Big Data Systems

Assignment, 2019

Big Data Architecture, HDFS and Hive

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[Assignment Overview 2](#_Toc6572718)

[General Instructions 2](#_Toc6572719)

[Useful Software 2](#_Toc6572720)

[Additional Material 2](#_Toc6572721)

[Purpose of the assignment 2](#_Toc6572722)

[Part 1: Theoretical questions 3](#_Toc6572723)

[Part 2: Data Warehousing using Hive 6](#_Toc6572724)

[Introduction 6](#_Toc6572725)

[Questions/Tasks 7](#_Toc6572726)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Assignment Overview

**Areas covered:**

* Big Data Architecture.
* Hadoop Distributed File System (HDFS).
* Apache Hive.

**Submission Deadline: 20/05/2019 00:00**

## General Instructions

The current assignment includes two parts of mandatory questions and tasks of 120 points, in total. The highest score that can be achieved in this assignment is 100 points (i.e., 20 points are considered as bonus points).

Note that although a certain dataset is provided and can be used to answer/complete the question/tasks of this assignment, it is not required to successfully complete the assignment. The questions could be answered even without using the dataset. However, it is recommended to use it (i.e., try to build the code requested over the given dataset, where it is feasible) in order to understand better what is asked, as well as practice over the areas considered in this assignment. The dataset is only used for practicing in Part 2.

*Please provide your answers into a document (e.g., answers could be provided in the corresponding “answer” sections). Note that in case the answers are provided into a different document, please add the corresponding question-numbers in order to be clear which is the question of each answer.*

## Useful Software

Use the Cloudera Quickstart (CDH, for short) to practice on the tasks of this assignment (for installation details see slides in SWInstallation.pdf uploaded in the space of the course). Note that the services (from CDH) that could be used for completing this assignment are limited to the following:

* Apache Hadoop (including MapReduce and HDFS),
* Apache Hive, and
* Hue

## Additional Material

* The given dataset (orders\_dataset.zip) includes the following files:
  + order\_details.csv: Details about orders.
  + orders.csv: Information about orders.
  + rates.zip: historical conversion rates to EUR.

## Purpose of the assignment

The purpose of this assignment is to:

1. Understand the main concepts and approaches discussed during the corresponding lectures (big data architectures, HDFS, storage formats and Hive), and
2. Learn/practice on building and utilizing a star schema using Apache Hive and HDFS.

# Part 1: Theoretical questions

<40 points>

Q1.1 <4pt.>: Which of the following issues may be caused by lot of small files in HDFS:

1. NameNode memory usage increases significantly.
2. Overall network load increases.
3. In MapReduce, the number of map tasks that need to process the same amount of data will be larger.
4. I/O rate will be faster.

Select the correct options (one or more) ***and explain*** your answer, **shortly**.

**Answer:**

Q1.2 <4pt.>: How is uniform data distribution across the servers achieved in HDFS?

1. By splitting files into blocks
2. By replication

Select one of the aforementioned options ***and explain*** your answer, **shortly**.

**Answer:**

Q1.3 <4pt.>: If you have a very important file and you aim to store it into HDFS, what is the best way to minimize the risk from losing it?

**Answer:**

Q1.4 <4pt.>: You were told that two servers in HDFS were down: Datanode and Namenode. Which would be your reaction:

* 1. It’s OK, replication factor is 3.
  2. Restore Datanode first.
  3. Restore Namenode first.

Select one of the aforementioned options ***and explain*** your answer, **shortly**.

**Answer:**

Q1.5 <4pt.>: You are writing a 10GB file into HDFS with a replication of 2 and block size of 64MB. How much **total** disk space will this file use? **E*xplain*** your answer, **shortly**.

**Answer:**

Q1.6 <4pt.>: Your colleague mistakenly dropped the table 'access\_log' created with the following statement, in Hive:

CREATE EXTERNAL TABLE access\_log (

id STRING,

user\_id STRING,

request STRING,

response STRING,

status\_code INT

) LOCATION “/data/access\_logs”

Which would be your reaction? **Explain** your answer**, shortly**.

**Answer:**

Q1.7 <4pt.>: You have a bunch of data in your local filesystem (which is **not** part of the cluster) and you need to load it in Hive. Describe the steps that you are going to follow in order the data to be accessible through Hive queries. *Code is not required, just a short description of the steps.*

**Answer:**

Q1.8 <4pt.>: You have an external table in Hive and want to store this data in more compact and efficient format. Your decision is:

1. Create a table in a new format with the CREATE TABLE statement and fill it from the external table with the INSERT INTO TABLE statement.
2. CREATE VIEW over an external table.
3. Create and fill a new table with the CREATE TABLE ... AS SELECT statement.

Select one of the options ***and explain*** your answer, **shortly**.

**In addition**, provide at least one type of format that could improve the performance on query aggregations. **E*xplain*** your answer, **shortly**.

**Answer:**

Q1.9 <4pt.>: Why does partitioning optimize Hive queries? Please provide a **short** answer.

**Answer:**

Q1.10 <4pt.>: You join two Hive tables: A(key INT, value STRING) and B(key INT, value STRING). Тable B is small enough to be stored in RAM of a single compute node in the cluster and A is much bigger than B (A exceeds the average RAM of a cluster node). Provide an efficient query optimizing the join (INNER JOIN over A.key=B.key) between A and B, in Hive. Assume that the query returns A.value and B.value.

**Answer:**

# Part 2: Data Warehousing using Hive

<80 points>

## Introduction

The purpose of this part is to apply transformations and analysis tasks over the data received through files. We assume that you receive data related to orders of an international company with multiple offices. Based on this data, you initially aim to structure the data into a simple star schema, and then apply further analysis. Please find below details about the files format and data considered as input.

**Main Orders File (file name: orders.csv):**

|  |  |
| --- | --- |
| **Field** | **Description** |
| ORDER\_NUMBER | Order unique identifier |
| ORDER\_DATE | Date of Order |
| SHIPPED\_DATE | Date the order shipped |
| STATUS | Order status |
| COMMENTS | Comments over the order |
| CUSTOMER\_NUMBER | Customer unique identifier |
| CUSTOMER\_NAME | Customer name |
| CUST\_CITY | City of customer |
| CUST\_STATE | State of customer |
| CUST\_COUNTRY | Country of customer |
| CUST\_COUNTRY\_ISO | ISO code of customer country |
| SALES\_CURRENCY | Currency of sales price |
| SALES\_REP\_ID | Sales representative id |
| SALES\_REP\_FIRSTNAME | First name of Sales representative |
| SALES\_REP\_LASTNAME | Surname of Sales representative |
| OFFICE\_CODE | Code of office the sales representative belongs to |
| REPORTING\_PATH | Sales representative reporting path |
| OFFICE\_CITY | City of office |
| OFFICE\_STATE | State of office |
| OFFICE\_TERRITORY | Territory of office |
| OFFICE\_COUNTRY | Office country |

**Order Details (file name: order\_details.csv):**

|  |  |
| --- | --- |
| **Field** | **Description** |
| ORDER\_NUMBER | Order unique identifier |
| PRODUCT\_CODE | Unique identifier of the product |
| PRODUCT\_NAME | Name of the product |
| PRODUCT\_CATEGORY | Category of the product |
| PRODUCT\_VENDOR | Vendor of the product |
| QUANTITY\_IN\_STOCK | Quantity in stock |
| BUY\_PRICE | Purchase price, per item |
| QUANTITY\_ORDERED | Number of items ordered, for the specific product |
| UNIT\_PRICE | Price of the product (per item) |

Note the following:

* The Sales currency included in the orders file applies to all the corresponding sales prices (UNIT\_PRICE) in the order\_details file. We consider that the currency of BUY\_PRICE is always EUR.
* REPORTING\_PATH refers to organizational structure of the sales department. i.e., the SALES\_REP\_ID of the line manager is the first element of the list, his/her line manager is included as second element, etc. The last element of the list is always -1, indicating that there is no line manager of the head of department.

**Currency conversion rates (from any currency to EUR):**

* Files included in zip file rates.zip.
* Each file includes the conversion rates for each date, in JSON format.
* The date included in both the file name and an element with key “date”.
* All the rates included in sub-element named “rates”.

## Questions/Tasks

Q2.1 <7pt.>: Provide the statements in order to create the following folder structure within the HDFS folder “*cloudera*” located in the path */user/cloudera.*

* RawData within the cloudera folder.
* OrdersData in RawData folder.
* Rates in RawData folder.

**Answer:**

Q2.2 <7pt.>: Provide the statements in order to load the following data into the corresponding HDFS folder*.*

|  |  |
| --- | --- |
| **Data type** | **HDFS Location** |
| orders.csv | /user/cloudera/RawData/OrdersData |
| order\_details.csv | /user/cloudera/RawData/OrdersData |
| Exchange rates | /user/cloudera/RawData/Rates |

Consider that the files have already been uploaded into a node having HDFS client set-up properly.

**Answer:**

Q2.3 <8pt.>: Provide the statements in order to make the following data accessible through Hive (i.e., the data could be queried through Hive).

* orders.csv
* order\_details.csv
* Exchange rates

Creation statement of the corresponding tables and “loading” statements are requested. **In addition,** initially, create a new database with name “order\_data” and then create all the tables in this database.

*Hints:*

* *There are multiple approaches/statements to load the data; each of them is accepted as answers.*
* *Use a complex data type for defining the column REPORTING\_PATH.*
* *Identify and define correctly field and collection delimiters.*
* *Do not forget to ignore the headers.*

**Answer:**

Q2.4 <7pt.>: Provide the statements in order to create a partitioned table named “stg\_orders” for storing historical orders data. Select proper partition columns.

**Answer:**

Q2.5 <7pt.>: Consider the following staging tables storing the historical, raw records for order\_details and rates, respectively:

CREATE TABLE IF NOT EXISTS order\_data.stg\_order\_details (

ORDER\_NUMBER BIGINT,

PRODUCT\_CODE STRING,

PRODUCT\_NAME STRING,

PRODUCT\_CATEGORY STRING,

PRODUCT\_VENDOR STRING,

QUANTITY\_IN\_STOCK INT,

BUY\_PRICE DECIMAL(15,2),

QUANTITY\_ORDERED INT,

UNIT\_PRICE DECIMAL(15,2)

)

STORED AS orc;

CREATE TABLE IF NOT EXISTS order\_data.stg\_conv\_rates (

INDATE DATE,

RATES STRING

)

STORED AS orc;

Consider also the external tables, ext\_order\_details and ext\_rates, over the corresponding raw data files, where all the columns are given in STRING data type. Provide the statements loading the data from the external tables to the stg\_ tables by converting the columns into the proper data type.

*Hint:*

* *Consider that the external table ext\_rates has a single column.*
* *Consider that each record of the external table ext\_rates has all the rates of a certain date.*

**Answer:**

Q2.6 <7pt.>: Provide the statements in order to build the surrogate keys of the dimensions dim\_status and dim\_product, where their definitions are given as follows:

CREATE TABLE order\_data.dim\_status (

STATUS\_KEY INT,

STATUS STRING

)

STORED AS orc;

CREATE TABLE order\_data.dim\_product (

PRODUCT\_KEY INT,

PRODUCT\_CODE STRING,

PRODUCT\_NAME STRING,

PRODUCT\_CATEGORY STRING,

PRODUCT\_VENDOR STRING

)

STORED AS orc;

Use the following table in order to store the surrogate keys:

*Hint:*

CREATE TABLE IF NOT EXISTS order\_data.dim\_keys (

dim\_key INT,

dim\_type STRING,

value STRING,

created\_at TIMESTAMP

)

STORED AS orc;

* *Provide the queries (including insertion clauses) over the tables* order\_data.stg\_order\_details and order\_data.stg\_orders defined in previous tasks.

**Answer:**

Q2.7 <7pt.>: Provide the statements in order to insert data into the following dimensions.

Use thetables order\_data.stg\_order\_details and order\_data.stg\_orders defined in previous tasks in order to retrieve the dimension data. Note that the surrogate keys CUST\_KEY and SALES\_REP\_KEY are defined in the following table (column dim\_key) for each distinct value of CUSTOMER\_NUMBER and SALES\_REP\_ID, respectively, in the corresponding columns in order\_data.stg\_order\_details and order\_data.stg\_orders. The surrogate keys are stored in the column “value”. The dim\_type for CUST\_KEY is “customer”, while the dim\_type for SALES\_REP\_ID is “salesrep”.

CREATE TABLE order\_data.dim\_customer (

CUST\_KEY BIGINT,

CUSTOMER\_NUMBER INT,

CUSTOMER\_NAME STRING,

CUST\_CITY STRING,

CUST\_STATE STRING,

CUST\_COUNTRY STRING,

CUST\_COUNTRY\_ISO STRING

)

STORED AS orc;

CREATE TABLE order\_data.dim\_sales\_rep (

SALES\_REP\_KEY INT,

SALES\_REP\_ID INT,

SALES\_REP\_FIRSTNAME STRING,

SALES\_REP\_LASTNAME STRING,

OFFICE\_CODE STRING,

OFFICE\_CITY STRING,

OFFICE\_STATE STRING,

OFFICE\_TERRITORY STRING,

OFFICE\_COUNTRY STRING

)

STORED AS orc;

*Hint:*

CREATE TABLE IF NOT EXISTS order\_data.dim\_keys (

dim\_key int,

dim\_type string,

value string,

created\_at TIMESTAMP

)

STORED AS orc;

* *Notice that* CUSTOMER\_NUMBER and SALES\_REP\_ID are of type INT, while they are stored in column “value” of dim\_keys. Hence, conversion of data types is required.

**Answer:**

Q2.8 <7pt.>: Provide the statements in order to create the following time/date dimension.

*Hint:*

CREATE TABLE IF NOT EXISTS order\_data.dim\_date (

date\_id INT,

date DATE,

day\_of\_week INT,

year INT,

month INT,

week\_of\_year INT

)

STORED AS orc;

* *Use the functions posexplode, split, repeat and datediff.*
* *Find the minimum date included either in ORDER\_DATE or in SHIPPED\_DATE of the table stg\_orders.*
* *Date functions:* [*link*](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-DateFunctions)*.*

**Answer:**

Q2.9 <8pt.>: Provide the statements in order to build the following fact table over the data stored in thetables order\_data.stg\_order\_details and order\_data.stg\_orders. Use also the following dimensions defined in the previous tasks:

* order\_data.dim\_status
* order\_data.dim\_product
* order\_data.dim\_customer
* order\_data.dim\_sales\_rep
* order\_data.dim\_date

Definition of fact table:

CREATE TABLE order\_data.fact\_orders (

ORDER\_DATE\_KEY INT, -- *Surrogate key of time dimension for ORDER\_DATE*

SHIPPED\_DATE\_KEY INT, -- *Surrogate key of time dimension for SHIPPED\_DATE*

CUST\_KEY BIGINT, -- *Surrogate key of customer dimension for CUSTOMER\_NUMBER*

STATUS\_KEY INT, -- *Surrogate key of status dimension for STATUS*

PRODUCT\_KEY INT, -- *Surrogate key of product dimension for PRODUCT\_CODE*

SALES\_REP\_KEY INT, -- *Surrogate key of sales\_rep dimension for SALES\_REP\_ID*

NUMBER\_OF\_ORDERS INT, -- *Number of distinct orders*

TOTAL\_QUANTITY\_ORDERED INT, -- *Total number of QUANTITY\_ORDERED*

SALES\_AMOUNT\_IN\_EUR DECIMAL(15,2), -- *Total amount paid by the customer in EUR*

MAX\_UNIT\_PRICE\_IN\_EUR DECIMAL(15,2), -- *Maximum unit price in EUR*

AVG\_UNIT\_PRICE\_IN\_EUR DECIMAL(15,2), -- *Average unit price in EUR*

MIN\_UNIT\_PRICE\_IN\_EUR DECIMAL(15,2), -- *Minimum unit price in EUR*

TOTAL\_PURCHASE\_AMOUNT\_IN\_EUR DECIMAL(15,2) -- *Total amount paid by the company in EUR*

)

STORED AS orc;

*Hints:*

* *The measures are given for each unique combination of keys.*
* *The calculation of measures SALES\_AMOUNT\_IN\_EUR, MAX\_UNIT\_PRICE\_IN\_EUR, AVG\_UNIT\_PRICE\_IN\_EUR, MIN\_UNIT\_PRICE\_IN\_EUR is based on QUANTITY\_ORDERED, UNIT\_PRICE and the corresponding conversion rates.*
* *The calculation of the measure TOTAL\_PURCHASE\_AMOUNT\_IN\_EUR is based on the QUANTITY\_ORDERED and BUY\_PRICE.*
* *Both ORDER\_DATE\_KEY INT, and SHIPPED\_DATE\_KEY get values from the dimension table order\_data.dim\_date.*

**Answer:**

Q2.10 <8pt.>: Consider the table product\_sales created as follows:

CREATE TABLE IF NOT EXISTS order\_data.products\_sales (

ORDER\_DATE DATE ,

PRODUCT\_CODE STRING,

SALES\_AMOUNT DECIMAL(15,2)

)

STORED AS orc;

insert into order\_data.products\_sales (order\_date,product\_code,sales\_amount)

select to\_date(o.order\_date) as order\_date,

d.product\_code,

sum(d.unit\_price\*d.quantity\_ordered) as SALES\_AMOUNT

from order\_data.stg\_order\_details d, order\_data.stg\_orders o

where d.order\_number=o.order\_number

group by d.product\_code,to\_date(o.order\_date)

Provide the statements answering the following questions:

1. For each product\_code and order\_date, find the SALES\_AMOUNT of the current, the previous and the next day.
2. For each product\_code and order\_date, find the following:
   * the SALES\_AMOUNT, and
   * the difference between the SALES\_AMOUNT of the product and the SALES\_AMOUNT of the product with **maximum** SALES\_AMOUNT in the given order\_date.
3. For each product\_code, find the cumulative sum of SALES\_AMOUNT over the order\_date.

*Hints:*

* *Window functions in Hive:* [*link*](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics)*.*
* *In (b), use windows function partitioning by order\_date and ordering properly.*
* *In (c), use windows function with corresponding ordering.*

**Answer:**

Q2.11 <7pt.>: In the following, it is requested to find the [association rules](https://en.wikipedia.org/wiki/Association_rule_learning) ([link1](http://infolab.stanford.edu/~ullman/mining/pdf/assoc-rules1.pdf), [link2](http://www.mmds.org/mmds/v2.1/ch06-assocrules.pdf)) of the following form:

* X => Y; i.e., if X is included in an order, the probability of having Y ordered, as well, is C%.

where X,Y are PRODUCT\_CODEs, and C is a given threshold.

Consider also the following:

Support: Supp(**X**)=Number of orders containing all the PRODUCT\_CODEs contained in the set **X.**

Confidence: conf(X=>Y)=supp({X} U {Y})/supp({X}).

In particular, provide the statement finding the pairs of products (X,Y), along with their confidence value, for conf(X=>Y) ≤ 80% and supp({X} U {Y}) > 1.

***Optional extension (for practice only):*** *Find the association rules of the form ({X,Y}=>Z), where X, Y, Z are PRODUCT\_CODEs.*

**Answer:**