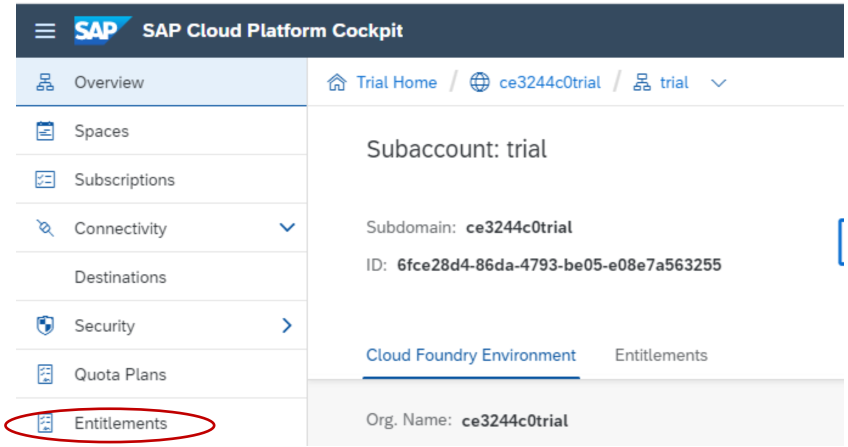
DAT 161 – Introduction to SAP HANA Cloud

Contents

HANA Cloud Hands on exercise

Pre-requisite

a. 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

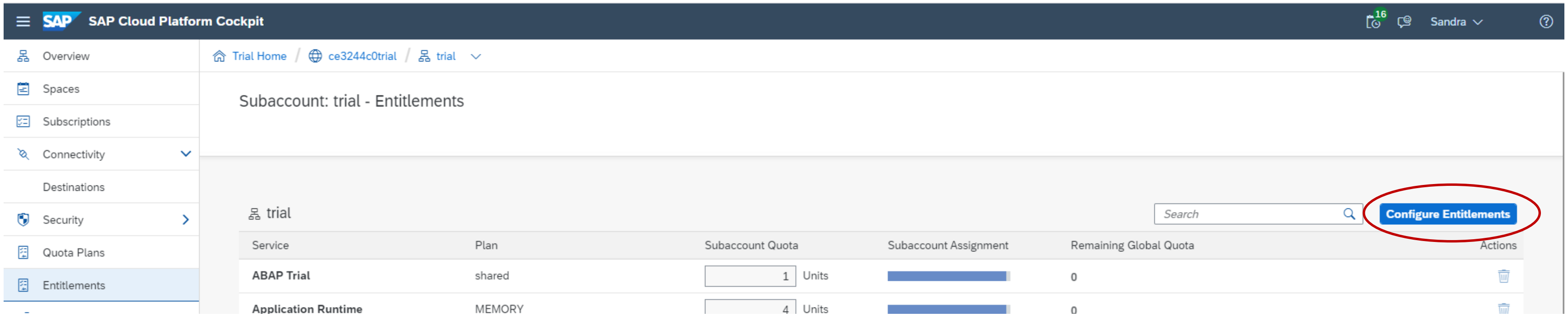


Subaccount: trial

Subdomain: ce3244c0trial

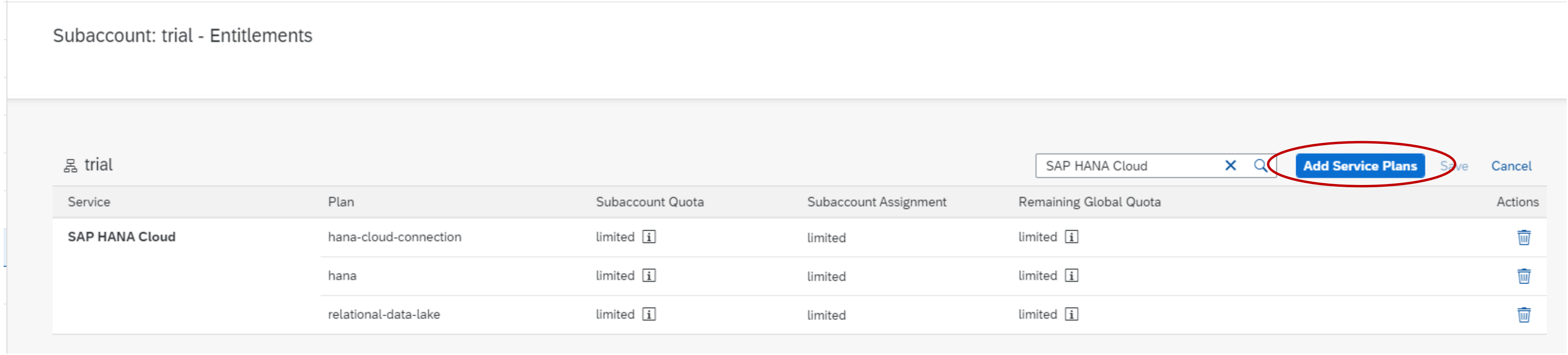
ID: 6fce28d4-86da-4793-be05-e08e7a563255

Org. Name: ce3244c0trial



Subaccount: trial - Entitlements

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

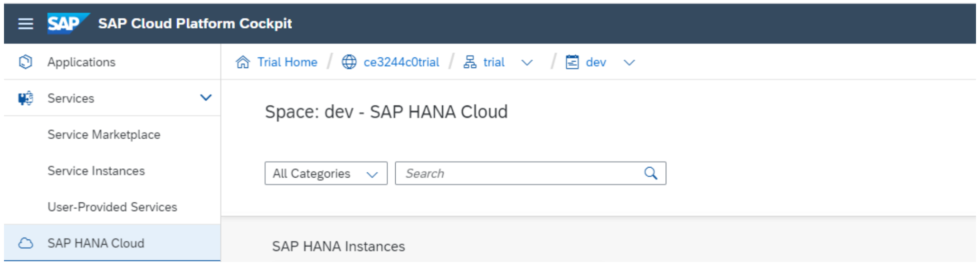


Subaccount: trial - Entitlements

trial

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



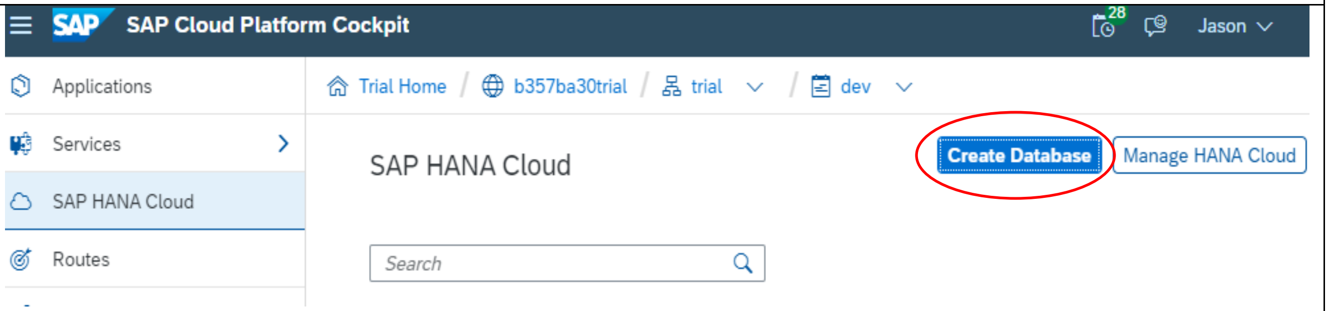
Space: dev - SAP HANA Cloud

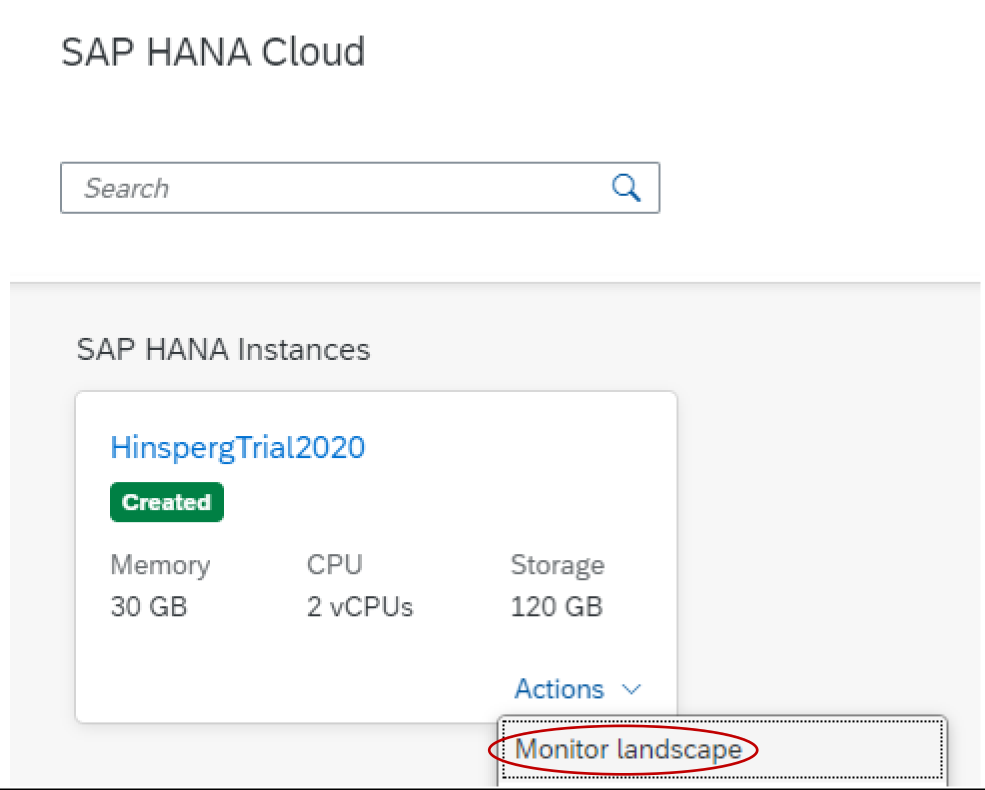
SAP HANA Instances

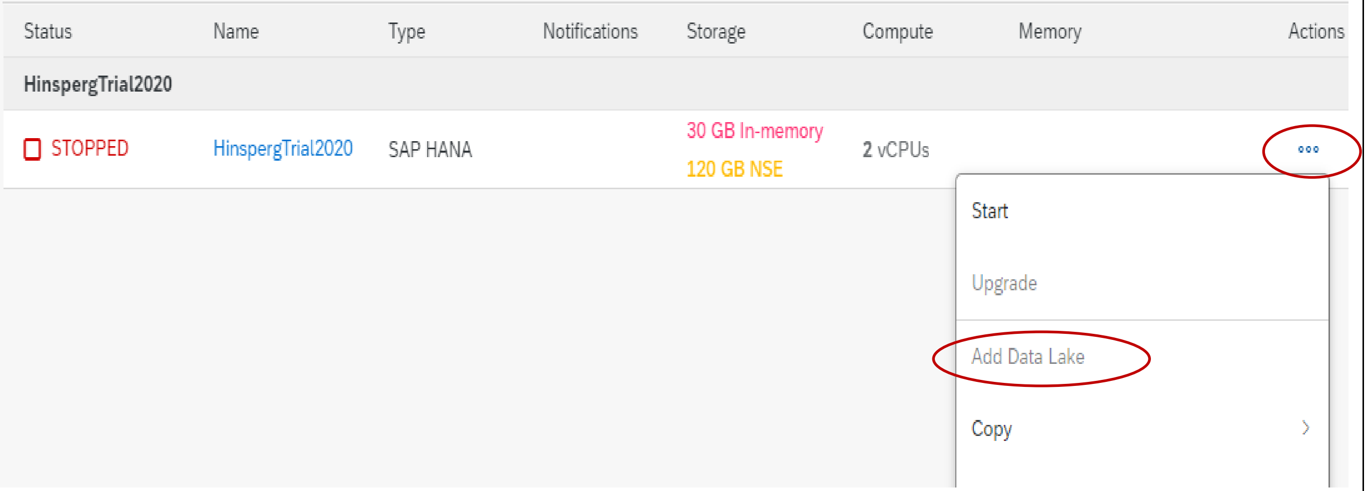
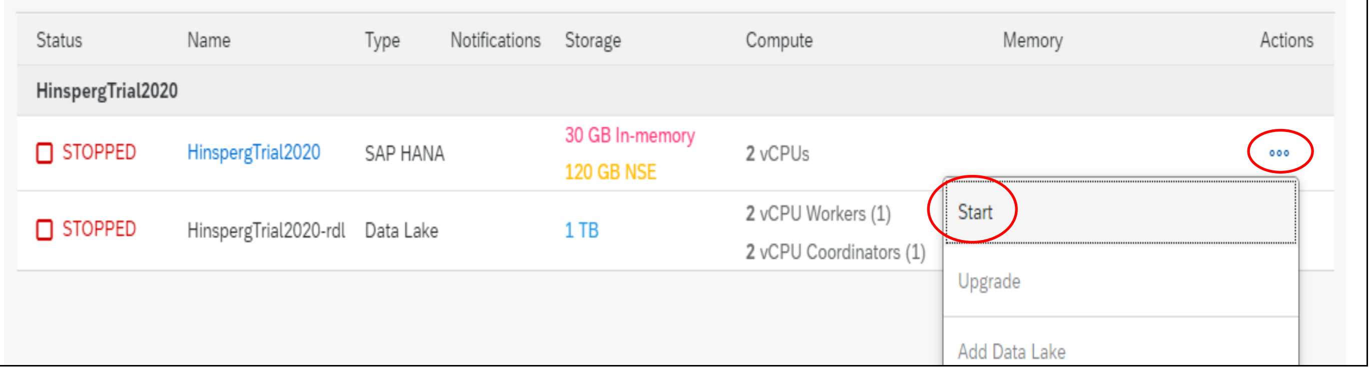
HANA Cloud, Provisioning

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.

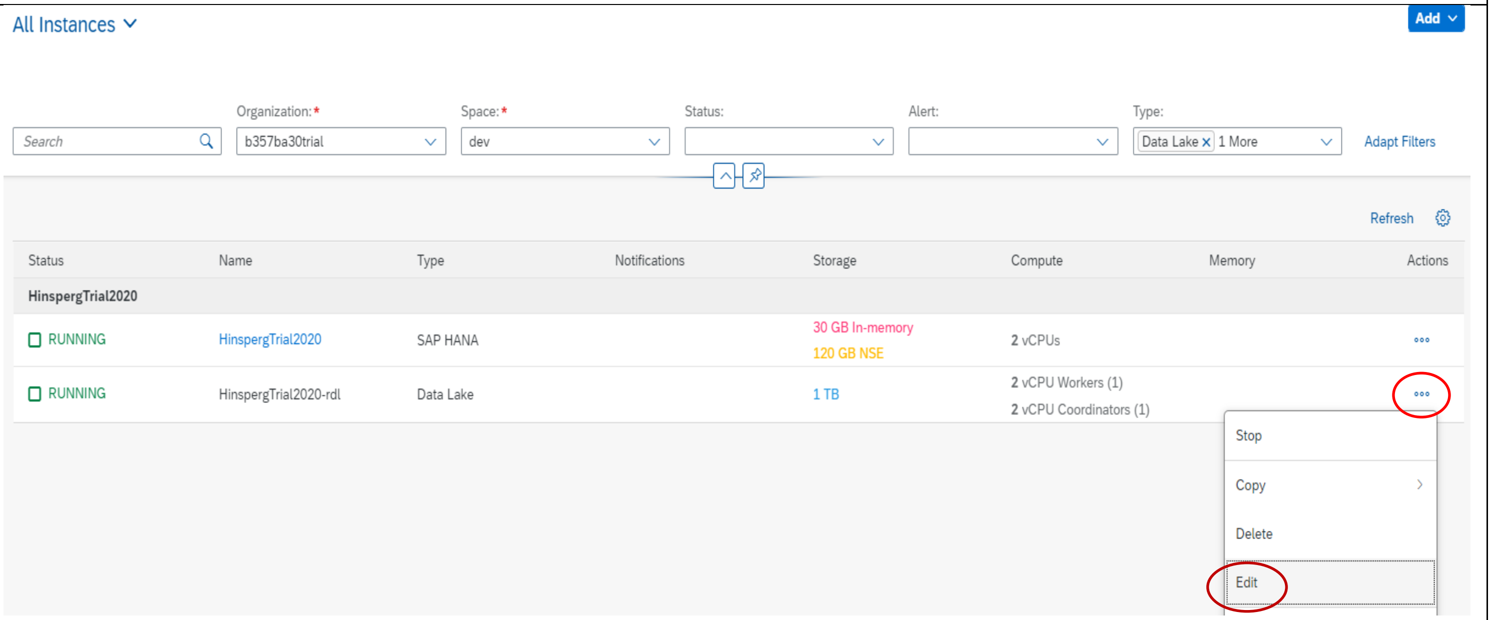
Step Explanation	Screenshot
1. From your SAP Cloud Platform trial account, open the “SAP HANA Cloud” item and click “Create Database” Follow through the wizard to create a HANA Cloud instance. Be sure to create a HANA data lake instance when given the option.	 SAP Cloud Platform Cockpit 28 Jason Applications Services SAP HANA Cloud Routes Trial Home b357ba30trial trial dev SAP HANA Cloud Create Database Manage HANA Cloud Search

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 Search SAP HANA Instances HinspergTrial2020 Created Memory 30 GB CPU 2 vCPUs Storage 120 GB Actions Monitor landscape

Step Explanation	Screenshot
3. When the HANA Cloud manager opens, choose “Add Data Lake” from the “Actions” menu in the landscape monitor.	 Status · Name · Type · Notifications · Storage · Compute · Memory · Actions HinspergTrial2020 STOPPED · HinspergTrial2020 · SAP HANA · 30 GB In-memory 120 GB NSE · 2 vCPUs
4. When the HANA Cloud instance is ready the status will change to *Running*. If the instance was previously created and is in the “Stopped” state, you can start it from the “Actions” menu.	 Status · Name · Type · Notifications · Storage · Compute · Memory · Actions HinspergTrial2020 STOPPED · HinspergTrial2020 · SAP HANA · 30 GB In-memory 120 GB NSE · 2 vCPUs STOPPED · HinspergTrial2020-rdl · Data Lake · 1 TB · 2 vCPU Workers (1) 2 vCPU Coordinators (1)

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.	

All Instances

Organization:* b357ba30trial

Space:* dev

Status:

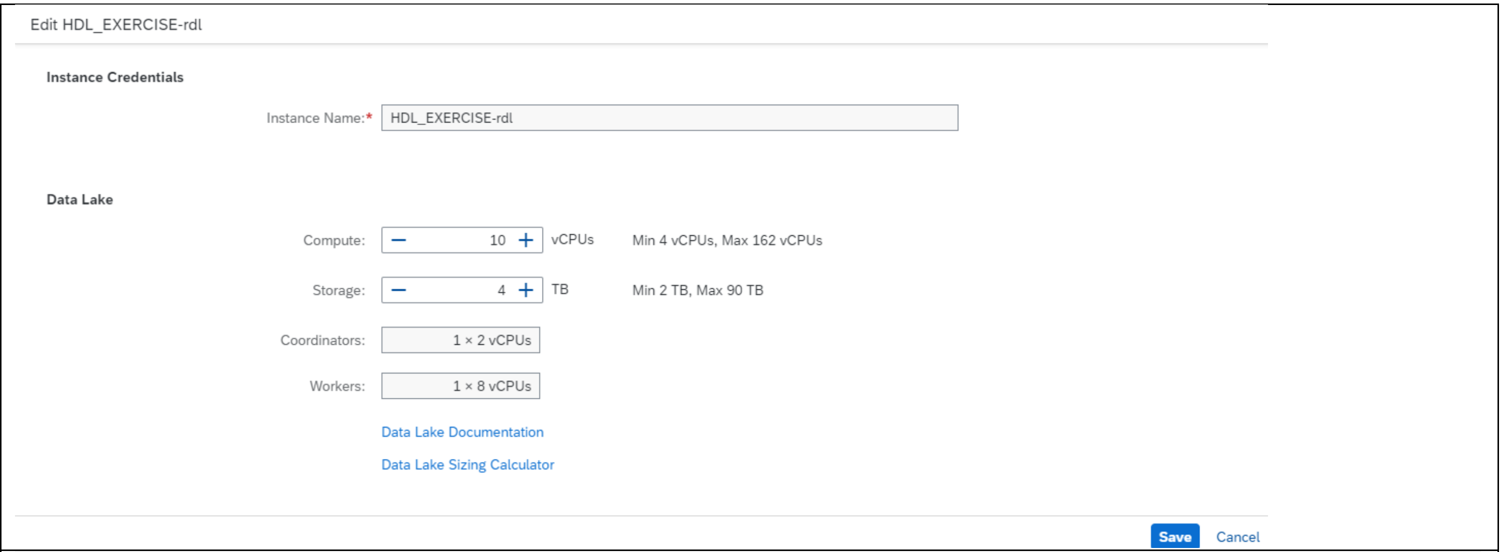
Alert:

Type: Data Lake 1 More

Status	Name	Type	Notifications	Storage	Compute	Memory	Actions
HinspergTrial2020							
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)		•••

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: 10 vCPUs Min 4 vCPUs, Max 162 vCPUs

Storage: 4 TB Min 2 TB, Max 90 TB

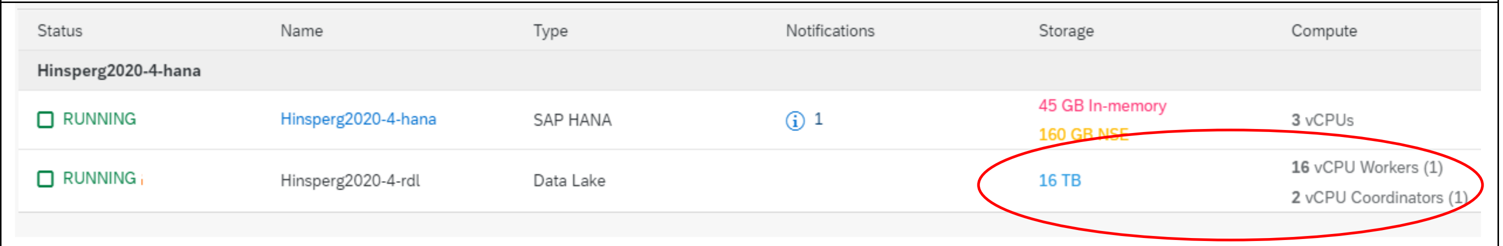
Coordinators: 1 × 2 vCPUs

Workers: 1 × 8 vCPUs

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.

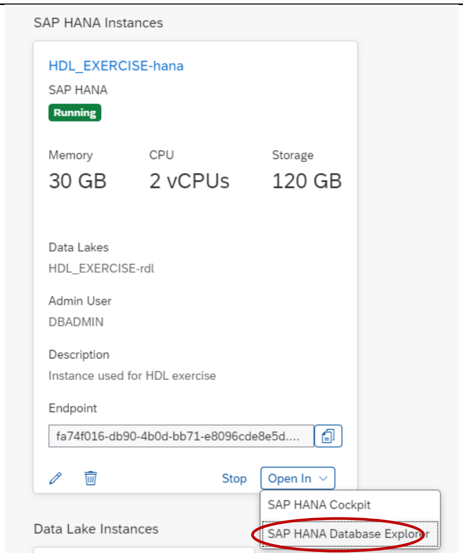
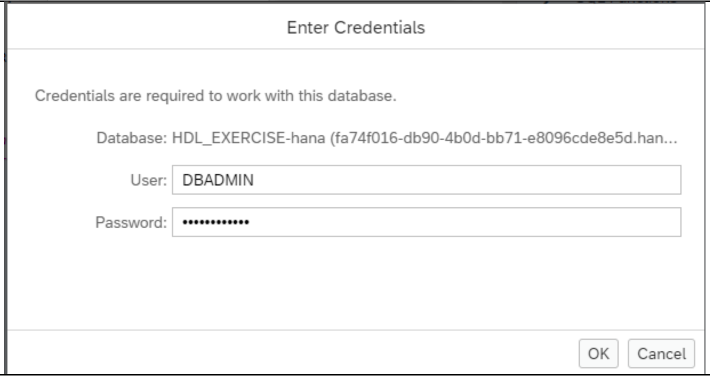


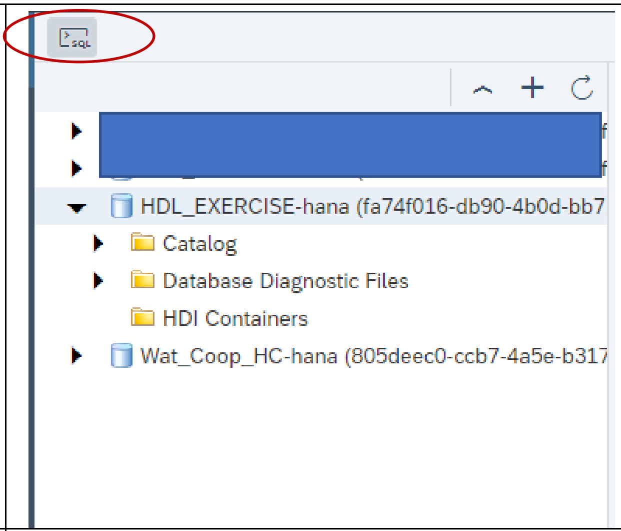
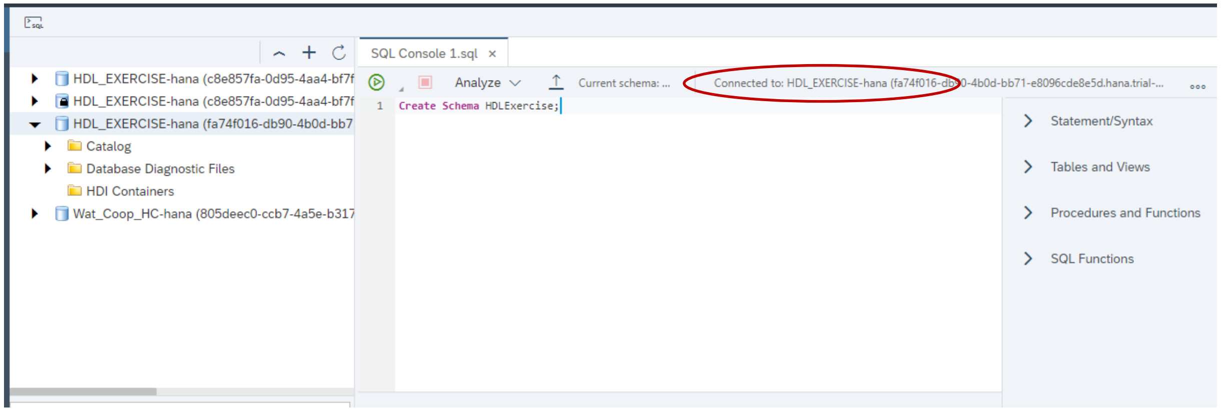
Status	Name	Type	Notifications	Storage	Compute
Hinsperg2020-4-hana					
□ RUNNING	Hinsperg2020-4-hana	SAP HANA	ⓘ 1	45 GB In-memory 160 GB NSE	**3** vCPUs
□ RUNNING	Hinsperg2020-4-rdl	Data Lake		16 TB	**16** vCPU Workers (1) **2** vCPU Coordinators (1)

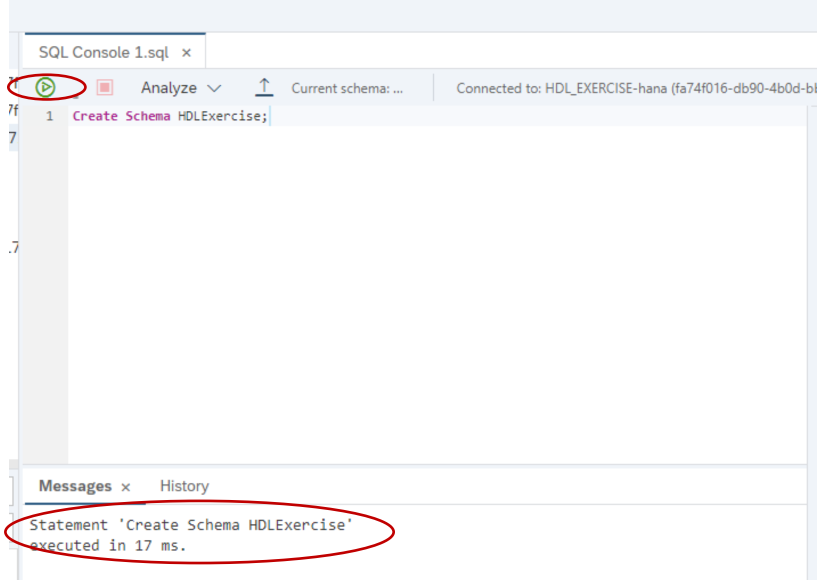
Introduction to HANA Cloud, Data Lake

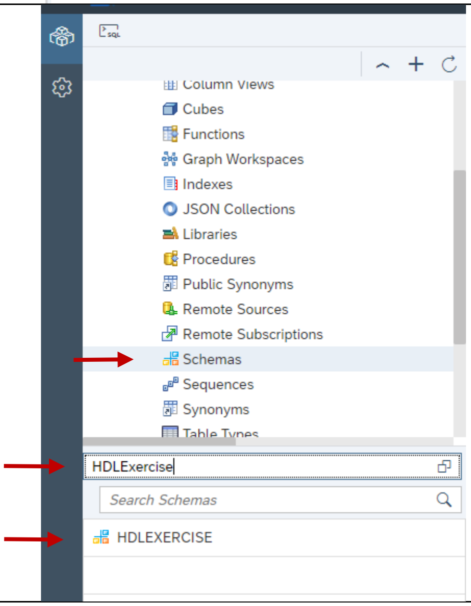
Exercise 1. Create Schema

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

Step Explanation	Screenshot
1. Navigate to the HANA Cloud instance we just created: *HDL_EXERCISE-hana*. Click on open in and select SAP HANA database explorer	 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 Instance used for HDL exercise Endpoint fa74f016-db90-4b0d-bb71-e8096cde8e5d.... Stop Open In SAP HANA Cockpit SAP HANA Database Explorer Data Lake Instances
2. Enter Credential if prompted: User: "DBADMIN" Password: "HDLexercise1"	 Enter Credentials Credentials are required to work with this database. Database: HDL_EXERCISE-hana (fa74f016-db90-4b0d-bb71-e8096cde8e5d.han... User: DBADMIN Password: •••••••••••• OK Cancel

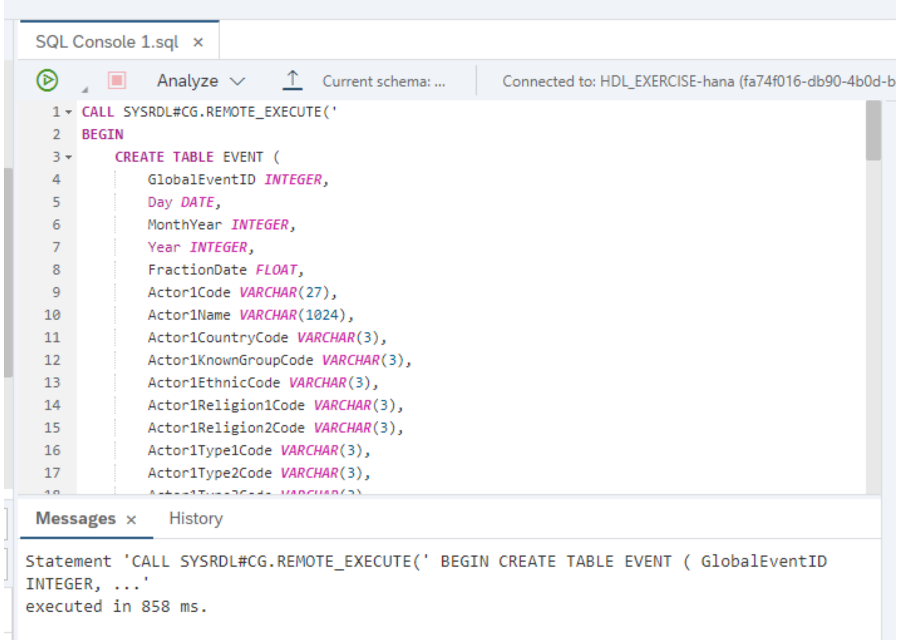
3. Click on HDL_EXERCISE-hana to ensure it is selected. Then click on the “SQL” icon on the top left corner to launch a SQL console.	 [redacted] HDL_EXERCISE-hana (fa74f016-db90-4b0d-bb7 Catalog Database Diagnostic Files HDI Containers Wat_Coop_HC-hana (805deec0-ccb7-4a5e-b317
4. Once the SQL Console is launched, type *Create Schema HDLExercise;* Note: *SQL Console 1.sql* is connected to HDL_EXERCISE-hana instance.	 HDL_EXERCISE-hana (c8e857fa-0d95-4aa4-bf7f HDL_EXERCISE-hana (c8e857fa-0d95-4aa4-bf7f HDL_EXERCISE-hana (fa74f016-db90-4b0d-bb7 Catalog Database Diagnostic Files HDI Containers Wat_Coop_HC-hana (805deec0-ccb7-4a5e-b317 SQL Console 1.sql × Analyze Current schema: ... Statement/Syntax Tables and Views Procedures and Functions SQL Functions Connected to: HDL_EXERCISE-hana (fa74f016-db90-4b0d-bb71-e8096cde8e5d.hana.trial-... `Create Schema HDLExercise;`

5.
Click on the green run button to execute the query.
 	
Make sure the query executes successfully by verifying in the *Messages* tab.
	
	 SQL Console 1.sql ×
	Analyze
	Current schema: ...
	Connected to: HDL_EXERCISE-hana (fa74f016-db90-4b0d-b
	Messages ×
	History
	`Create Schema HDLExercise;`
	`Statement 'Create Schema HDLExercise' executed in 17 ms.`

6. Check the Schema "HDLExercise" has been created by navigating to Catalog, Schemas, and then type "HDLExercise" in the Choose schema input form. Ensure that HDLExercise exists.	 Column Views Cubes Functions Graph Workspaces Indexes JSON Collections Libraries Procedures Public Synonyms Remote Sources Remote Subscriptions Schemas Sequences Synonyms Table Types Search Schemas HDLExercise HDLEXERCISE

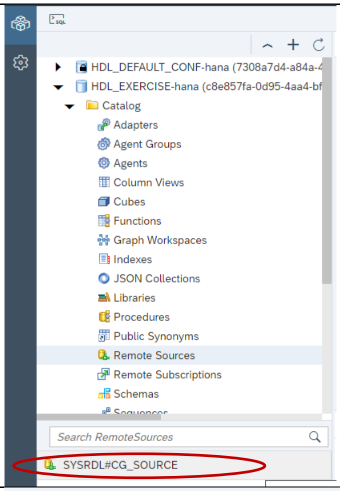
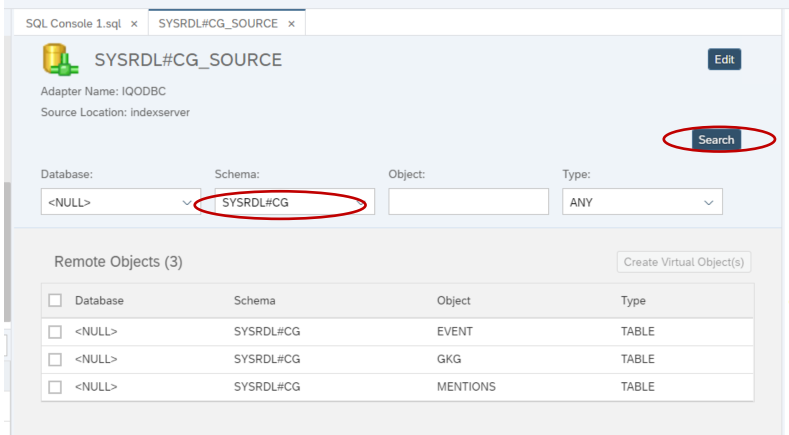
Exercise 2. Create physical tables in Data Lake

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

Step Explanation	Screenshot
1. Copy and paste the command from DAT161*Exercise_CreateTable.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully. This command creates the following tables in data lake: EVENT, GKG, MENTIONS. Note the create table statements are wrapped inside the *SYSRDL#CG.remote_execute* function. *remote_execute()* is a stored procedure which allows you to execute SQL natively in data lake.	

```
CALL SYSRDL#CG.REMOTE_EXECUTE('
BEGIN
    CREATE TABLE EVENT (
        GlobalEventID INTEGER,
        Day DATE,
        MonthYear INTEGER,
        Year INTEGER,
        FractionDate FLOAT,
        Actor1Code VARCHAR(27),
        Actor1Name VARCHAR(1024),
        Actor1CountryCode VARCHAR(3),
        Actor1KnownGroupCode VARCHAR(3),
        Actor1EthnicCode VARCHAR(3),
        Actor1Religion1Code VARCHAR(3),
        Actor1Religion2Code VARCHAR(3),
        Actor1Type1Code VARCHAR(3),
        Actor1Type2Code VARCHAR(3),
        Actor1Type3Code VARCHAR(3),
```

```
Statement 'CALL SYSRDL#CG.REMOTE_EXECUTE(' BEGIN CREATE TABLE EVENT ( GlobalEventID
INTEGER, ...'
executed in 858 ms.
```

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*.	 HDL_DEFAULT_CONF-hana (7308a7d4-a84a-4 HDL_EXERCISE-hana (c8e857fa-0d95-4aa4-bf Catalog Adapters Agent Groups Agents Column Views Cubes Functions Graph Workspaces Indexes JSON Collections Libraries Procedures Public Synonyms Remote Sources Remote Subscriptions Schemas Sequences Search RemoteSources 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.	 SQL Console 1.sql × SYSRDL#CG_SOURCE × SYSRDL#CG_SOURCE Edit Search Create Virtual Object(s)

Adapter Name: IQODBC
Source Location: indexserver

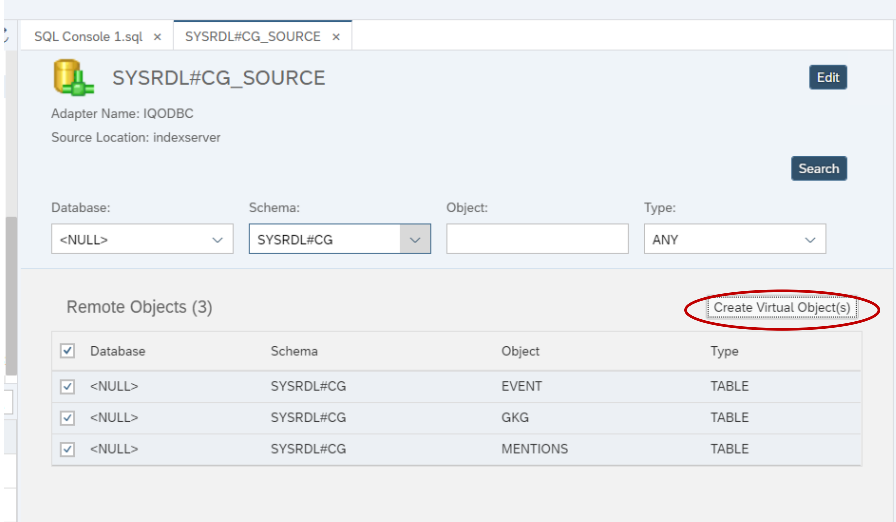
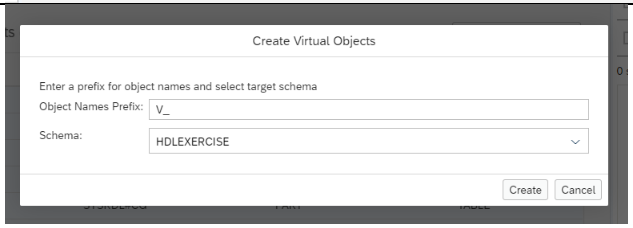
Database: <NULL>
Schema: SYSRDL#CG
Object:
Type: ANY

Remote Objects (3)

☐	Database	Schema	Object	Type
☐	<NULL>	SYSRDL#CG	EVENT	TABLE
☐	<NULL>	SYSRDL#CG	GKG	TABLE
☐	<NULL>	SYSRDL#CG	MENTIONS	TABLE

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
1. From the same interface as the previous exercise, select all the tables, and click on *Create Virtual Objects*.	 SQL Console 1.sql × SYSRDL#CG_SOURCE × SYSRDL#CG_SOURCE Edit Adapter Name: IQODBC Source Location: indexserver Search Database: <NULL> Schema: SYSRDL#CG Object: Type: ANY Remote Objects (3) Create Virtual Object(s) ☑ Database Schema Object Type ☑ <NULL> SYSRDL#CG EVENT TABLE ☑ <NULL> SYSRDL#CG GKG TABLE ☑ <NULL> SYSRDL#CG MENTIONS TABLE
2. Set *Object Names Prefix* to V_ Then select the schema created from the previous exercise *HDLEXERCISE*, and click on create. 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. The virtual tables are associated with the schema HDLEXERCISE.	 Create Virtual Objects Enter a prefix for object names and select target schema Object Names Prefix: V_ Schema: HDLEXERCISE Create Cancel

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.

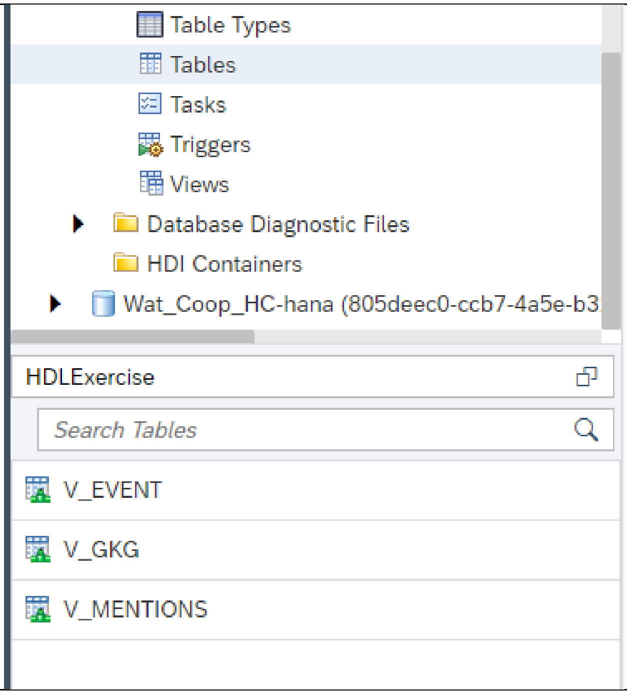


Table Types
Tables
Tasks
Triggers
Views
Database Diagnostic Files
HDI Containers
Wat_Coop_HC-hana (805deec0-ccb7-4a5e-b3
Search Tables

HDLExercise

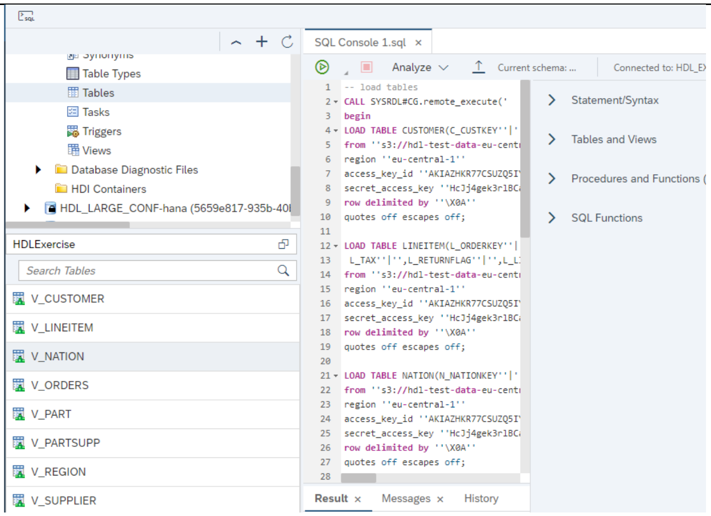
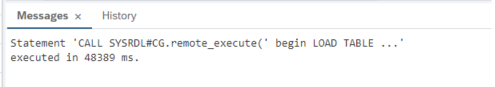
V_EVENT

V_GKG

V_MENTIONS

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
1. Copy and paste the command from DAT161Exercise_*LoadTable.sql* into the SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully. This command loads data from a public AWS s3 bucket into the following tables in data lake: EVENT, GKG, MENTIONS. 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.	
2. This process could take a few minutes. Verify that the statements execute successfully.	

Screenshot for step 1:

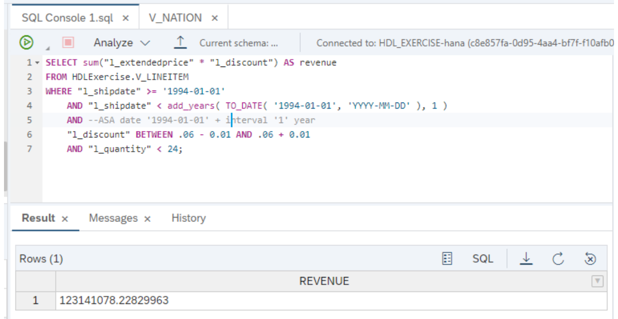
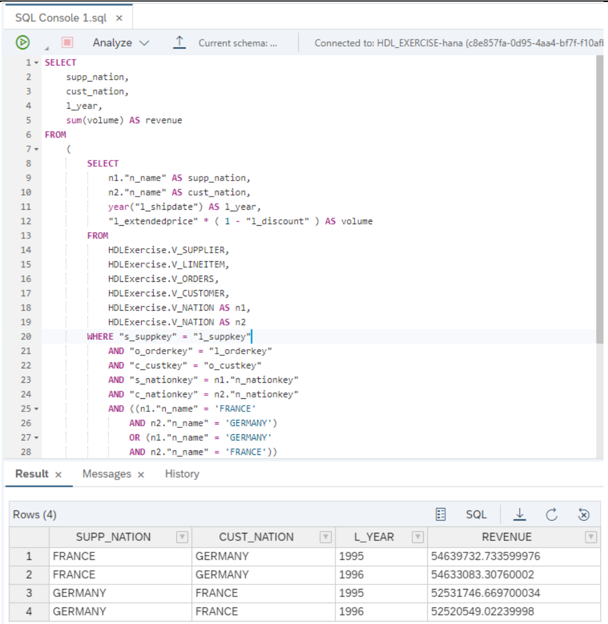
```
-- load tables
CALL SYSRDL#CG.remote_execute('
begin
LOAD TABLE CUSTOMER(C_CUSTKEY''|'
from ''s3://hdl-test-data-eu-centr
region ''eu-central-1''
access_key_id ''AKIAZHKR77CSUZQ5I
secret_access_key ''HcJj4gek3rlBCa
row delimited by ''\X0A''
quotes off escapes off;

LOAD TABLE LINEITEM(L_ORDERKEY''|
 L_TAX''|'',L_RETURNFLAG''|'',L_LI
from ''s3://hdl-test-data-eu-centr
region ''eu-central-1''
access_key_id ''AKIAZHKR77CSUZQ5I
secret_access_key ''HcJj4gek3rlBCa
row delimited by ''\X0A''
quotes off escapes off;

LOAD TABLE NATION(N_NATIONKEY''|'
from ''s3://hdl-test-data-eu-centr
region ''eu-central-1''
access_key_id ''AKIAZHKR77CSUZQ5I
secret_access_key ''HcJj4gek3rlBCa
row delimited by ''\X0A''
quotes off escapes off;
```

Screenshot for step 2:

```
Statement 'CALL SYSRDL#CG.remote_execute(' begin LOAD TABLE ...'
executed in 48389 ms.
```

Exercise 5. Query data from Virtual Table

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

Step Explanation	Screenshot
1. Copy and paste the command from *Query1.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.	
2. Copy and paste the command from *Query2.sql* into SQL console. Execute the command by pressing the run button. Ensure the commands execute successfully.	

```sql
SELECT sum("l_extendedprice" * "l_discount") AS revenue
FROM HDLExercise.V_LINEITEM
WHERE "l_shipdate" >= '1994-01-01'
    AND "l_shipdate" < add_years( TO_DATE( '1994-01-01', 'YYYY-MM-DD' ), 1 )
    AND --ASA date '1994-01-01' + interval '1' year
    "l_discount" BETWEEN .06 - 0.01 AND .06 + 0.01
    AND "l_quantity" < 24;
```

	REVENUE
1	123141078.22829963

```sql
SELECT
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) AS revenue
FROM
    (
        SELECT
            n1."n_name" AS supp_nation,
            n2."n_name" AS cust_nation,
            year("l_shipdate") AS l_year,
            "l_extendedprice" * ( 1 - "l_discount" ) AS volume
        FROM
            HDLExercise.V_SUPPLIER,
            HDLExercise.V_LINEITEM,
            HDLExercise.V_ORDERS,
            HDLExercise.V_CUSTOMER,
            HDLExercise.V_NATION AS n1,
            HDLExercise.V_NATION AS n2
        WHERE "s_suppkey" = "l_suppkey"
            AND "o_orderkey" = "l_orderkey"
            AND "c_custkey" = "o_custkey"
            AND "s_nationkey" = n1."n_nationkey"
            AND "c_nationkey" = n2."n_nationkey"
            AND ((n1."n_name" = 'FRANCE'
                AND n2."n_name" = 'GERMANY')
                OR (n1."n_name" = 'GERMANY'
                AND n2."n_name" = 'FRANCE'))
```

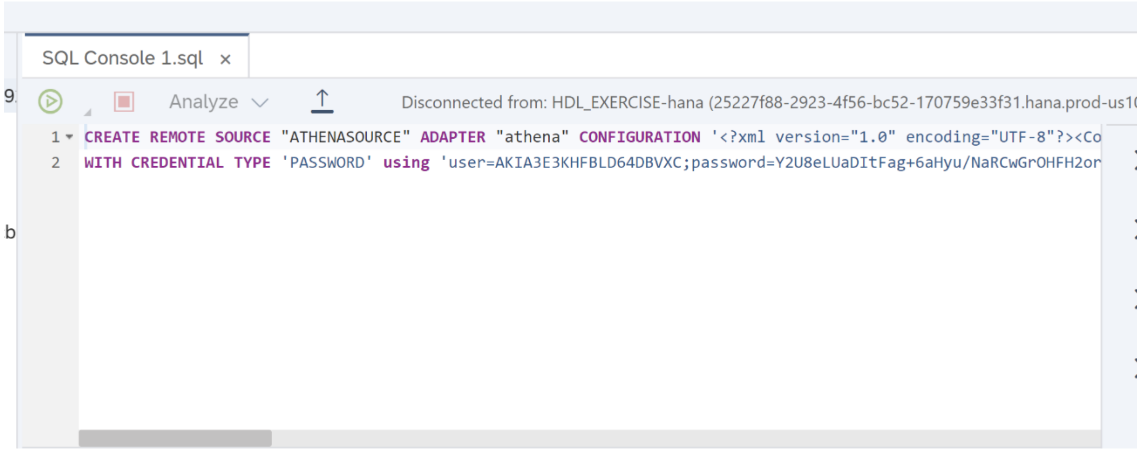
	SUPP_NATION	CUST_NATION	L_YEAR	REVENUE
1	FRANCE	GERMANY	1995	54639732.733599976
2	FRANCE	GERMANY	1996	54633083.30760002
3	GERMANY	FRANCE	1995	52531746.669700034
4	GERMANY	FRANCE	1996	52520549.02239998

Setting up a Remote Source to Athena

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

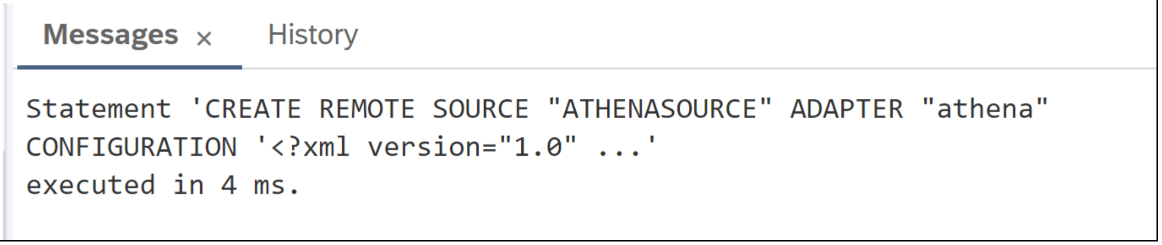
The goal of this exercise is to set up a remote source to Athena and create virtual tables using the SAP HANA database explorer.

Step Explanation	Screenshot
1. Create a remote source called “ATHENASOURCE” by running the first statement in DAT161*Exercise_Athena.sql* under *“-- Create Remote Source”* in a SQL console	



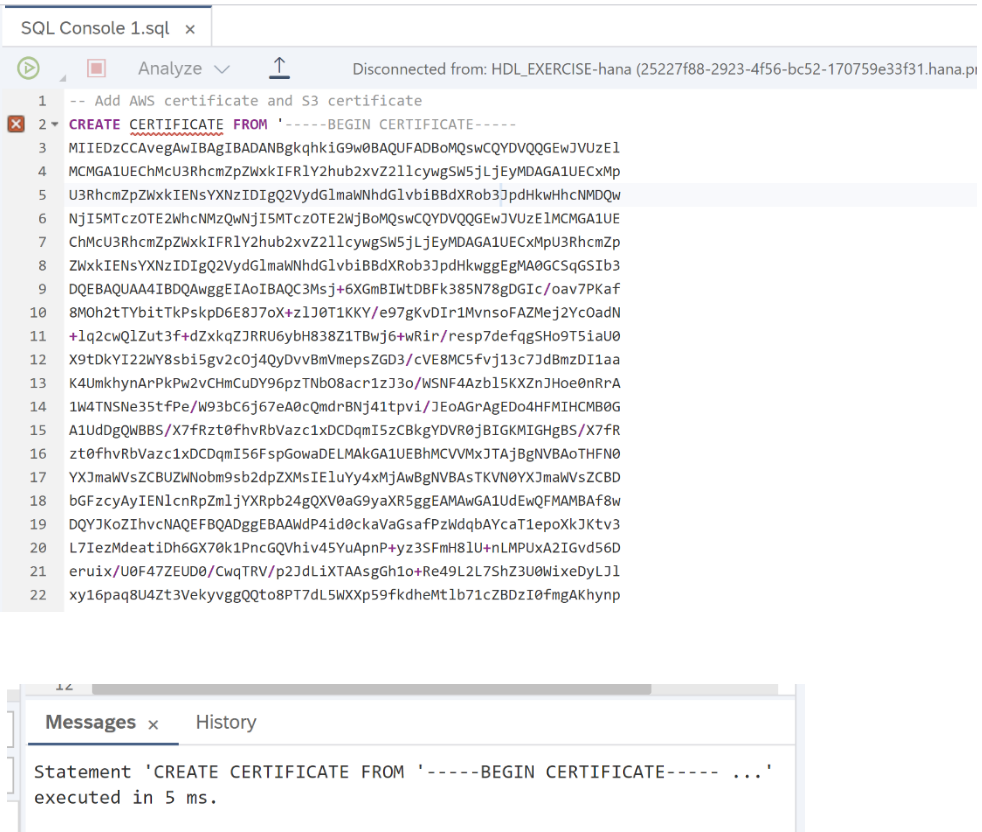
SQL Console 1.sql ×
Analyze
Disconnected from: HDL_EXERCISE-hana (25227f88-2923-4f56-bc52-170759e33f31.hana.prod-us1(

```
CREATE REMOTE SOURCE "ATHENASOURCE" ADAPTER "athena" CONFIGURATION '<?xml version="1.0" encoding="UTF-8"?><Co
WITH CREDENTIAL TYPE 'PASSWORD' using 'user=AKIA3E3KHFBLD64DBVXC;password=Y2U8eLUaDItFag+6aHyu/NaRCwGrOHFH2or
```



```
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

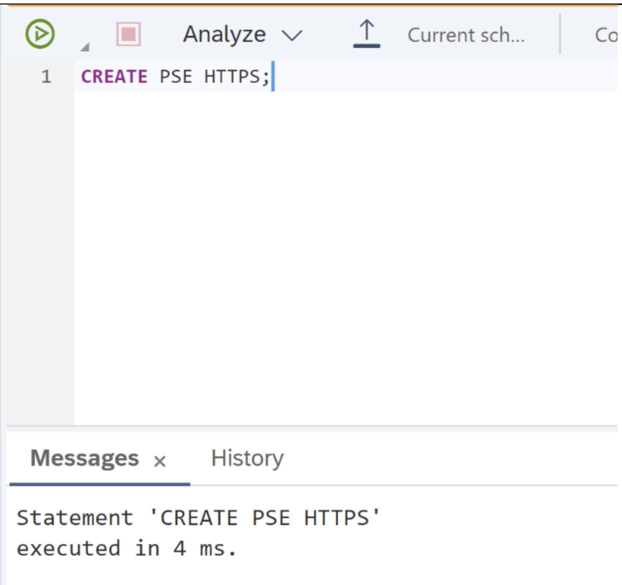

SQL Console 1.sql ×
Analyze
Disconnected from: HDL_EXERCISE-hana (25227f88-2923-4f56-bc52-170759e33f31.hana.pr
Messages ×
History

```
-- Add AWS certificate and S3 certificate
CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE-----
MIIEDzCCAvegAwIBAgIBADANBgkqhkiG9w0BAQUFADBoMQswCQYDVQQGEwJVUzEl
MCMGA1UEChMcU3RhcmZpZWxkIFRlY2hub2xvZ2llcywgSW5jLjEyMDAGA1UECxMp
U3RhcmZpZWxkIENsYXNzIDIgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkwHhcNMDQw
NjI5MTczOTE2WhcNMzQwNjI5MTczOTE2WjBoMQswCQYDVQQGEwJVUzElMCMGA1UE
ChMcU3RhcmZpZWxkIFRlY2hub2xvZ2llcywgSW5jLjEyMDAGA1UECxMpU3RhcmZp
ZWxkIENsYXNzIDIgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkwggEgMA0GCSqGSIb3
DQEBAQUAA4IBDQAwggEIAoIBAQC3Msj+6XGmBIWtDBFk385N78gDGIc/oav7PKaf
8MOh2tTYbitTkPskpD6E8J7oX+zlJ0T1KKY/e97gKvDIr1MvnsoFAZMej2YcOadN
+lq2cwQlZut3f+dZxkqZJRRU6ybH838Z1TBwj6+wRir/resp7defqgSHo9T5iaU0
X9tDkYI22WY8sbi5gv2cOj4QyDvvBmVmepsZGD3/cVE8MC5fvj13c7JdBmzDI1aa
K4UmkhynArPkPw2vCHmCuDY96pzTNbO8acr1zJ3o/WSNF4Azbl5KXZnJHoe0nRrA
1W4TNSNe35tfPe/W93bC6j67eA0cQmdrBNj41tpvi/JEoAGrAgEDo4HFMIHCMB0G
A1UdDgQWBBS/X7fRzt0fhvRbVazc1xDCDqmI5zCBkgYDVR0jBIGKMIGHgBS/X7fR
zt0fhvRbVazc1xDCDqmI56FspGowaDELMAkGA1UEBhMCVVMxJTAjBgNVBAoTHFN0
YXJmaWVsZCBUZWNobm9sb2dpZXMsIEluYy4xMjAwBgNVBAsTKVN0YXJmaWVsZCBD
bGFzcyAyIENlcnRpZmljYXRpb24gQXV0aG9yaXR5ggEAMAwGA1UdEwQFMAMBAf8w
DQYJKoZIhvcNAQEFBQADggEBAAWdP4id0ckaVaGsafPzWdqbAYcaT1epoXkJKtv3
L7IezMdeatiDh6GX70k1PncGQVhiv45YuApnP+yz3SFmH8lU+nLMPUxA2IGvd56D
eruix/U0F47ZEUD0/CwqTRV/p2JdLiXTAAsgGh1o+Re49L2L7ShZ3U0WixeDyLJl
xy16paq8U4Zt3VekyvggQQto8PT7dL5WXXp59fkdheMtlb71cZBDzI0fmgAKhynp
```

```
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;



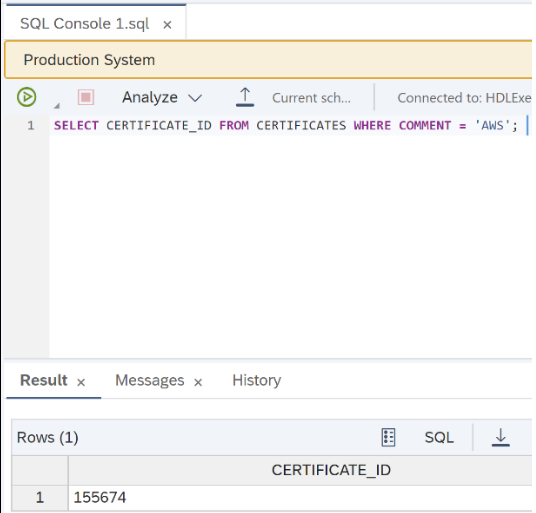
```
CREATE PSE HTTPS;
```

```
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



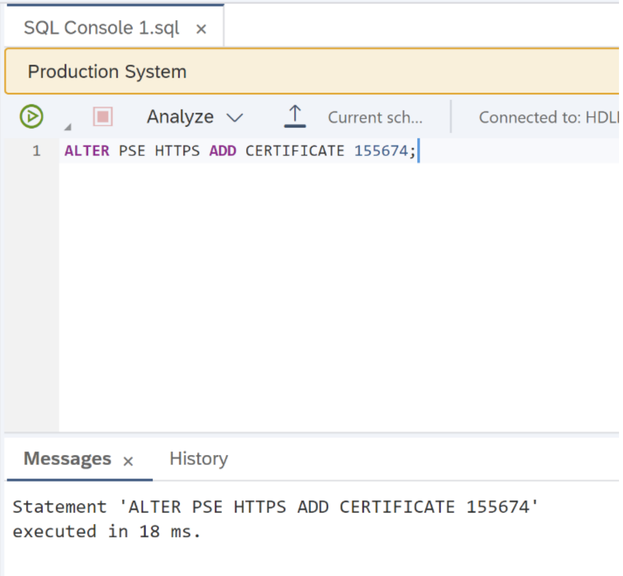
```
SELECT CERTIFICATE_ID FROM CERTIFICATES WHERE COMMENT = 'AWS';
```

	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



```
ALTER PSE HTTPS ADD CERTIFICATE 155674;
```

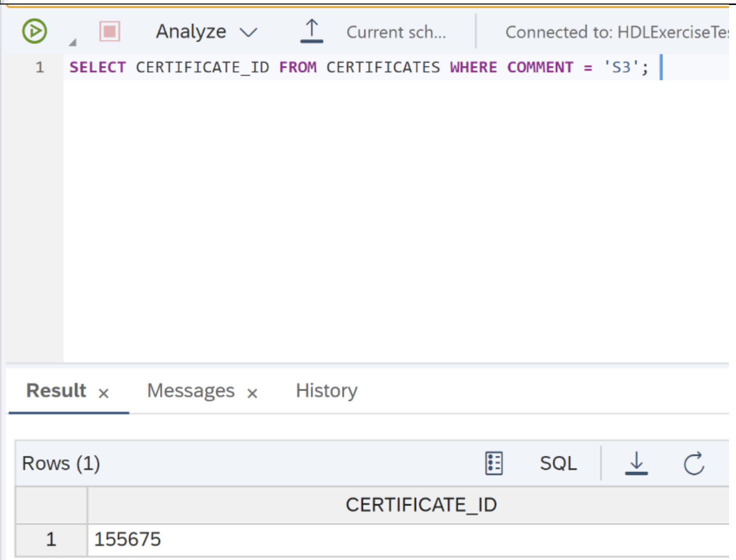
```
Statement 'ALTER PSE HTTPS ADD CERTIFICATE 155674'
executed in 18 ms.
```

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



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

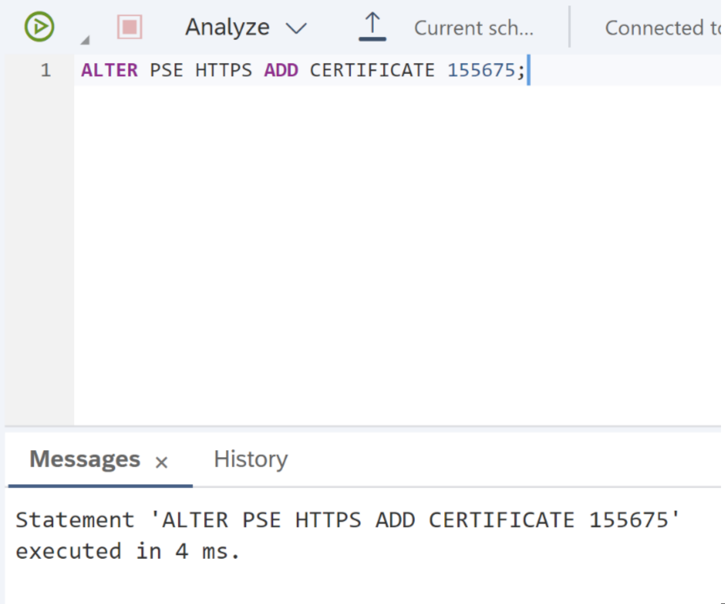
Rows (1)

	CERTIFICATE_ID
1	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

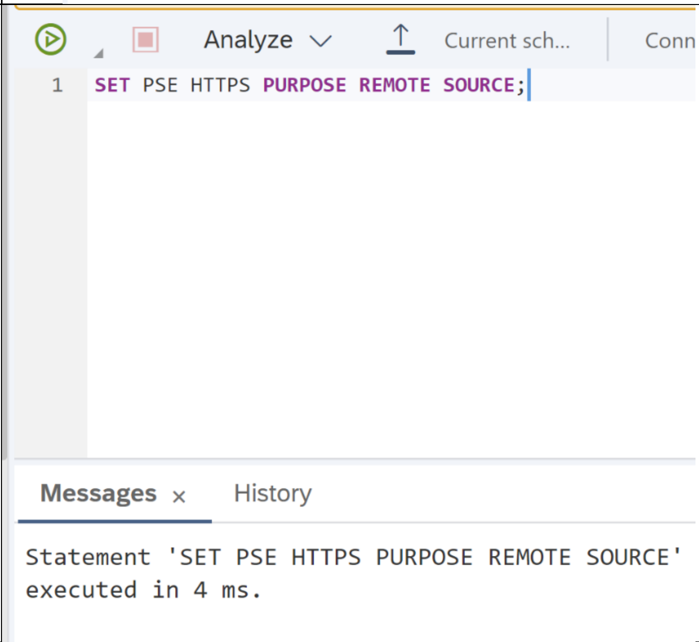


```
ALTER PSE HTTPS ADD CERTIFICATE 155675;
```

```
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:

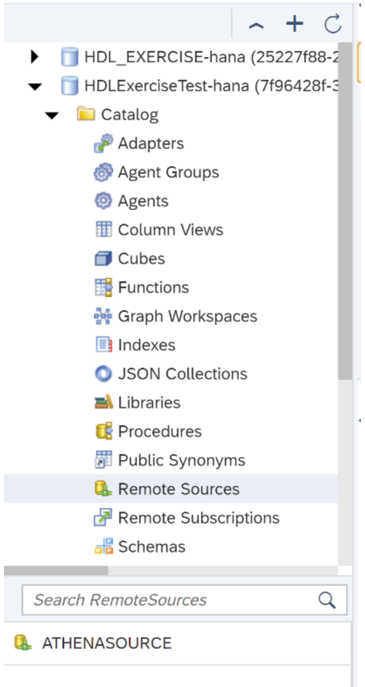
SET PSE HTTPS PURPOSE REMOTE SOURCE;



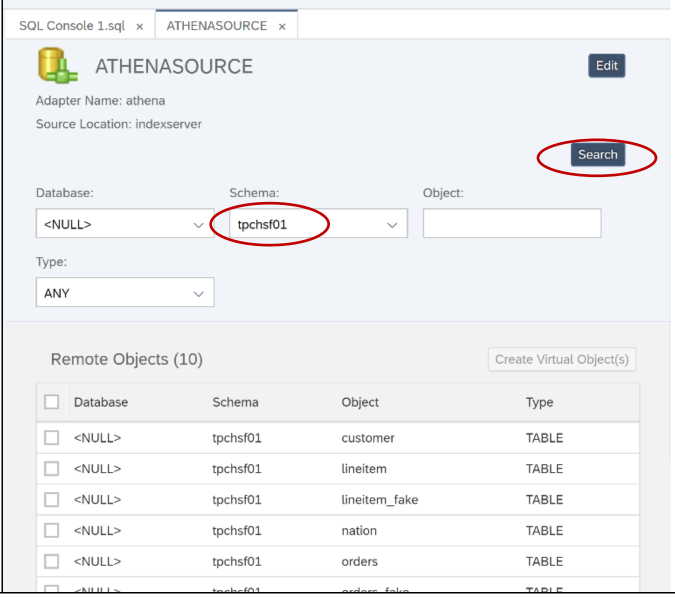
```
SET PSE HTTPS PURPOSE REMOTE SOURCE;
```

```
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.



ATHENASOURCE

Adapter Name: athena

Source Location: indexserver

Database: <NULL>

Schema: tpchsf01

Object:

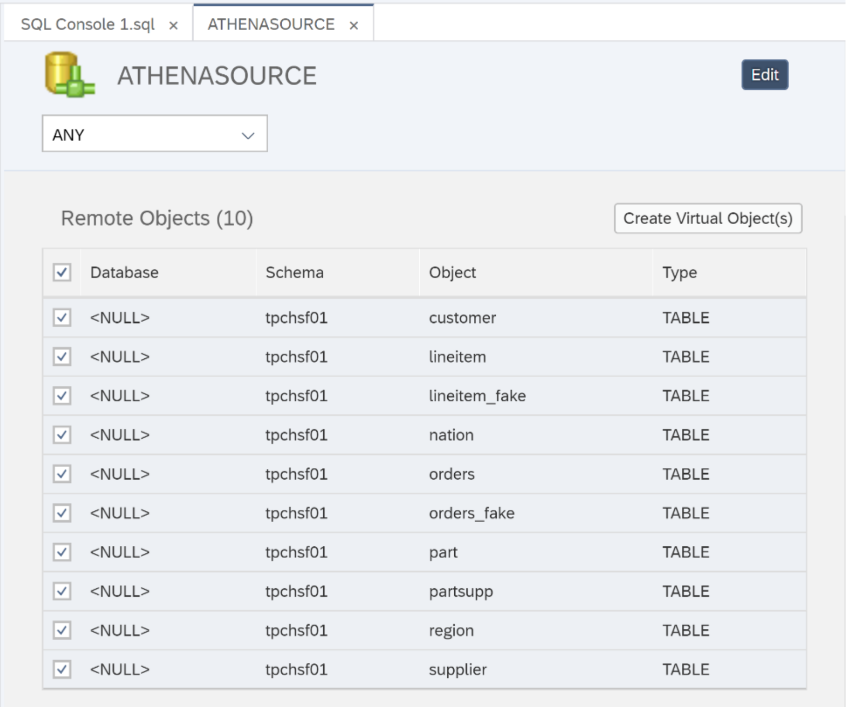
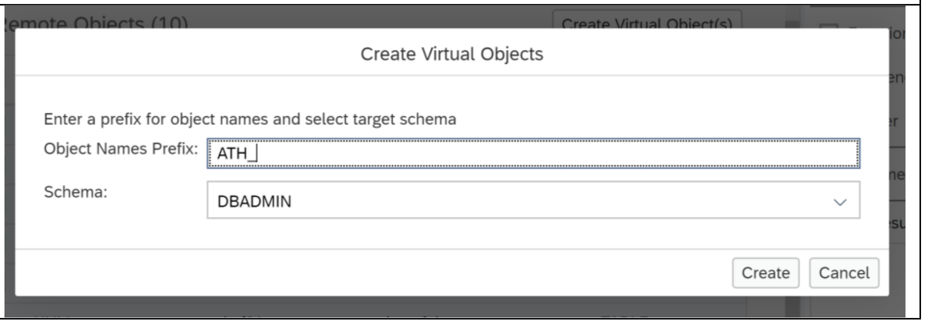
Type: ANY

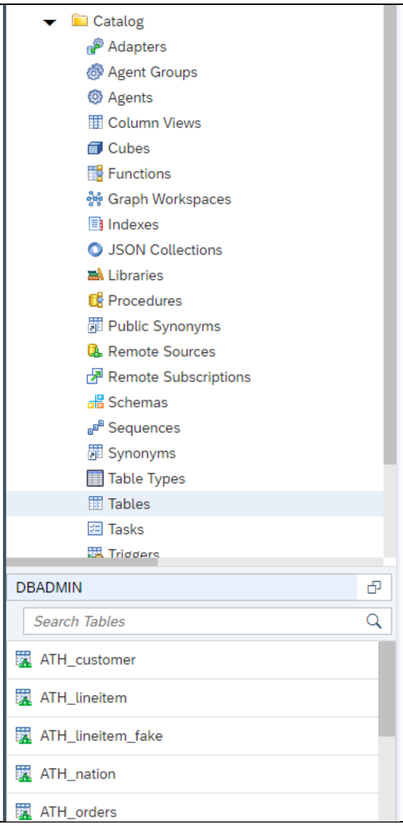
Remote Objects (10)

□	Database	Schema	Object	Type
□	<NULL>	tpchsf01	customer	TABLE
□	<NULL>	tpchsf01	lineitem	TABLE
□	<NULL>	tpchsf01	lineitem_fake	TABLE
□	<NULL>	tpchsf01	nation	TABLE
□	<NULL>	tpchsf01	orders	TABLE
□	<NULL>	tpchsf01	orders_fake	TABLE

Exercise 2. Create virtual tables

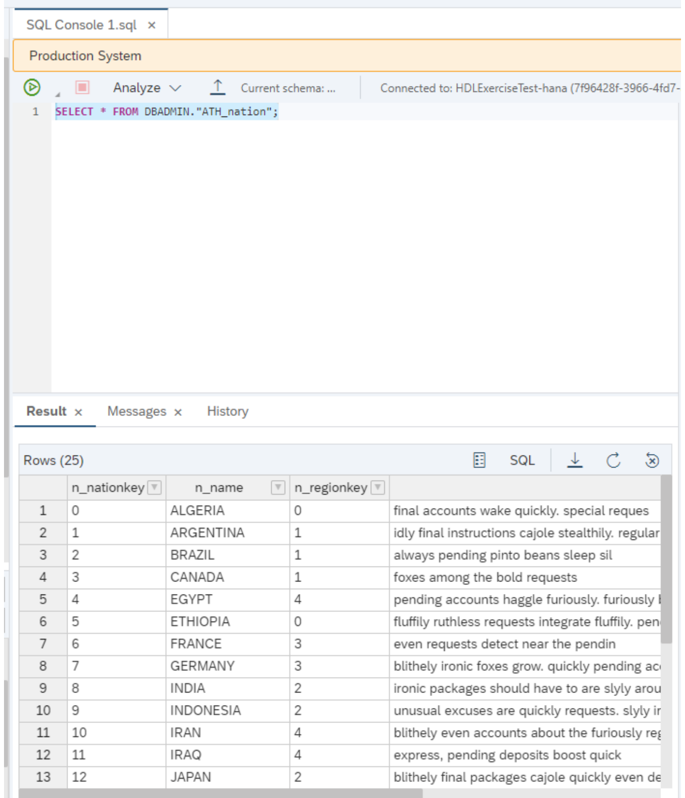
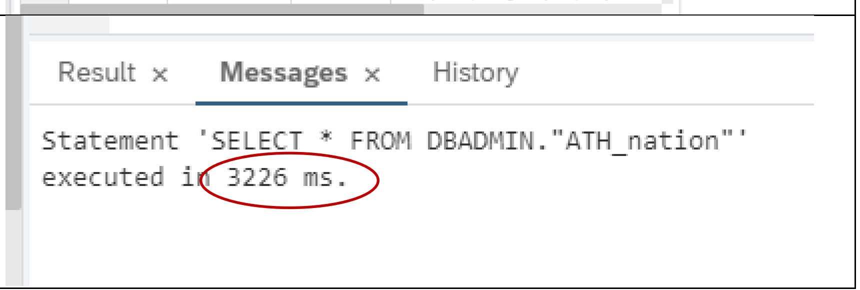
The goal of this exercise is to create virtual tables pointing to remote objects in ATHENASOURCE.

Step Explanation	Screenshot
1. Select all object in the remote source and click on the `Create Virtual Object(s)` button	 SQL Console 1.sql × ATHENASOURCE × ATHENASOURCE Edit ANY Remote Objects (10) Create Virtual Object(s) ☑ Database | Schema | Object | Type ☑ <NULL> | tpchsf01 | customer | TABLE ☑ <NULL> | tpchsf01 | lineitem | TABLE ☑ <NULL> | tpchsf01 | lineitem_fake | TABLE ☑ <NULL> | tpchsf01 | nation | TABLE ☑ <NULL> | tpchsf01 | orders | TABLE ☑ <NULL> | tpchsf01 | orders_fake | TABLE ☑ <NULL> | tpchsf01 | part | TABLE ☑ <NULL> | tpchsf01 | partsupp | TABLE ☑ <NULL> | tpchsf01 | region | TABLE ☑ <NULL> | tpchsf01 | supplier | TABLE
2. Set prefix as ATH_ and leave Schema as DBADMIN. Click on Create.	 Create Virtual Objects Enter a prefix for object names and select target schema Object Names Prefix: ATH_ Schema: DBADMIN Create Cancel

3. Once the virtual objects are created, navigate to Catalog, Tables to verify the tables are created successfully.	 Catalog Adapters Agent Groups Agents Column Views Cubes Functions Graph Workspaces Indexes JSON Collections Libraries Procedures Public Synonyms Remote Sources Remote Subscriptions Schemas Sequences Synonyms Table Types Tables Tasks Triggers DBADMIN Search Tables ATH_customer ATH_lineitem ATH_lineitem_fake ATH_nation ATH_orders

Exercise 3. Query Virtual Tables

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

Step Explanation	Screenshot
1. Select all the rows in the ATH_nation virtual table that we have just created by running the following statement: SELECT * FROM DBADMIN."ATH_nation";	 Production System Connected to: HDLExerciseTest-hana (7f96428f-3966-4fd7- `SELECT * FROM DBADMIN."ATH_nation";` Rows (25)
2. Note the performance of the statement which queries a virtual table (3226 ms).	 Statement 'SELECT * FROM DBADMIN."ATH_nation"' executed in 3226 ms.

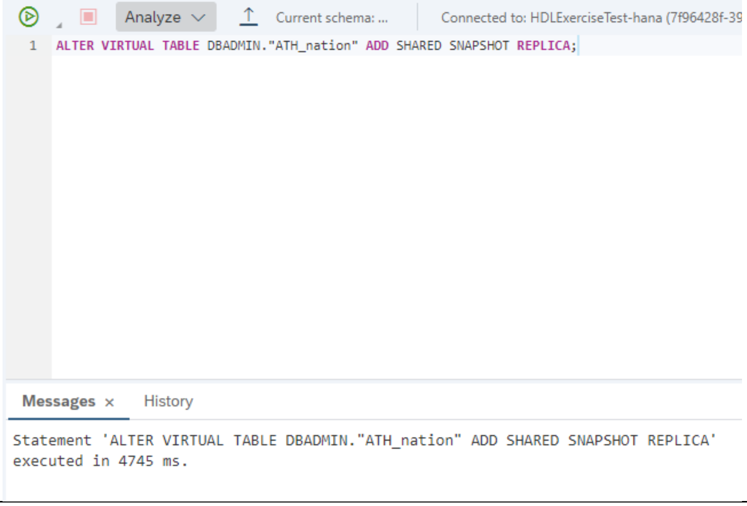
Rows (25)

	n_nationkey	n_name	n_regionkey	
1	0	ALGERIA	0	final accounts wake quickly. special reques
2	1	ARGENTINA	1	idly final instructions cajole stealthily. regular
3	2	BRAZIL	1	always pending pinto beans sleep sil
4	3	CANADA	1	foxes among the bold requests
5	4	EGYPT	4	pending accounts haggle furiously. furiously
6	5	ETHIOPIA	0	fluffily ruthless requests integrate fluffily. pen
7	6	FRANCE	3	even requests detect near the pendin
8	7	GERMANY	3	blithely ironic foxes grow. quickly pending ac
9	8	INDIA	2	ironic packages should have to are slyly arou
10	9	INDONESIA	2	unusual excuses are quickly requests. slyly ir
11	10	IRAN	4	blithely even accounts about the furiously re
12	11	IRAQ	4	express, pending deposits boost quick
13	12	JAPAN	2	blithely final packages cajole quickly even de

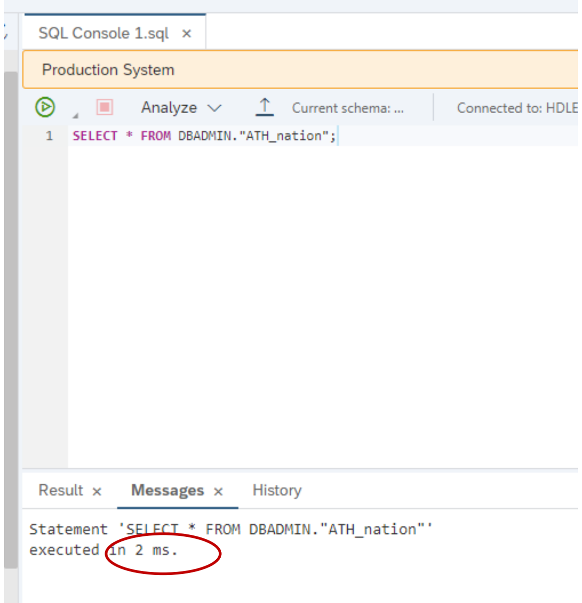
Exercise 4. Turn on Table Replication

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

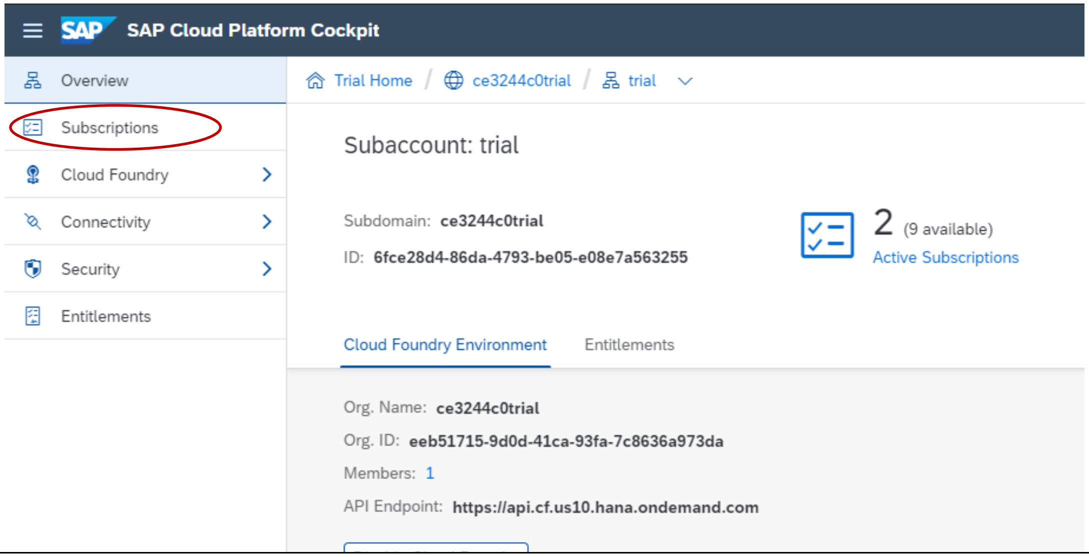
1. Replicate the table in HANA Cloud by running the following statement:
ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA;
Note: to avoid excessive memory usage this statement should ideally be used with smaller tables
	
	 Analyze
	Current schema: ...
	Connected to: HDLExerciseTest-hana (7f96428f-39
	Messages ×
	History
	`ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA;`
	`Statement 'ALTER VIRTUAL TABLE DBADMIN."ATH_nation" ADD SHARED SNAPSHOT REPLICA' executed in 4745 ms.`

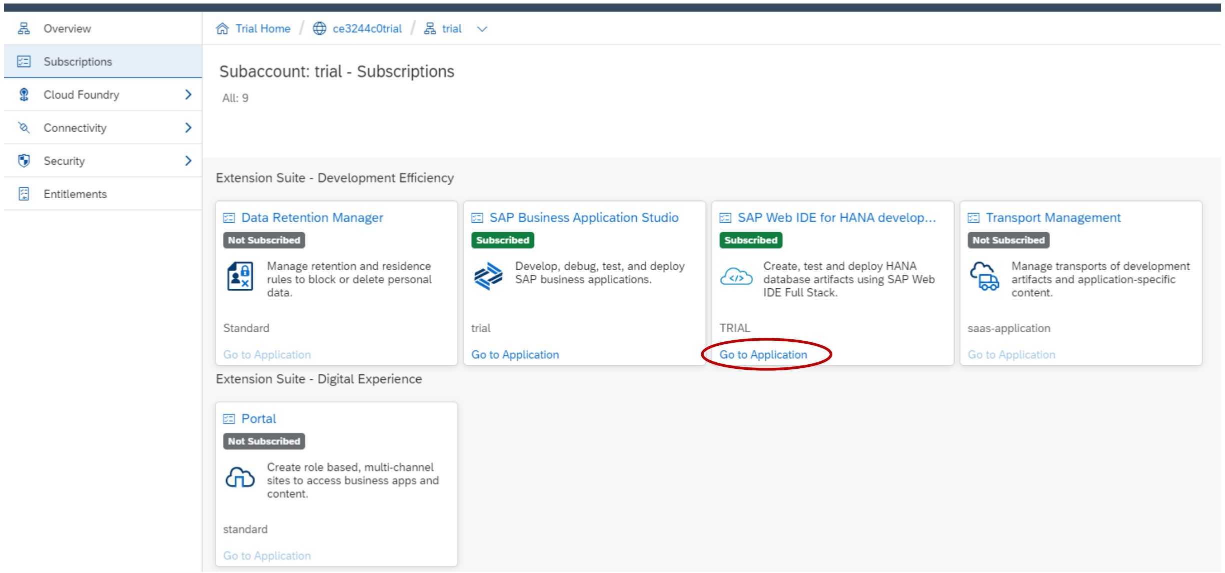
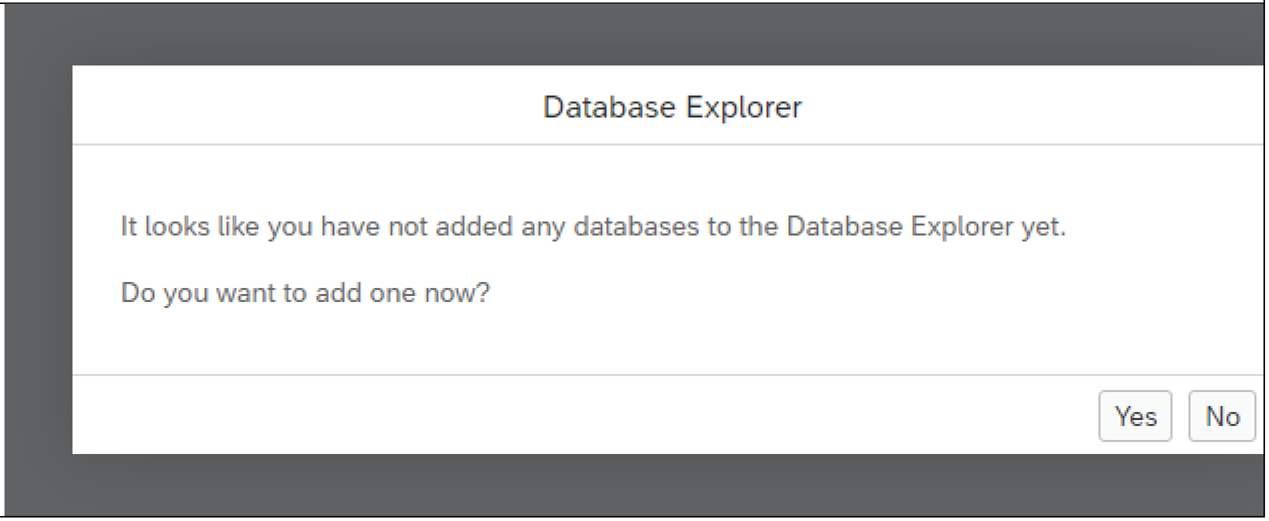
Exercise 5. Re-Run Query and Note Performance Difference

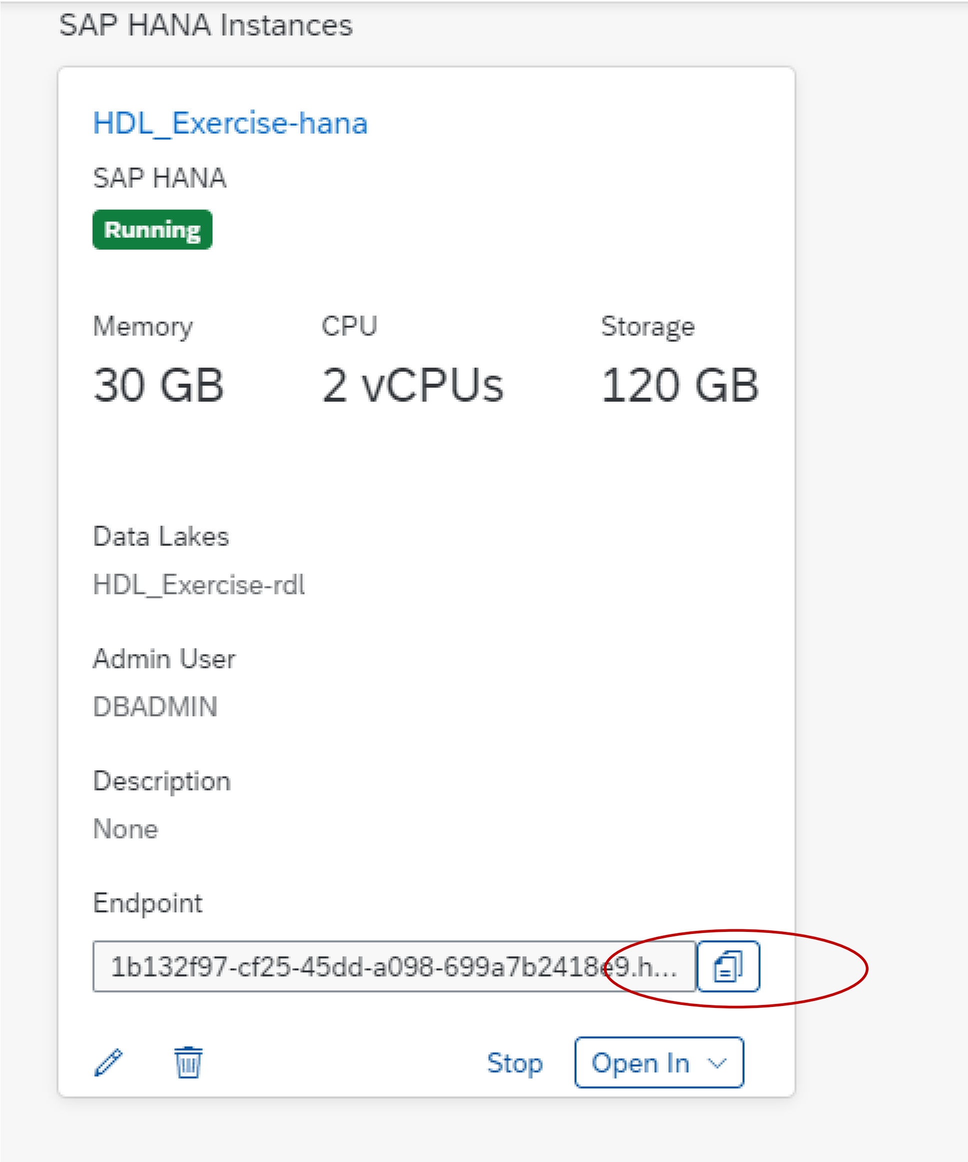
Step Explanation	Screenshot
1. Rerun the statement SELECT * FROM DBADMIN."ATH_nation"; Note: since the table is now in HANA Cloud's main memory, the query should execute much faster. The performance difference between Exercise 3 and Exercise 5, while executing the same query, **went from 3226 ms to 2 ms.**	 SQL Console 1.sql × Production System Analyze Current schema: ... Connected to: HDLE Result × Messages × History SELECT * FROM DBADMIN."ATH_nation"; Statement 'SELECT * FROM DBADMIN."ATH_nation"' executed in 2 ms.

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	 SAP Cloud Platform Cockpit Overview Subscriptions Cloud Foundry Connectivity Security Entitlements Trial Home / ce3244c0trial / trial Cloud Foundry Environment Entitlements Subaccount: trial Subdomain: ce3244c0trial ID: 6fce28d4-86da-4793-be05-e08e7a563255 2 (9 available) Active Subscriptions Org. Name: ce3244c0trial Org. ID: eeb51715-9d0d-41ca-93fa-7c8636a973da Members: 1 API Endpoint: https://api.cf.us10.hana.ondemand.com

2. Find SAP Web IDE tile, and click on Go to Application	 Overview Subscriptions Cloud Foundry Connectivity Security Entitlements Trial Home / ce3244c0trial / trial Subaccount: trial - Subscriptions All: 9 Extension Suite - Development Efficiency Data Retention Manager Not Subscribed Manage retention and residence rules to block or delete personal data. Standard Go to Application SAP Business Application Studio Subscribed Develop, debug, test, and deploy SAP business applications. trial Go to Application SAP Web IDE for HANA develop... Subscribed Create, test and deploy HANA database artifacts using SAP Web IDE Full Stack. TRIAL Go to Application Transport Management Not Subscribed Manage transports of development artifacts and application-specific content. saas-application Go to Application Extension Suite - Digital Experience Portal Not Subscribed Create role based, multi-channel sites to access business apps and content. standard Go to Application
3. When prompted to add a database to the database explorer, click on yes	 Database Explorer It looks like you have not added any databases to the Database Explorer yet. Do you want to add one now?

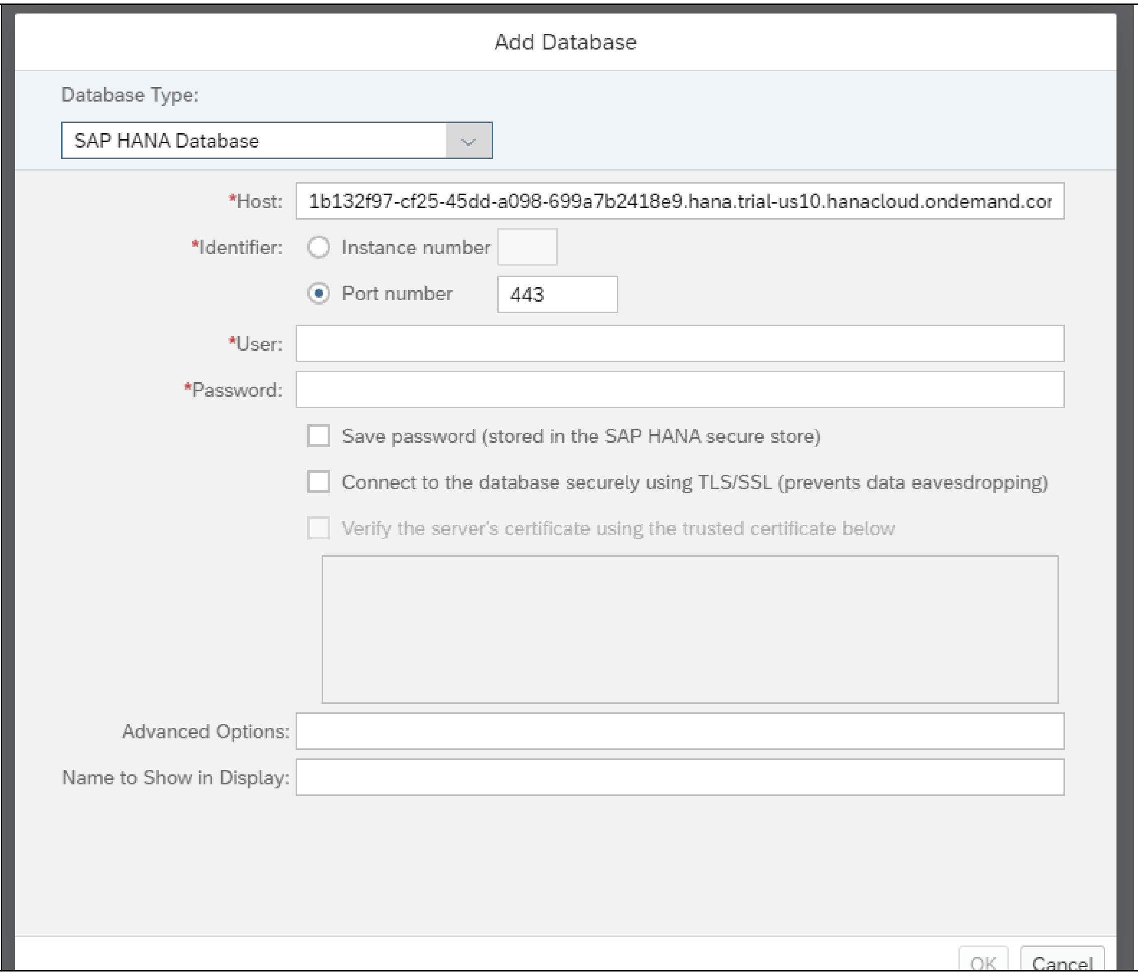
4. Navigate to your HANA Cloud instance and copy the end point	 SAP HANA Instances HDL_Exercise-hana SAP HANA Running Memory 30 GB CPU 2 vCPUs Storage 120 GB Data Lakes HDL_Exercise-rdl Admin User DBADMIN Description None Endpoint 1b132f97-cf25-45dd-a098-699a7b2418e9.h...

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.cor

*Identifier: ○ Instance number

● Port number 443

*User:

*Password:

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

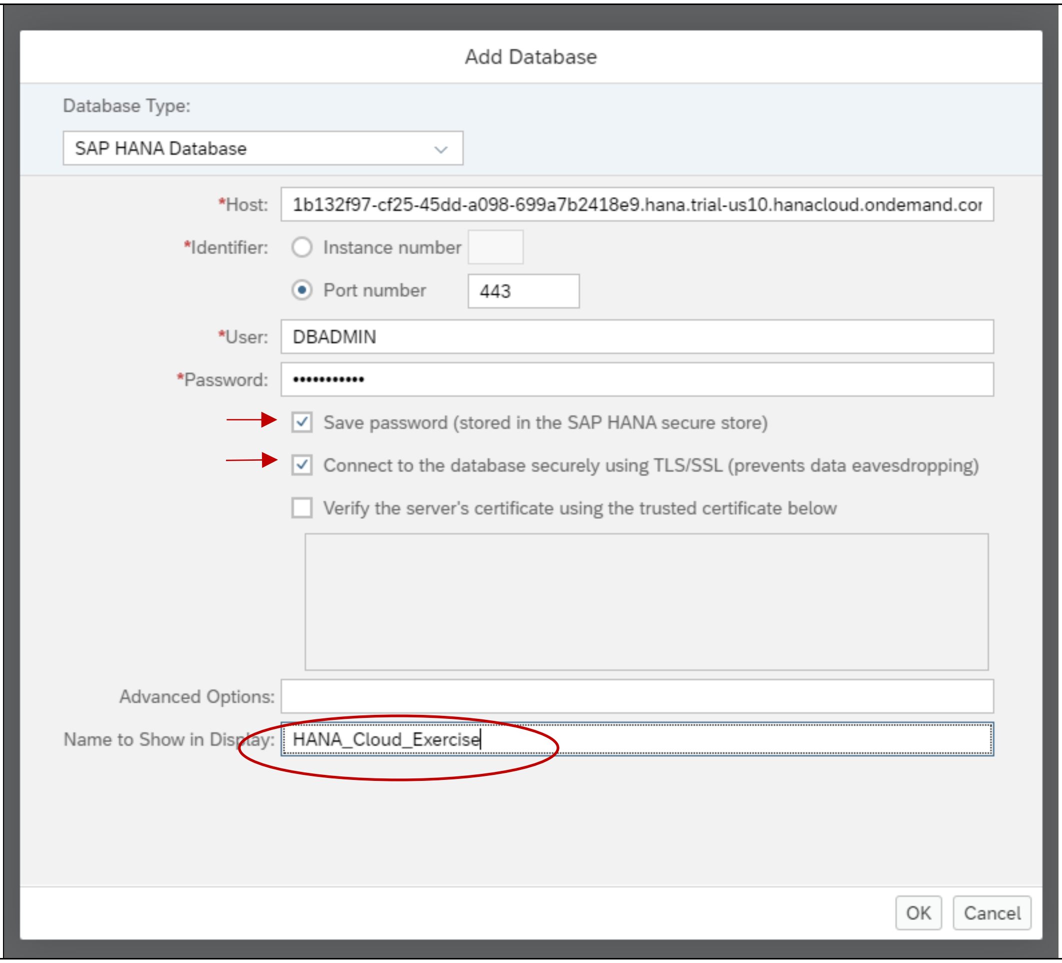
□ Connect to the database securely using TLS/SSL (prevents data eavesdropping)

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

Advanced Options:

Name to Show in Display:

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.



Add Database

Database Type: SAP HANA Database

*Host: 1b132f97-cf25-45dd-a098-699a7b2418e9.hana.trial-us10.hanacloud.ondemand.cor

*Identifier: ○ Instance number

● Port number 443

*User: DBADMIN

*Password: •••••••••••

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

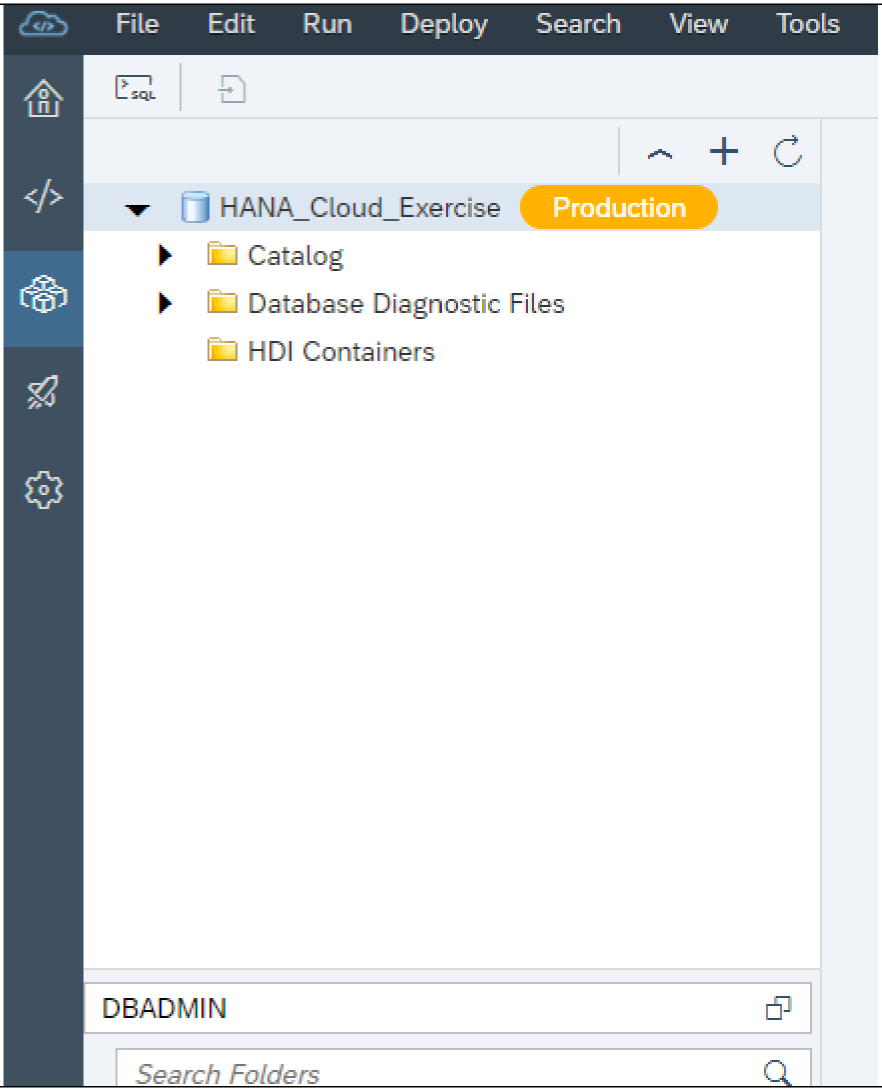
☑ Connect to the database securely using TLS/SSL (prevents data eavesdropping)

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

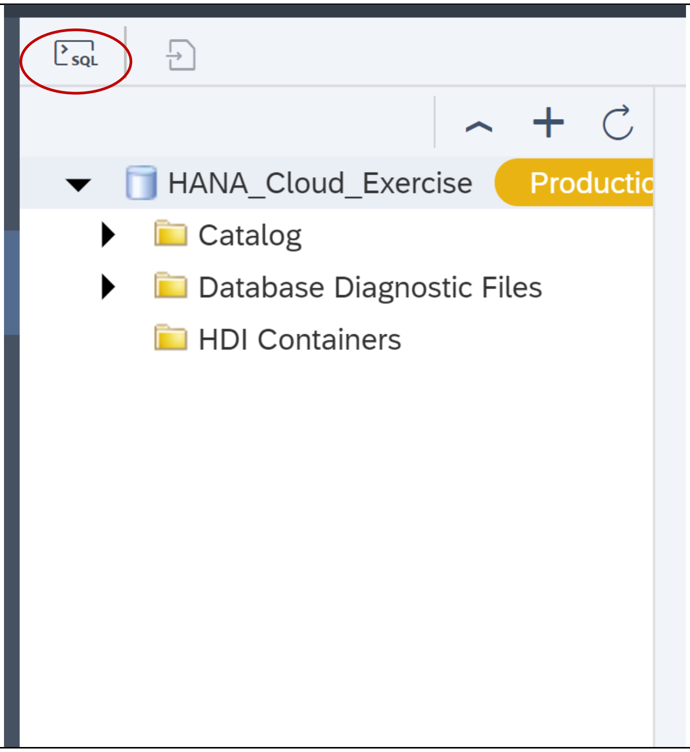
Advanced Options:

Name to Show in Display: HANA_Cloud_Exercise

7. You should now see the database

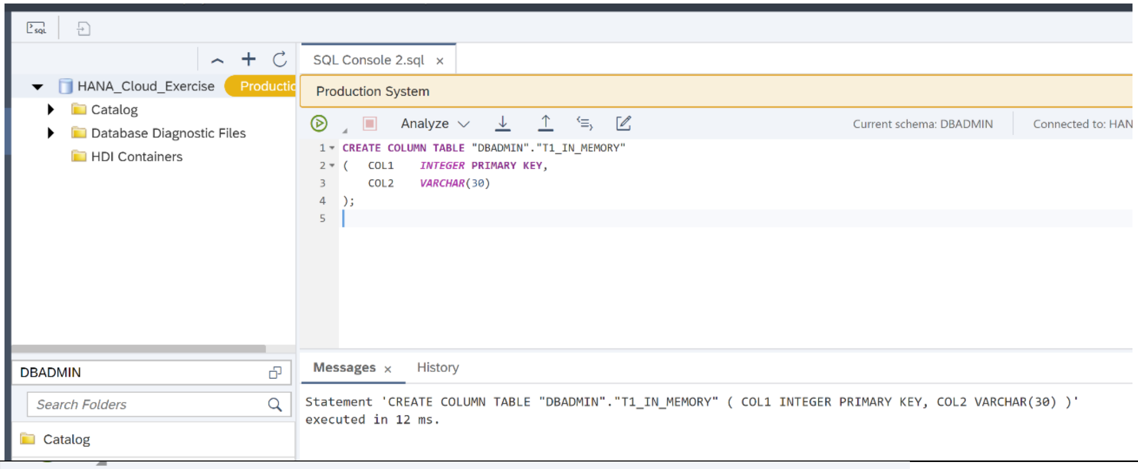


Exercise 1. Execute SQL to create a table in NSE storage

Step Explanation	Screenshot
1. Open a SQL console by selecting the database and clicking on the SQL icon	 SQL HANA_Cloud_Exercise Productio Catalog Database Diagnostic Files HDI Containers

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)
);
```



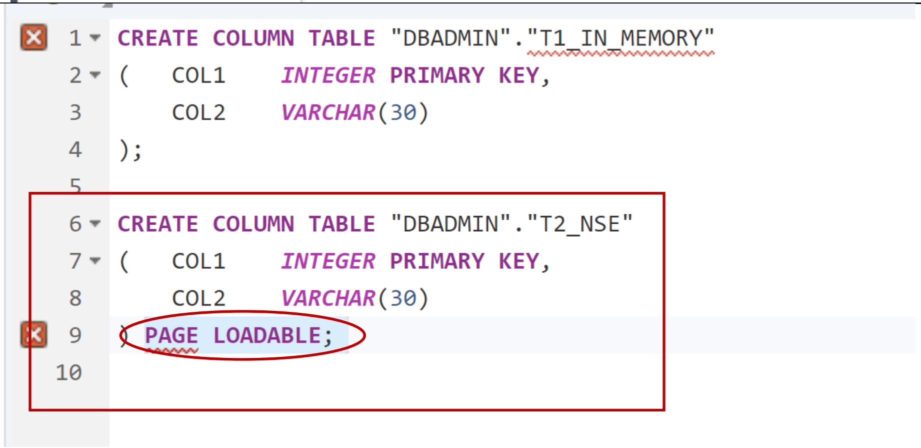
```
CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY"
(    COL1    INTEGER PRIMARY KEY,
     COL2    VARCHAR(30)
);
```

```
Statement 'CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY" ( COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30) )'
executed in 12 ms.
```

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

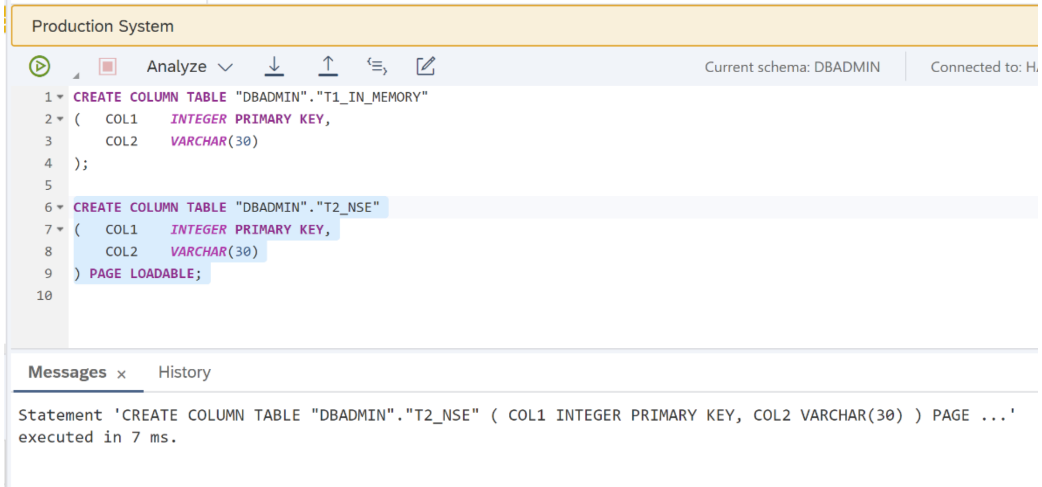
```
CREATE COLUMN TABLE "DBADMIN"."T2_NSE"
(	COL1	INTEGER PRIMARY KEY,
	COL2	VARCHAR(30)
) PAGE LOADABLE;
```

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



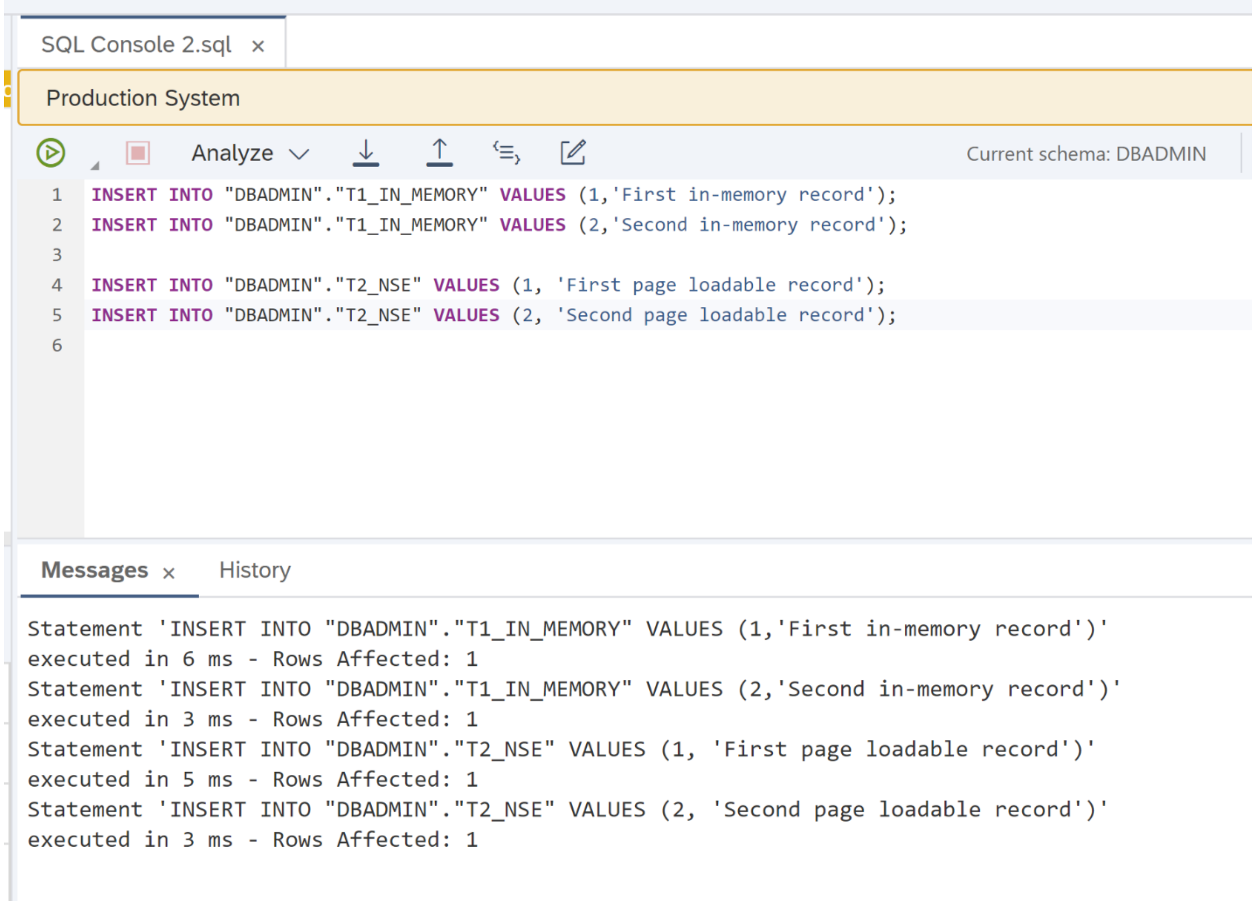
```
CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY"
(    COL1    INTEGER PRIMARY KEY,
     COL2    VARCHAR(30)
);

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

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



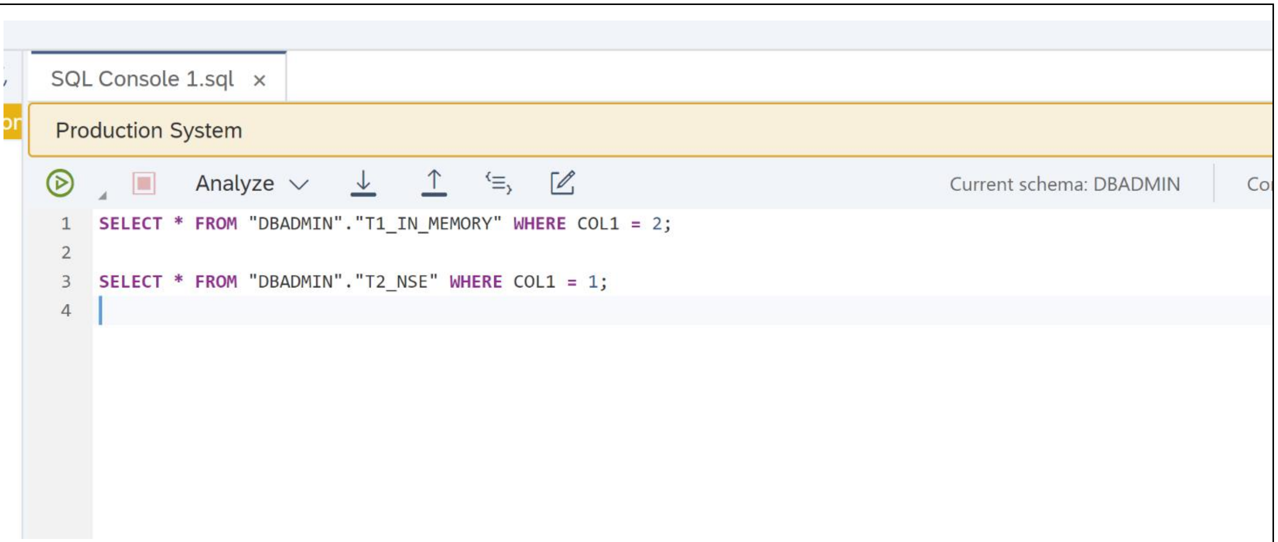
```
CREATE COLUMN TABLE "DBADMIN"."T1_IN_MEMORY"
(    COL1     INTEGER PRIMARY KEY,
     COL2     VARCHAR(30)
);

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

```

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

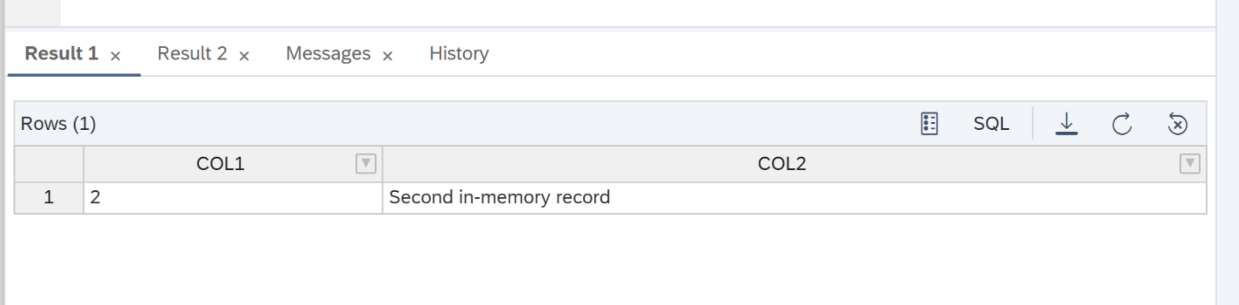
5. Insert values into both tables by running the following statements: INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (1,'First in-memory record'); INSERT INTO "DBADMIN"."T1_IN_MEMORY" VALUES (2,'Second in-memory record'); INSERT INTO "DBADMIN"."T2_NSE" VALUES (1, 'First page loadable record'); INSERT INTO "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.	 SQL Console 2.sql × Production System Analyze Current schema: DBADMIN 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'); 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
6. Query the tables by running the following SELECT statements: SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;	

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”.

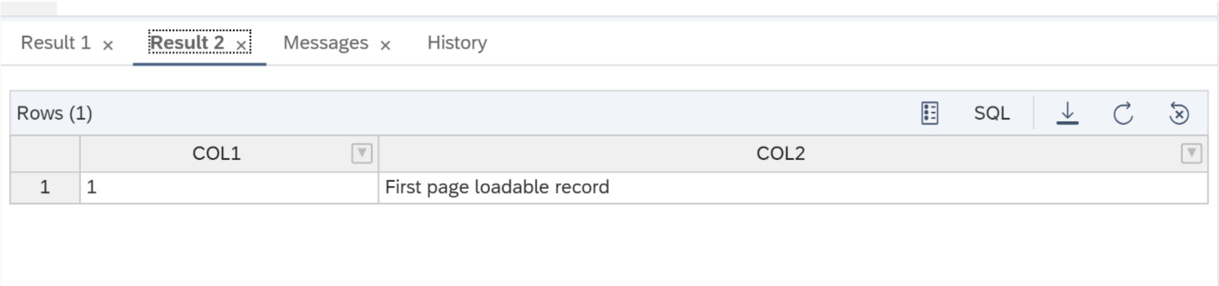


```
SELECT * FROM "DBADMIN"."T1_IN_MEMORY" WHERE COL1 = 2;

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



Rows (1)

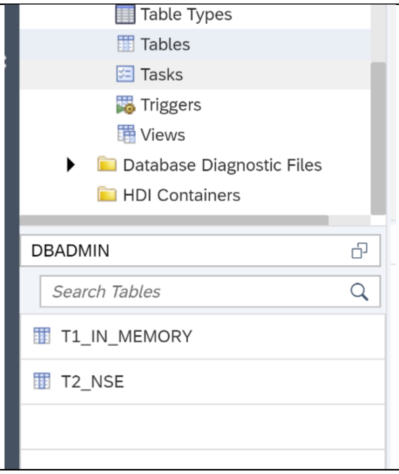
	COL1	COL2
1	2	Second in-memory record



Rows (1)

	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

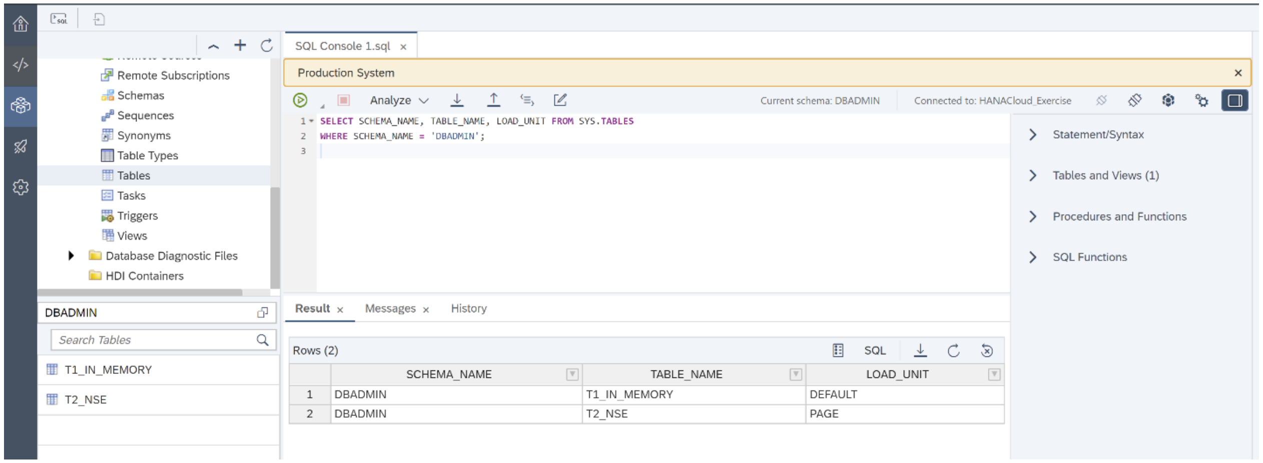


Exercise 2. View the Load Unit configuration

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.

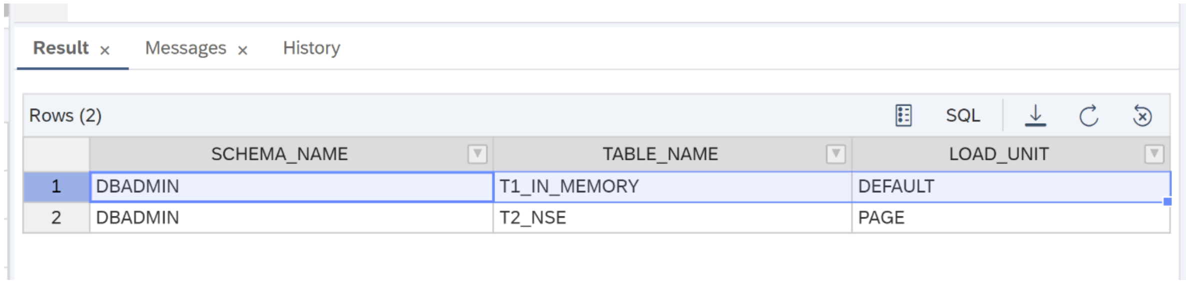


```
SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES
WHERE SCHEMA_NAME = 'DBADMIN';
```

Rows (2)

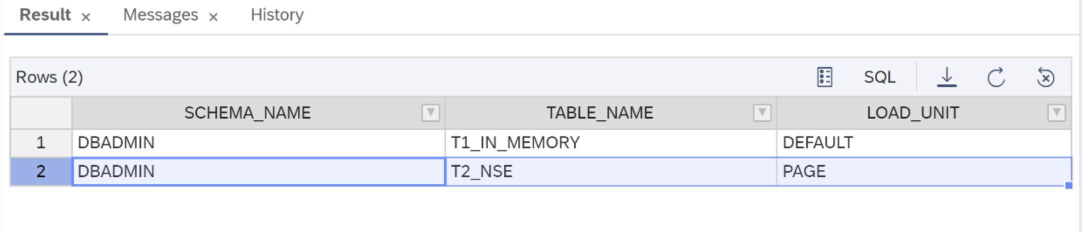
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT
1	DBADMIN	T1_IN_MEMORY	DEFAULT
2	DBADMIN	T2_NSE	PAGE

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**.



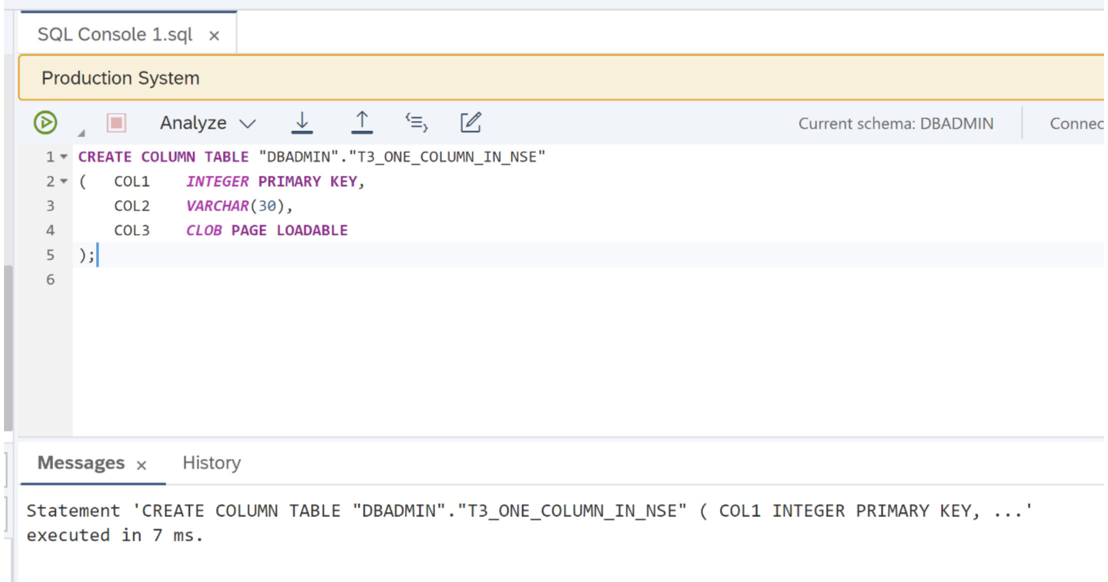
	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT
1	DBADMIN	T1_IN_MEMORY	DEFAULT
2	DBADMIN	T2_NSE	PAGE

3. click on “2”.
The LOAD UNIT for the **T2_NSE** table is reported as **PAGE** indicating that this is a page loadable table.



	SCHEMA_NAME	TABLE_NAME	LOAD_UNIT
1	DBADMIN	T1_IN_MEMORY	DEFAULT
2	DBADMIN	T2_NSE	PAGE

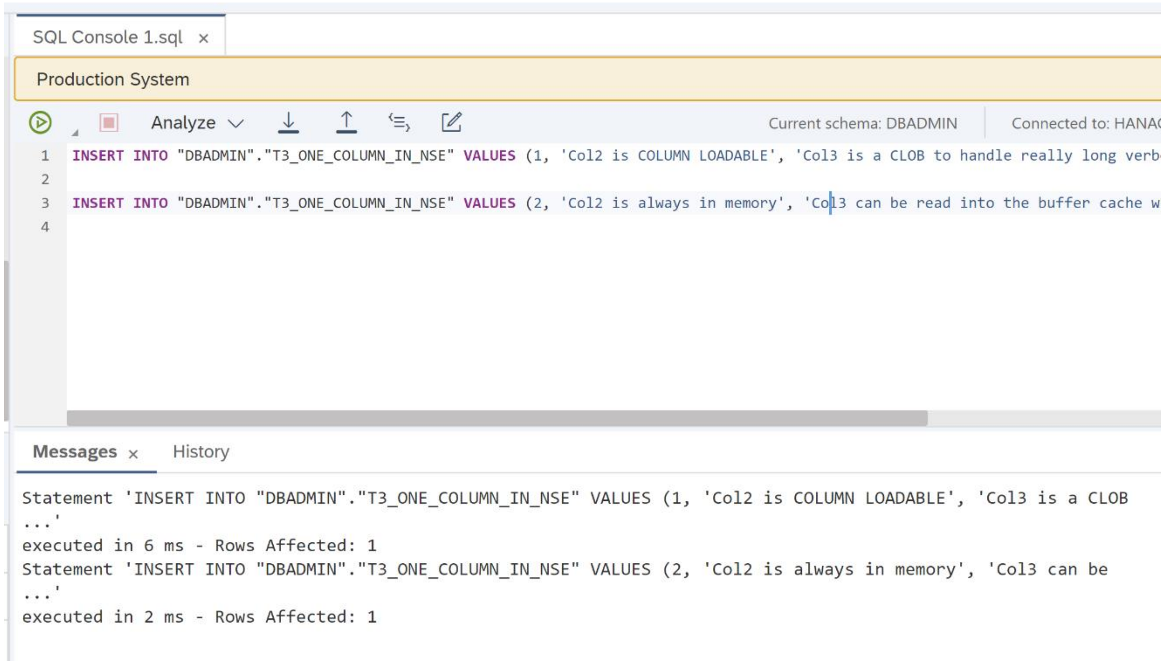
Exercise 3. Create and use a table with a specific column configured as page loadable

Step Explanation	Screenshot
1. Clear the SQL console and run the following statement: CREATE COLUMN TABLE "DBADMIN"."T3_ONE_COLUMN_IN_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30), COL3 CLOB PAGE LOADABLE); 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.	 SQL Console 1.sql × Production System Analyze Current schema: DBADMIN Connec Messages × History CREATE COLUMN TABLE "DBADMIN"."T3_ONE_COLUMN_IN_NSE" (COL1 INTEGER PRIMARY KEY, COL2 VARCHAR(30), COL3 CLOB PAGE LOADABLE); Statement 'CREATE COLUMN TABLE "DBADMIN"."T3_ONE_COLUMN_IN_NSE" (COL1 INTEGER PRIMARY KEY, ...' executed in 7 ms.

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');

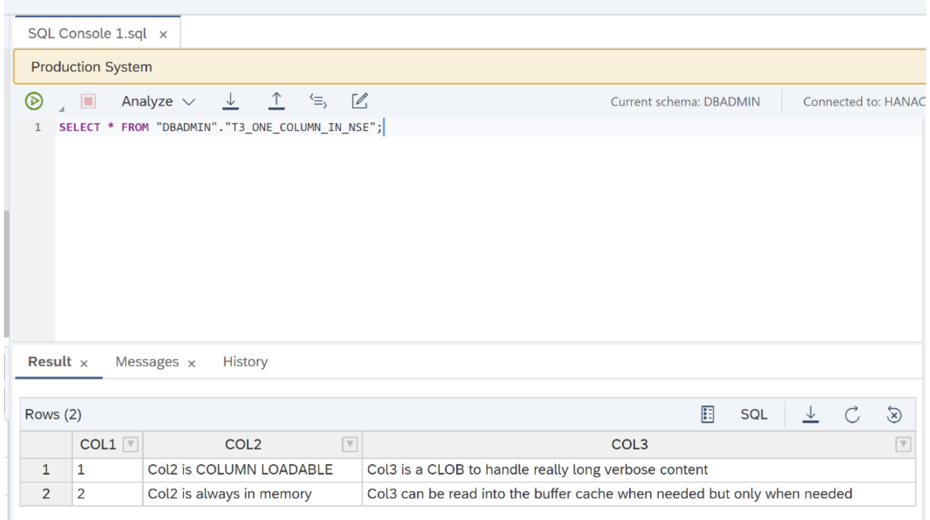


```
INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (1, 'Col2 is COLUMN LOADABLE', 'Col3 is a CLOB to handle really long verb

INSERT INTO "DBADMIN"."T3_ONE_COLUMN_IN_NSE" VALUES (2, 'Col2 is always in memory', 'Col3 can be read into the buffer cache w
```

```
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";



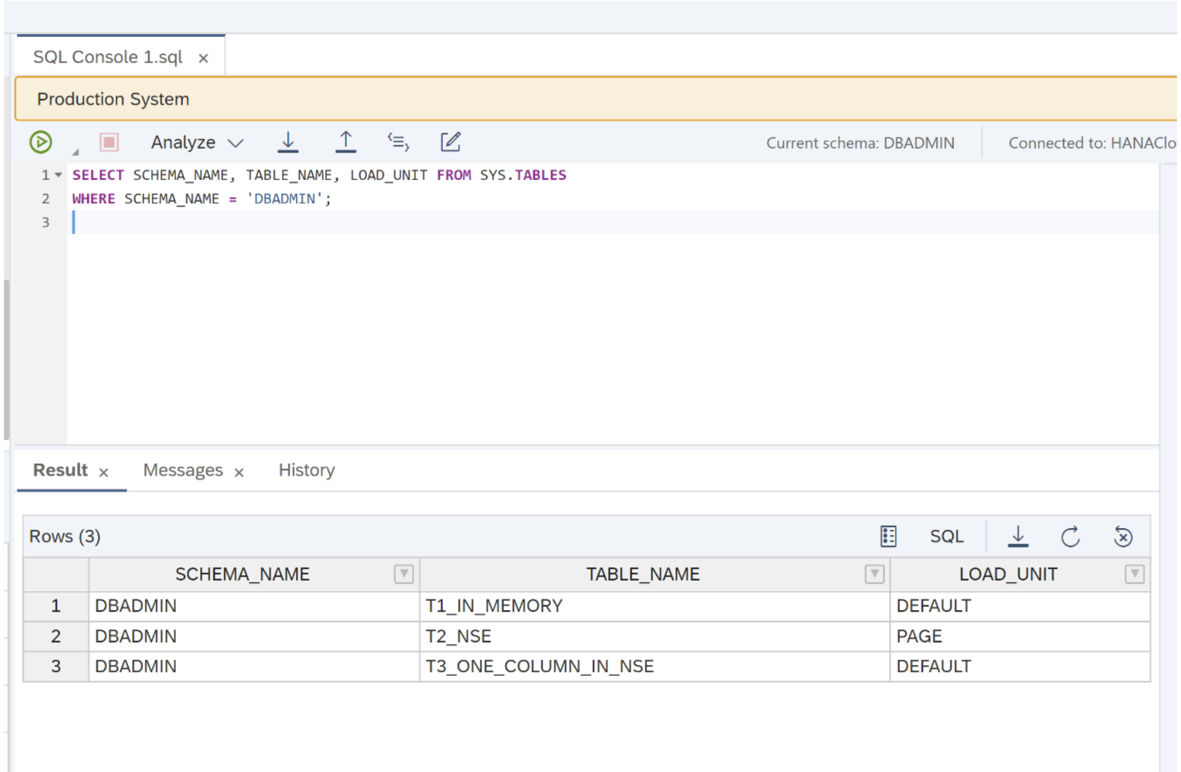
```
SELECT * FROM "DBADMIN"."T3_ONE_COLUMN_IN_NSE";
```

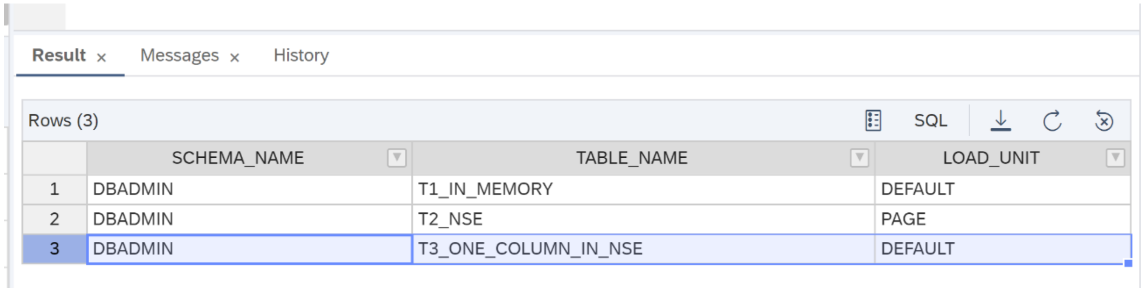
Rows (2)

	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

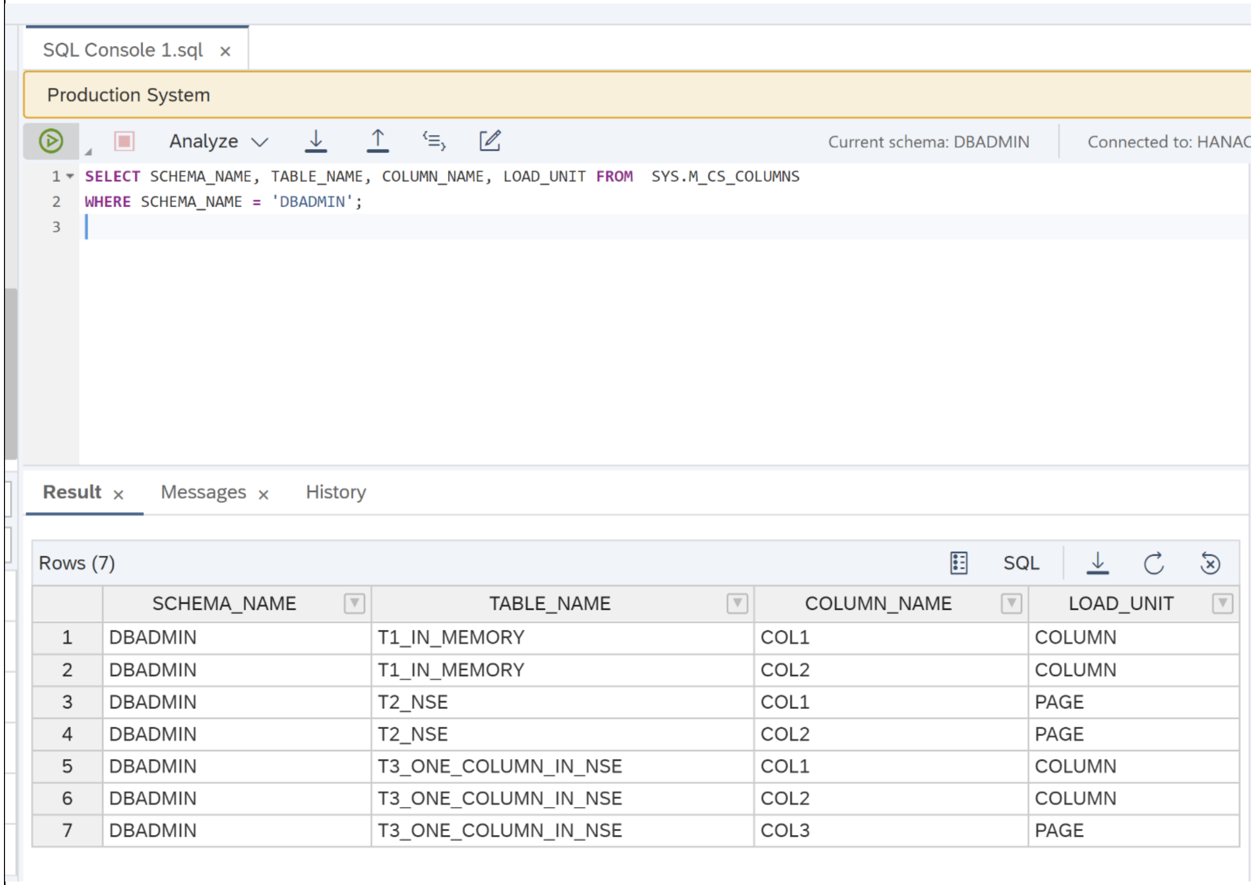
Step Explanation	Screenshot

1. Query the SYS.TABLES view to see what LOAD UNIT is configured at the table level by running the following statement:
SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES
WHERE SCHEMA_NAME = 'DBADMIN';
	
	 SQL Console 1.sql ×
	Production System
	Analyze
	Current schema: DBADMIN
	Connected to: HANAClo
	Result ×
	Messages ×
	History
	SQL
	`SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES`
	`WHERE SCHEMA_NAME = 'DBADMIN';`
	Rows (3)
	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

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.	 Rows (3) 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';

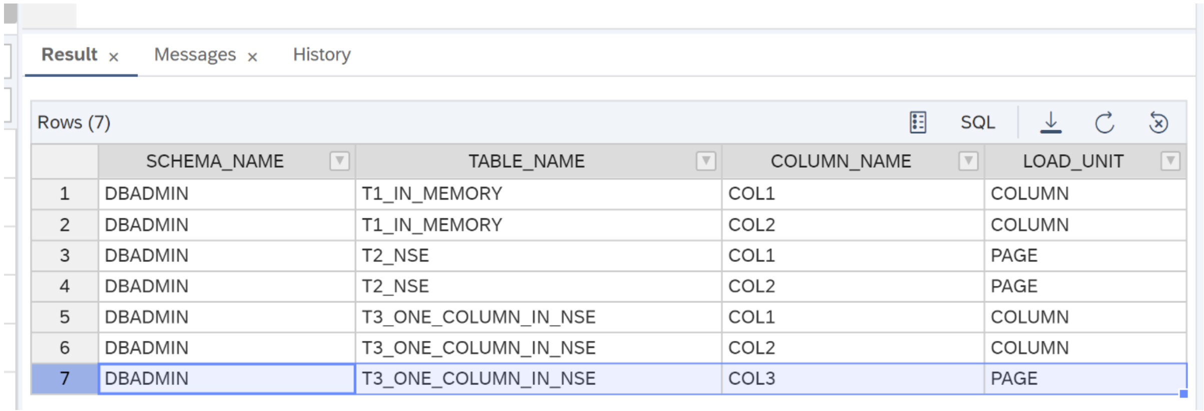


```
SELECT SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, LOAD_UNIT FROM  SYS.M_CS_COLUMNS
WHERE SCHEMA_NAME = 'DBADMIN';
```

	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.

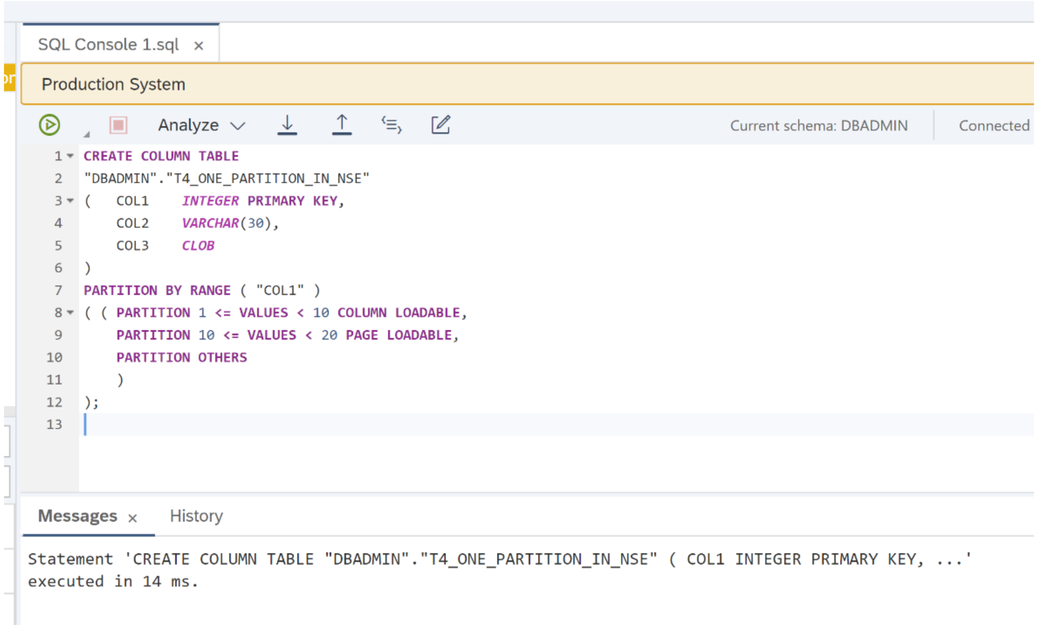
Also note that the **LOAD UNIT** for the first 2 columns of **T3_ONE_COLUMN_IN_NSE** (row 6 and 7) are shown as **COLUMN** because that is the default LOAD UNIT and no other LOAD UNIT was specified for these columns in the CREATE TABLE statement.



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

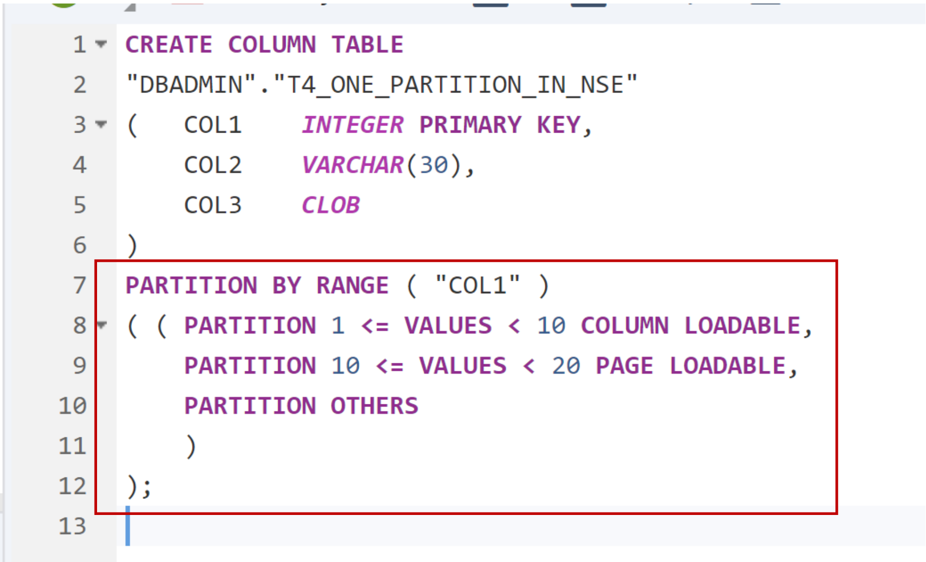
Exercise 5. Create and use a table with a specific partition configured as page loadable

Step Explanation	Screenshot
1. Run the following statement to create a table: 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)); 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**. 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.	 SQL Console 1.sql × Production System Analyze Current schema: DBADMIN Connected Messages × History 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)); Statement 'CREATE COLUMN TABLE "DBADMIN"."T4_ONE_PARTITION_IN_NSE" (COL1 INTEGER PRIMARY KEY, ...' executed in 14 ms.

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**.



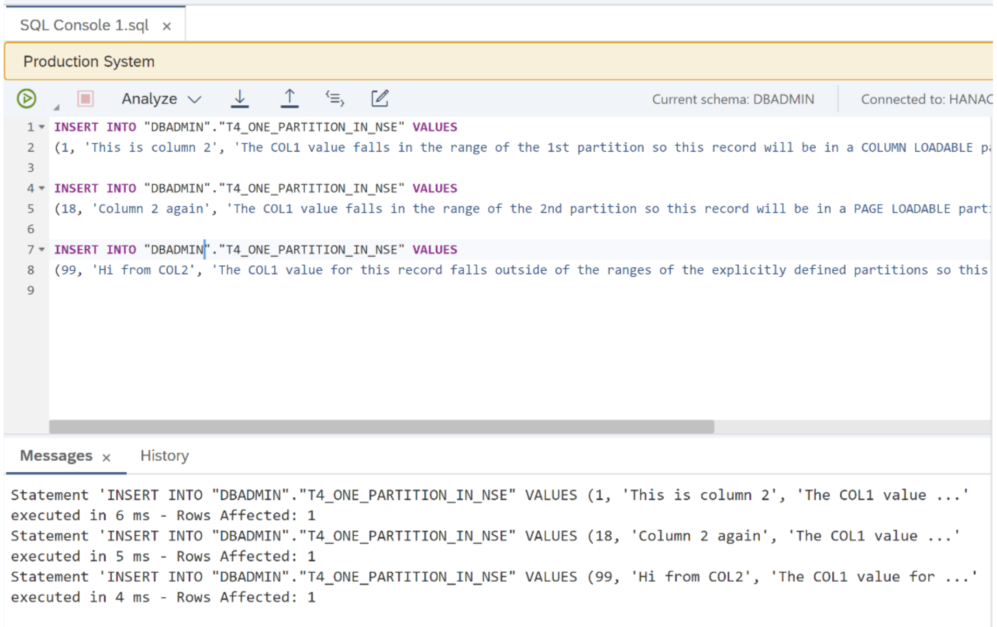
```
CREATE COLUMN TABLE
"DBADMIN"."T4_ONE_PARTITION_IN_NSE"
(   COL1    INTEGER PRIMARY KEY,
    COL2    VARCHAR(30),
    COL3    CLOB
)
PARTITION BY RANGE ( "COL1" )
( ( PARTITION 1 <= VALUES < 10 COLUMN LOADABLE,
    PARTITION 10 <= VALUES < 20 PAGE LOADABLE,
    PARTITION OTHERS
    )
);
```

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');

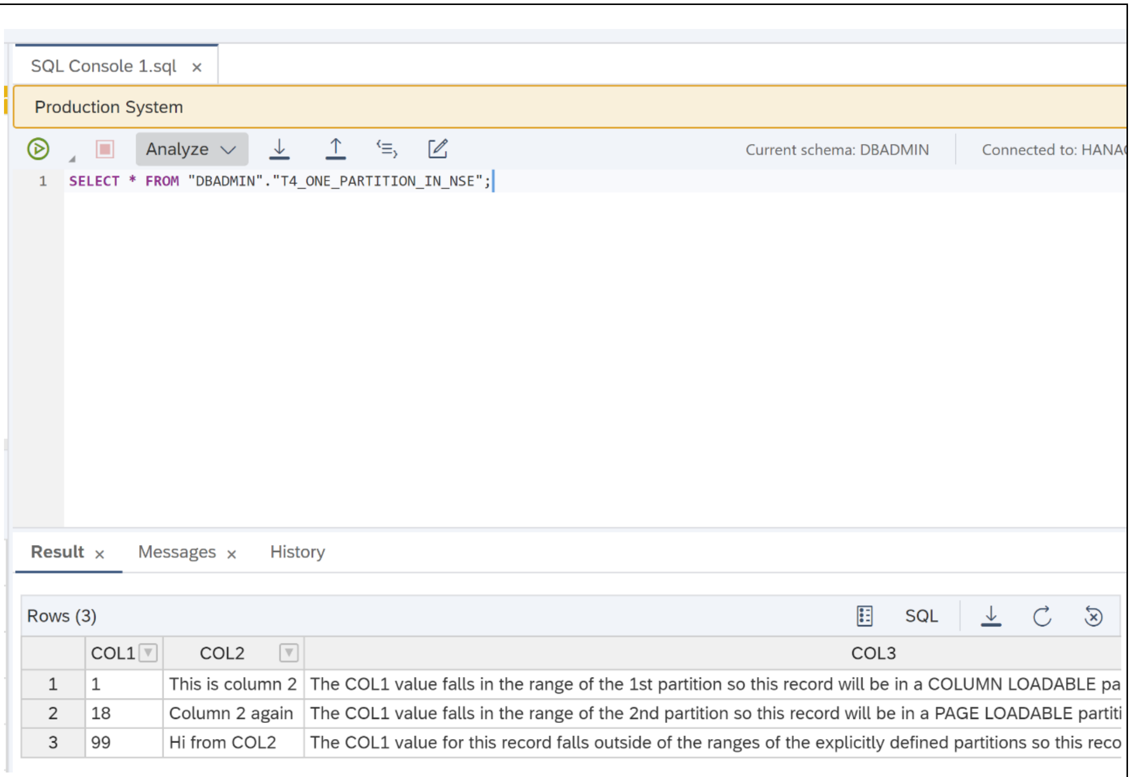


```
INSERT INTO "DBADMIN"."T4_ONE_PARTITION_IN_NSE" VALUES
(1, 'This is column 2', 'The COL1 value falls in the range of the 1st partition so this record will be in a COLUMN LOADABLE pa

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 part

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
```

```
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";
```



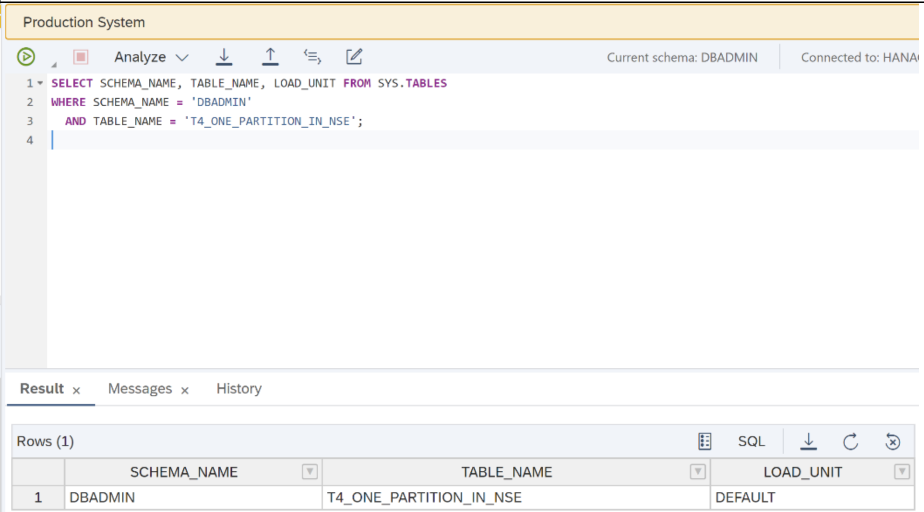
```
1  SELECT * FROM "DBADMIN"."T4_ONE_PARTITION_IN_NSE";
```

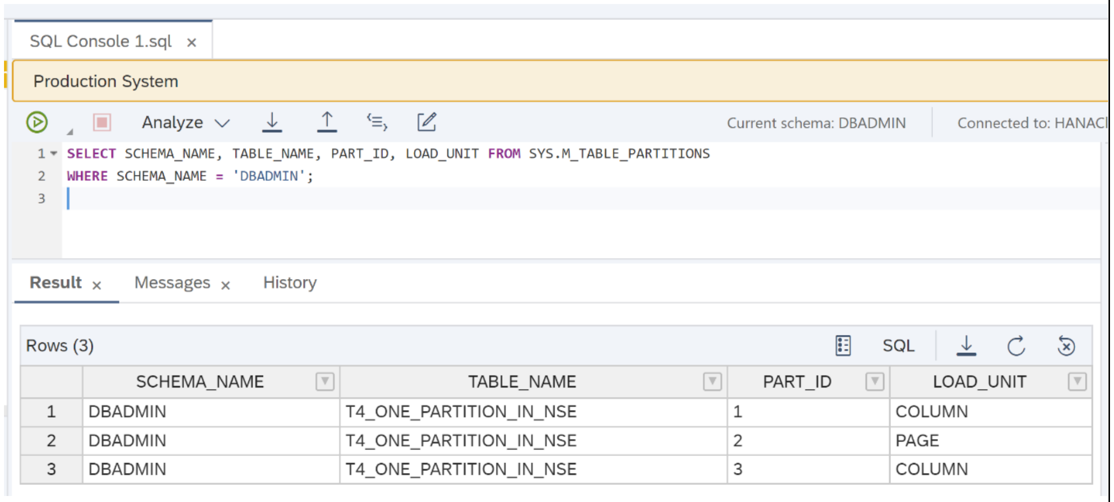
Rows (3)

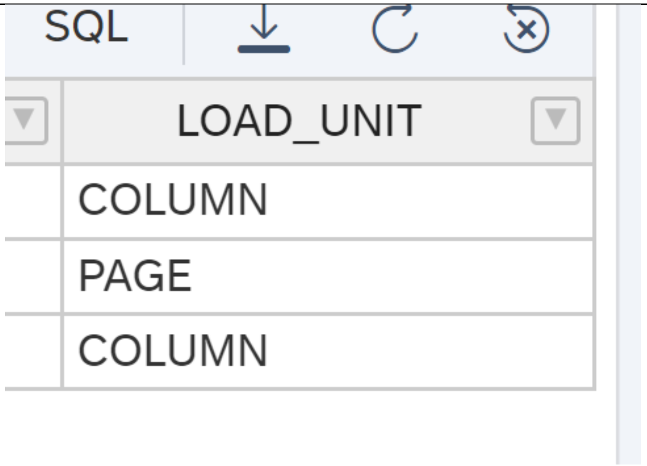
	COL1	COL2	COL3
1	1	This is column 2	The COL1 value falls in the range of the 1st partition so this record will be in a COLUMN LOADABLE pa
2	18	Column 2 again	The COL1 value falls in the range of the 2nd partition so this record will be in a PAGE LOADABLE partiti
3	99	Hi from COL2	The COL1 value for this 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

1. Review the table level LOAD UNIT for the T4 table by running the following statement:
 	
SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES
WHERE SCHEMA_NAME = 'DBADMIN'
  AND TABLE_NAME = 'T4_ONE_PARTITION_IN_NSE';
 	
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**.
	
	Production System
	
	Current schema: DBADMIN
	Connected to: HANA
	SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES
	WHERE SCHEMA_NAME = 'DBADMIN'
	  AND TABLE_NAME = 'T4_ONE_PARTITION_IN_NSE';
	
	Rows (1)
	
	SCHEMA_NAME, TABLE_NAME, LOAD_UNIT
	1 DBADMIN, T4_ONE_PARTITION_IN_NSE, DEFAULT

2. Review the details of individual table partitions, including the LOAD UNIT configuration by running the following statement:
 	
SELECT SCHEMA_NAME, TABLE_NAME, PART_ID, LOAD_UNIT FROM SYS.M_TABLE_PARTITIONS
WHERE SCHEMA_NAME = 'DBADMIN';
 	
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.
	
	
	Production System
	
	Current schema: DBADMIN
	Connected to: HANACI
	SELECT SCHEMA_NAME, TABLE_NAME, PART_ID, LOAD_UNIT FROM SYS.M_TABLE_PARTITIONS
	WHERE SCHEMA_NAME = 'DBADMIN';
	
	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