Logo, company name

Description automatically generated

**Cloud Computing And Big Data Project Report**

**Prof. Anwar Hossain**

**Group 16**

**Reported By:**

Ahmed Elsayed Salama (21aes20)

Fatma Eldesouky (21feae)

Mahmoud Adel Khorshed (21mamm2)

Zeyad Tarek Mohamed (21ztem)

**July 2022**

**Introduction And Objective**

The Apache Hive ™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage.

Apache Spark is a lightning-fast cluster computing designed for fast computation. It was built on top of Hadoop MapReduce, and it extends the MapReduce model to efficiently use more types of computations which includes Interactive Queries and Stream Processing. This is a brief tutorial that explains the basics of Spark Core programming.

Our objective here we want to create a database have three tables the **(employee-data-hive, department-data-hive, employee\_ptn)** and we have the csv data **(Chicago employee)**, so we want to do some data analysis on these tables to have get useful insights from their data using hive commands or spark commands.

**Before starting to do our project, we need to setup and do some important steps to make doing our tasks possible.**

1. First, we create a database as the namespace of our tables that we want to create and use it.

Text

Description automatically generated with low confidence

1. We moved the **employee-data.csv** file from **local machine** to local folder in VM OS through WinSCP software.

Table

Description automatically generated with low confidence

1. Then We copied this csv file into a hdfs folder because we will need later in the last task.



1. We changed some settings in hive to enable us using **DML** operations and we target to use **UPDATE** clause in **task 4**.

Add these properties with these values in tag format in etc/gedit/hive-site.xml file or set them directly in hive shell as commands.

1. **hive.support.concurrency – true**
2. **hive.enforce.buckting – true**
3. **hive.exec.dynamic.partition.mode – nonstrict**
4. **hive.txn.manager – org.apache.hadoop.hive.ql.lockmgr.DbTxnManager**
5. **hive.compactor.initiator.on – true**
6. **hive.compactor.worker.threads – 1**

**Note:** to preform hive **CRUD** using **ACID** operations, your **hive** version must be **0.14** or above the table format must be **ORC** file format with **TBLPROPERTEIS(‘transactional’ = ‘true’)** , also the table on which you want to perform the update operation must be **CLUSTERD BY** with some **buckets**.

**Task 1**

**Goal: Create a Hive table named employee-data-hive based on the given dataset.**

**Solution**

1. we used (drop table) command to drop any other tables that had the same name to prevent overwriting on it.

Text

Description automatically generated

1. Now we created (**EmployeeDataHive**) table to store employee’s data.
2. The (**employee-data.CSV**) file is separated by comma but there is a **Name** column separate between first and last name by comma, also we found in row **24401** in **JobTitles** column there is value separated by comma too, so it will be conflicted when loading data into a table and separate it by comma.
3. To solve this issue, we used **OpenCSVSedre** library.
4. The **OpenCSVSerde** library has the following characteristics for string data: Uses double quotes (“) as the default quote character, and allows you to specify separator, quote, and escape characters
5. Then we used **tblproperties** library to skip header in first row.

Text

Description automatically generated

1. Finally, we did an information retrieval query after we loaded our csv data into the table (**EmployeeDataHive**), the number of records is **32928 (header doesn’t included)**.

A screenshot of a computer

Description automatically generated with medium confidence

**The Final script for task 1**

Graphical user interface, text, application, email

Description automatically generated

**Task 2**

**Goal: Create a department-data-hive table by selecting unique department names from the employee-data-hive and adding a column named DeptID in the new department-data-hive table and put unique values in the DeptID column.**

**Solution**

1. We first get unique department name from EmployeeDataHive table and show the unique department names.



Note: the output here is only the unique departments names without and key or referred number column.

1. Then we took the result of the SELECT query manually and put it into an **EXCEL** sheet, then we generate an auto numeric incremental column started from 1 to the last department to refer to each department by a unique number this column name “**deptID**” by that the format of that sheet became (**depart\_name**, **DeptID**), we saved that file as **Department.csv** and export that file into a csv file and move it to **VMware OS** to create a new table from that csv file.

Table

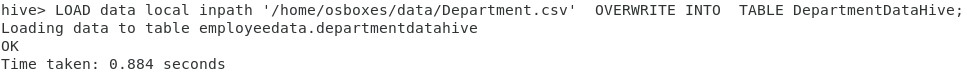
Description automatically generated

1. We created new table (**DepartmentDataHive**) from the new imported csv file to store the new data of Department into it.

Graphical user interface, text, application

Description automatically generated

1. Load the Department.csv data file from local path to the table(**DepartmentDataHive**)



1. Then we showed our data after loaded into table (**DepartmentDataHive**), the number of departments is **36**.

Table

Description automatically generated

**The Final script for task 2**

Graphical user interface, text, application

Description automatically generated

**Task 3**

**Goal 3.1 Update the employee-data-hive table by replacing the department field data with the deptID values as created in the department-data-hive table.**

In this task, we need to create relation as represented in the below graph between **EmployeeDataHive** table and **DepartmentDataHive** to know the **ID** of each department name in **EmployeeDataHive** and to get that relationship with will use **JOIN CLAUSE**.

Diagram

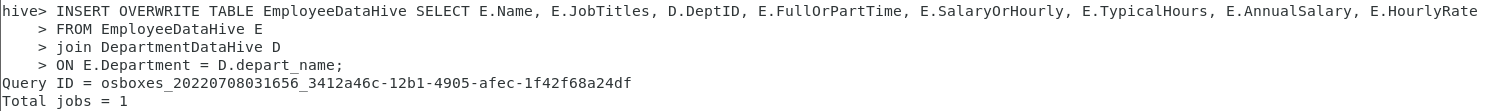
Description automatically generated

In this task we have 2 solutions:-

1. Using **INSERT OVERWRITE with JOIN clause**.
2. Using **CTAS** with  **JOIN** clause.

**First Solution**

1. Overwrite the data with **INSERT OVERWRTIE** clause.
2. Select the data that we want to replace it the old columns and it’s same as its, but we’ll replace only department column with DeptID column which is existing in **DepartmentDataHive** table.
3. We want to join **EmployeeDataHive** table and **DepartmentDataHive** table to find a relationship between these tables and that’s relationship is based on **department** column in **EmployeeDataHive** table and **depart\_name** column in **DepartmentDataHive** table, by that we will be able to know what’s the **DeptID** for each department name in **EmployeeDataHive** table.



1. Verify that the department name is relaced with their IDs correctly.

A screenshot of a computer

Description automatically generated with low confidence

**The Final script for task 3.1.1**

Text

Description automatically generated with medium confidence

===============================================

**Second Solution**

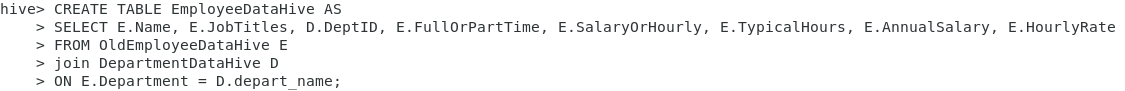
1. We **renamed** the **EmployeeDataHive** table to**OldEmployeeDataHive** to use the **EmployeeDataHive** tableto store the new update.

Graphical user interface, text

Description automatically generated

Note: Drop operation here just to not give us an error if **OldEmployeeDataHive** if it’s already existing.

1. Then we used **CREATE AS SELECT** to create the new empty **EmployeeDataHive** table and we added the required columns and replaced the **department** column by joining the two tables (**OldEmployeeDataHive, DepartmentDataHive**) based on Department name, then select specific columns from the new table (remove department name from **EmployeeDataHive table** and replace it **DeptID** instead).



1. Then we showed our data after updating the department name into **DeptID** from the table (**EmployeeDataHive**).

A screenshot of a computer

Description automatically generated with low confidence

1. Drop **OldEmployeeDataHive** because we don’t need it or its data anymore.



**The Final script for task 3.1.2**

**Graphical user interface, text, application, email

Description automatically generated**

**Task 3.2**

**Goal:** Also update the employee-data-hive table 'annual salary' field based on the 'Typical Hours' \* 'Hourly Rate' \* 52 if the annual salary field is empty.

this task we have 2 solutions:-

1. Using **UPDATE clause.**
2. Using **CASE WHEN THEN** clause.

**First Solution**

1. Since ACID operations only is performed on **ORC** tables, so I renamed the **EmployeeDataHive** table to **OldEmployeeDataHive** to create a new empty ORC table.

Graphical user interface, text

Description automatically generated

1. Create an **ORC** table called **EmployeeDataHive CULSTERD BY** **JobtTitles** column and bucketed into **3** buckets (**3 reducers will be used to perform an ACID operations**).

Text

Description automatically generated

1. **INSERT** data into an ORC (**EmployeeDataHive**) table from **OldEmployeeDataHive.**



1. Update **Annualsalary** column if its record is **NULL** by multiplying the value of **TypicalHours** by **HourlyRate** \* 52 and it’s not **NULL** keep it as it is.



1. Verify that **Annualsalary** is updated in correct way.

Text, letter

Description automatically generated

1. Drop **OldEmployeeDataHive** because we don’t need it or its data anymore.



**The Final script for task 3.2.1**

**Graphical user interface, text, application, email

Description automatically generated**

=========================================

**Second Solution**

1. We **renamed** the **EmployeeDataHive** table to **OldEmployeeDataHive** to use the **EmployeeDataHive** tableto store the new update.

Graphical user interface, text

Description automatically generated

Note: Drop operation here just to not give us an error if **OldEmployeeDataHive** if it’s already existing.

1. we used **(CREATE TABLE AS SELECT)** to update **Annualsalary** from table **EmployeeDataHive** based on the **TypicalHours** \* **HourlyRate** \* 52 if the annual salary field is **NULL**.
2. We used **CASE WHEN THEN** clause to check the condition of **Annualsalary NULL** or not If **NULL** it will apply the following calculation, else it will be the same value.

Text

Description automatically generated

1. Then we showed our data after updating the **Annualsalary** values.

A screenshot of a computer

Description automatically generated with low confidence

1. Drop **OldEmployeeDataHive** because we don’t need it or its data anymore.



**The Final script for task 3.2.2**

**Table

Description automatically generated with medium confidence**

**Task 4**

**Task 4.1**

Goal: Display all employees list with salary more than $100,000 based on employee-data-hive table.

**Solution**

1. We selected the name of employees and their annual salaries who have **Annualsalary** greater than 100000 by **SELECT** command and **WHERE** clause to set a condition.



1. This is the sample of result because the data is large

A picture containing text, newspaper, receipt

Description automatically generated

We got **7561** employees who have **Annualsalary** more than **100000,** and we selected **Annualsalary** to ensure that **WHERE** clauseis worked correctly.

**The Final script for task 4.1**

Text

Description automatically generated

**Task 4.2**

**Goal**: join the employee-data-hive and department-data-hive table to show the average salary of employees by department name.

1. To get the **average salary** of employees by name of department, we will join two tables by INNER **JOIN** command then group by department name and we finally select that column (**depart\_name and average salary**)

**Text

Description automatically generated**

1. This sample of result

**Text

Description automatically generated**

We got **36 departments** with average salaries.

**The Final script for task 4.2**

Graphical user interface, text

Description automatically generated with medium confidence

**Task 5**

**Goal:** Create **5** partitions in **employees\_ptn** table to store **5 departments** in the appropriate partition. Display the partition structure.

1. Edit dynamic partitioning in hive.



1. We created table **(employees\_ptn)** to store **5** appropriate partitions and we partitioned by **department** column as string.



1. Now we’ll insert the appropriate data to these partitions from **EmployeeDataHive,** we will have **5 WHERE** clauses each **WHERE** clause will extract the required data from **EmployeeDataHive** to each **partition**.

A picture containing text

Description automatically generated

1. Display every partition structure by using DESCRIBE FORMATTED employees\_ptn PARTITION(department= partition\_number);



These are results samples.

All partitions in the table.

Chart, text

Description automatically generated with medium confidence

Partition department = 1 sturcture.

Text

Description automatically generated

The above query will print another 4 results like above images but for other partitions.

**The Final script for task 5**

**A picture containing table

Description automatically generated**

**Task 6**

**Goal: Create spark DataFrame based on the given dataset. Identify # of records in the DataFrame and show top 10 records.**

**Solution**

1. Open spark-shell to start write commands.



1. Create required **DataFrame** with header argument true to consider the first row of the data as header and load the data from csv file which is located into hdfs.



1. Start counting the number of rows of the csv file.



We got **32928 employees** (header doesn’t included).

1. We displayed the top 10 records.

Table

Description automatically generated

**Workload**

Every member in the team share in the project in equal efforts, managed by **Asana software** to deliver and handling their works.

Graphical user interface, application

Description automatically generated

1 - Ahmed Elsayed Salama

* Create the **ERD** Diagram of the project.
* Modify in **EXCEL** sheet.
* **Reported** and **reviewed** the project.
* Worked on **Task2 (Part2**)**, Task 3, Task4 (Part2)**, **Task6**.

2 - Fatma Eldesouky

* She exerts more effort in the **report**.
* Worked on **Task 5 and Task 4** part 1.
* Was responsible for getting resources for our task along with **Mahmoud**.

3 - Mahmoud Khorshed

* He **shared** in all the project in each part.
* **Worked** on all tasks**.**
* **Was responsible for getting resources for our task along with Fatma.**

4 **-** Zeyad Tarek Mohamed

* He worked on **Task 3** the first solution.
* He worked on **Task 1** by finding smart solution, **Task6**, **Task2.**
* **Reported** and **reviewed** the project**.**

**References**

1. <https://hive.apache.org/>
2. <https://www.tutorialspoint.com/apache_spark/index.htm>
3. <https://app.dbdesigner.net/designer/schema/0-project2_erd>.
4. <https://stackoverflow.com/questions/39032279/hive-update-with-subquery>
5. <https://stackoverflow.com/questions/18432925/handling-null-values-in-hive>
6. <https://riptutorial.com/hive/example/15796/orc#:~:text=The%20Optimized%20Row%20Columnar%20>
7. <https://sparkbyexamples.com/apache-hive/hive-enable-and-use-acid-transactions/>
8. <https://www.revisitclass.com/hadoop/case-statement-in-hive-with-examples/>