

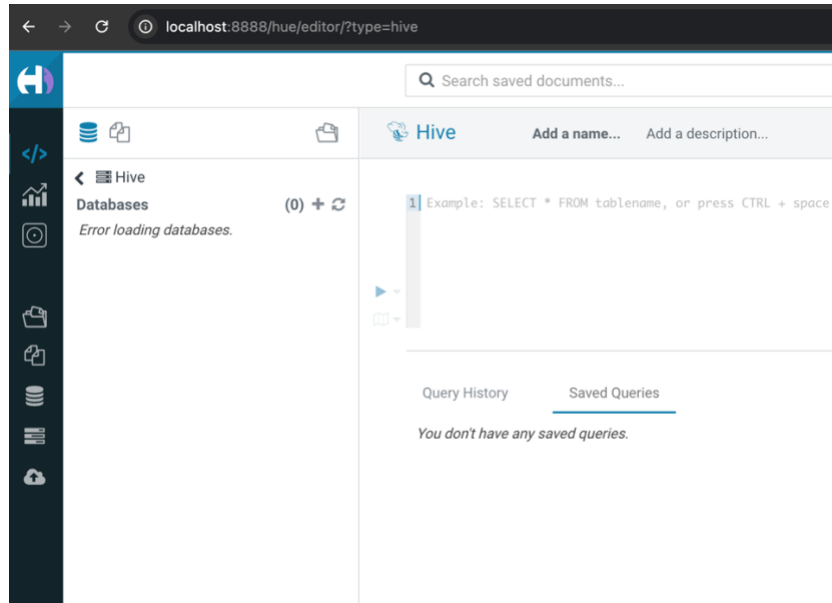
Hive Practice

CONTENTS

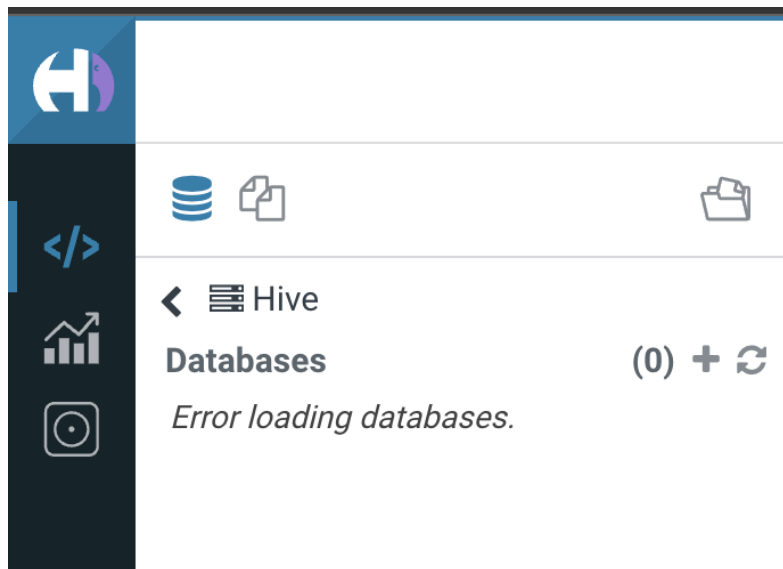
1.	Basic Hive Interaction Using Hue (15 points)	2
2.	Basic Hive Interaction Using Beeline (10 points)	9
3.	Managed and External Tables Using Beeline (20 points)	10
4.	Partitioned Tables (35 points)	14
5.	Hive ACID Tables (20 points).....	18

BASIC HIVE INTERACTION USING HUE (15 POINTS)

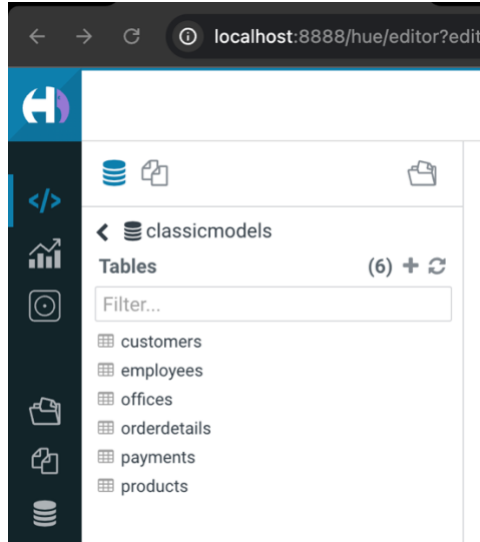
- Connect to Hue using “admin/admin”



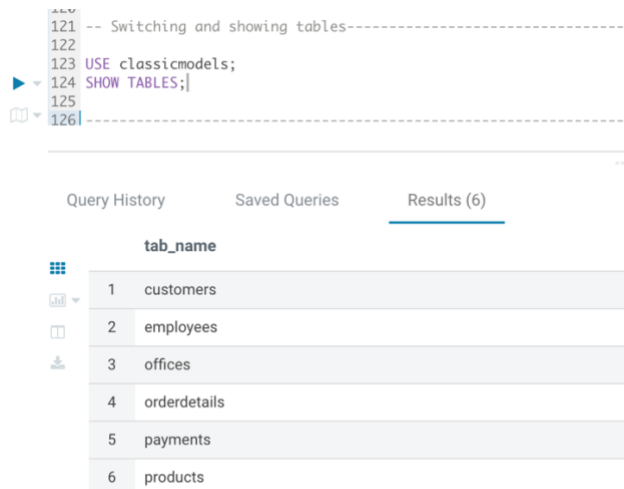
- Browse over to the Hive editor and perform the following:
 - Show available databases under Hive



- Create the “classicmodels” database and upload the database from the attachment of the Homework module; verify that the separator is considered successfully



- Switch to it
- Show all the available tables in the database



- Expand the “Customers” table and view its columns and data types

The screenshot shows the Hive Table Browser interface. On the left, a sidebar lists various tables under the 'classicmodels' database, including 'customers', 'employees', 'offices', 'orderdetails', 'payments', and 'products'. The 'customers' table is selected. The main panel displays the table's properties, schema, and a sample of data. The schema lists 13 columns: customer_id, business_name, last_name, first_name, phone_number, address_line1, address_line2, city, state, postal_code, country, country_code, and credit_limit. The sample data shows rows for customers like 'Signal Gift Stores', 'King', 'Jean', '7025551838', '8489 Strong St', 'Las Vegas', 'NV', '89030', 'USA', and '1166'.

Column (13)	Type	Description	Sample
customer_id	string	from deserializer	103
business_name	string	from deserializer	Atelier graphique
last_name	string	from deserializer	Schmitt
first_name	string	from deserializer	Carine
phone_number	string	from deserializer	40.32.2555
address_line1	string	from deserializer	54, rue Royale
address_line2	string	from deserializer	null
city	string	from deserializer	Nantes
state	string	from deserializer	null
postal_code	string	from deserializer	44000
country	string	from deserializer	France
country_code	string	from deserializer	1378
credit_limit	string	from deserializer	21000.00

I've tried both populating after table creation, as well as populating while creating. Both methods drop all columns data type to string. I found that it is SerDe behaviour, so in general production environment it is better to create staging table with string types and then copy from there to a typed schema table of ORC or Parquet storage. Currently working with untyped tables (all-string tables), but all math operations will work, since Hive manages casting implicitly while comparing.

- Perform the following queries:
 - Query all rows from the “Employees” table

The screenshot shows the Hive Query Editor with a SQL query: `select * from classicmodels.employees;`. Below the query, the results are displayed in a table with 5 columns: employees.employee_id, employees.last_name, employees.first_name, employees.extension, and employees.email. The results show 11 rows of employee data.

	employees.employee_id	employees.last_name	employees.first_name	employees.extension	employees.email
1	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com
2	1056	Patterson	Mary	x4611	mpatterson@classicmodelcars.com
3	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com
4	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com
5	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com
6	1143	Bow	Anthony	x5428	abow@classicmodelcars.com
7	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com
8	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com
9	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com
10	1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com
11	1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com

- Alter the previous query to fetch only the first 10 rows

```

106 select * from classicmodels.employees
107 limit 10
108 ;

```

	employees.employee_id	employees.last_name	employees.first_name	employees.extension	employees.email
1	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com
2	1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com
3	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com
4	1088	Patterson	William	x4871	wpatterson@classicmodelcars.com
5	1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com
6	1143	Bow	Anthony	x5428	abow@classicmodelcars.com
7	1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com
8	1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com
9	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com
10	1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com

- Write a query to fetch the following:
 - The employee ID: first name and last name
 - The employee number should be between 1002 and 1100
 - Order by last name in descending order
 - Fetch only first five rows

```

111
112 select employee_id, first_name, last_name
113 from classicmodels.employees
114 where employee_id between 1002 and 1100
115 order by last_name desc
116 limit 5
117 ;
118
119

```

	employee_id	first_name	last_name
1	1088	William	Patterson
2	1056	Mary	Patterson
3	1002	Diane	Murphy
4	1076	Jeff	Firrelli

- Write a query to fetch the number of employees per job title, ordered by number of employees in descending order

```

120
121 SELECT
122     job_title,
123     COUNT(*) AS employees_count
124 FROM classicmodels.employees
125 GROUP BY job_title
126 ORDER BY employees_count DESC;
127

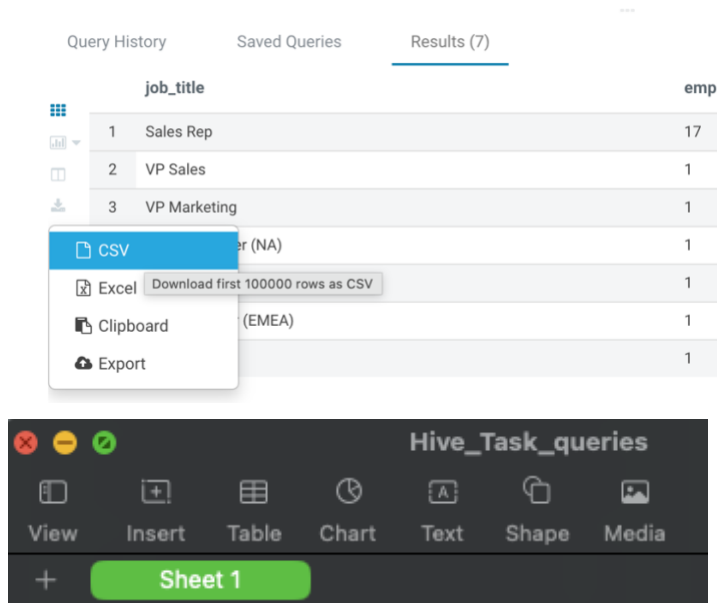
```

WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.

Query History Saved Queries Results (7)

	job_title	employees_count
1	Sales Rep	17
2	VP Sales	1
3	VP Marketing	1
4	Sales Manager (NA)	1
5	Sales Manager (APAC)	1
6	Sale Manager (EMEA)	1
7	President	1

- Export the query output to a text file



Query History Saved Queries Results (7)

	job_title	emp
1	Sales Rep	17
2	VP Sales	1
3	VP Marketing	1
	Sales Manager (NA)	1
	Sales Manager (APAC)	1
	Sale Manager (EMEA)	1
	President	1

CSV Excel Clipboard Export

Download first 100000 rows as CSV

Hive_Task_queries

View Insert Table Chart Text Shape Media

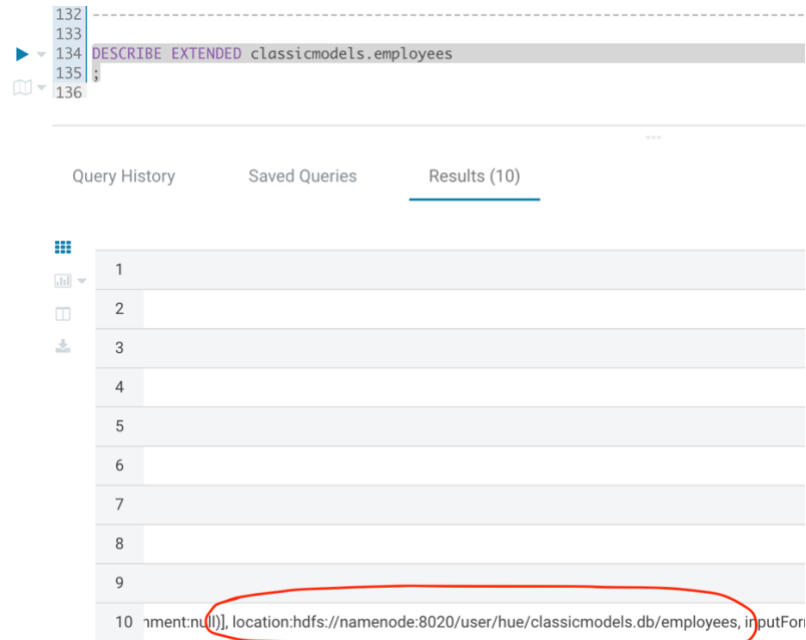
Sheet 1

Hive_Task_queries

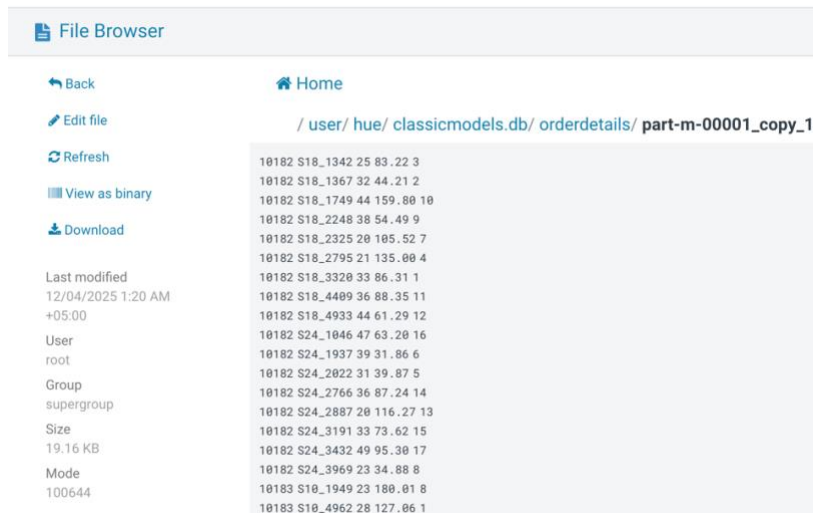
job_title	employees_count
Sales Rep	17
VP Sales	1
VP Marketing	1
Sales Manager (NA)	1
Sales Manager (APAC)	1
Sale Manager (EMEA)	1
President	1

- Check which HDFS folder the “employees” table points at

Hint: Use the practice guide to see how you can view such details for Hive tables.



- HDFS Browse
 - Use the HDFS browser in Hue to browse over to the HDFS folder and examine its contents
 - Click on one of the files to examine the file contents. Check the following:



- If the file is human-readable: **YES**
- Which Hive table property is responsible for this: **ROW FORMAT and SerDe controls how to read files.**

BASIC HIVE INTERACTION USING BEELINE (10 POINTS)

- Open a BASH session to the practice environment and connect to Hive using Beeline

```
batyagg@BatyrbhansMBP4 ~ % docker exec -it docker-hive-hive-server-1 bash
root@06b18bd0be9b:/opt# beeline -u jdbc:hive2://localhost:10000
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanat
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactor
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 2.3.2)
Driver: Hive JDBC (version 2.3.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.2 by Apache Hive
0: jdbc:hive2://localhost:10000> █
```

- Show available databases (Verify you see the “ClassisModels” database)

```
0: jdbc:hive2://localhost:10000> show databases;
+-----+
| database_name |
+-----+
| classicmodels |
| default       |
+-----+
```

- Switch to use the “ClassisModels” database

```
0: jdbc:hive2://localhost:10000> use classicmodels;
No rows affected (0.536 seconds)
0: jdbc:hive2://localhost:10000> select current_database();
+-----+
| _c0    |
+-----+
| classicmodels |
+-----+
1 row selected (0.819 seconds)
```

```
0: jdbc:hive2://localhost:10000> show tables;
+-----+
| tab_name |
+-----+
| customers |
| employees |
| offices   |
| orderdetails |
| payments  |
| products  |
+-----+
6 rows selected (0.393 seconds)
```

- Show all tables in this database

- Create a new database called "newdb" and verify the database was created

```
0: jdbc:hive2://localhost:10000> CREATE DATABASE newdb;
No rows affected (0.593 seconds)
0: jdbc:hive2://localhost:10000> SHOW DATABASES;
+-----+
| database_name |
+-----+
| classicmodels |
| default       |
| newdb         |
+-----+
3 rows selected (0.386 seconds)
```

- Create a table "new_emp" in "newdb" identical to the "Employees" table in "ClassicModels" database (Both schema and data), and run a COUNT(*) to verify the table is populated

```
0: jdbc:hive2://localhost:10000> use newdb;
No rows affected (0.277 seconds)
0: jdbc:hive2://localhost:10000> select count(*) from new_emp;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available
+-----+
| _c0 |
+-----+
| 23 |
+-----+
1 row selected (5.196 seconds)
```

MANAGED AND EXTERNAL TABLES USING BEELINE (20 POINTS)

- Open a BASH session to the practice environment and connect to Hive using Beeline
- Run a COUNT(*) to verify that "newdb.new_emp" is in place and populated

```
0: jdbc:hive2://localhost:10000> use newdb;
No rows affected (0.277 seconds)
0: jdbc:hive2://localhost:10000> select count(*) from new_emp;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available
+-----+
| _c0 |
+-----+
| 23 |
+-----+
1 row selected (5.196 seconds)
```

- Check using the table properties (without browsing HDFS):

```
0: jdbc:hive2://localhost:10000> describe extended new_emp;
+-----+-----+-----+
| col_name | data_type | comment |
+-----+-----+-----+
| employee_id | int | from deserializer |
| last_name | string | from deserializer |
| first_name | string | from deserializer |
| extension | string | from deserializer |
| email | string | from deserializer |
| office_code | string | from deserializer |
| reports_to | int | from deserializer |
| job_title | string | from deserializer |
| | NULL | NULL |
+-----+-----+-----+
Detailed Table Information | Table(tableName=new_emp, dbName=newdb, owner=root, createTime=1764860698, lastAccessTime=0, retention=0, sd=StorageDescriptor(cols=[FieldSchema(name=employee_id, type=int, comment=null), FieldSchema(name=last_name, type:string, comment=null), FieldSchema(name=first_name, type:string, comment=null), FieldSchema(name=extension, type:string, comment=null), FieldSchema(name=email, type:string, comment=null), FieldSchema(name=office_code, type:int, comment=null), FieldSchema(name=reports_to, type:int, comment=null), FieldSchema(name=job_title, type:string, comment=null)], location=hdfs://namenode:8020/user/hue/classicmodels.db/employees, inputFormat=org.apache.hadoop.mapred.TextInputFormat, outputFormat=org.apache.hadoop.hiveql.io.HiveIgnoreKeyTextOutputFormat, compressed=false, numBuckets=1, serdeInfo=SerDeInfo(name=null, serializationLib=org.apache.hadoop.hive.serde2.OpenCSVSerde, parameters={serialization.format=1, separator=,}), bucketCols=[], sortCols=[], parameters={}, skewedInfo=SkewedInfo(skewedColNames=[], skewedColValues=[], skewedColValueLocations=[]), storedAsSubDirectories=false), partitionKeys=[], parameters={transient_lastDdlTime=1764860698, totalSize=1661, EXTERNAL=TRUE, numFiles=4}, viewOriginalText=null, viewExpandedText=null, tableType=EXTERNAL_TABLE, rewriteEnabled=false) |
+-----+-----+-----+
10 rows selected (0.603 seconds)
```

Text in the “Detailed Table Information” is as follows:

```
Table(tableName:new_emp, dbName:newdb, owner:root, createTime:1764860698, lastAccessTime:0,
retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:employee_id, type:int, comment:null),
FieldSchema(name:last_name, type:string, comment:null), FieldSchema(name:first_name, type:string,
comment:null), FieldSchema(name:extension, type:string, comment:null), FieldSchema(name:email,
type:string, comment:null), FieldSchema(name:office_code, type:int, comment:null),
FieldSchema(name:reports_to, type:int, comment:null), FieldSchema(name:job_title, type:string,
comment:null)], location:hdfs://namenode:8020/user/hue/classicmodels.db/employees,
inputFormat:org.apache.hadoop.mapred.TextInputFormat,
outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false,
numBuckets:-1, serDeInfo:SerDeInfo(name:null,
serializationLib:org.apache.hadoop.hive.serde2.OpenCSVSerde, parameters:{serialization.format=1,
separatorChar=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[],
skewedColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[],
parameters:{transient_lastDdlTime=1764860698, totalSize=1661, EXTERNAL=TRUE, numFiles=4},
viewOriginalText:null, viewExpandedText:null, tableType:EXTERNAL_TABLE, rewriteEnabled:false)
I highlighted parts of the text with answers to the above questions.
```

- Where in HDFS the data for this table is located:

location:hdfs://namenode:8020/user/hue/classicmodels.db/employees
 - The file type for this table: Text Type table both input and output (CSV is also considered as Text).

We can also check SerDe, which is OpenCSVSerde meaning that the physical files are text csv files.
 - If the table Managed or External? EXTERNAL_TABLE
 - How many physical data files belong to this table. numFiles=4
- Exit Beeline and use the HDFS CLI to examine the HDFS directory for this table. Do the following:
- Check how many files there are
 - View the file contents and see if they are readable; explain why

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hue/classicmodels.db/employees
Found 5 items
-rw-r--r-- 3 root supergroup 0 2025-12-03 20:20 /user/hue/classicmodels.db/employees/_SUCCESS
-rw-r--r-- 3 root supergroup 614 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00000_copy_1
-rw-r--r-- 3 root supergroup 361 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00001_copy_1
-rw-r--r-- 3 root supergroup 284 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00002_copy_1
-rw-r--r-- 3 root supergroup 402 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00003_copy_1
root@d0ba18e61f19:/# hdfs dfs -cat /user/hue/classicmodels.db/employees/part-m-00000_copy_1
1002MurphyDianex5800dmurphy@classicmodelcars.com1nullPresident
1056PattersonMaryx4611mpatterson@classicmodelcars.com11002VP Sales
1076FirrelliJeffx9273jfirrelli@classicmodelcars.com11002VP Marketing
1088PattersonWilliamx4871wpatterson@classicmodelcars.com61056Sales Manager (APAC)
1102BondurGerardx5408gbondur@classicmodelcars.com41056Sale Manager (EMEA)
1143BowAnthonyx5428abow@classicmodelcars.com11056Sales Manager (NA)
1165JenningsLesliex3291ljennings@classicmodelcars.com11143Sales Rep
1166ThompsonLesliex4065lthompson@classicmodelcars.com11143Sales Rep
root@d0ba18e61f19:/#
```

There are 5 files, 1 of which is empty _SUCCESS file. So there are 4 data files of which the table is constructed, getting schema from them to be stored in metadata. Contents are readable, we could clarify that with -cat command.

- Go back to Beeline and drop the “new_emp” table

```
0: jdbc:hive2://localhost:10000> drop table new_emp;
No rows affected (0.976 seconds)
0: jdbc:hive2://localhost:10000> show tables;
+-----+
| tab_name |
+-----+
+-----+
```

No rows affected since it is EXTERNAL one.

- Check in the HDFS CLI again if the HDFS directory and files still exist; explain why

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hue/classicmodels.db/employees
Found 5 items
-rw-r--r-- 3 root supergroup 0 2025-12-03 20:20 /user/hue/classicmodels.db/employees/_SUCCESS
-rw-r--r-- 3 root supergroup 614 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00000_copy_1
-rw-r--r-- 3 root supergroup 361 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00001_copy_1
-rw-r--r-- 3 root supergroup 284 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00002_copy_1
-rw-r--r-- 3 root supergroup 402 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00003_copy_1
root@d0ba18e61f19:/#
```

Still there, since it was EXTERNAL TABLE. The contents of source directory are not managed by Hive, so it is not altered in any way.

- Go back to Beeline and create the “new_emp” table again; this time, create it as an EXTERNAL table. Check what error you received and explain why.

I’ve already done the first part of the task using EXTERNAL table. Nothing was written about the type of the table, it was just mentioned to create in “identical to the...” manner. My “identical” way was using EXTERNAL table. That’s why I decided to create INTERNAL table this time to capture the difference.

Dropped the table and realized that all files and the folder itself were lost, and that’s why I would have error.

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hue/classicmodels.db/employees
Found 5 items
-rw-r--r-- 3 root supergroup 0 2025-12-03 20:20 /user/hue/classicmodels.db/employees/_SUCCESS
-rw-r--r-- 3 root supergroup 614 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00000_copy_1
-rw-r--r-- 3 root supergroup 361 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00001_copy_1
-rw-r--r-- 3 root supergroup 284 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00002_copy_1
-rw-r--r-- 3 root supergroup 402 2025-12-03 20:20 /user/hue/classicmodels.db/employees/part-m-00003_copy_1
root@d0ba18e61f19:/# hdfs dfs -ls /user/hue/classicmodels.db/employees
ls: '/user/hue/classicmodels.db/employees': No such file or directory
```

- Create the table as MANAGED (this is the default), and change it manually to EXTERNAL after its creation.

```

160 CREATE TABLE newdb.offices (
161     office_code INT,
162     city STRING,
163     phone STRING,
164     address_line1 STRING,
165     address_line2 STRING,
166     state STRING,
167     country STRING,
168     postal_code STRING,
169     territory STRING
170 )
171 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
172 WITH SERDEPROPERTIES (
173     "separatorChar" = "\u0001"
174 )
175 STORED AS TEXTFILE;
176
177 LOAD DATA INPATH '/user/hue/classicmodels.db/offices'
178 INTO TABLE newdb.offices;
179
180 ALTER TABLE newdb.offices SET TBLPROPERTIES('EXTERNAL'='TRUE');
```

| Detailed Table Information | Table(tableName:offices, dbName:newdb, owner:root, createTime:1764868765, lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:office_code, type:int, comment:null), FieldSchema(name:city, type:string, comment:null), FieldSchema(name:phone, type:string, comment:null), FieldSchema(name:address_line1, type:string, comment:null), FieldSchema(name:address_line2, type:string, comment:null), FieldSchema(name:state, type:string, comment:null), FieldSchema(name:country, type:string, comment:null), FieldSchema(name:postal_code, type:string, comment:null), FieldSchema(name:territory, type:string, comment:null)], location:hdfs://namenode:8020/user/hive/warehouse/newdb.db/offices, inputFormat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1, serDeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.OpenCSVSerde, parameters:{serialization.format=1, separatorChar=}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters:{totalSize=522, last_modified_time=1764869031, numRows=0, rawDataSize=0, EXTERNAL=TRUE, numFiles=4, transient_lastDdlTime=1764869031, last_modified_by=root}, viewOriginalText:null, viewExpandedText:null, tableType:EXTERNAL_TABLE, rewriteEnabled:false)

- Check again:
 - Where in HDFS the data for this table is located? In Hive user folder, inside warehouse folder.
/user/hive/warehouse/newdb.db/offices
 - The file type for this table. Input: TextInputFormat, Output: HiveIgnoreKeyTextOutputFormat, SerDe: serializationLib:org.apache.hadoop.hive.serde2.OpenCSVSerde
 - If the table Managed or External EXTERNAL_TABLE

- Exit Beeline and use the HDFS CLI to examine the HDFS directory for this table; check how many files there are. 4 files are there.

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hive/warehouse/newdb.db/offices
Found 4 items
-rwxrwxr-x  3 root supergroup      144 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00000_copy_1
-rwxrwxr-x  3 root supergroup       67 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00001_copy_1
-rwxrwxr-x  3 root supergroup      150 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00002_copy_1
-rwxrwxr-x  3 root supergroup      161 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00003_copy_1
root@d0ba18e61f19:/#
```

- Go back to Beeline, and drop the “new_emp” table (I’ve created offices table, instead of new_emp)

```
0: jdbc:hive2://localhost:10000> drop table classicmodels.offices;
No rows affected (0.629 seconds)
```

- Check again in the HDFS CLI if the HDFS directory and file still exist; explain why

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hive/warehouse/newdb.db/offices
Found 4 items
-rwxrwxr-x  3 root supergroup      144 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00000_copy_1
-rwxrwxr-x  3 root supergroup       67 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00001_copy_1
-rwxrwxr-x  3 root supergroup      150 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00002_copy_1
-rwxrwxr-x  3 root supergroup      161 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00003_copy_1
root@d0ba18e61f19:/#
```

The files are still there, since it is external table (that is the correct behaviour). Even though the files are still in warehouse (which is Hive home folder for data), appropriate records are removed from metadata, and this table and files are not tracked by Hive anymore.

- Remove the HDFS folder manually using the HDFS command CLI (Careful...). and verify that the directory does not exist

```
root@d0ba18e61f19:/# hdfs dfs -ls /user/hive/warehouse/newdb.db/offices
Found 4 items
-rwxrwxr-x  3 root supergroup      144 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00000_copy_1
-rwxrwxr-x  3 root supergroup       67 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00001_copy_1
-rwxrwxr-x  3 root supergroup      150 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00002_copy_1
-rwxrwxr-x  3 root supergroup      161 2025-12-03 20:20 /user/hive/warehouse/newdb.db/offices/part-m-00003_copy_1
root@d0ba18e61f19:/# hdfs dfs -rm -R /user/hive/warehouse/newdb.db/offices
25/12/04 17:38:47 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /user/hive/warehouse/newdb.db/offices
root@d0ba18e61f19:/# hdfs dfs -ls /user/hive/warehouse/newdb.db/offices
ls: '/user/hive/warehouse/newdb.db/offices': No such file or directory
root@d0ba18e61f19:/#
```

PARTITIONED TABLES (35 POINTS)

You can complete this task through Hue or Beeline. Use the command prompt to perform HDFS tasks if the HDFS browser is not available in your Hue environment.

- Query the customers using a simple “SELECT *” to view the sample data

```
188 select * from classicmodels.customers;
```

	customers.customer_id	customers.business_name	customers.last_name	customers.first_name	customers.phone_number
1	103	Atelier graphique	Schmitt	Carine	40.32.25
2	112	Signal Gift Stores	King	Jean	702.5551
3	114	Australian Collectors, Co.	Ferguson	Peter	03 9520
4	119	La Rochelle Gifts	Labruno	Janine	40.67.85
5	121	Baane Mini Imports	Bergulfsen	Jonas	07-98 95
6	124	Mini Gifts Distributors Ltd.	Nelson	Susan	415.5551
7	125	Havel & Zbyszek Co	Piestrzeniewicz	Zbyszek	(26) 642
8	128	Blaauw Goud & Zilver, Inc.	Kaizer	Brian	440.60.66

- Run another query, this time to see the number of customers in each country (GROUP BY..)

```
188
189 select country, count(*) as customers_n from classicmodels.customers
190 group by country
191 ;
```

WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using the alternatives: tez, mapred, mrpark, tez) or using Hive 1.X releases.

Query History Saved Queries **Results (28)**

	country	customers_n
1	Australia	5
2	Austria	2
3	Belgium	2
4	Canada	3
5	Denmark	2
6	Finland	3
7	France	12
8	Germany	13

- Get the DDL of the customers' table

```
194
195 show create table classicmodels.customers;
196
```

Query History Saved Queries **Results (28)**

	createtab_stmt
1	CREATE EXTERNAL TABLE `classicmodels.customers` (
2	`customer_id` string COMMENT 'from deserializer',
3	`business_name` string COMMENT 'from deserializer',
4	`last_name` string COMMENT 'from deserializer',
5	`first_name` string COMMENT 'from deserializer',
6	`phone_number` string COMMENT 'from deserializer',
7	`address_line1` string COMMENT 'from deserializer',

- Copy only the main CREATE TABLE section to a text editor, without all the properties:
 - *CREATE TABLE customers*
 - *(customernumber int,*
 - *..*
 - *creditlimit double);*
- Modify the CREATE TABLE command, to create a new table with the following characteristics:
 - DB Name: "ClassicModels"
 - Table name: "cust_country"

- Partitioned By: “country” column
- File type: AVRO

```
198
199 CREATE EXTERNAL TABLE `classicmodels.cust_country` (
200   `customer_id` string COMMENT 'from deserializer',
201   `business_name` string COMMENT 'from deserializer',
202   `last_name` string COMMENT 'from deserializer',
203   `first_name` string COMMENT 'from deserializer',
204   `phone_number` string COMMENT 'from deserializer',
205   `address_line1` string COMMENT 'from deserializer',
206   `address_line2` string COMMENT 'from deserializer',
207   `city` string COMMENT 'from deserializer',
208   `state` string COMMENT 'from deserializer',
209   `postal_code` string COMMENT 'from deserializer',
210   `country_code` string COMMENT 'from deserializer',
211   `credit_limit` string COMMENT 'from deserializer'
212 )
213 PARTITIONED BY (country string)
214 STORED AS AVRO;
```

- Verify the table was created properly and view its properties

classicmodels (Hive) Rowser

Hive ▾

Databases > classicmodels > cust_country

Overview Partitions (0) Sample (0) Details

PROPERTIES Partitioned Table External and stored in location Created by root on 12/07/2025 3:23 PM +05:00	STATS Files 0 Rows 0 Total size 0 B Data last updated on 12/07/2025 3:23 PM +05:00
---	---

SCHEMA

Filter...

Column (13)	Type	Description
country	string	Add a description...
customer_id	string	from deserializer
business_name	string	from deserializer
last_name	string	from deserializer
first_name	string	from deserializer

- Insert data into the “Cust_Country” from the “Customers” table (Limit to 50 rows) so that partitions will be generated and populated dynamically.

```
216 INSERT INTO TABLE classicmodels.cust_country
217 PARTITION (country)
218 SELECT
219   customer_id,
220   business_name,
221   last_name,
222   first_name,
223   phone_number,
224   address_line1,
225   address_line2,
226   city,
227   state,
228   postal_code,
229   country_code,
230   credit_limit,
231   country
232 FROM classicmodels.customers
233 LIMIT 50;
234
235
```

✓ Success.

- Run a query from “Cust_Country” to view all customers from “USA”

```
236 select * from classicmodels.cust_country
237 where country = 'USA'
238 ;
```

	cust_country.state	cust_country.postal_code	cust_country.country_code	cust_country.credit_limit	cust_country.country
1	CA	91217	1166	105000.00	USA
2	CA	92561	1166	11000.00	USA
3	CA	90003	1166	90700.00	USA
4	MA	58339	1188	68700.00	USA
5	MA	58339	1216	23000.00	USA
6	NY	10022	1286	76400.00	USA
7	CT	97562	1323	84300.00	USA

- Examine the execution plan of the query to verify partition elimination has occurred. Answer the following questions:

```
236 explain dependency select * from classicmodels.cust_country
237 where country = 'USA'
238 ;
```

	me:"classicmodels@cust_country",tabletype:"EXTERNAL_TABLE",input_partitions:[{"partitionName":"classicmodels@cust_country@country=USA"}]
1	

- Which EXPLAIN command was required to view partition related details? **EXPLAIN DEPENDENCY**
- Why is partitioning so important for query performance? **To prune partition paths avoiding unnecessary operations. It dramatically improves performance of queries decreasing the number of unnecessary data reads and transformations by avoiding visiting unnecessary partition folders.**
- Go over to HDFS and see the directory structure created for the partitioned table. Answer the following questions:

- What are the contents of the main directory for this table? **Folders for all unique countries in the table.**
- What are the names of the subdirectories? **country=CountryName**

File Browser

Search for file name Actions Copy Path Open in Importer

Home /user/hive/warehouse/classicmodels.db/cust_country

	Name	Size
	.	
	country=Australia	
	country=Canada	
	country=Denmark	
	country=Finland	
	country=France	
	country=Germany	
	country=Hong Kong	

HIVE ACID TABLES (20 POINTS)

This task can be completed through Hue or Beeline.

Note:

- The practice environment requires setting the following to support transactions:


```
SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
SET hive.support.concurrency=true;
SET hive.enforce.bucketing=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```
- Please run these SET commands in the Hue/Beeline window prior to performing this exercise.

- Create a new transactional table called “my_emp” with the following properties:

- Columns:

- ID – INT
- Name – STRING
- Salary – INT

- File type: ORC

```
247 CREATE TABLE classicmodels.my_emp (
248   id      INT,
249   name    STRING,
250   salary  INT
251 )
252 CLUSTERED BY (id) INTO 5 BUCKETS
253 STORED AS ORC
254 TBLPROPERTIES (
255   'transactional' = 'true'
256 );
257
```

✓ Success.

- Transactional...

- Check if this table supports DML operations and which DESCRIBE operation is required

```
256 );
257
258 DESCRIBE EXTENDED classicmodels.my_emp;
```

Query History

Saved Queries

Results (5)



1

2

3

4

5

ansient_lastDdlTime=1765106106, transactional=true), viewOrig

Also, it is ORC file storage and bucketed, which

are mandatory properties of transactional (ACID) tables in Hive. Below is the full output text of the field.

```
Table(tableName:my_emp, dbName:classicmodels, owner:root, createTime:1765106106, lastAccessTime:0,
retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:id, type:int, comment:null), FieldSchema(name:name,
type:string, comment:null), FieldSchema(name:salary, type:int, comment:null)],
location:hdfs://namenode:8020/user/hive/warehouse/classicmodels.db/my_emp,
inputFormat:org.apache.hadoop.hive ql.io.orc.OrcInputFormat,
outputFormat:org.apache.hadoop.hive ql.io.orc.OrcOutputFormat, compressed:false, numBuckets:5,
serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive ql.io.orc.OrcSerde,
parameters:{serialization.format=1}), bucketCols:[id], sortCols:[], parameters:{},
skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}),
storedAsSubDirectories:false), partitionKeys:[], parameters:{totalSize=0, numRows=0, rawDataSize=0,
COLUMN_STATS_ACCURATE={"BASIC_STATS"."true"}, numFiles=0, transient_lastDdlTime=1765106106,
transactional=true}, viewOriginalText:null, viewExpandedText:null, tableType:MANAGED_TABLE,
rewriteEnabled:false)
```

- Insert 3 rows to this table in a single INSERT command:

- 1, John, 10000
- 2, Sara, 12000
- 3, Adam, 8000

```
259
260 INSERT INTO my_emp VALUES
261 (1, 'John', 10000),
262 (2, 'Sara', 12000),
263 (3, 'Adam', 8000)
264 ;
```

✓ Success.

- Query the table to verify all rows were inserted

```
266
267 select * from classicmodels.my_emp;
```

Query History		Saved Queries		Results (3)	
				my_emp.id	my_emp.name
				my_emp.salary	
1	1			John	10000
2	2			Sara	12000
3	3			Adam	8000

- Update Adam's salary in "my_emp" to 9000

```
268 UPDATE my_emp
269 SET salary = 9000
270 WHERE name = 'Adam';
271
```

✓ Success.

- Insert a new row to “my_emp”—4, Alex, 13000

```
270 WHERE name = 'Adam';
271
272 INSERT INTO my_emp VALUES (4, 'Alex', 13000);
```

✓ Success.

- Delete John from the table.

```
273
274 DELETE FROM my_emp
275 WHERE name = 'John';
```

✓ Success.

- Query “my_emp” to verify you see all changes performed.

```
274
275 select * from classicmodels.my_emp;
276
```

Query History	Saved Queries	Results (3)
my_emp.id		my_emp.name
		my_emp.salary
1	2	Sara
2	3	Adam
3	4	Alex

SQL QUERIES FILE USED FOR ALL TASKS

```
-- Creating and loading tables
CREATE EXTERNAL TABLE classicmodels.customers (
  customer_id INT,
  business_name STRING,
  last_name STRING,
  first_name STRING,
  phone_number STRING,
  address_line1 STRING,
  address_line2 STRING,
  city STRING,
  state STRING,
  postal_code STRING,
  country STRING,
  country_code INT,
```

```
    credit_limit DECIMAL(15,2)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
    "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/customers';

CREATE EXTERNAL TABLE classicmodels.employees (
    employee_id INT,
    last_name STRING,
    first_name STRING,
    extension STRING,
    email STRING,
    office_code INT,
    reports_to INT,
    job_title STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
    "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/employees';

CREATE EXTERNAL TABLE classicmodels.offices (
    office_code INT,
    city STRING,
    phone STRING,
    address_line1 STRING,
    address_line2 STRING,
    state STRING,
    country STRING,
    postal_code STRING,
    territory STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
    "separatorChar" = "\u0001"
)
```

```
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/offices';

CREATE EXTERNAL TABLE classicmodels.orderdetails (
  order_number INT,
  product_code STRING,
  quantity_ordered INT,
  price_each DECIMAL(10,2),
  order_line_number INT
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/orderdetails';

CREATE EXTERNAL TABLE classicmodels.payments (
  customer_number INT,
  check_number STRING,
  payment_date DATE,
  amount DECIMAL(10,2)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/payments';

CREATE EXTERNAL TABLE classicmodels.products (
  product_code STRING,
  product_name STRING,
  product_line STRING,
  product_scale STRING,
  product_vendor STRING,
  product_description STRING,
  quantity_in_stock INT,
  buy_price DECIMAL(10,2),
  msrp DECIMAL(10,2)
)
```

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/products';
```

```
-----

select * from classicmodels.employees
limit 10
;
```

```
-----

select employee_id, first_name, last_name
from classicmodels.employees
where employee_id between 1002 and 1100
order by last_name desc
limit 5
;
```

```
-----

SELECT
  job_title,
  COUNT(*) AS employees_count
FROM classicmodels.employees
GROUP BY job_title
ORDER BY employees_count DESC
;
```

```
-----

SHOW CREATE TABLE classicmodels.employees;
```

```
-----

DESCRIBE EXTENDED classicmodels.employees;
-----
```

```
select * from newdb.new_emp;

-----

CREATE TABLE newdb.new_emp (
  employee_id INT,
  last_name STRING,
  first_name STRING,
  extension STRING,
  email STRING,
  office_code INT,
  reports_to INT,
  job_title STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = "\u0001"
)
STORED AS TEXTFILE
LOCATION '/user/hue/classicmodels.db/employees';

CREATE TABLE newdb.offices (
  office_code INT,
  city STRING,
  phone STRING,
  address_line1 STRING,
  address_line2 STRING,
  state STRING,
  country STRING,
  postal_code STRING,
  territory STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = "\u0001"
)
STORED AS TEXTFILE;

LOAD DATA INPATH '/user/hue/classicmodels.db/offices'
INTO TABLE newdb.offices;
```

```
ALTER TABLE newdb.offices SET TBLPROPERTIES('EXTERNAL'='TRUE');
```

```
DROP TABLE newdb.offices;
```

```
DESCRIBE EXTENDED newdb.offices;
```

```
-----  
  
select country, count(*) as customers_n from classicmodels.customers  
group by country  
;  
  
-----
```

```
set hive.exec.dynamic.partition.mode=nonstrict;
```

```
show create table classicmodels.customers;
```

```
CREATE EXTERNAL TABLE `classicmodels.cust_country`(  
  `customer_id` string COMMENT 'from deserializer',  
  `business_name` string COMMENT 'from deserializer',  
  `last_name` string COMMENT 'from deserializer',  
  `first_name` string COMMENT 'from deserializer',  
  `phone_number` string COMMENT 'from deserializer',  
  `address_line1` string COMMENT 'from deserializer',  
  `address_line2` string COMMENT 'from deserializer',  
  `city` string COMMENT 'from deserializer',  
  `state` string COMMENT 'from deserializer',  
  `postal_code` string COMMENT 'from deserializer',  
  `country_code` string COMMENT 'from deserializer',  
  `credit_limit` string COMMENT 'from deserializer'  
)  
PARTITIONED BY (country string)  
STORED AS AVRO;  
  
INSERT INTO TABLE classicmodels.cust_country  
PARTITION (country)  
SELECT  
  customer_id,
```

```
business_name,
last_name,
first_name,
phone_number,
address_line1,
address_line2,
city,
state,
postal_code,
country_code,
credit_limit,
country
FROM classicmodels.customers
LIMIT 50
;

explain dependency select * from classicmodels.cust_country
where country = 'USA'
;

-----

SET hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
SET hive.support.concurrency=true;
SET hive.enforce.bucketing=true;
SET hive.exec.dynamic.partition.mode=nonstrict;

CREATE TABLE classicmodels.my_emp (
  id INT,
  name STRING,
  salary INT
)
CLUSTERED BY (id) INTO 5 BUCKETS
STORED AS ORC
TBLPROPERTIES (
  'transactional' = 'true'
);

DESCRIBE EXTENDED classicmodels.my_emp;

INSERT INTO my_emp VALUES
```

```
(1, 'John', 10000),
(2, 'Sara', 12000),
(3, 'Adam', 8000)
;

UPDATE my_emp
SET salary = 9000
WHERE name = 'Adam';

INSERT INTO my_emp VALUES (4, 'Alex', 13000);

DELETE FROM my_emp
WHERE name = 'John';

select * from classicmodels.my_emp;

DESCRIBE EXTENDED classicmodels.offices;
```
