hi from COL2	The COL1 value for this record falls outside of the ranges of the explicitly defined partitions so this reco

Exercise 6. View the load unit configuration of individual partitions

Step Explanation	Screenshot																				
<p>1. 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>Production System</p> <p>Current schema: DBADMIN Connected to: HANA</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>Result x Messages x History</p> <table><tr><th>Rows (1)</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>	Rows (1)	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT	1	DBADMIN	T4_ONE_PARTITION_IN_NSE	DEFAULT												
Rows (1)	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT																		
1	DBADMIN	T4_ONE_PARTITION_IN_NSE	DEFAULT																		
<p>2. 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>SQL Console 1.sql x</p> <p>Production System</p> <p>Current schema: DBADMIN Connected to: HANA</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>Result x Messages x History</p> <table><tr><th>Rows (3)</th><th>SCHEMA_NAME</th><th>TABLE_NAME</th><th>PART_ID</th><th>LOAD_UNIT</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>	Rows (3)	SCHEMA_NAME	TABLE_NAME	PART_ID	LOAD_UNIT	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
Rows (3)	SCHEMA_NAME	TABLE_NAME	PART_ID	LOAD_UNIT																	
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																	

3. 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**.

The LOAD UNIT of the second partition was specifically configured as **PAGE LOADABLE** in the CREATE TABLE statement, so it is shown as **PAGE**.

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).

SQL	↓	↺	ⓧ
▼	LOAD_UNIT		▼
	COLUMN		
	PAGE		
	COLUMN		