**Hive Practice**

CONTENTS

[1. Basic Hive Interaction Using Hue (15 points) 3](#_Toc126593965)

[2. Basic Hive Interaction Using Beeline (10 points) 4](#_Toc126593966)

[3. Managed and External Tables Using Beeline (20 points) 5](#_Toc126593967)

[4. Partitioned Tables (35 points) 6](#_Toc126593968)

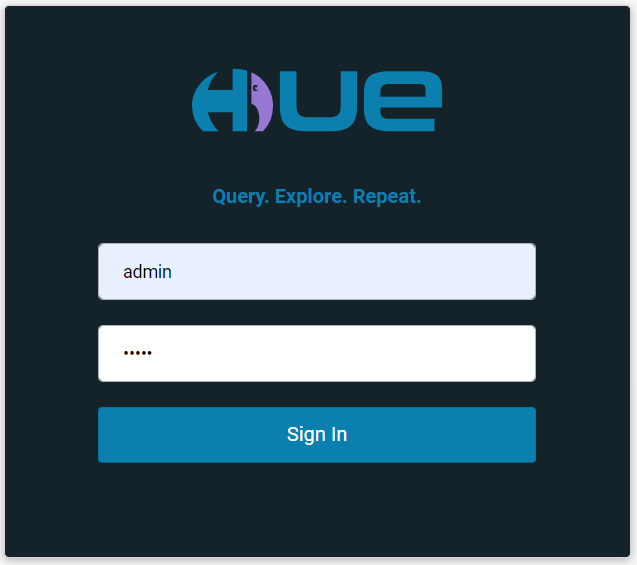
[5. Hive ACID Tables (20 points) 8](#_Toc126593969)

Before I want to say that I’ve changed the image of hive-metastore-postgresql from “peerlesscnn/hive-metastore-postgresql:2.3.0” to “bde2020/hive-metastore-postgresql:2.3.0” in the docker-compose.yml. And after that I created hue user for postgres manually from sql file. Because It doesn’t work without it. As a consequence it works perfectly.

***Note: The requirements for the practical task are highlighted in bold, the other simple text is my answers.***

**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**

A screenshot of a computer

Description automatically generated

After installing all these things I noticed that there is no tables in default scheme.

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

I found the classicmodels.db in the <https://www.mysqltutorial.org/mysql-sample-database.aspx>. Because I needed the column names. And this is really hard to do without it.

Also, I prepared python code for parsing and replacing separator symbol to tab, that means I created CSV files(Actually it is TSV(Tabular separated values)).

Code:

import zipfile

import os

customers = 'customerNumber\tcustomerName\tcontactLastName\tcontactFirstName\tphone\taddressLine1\taddressLine2\tcity\tstate\tpostalCode\tcountry\tsalesRepEmployeeNumber\tcreditLimit'

employees = 'employeeNumber\tlastName\tfirstName\textension\temail\tofficeCode\treportsTo\tjobTitle'

offices = 'officeCode\tcity\tphone\taddressLine1\taddressLine2\tstate\tcountry\tpostalCode\tterritory'

orderdetails = 'orderNumber\tproductCode\tquantityOrdered\tpriceEach\torderLineNumber'

payments = 'customerNumber\tcheckNumber\tpaymentDate\tamount'

products = 'productCode\tproductName\tproductLine\tproductScale\tproductVendor\tproductDescription\tquantityInStock\tbuyPrice\tMSRP'

table\_headers = (customers, employees, offices, orderdetails, payments, products)

zip\_path = 'classicmodels.db\_\_1\_.zip'

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

zip\_ref.extractall(f'dataset/')

path = 'dataset/classicmodels.db'

os\_walk = list(os.walk(path))

tables = list(zip(table\_headers, os\_walk[0][1], [i[2][:-1] for i in os\_walk[1:]]))

target\_path = 'classicmodels.db'

if not os.path.exists(target\_path):

os.mkdir(target\_path)

for table in tables:

parts = f'{table[0]}\n'

for table\_part in table[2]:

file\_path = os.path.join(path, table[1], table\_part)

with open(file\_path, 'r') as f:

file\_data = f.readlines()

parts += ''.join(file\_data)

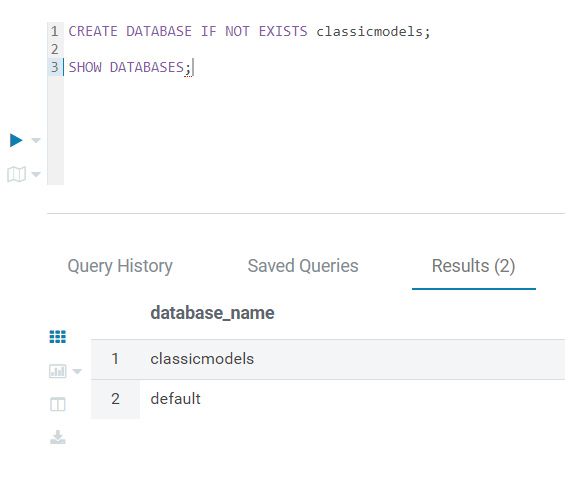
with open(os.path.join(target\_path, f'{table[1]}.csv'), 'w') as f:

f.write(parts.replace('\x01', '\t'))

Imported files:

A screenshot of a computer

Description automatically generated



* + **Switch to it**
  + **Show all the available tables in the database**
  + **Expand the “Customers” table and view its columns and data types**
* **Perform the following queries:**
  + **Query all rows from the “Employees” table**
  + **Alter the previous query to fetch only the first 10 rows**
  + **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**
  + **Write a query to fetch the number of employees per job title, ordered by number of employees in descending order**
  + **Export the query output to a text file**
* **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**
    - **Which Hive table property is responsible for this**

**Note: Use the CLI to browse HDFS and view file contents in case Hue HDFS browser is not working in your environment (If ports 50070 and 50075 were not mapped, this will be causing an error message)**

**Basic Hive Interaction Using Beeline (10 points)**

* **Open a BASH session to the practice environment and connect to Hive using Beeline**
* **Show available databases (Verify you see the “ClassisModels” database)**
* **Switch to use the “ClassisModels” database**
* **Show all tables in this database**
* **Create a new database called “newdb” and verify the database was created**
* **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**

**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**
* **Check using the table properties (without browsing HDFS):**
  + **Where in HDFS the data for this table is located**
  + **The file type for this table**
  + **If the table Managed or External?**
  + **How many physical data files belong to this table**

**Hints:**

* + - **Use the practice guide to see how you can view such details for Hive tables**
    - **Use a more verbose command to view all properties**
* **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**
* **Go back to Beeline and drop the “new\_emp” table**
* **Check in the HDFS CLI again if the HDFS directory and files still exist; explain why**
* **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.**
* **Create the table as MANAGED (this is the default), and change it manually to EXTERNAL after its creation.**
* **Check again:**
  + **Where in HDFS the data for this table is located**
  + **The file type for this table**
  + **If the table Managed or External**
* **Exit Beeline and use the HDFS CLI to examine the HDFS directory for this table; check how many files there are**
* **Go back to Beeline, and drop the “new\_emp” table**
* **Check again in the HDFS CLI if the HDFS directory and file still exist; explain why**
* **Remove the HDFS folder manually using the HDFS command CLI (Careful…). and verify that the directory does not exist**

**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**
* **Run another query, this time to see the number of customers in each country  
  (GROUP BY..)**
* **Get the DDL of the customers' table**

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

* **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**
* **Verify the table was created properly and view its properties**
* **Insert data into the “Cust\_Country” from the “Customers” table (Limit to 50 rows) so that partitions will be generated and populated dynamically.**

**Note: If you get an error related to “MoveTask” you can ignore it.   
This error is related to the practice environment. Using the following practices, you can verify that rows and partitions were actually created in the target table. This is what is required for this practice.**

* **Run a query from “Cust\_Country” to view all customers from “USA”**
* **Examine the execution plan of the query to verify partition elimination has occurred. Answer the following questions:**
  + **Which EXPLAIN command was required to view partition related details?**
  + **Why is partitioning so important for query performance?**
* **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?**
  + **What are the names of the subdirectories?**

**Hive ACID Tables (20 points)**

**This task can be completed through Hue or Beeline.**

**Note:**

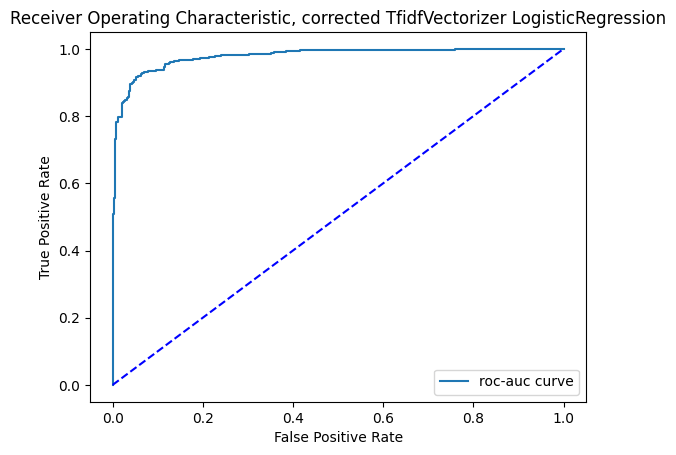
* **The practice environment requires setting the following to support transactions:**

***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**
  + **Transactional…**
* **Check if this table supports DML operations and which DESCRIBE operation is required**
* **Insert 3 rows to this table in a single INSERT command:**
  + **1, John, 10000**
  + **2, Sara, 12000**
  + **3, Adam, 8000**
* **Query the table to verify all rows were inserted**
* **Update Adam’s salary in “my\_emp” to 9000**
* **Insert a new row to “my\_emp”—4, Alex, 13000**
* **Delete John from the table.**
* **Query “my\_emp” to ver**  **ify you see all changes performed.**