Visa

Big Data 101

Student Lab 4

Web Age Solutions Inc.
USA: 1-877-517-6540
Canada: 1-866-206-4644
Web: http://www.webagesolutions.com

For customizations of this book or other sales inquiries, please contact us at:
USA: 1-877-517-6540, email: getinfousa@webagesolutions.com
Canada: 1-866-206-4644 toll free, email: getinfo@webagesolutions.com

Contents

Lab 4 –HiveQL Joins and Mini-Project - 1 ½ hours

Overview

HiveQL Specialized Queries and Mini-Project

In this lab we start off with some new ways of working with variables in the Hive Environment. We will then work with the Hive CLI and apply what we have already learned throughout the class to solve various issues with the data and to build upon our reporting skillset.

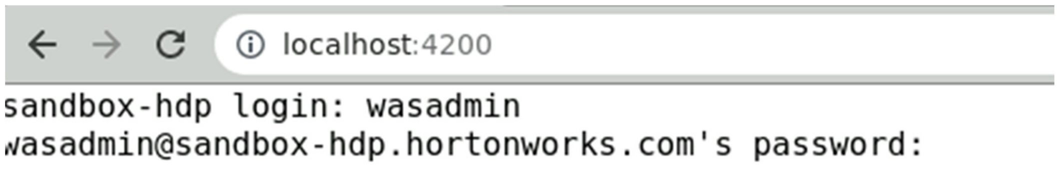
Step 1. Connect to our Lab Environment

Please connect to the assigned Nuvepro Virtual Machine where the labs will be run from.

On the Linux desktop you will see the HDP VM has already been started and is showing a couple of ways to login to the system.

Open a web browser on the Nuvepro Virtual Machine. Enter this address into the URL to open the SSH connection to HortonWorks HDP Sandbox.

`http://localhost:4200`



```
sandbox-hdp login: wasadmin
wasadmin@sandbox-hdp.hortonworks.com's password:
```

Login to the console and provide the password when prompted.

Login username: **`wasadmin`**
Password: **`wasadmin`**

We will now list the directory information using the Linux ll command (lowercase letter L) for long listing. You should see LabFiles folder.

```
ll
```

Use the following command to change to that directory.

```
cd LabFiles
```

Start the Hive CLI

```
hive
```

Set the database to use wasadmin_db

```
use wasadmin_db;
```

Make sure the tables are in the database.

```
show tables;
```

Step 2. Joins

The Hive QL syntax for joining is like SQL with a few added conditional operations like MAPJOIN (bucketmapjoin, sortedmerge) and enforcement of parenthesis in the on condition syntax.
If all but one of the tables being joined are small, the join can be performed as a map only job saving resources and time.
https://cwiki.apache.org/confluence/display/hive/languagemanual+joins

Hive supports the following syntax for joining tables:

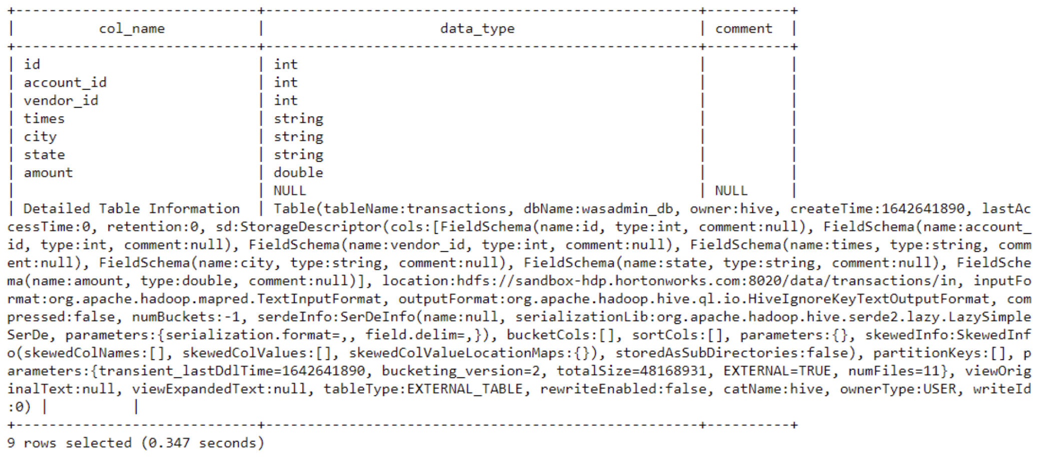
```
join_table:
    table_reference [INNER] JOIN table_factor [join_condition]
```

```
  | table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference join_condition
  | table_reference LEFT SEMI JOIN table_reference join_condition
  | table_reference CROSS JOIN table_reference [join_condition] (as of Hive 0.10)

table_reference:
    table_factor
  | join_table

table_factor:
    tbl_name [alias]
  | table_subquery alias
  | ( table_references )

join_condition:
    ON expression
```

To combine and retrieve the records from multiple tables we use a Hive Join clause taking into consideration how the tables are related. That means knowing the many data tables' columns and the commonality between the tables to join is a must. First let's look at the tables we want to join and find the common columns.

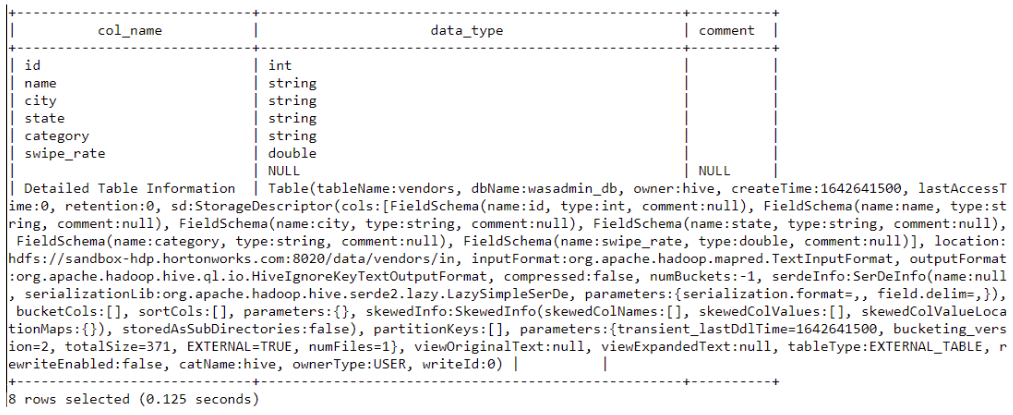
Start by using the describe command. Here we will also look at additional information on both transactions and vendors tables.

describe extended transactions;



```
+----------------------------+----------------------------------------------------+----------+
|          col_name          |                     data_type                      | comment  |
+----------------------------+----------------------------------------------------+----------+
| id                         | int                                                |          |
| account_id                 | int                                                |          |
| vendor_id                  | int                                                |          |
| times                      | string                                             |          |
| city                       | string                                             |          |
| state                      | string                                             |          |
| amount                     | double                                             |          |
|                            | NULL                                               | NULL     |
| Detailed Table Information | Table(tableName:transactions, dbName:wasadmin_db, owner:hive, createTime:1642641890, lastAc
cessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:id, type:int, comment:null), FieldSchema(name:account_
id, type:int, comment:null), FieldSchema(name:vendor_id, type:int, comment:null), FieldSchema(name:times, type:string, comm
ent:null), FieldSchema(name:city, type:string, comment:null), FieldSchema(name:state, type:string, comment:null), FieldSche
ma(name:amount, type:double, comment:null)], location:hdfs://sandbox-hdp.hortonworks.com:8020/data/transactions/in, inputFo
rmat:org.apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, com
pressed:false, numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimple
SerDe, parameters:{serialization.format=,, field.delim=,}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInf
o(skewedColNames:[], skewedColValues:[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[], p
arameters:{transient_lastDdlTime=1642641890, bucketing_version=2, totalSize=48168931, EXTERNAL=TRUE, numFiles=11}, viewOrig
inalText:null, viewExpandedText:null, tableType:EXTERNAL_TABLE, rewriteEnabled:false, catName:hive, ownerType:USER, writeId
:0) |          |
+----------------------------+----------------------------------------------------+----------+
9 rows selected (0.347 seconds)
```

describe extended vendors;



```
+-----------------------------+-----------------------------------------------------+----------+
|          col_name           |                      data_type                      | comment  |
+-----------------------------+-----------------------------------------------------+----------+
| id                          | int                                                 |          |
| name                        | string                                              |          |
| city                        | string                                              |          |
| state                       | string                                              |          |
| category                    | string                                              |          |
| swipe_rate                  | double                                              |          |
|                             | NULL                                                | NULL     |
| Detailed Table Information  | Table(tableName:vendors, dbName:wasadmin_db, owner:hive, createTime:1642641500, lastAccessT
ime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:id, type:int, comment:null), FieldSchema(name:name, type:st
ring, comment:null), FieldSchema(name:city, type:string, comment:null), FieldSchema(name:state, type:string, comment:null),
 FieldSchema(name:category, type:string, comment:null), FieldSchema(name:swipe_rate, type:double, comment:null)], location:
hdfs://sandbox-hdp.hortonworks.com:8020/data/vendors/in, 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.lazy.LazySimpleSerDe, parameters:{serialization.format=,, field.delim=,}),
 bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[], skewedColValueLoca
tionMaps:{}), storedAsSubDirectories:false), partitionKeys:[], parameters:{transient_lastDdlTime=1642641500, bucketing_vers
ion=2, totalSize=371, EXTERNAL=TRUE, numFiles=1}, viewOriginalText:null, viewExpandedText:null, tableType:EXTERNAL_TABLE, r
ewriteEnabled:false, catName:hive, ownerType:USER, writeId:0) |          |
+-----------------------------+-----------------------------------------------------+----------+
8 rows selected (0.125 seconds)
```

In the description of the columns, you can see there is an id column in vendors table and a vendor_id column in the transactions table. While not using the same name for the columns these are the common columns between these tables.

Q: How could we confirm that?
A: Check the data in those fields by writing simple queries.

select * from transactions limit 10;

select * from vendors limit 10;

Step 3. Inner Joins

Time to try our command to retrieve the columns and common data between the two tables. In this case we must set the on condition. Here we join 'transactions' and 'vendors' tables.

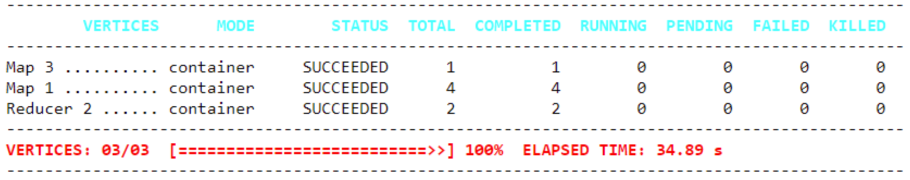
select transactions.*, vendors.* from transactions join vendors on (transactions.vendor_id = vendors.id) limit 10;

Generally, it is better to return a projection of columns when querying data. That means selecting only specific columns to be returned using comma separation between column names.

Find the sum of all amounts by account_id:

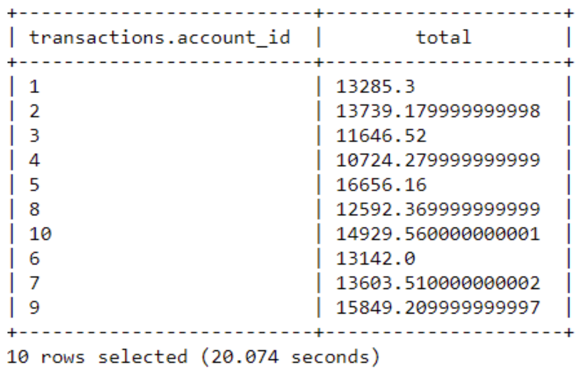
select transactions.account_id, SUM(amount) as total from transactions join vendors on (transactions.vendor_id = vendors.id) group by account_id limit 10;

Did you notice the extra reducer required for the group by aggregation?



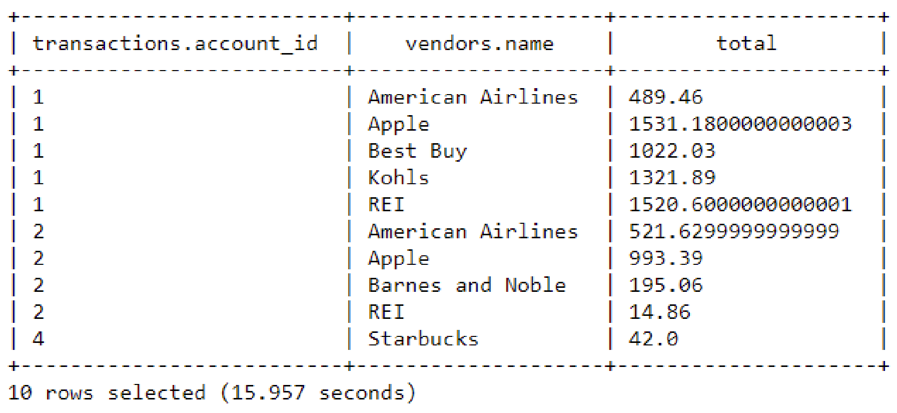
```
----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 3 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 1 .......... container     SUCCEEDED      4          4        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 34.89 s
----------------------------------------------------------------------------------------------
```

Here the columns are easier to read since we have chosen a smaller number to be returned (projected) as the resultset.



```
+--------------------------+----------------------+
| transactions.account_id  |        total         |
+--------------------------+----------------------+
| 1                        | 13285.3              |
| 2                        | 13739.179999999998   |
| 3                        | 11646.52             |
| 4                        | 10724.279999999999   |
| 5                        | 16656.16             |
| 8                        | 12592.369999999999   |
| 10                       | 14929.560000000001   |
| 6                        | 13142.0              |
| 7                        | 13603.510000000002   |
| 9                        | 15849.209999999997   |
+--------------------------+----------------------+
10 rows selected (20.074 seconds)
```

Q: How would you find the sum of all amounts by vendors name??
A: Add the vendors.name column to the projection, and don't forget to group by it too.



```
+--------------------------+--------------------+----------------------+
| transactions.account_id  |    vendors.name    |        total         |
+--------------------------+--------------------+----------------------+
| 1                        | American Airlines  | 489.46               |
| 1                        | Apple              | 1531.1800000000003   |
| 1                        | Best Buy           | 1022.03              |
| 1                        | Kohls              | 1321.89              |
| 1                        | REI                | 1520.6000000000001   |
| 2                        | American Airlines  | 521.6299999999999    |
| 2                        | Apple              | 993.39               |
| 2                        | Barnes and Noble   | 195.06               |
| 2                        | REI                | 14.86                |
| 4                        | Starbucks          | 42.0                 |
+--------------------------+--------------------+----------------------+
10 rows selected (15.957 seconds)
```

select transactions.account_id, vendors.name, SUM(amount) as total from transactions join vendors on (transactions.vendor_id = vendors.id) group by account_id, vendors.name limit 10;

Challenge 1: (Solutions are not shown, confer with others if stuck on a question)

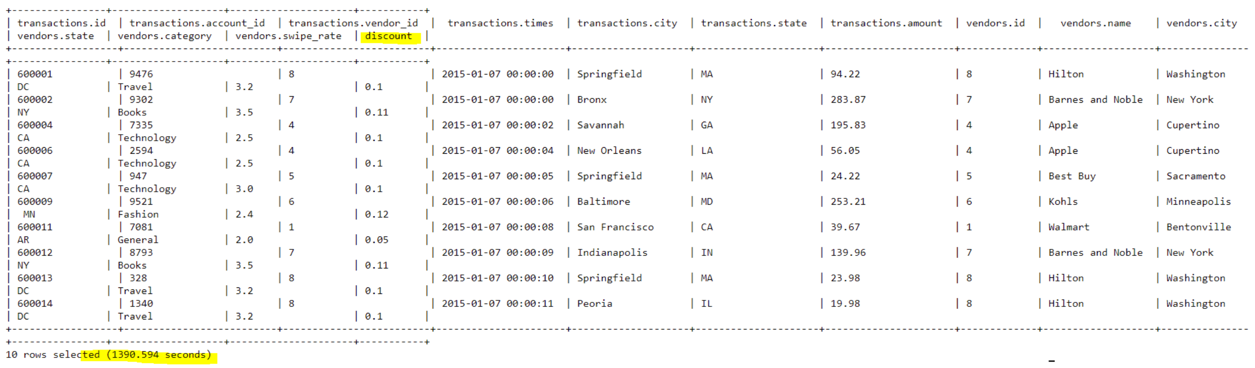
Find the discount offered for the top 5 transactions (highest spent). Hint: Join offers and transactions tables.

Use the offers table discount field. Calculate the average amount spent on each category. Show the top 5 results with the highest discount.

What question does the following query answer?
select vendors.category, SUM(transactions.amount) as total from transactions join vendors on (transactions.vendor_id = vendors.id) group by vendors.category limit 10;

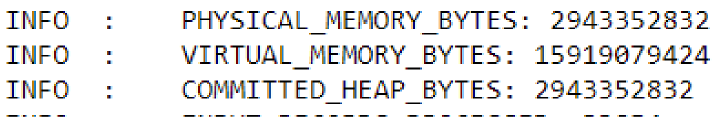
HiveQL can join multiple tables as required if the configuration within the system allocates enough memory. Like SQL engines, Hive will try to load the smallest table in memory then use caching and buffering as necessary to work with the additional tables.

select transactions.*, vendors.*, offers.discount as discount from transactions join vendors on (transactions.vendor_id = vendors.id) join offers on (offers.vendor_id = vendors.id) limit 10



```
+-----------------+---------------------------+------------------------+-----------+
| transactions.id  | transactions.account_id  | transactions.vendor_id  |  transactions.times    | transactions.city    | transactions.state    | transactions.amount    | vendors.id  |    vendors.name     | vendors.city
| vendors.state  | vendors.category  | vendors.swipe_rate  | discount  |
+-----------------+---------------------------+------------------------+-----------+---------------------+----------------------+-----------------------+------------------------+-------------+---------------------+---------------
+-----------------+---------------------------+------------------------+-----------+
| 600001          | 9476                     | 8                       | 2015-01-07 00:00:00   | Springfield          | MA                    | 94.22                  | 8           | Hilton              | Washington
| DC             | Travel            | 3.2                 | 0.1       |
| 600002          | 9302                     | 7                       | 2015-01-07 00:00:00   | Bronx                | NY                    | 283.87                 | 7           | Barnes and Noble    | New York
| NY             | Books             | 3.5                 | 0.11      |
| 600004          | 7335                     | 4                       | 2015-01-07 00:00:02   | Savannah             | GA                    | 195.83                 | 4           | Apple               | Cupertino
| CA             | Technology        | 2.5                 | 0.1       |
| 600006          | 2594                     | 4                       | 2015-01-07 00:00:04   | New Orleans          | LA                    | 56.05                  | 4           | Apple               | Cupertino
| CA             | Technology        | 2.5                 | 0.1       |
| 600007          | 947                      | 5                       | 2015-01-07 00:00:05   | Springfield          | MA                    | 24.22                  | 5           | Best Buy            | Sacramento
| CA             | Technology        | 3.0                 | 0.1       |
| 600009          | 9521                     | 6                       | 2015-01-07 00:00:06   | Baltimore            | MD                    | 253.21                 | 6           | Kohls               | Minneapolis
|  MN            | Fashion           | 2.4                 | 0.12      |
| 600011          | 7081                     | 1                       | 2015-01-07 00:00:08   | San Francisco        | CA                    | 39.67                  | 1           | Walmart             | Bentonville
| AR             | General           | 2.0                 | 0.05      |
| 600012          | 8793                     | 7                       | 2015-01-07 00:00:09   | Indianapolis         | IN                    | 139.96                 | 7           | Barnes and Noble    | New York
| NY             | Books             | 3.5                 | 0.11      |
| 600013          | 328                      | 8                       | 2015-01-07 00:00:10   | Springfield          | MA                    | 23.98                  | 8           | Hilton              | Washington
| DC             | Travel            | 3.2                 | 0.1       |
| 600014          | 1340                     | 8                       | 2015-01-07 00:00:11   | Peoria               | IL                    | 19.98                  | 8           | Hilton              | Washington
| DC             | Travel            | 3.2                 | 0.1       |
+-----------------+---------------------------+------------------------+-----------+---------------------+----------------------+-----------------------+------------------------+-------------+---------------------+---------------
+-----------------+---------------------------+------------------------+-----------+
10 rows selected (1390.594 seconds)
```

Without using partitioning or bucketing, Hive reads all the data in the directory and applies the query filters on it. This is *slow and expensive* for memory since all data must be read in each of the three tables.



```
INFO  :    PHYSICAL_MEMORY_BYTES: 2943352832
INFO  :    VIRTUAL_MEMORY_BYTES: 15919079424
INFO  :    COMMITTED_HEAP_BYTES: 2943352832
```

Step 4. Left Outer Join

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.
https://cwiki.apache.org/confluence/display/Hive/OuterJoinBehavior

The following query demonstrates Left Outer Join between transactions and offers tables, notice the result shows null values.

select transactions.*, offers.* from transactions left outer join offers on (transactions.vendor_id = offers.vendor_id) limit 10;

Challenge 2: (Solutions are not shown, confer with others if stuck on a question)

Picking two tables at a time identify any missing information when joining these tables; Accounts, Offers, Transactions and Vendors.

Right Outer Join

A right outer join behaves the opposite to a left outer join. In this join, all the rows from the right table and the matching rows from the left table are displayed. All the unmatched rows from the table on the left will be dropped and null will be displayed.

Full Outer Join

The major purpose of the HiveQL Full Outer Join is it combines the records of both the left and the right outer tables which fulfills the Hive JOIN condition. Moreover, this joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side.

Step 5. Additional Join Optimization

Hive automatically recognizes various use cases and optimizes for them. Hive 0.11 improves the optimizer for these cases:

- Joins where one side fits in memory. In the new optimization:
    - that side is loaded into memory as a hash table
    - only the larger table needs to be scanned
    - fact tables have a smaller footprint in memory
- Star-schema joins
- Hints are no longer needed for many cases.
- Map joins are automatically picked up by the optimizer.

Review the documentation here:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+JoinOptimization

Star Join Optimization

A simple schema for decision support systems or data warehouses is the star schema, where events are collected in large *fact tables*, while smaller supporting tables (*dimensions*) are used to describe the data.

Hive supports MAPJOINs, which are well suited for this scenario – at least for dimensions small enough to fit in memory (less than 25MB). MAPJOINs are processed by loading the smaller table into an in-memory hash map and matching keys with the larger table as they are streamed through.

```
select /*+ MAPJOIN(b) */ a.key, a.value from a join b on a.key = b.key
```

```
select accounts.* from accounts a, offers b where a.account_id = b.account_id;
```

Challenge 3:

Calculate total invoice for each account. Hint : group by account_id

Find top 10 spending customers by name. What is the query?

Calculate accounts totals by month

Exit from the hive shell once completed.

!q

Mini-Project

In this Mini-Project you'll learn to work with n-grams on a text file to identify aspects within the file itself. N-grams of texts are extensively used in text mining and natural language processing tasks. They are basically a set of co-occurring words within a given window and when computing the n-grams you typically move one word forward (although you can move X words forward in more advanced scenarios).
https://en.wikipedia.org/wiki/N-gram

Hive uses command formula to create the ngrams in two ways; ngrams and context_ngrams. We will look at both in the Mini-Project.

```
SELECT ngrams(sentences(lower(tweet)), 2, 100 [, 1000]) FROM twitter;

SELECT context_ngrams(sentences(lower(tweet)), array("i","love",null), 100,
[, 1000]) FROM twitter;
```

For more information see:
https://cwiki.apache.org/confluence/display/hive/statisticsanddatamining#StatisticsAndDataMining-ngrams()andcontext_ngrams():N-gramfrequencyestimation

We will start by finding the file within the HDFS file system. The file is called ***moby-dick.txt***, the classic novel.

1. Use Hadoop CLI commands to find the file and the folder path or browse the file system using the Browser Utility. Make a note of the path.

2. Start Hive CLI and switch to the wasadmin_db database.

3. Check to see if the table the 'moby' table already exists.

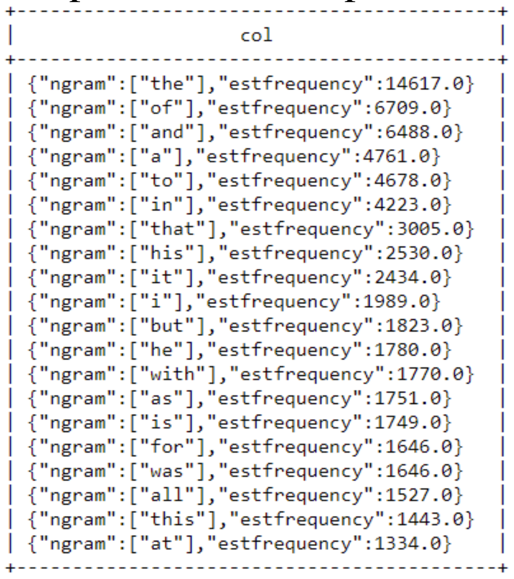
4. If the table exists use it, otherwise create the external table with the following properties:

    a. Column name line, data type is string
    b. Rows are delimited
    c. All fields have a newline character at the end
    d. It is stored as a text file
    e. Found in the path you identified within HDFS

5. Ensure the data from the moby-dick.txt file was loaded properly into the hive 'moby' table you created. Examine the data in the table.

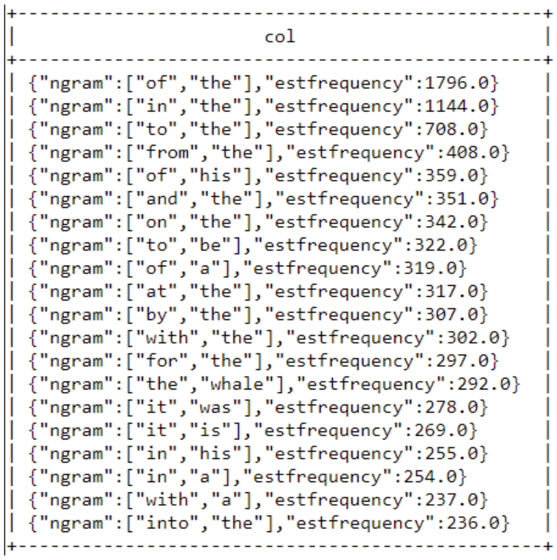
6. Find the 20 most occurring ngrams (one) word in the text. https://en.wikipedia.org/wiki/N-gram. In Hive we can explode, format the output of, the 'ngrams' by reading in each line of text from the file and applying the value of how many, in this case 1)

   You should be able to produce this output:



```
+-----------------------------------------------+
|                     col                       |
+-----------------------------------------------+
| {"ngram":["the"],"estfrequency":14617.0}      |
| {"ngram":["of"],"estfrequency":6709.0}        |
| {"ngram":["and"],"estfrequency":6488.0}       |
| {"ngram":["a"],"estfrequency":4761.0}         |
| {"ngram":["to"],"estfrequency":4678.0}        |
| {"ngram":["in"],"estfrequency":4223.0}        |
| {"ngram":["that"],"estfrequency":3005.0}      |
| {"ngram":["his"],"estfrequency":2530.0}       |
| {"ngram":["it"],"estfrequency":2434.0}        |
| {"ngram":["i"],"estfrequency":1989.0}         |
| {"ngram":["but"],"estfrequency":1823.0}       |
| {"ngram":["he"],"estfrequency":1780.0}        |
| {"ngram":["with"],"estfrequency":1770.0}      |
| {"ngram":["as"],"estfrequency":1751.0}        |
| {"ngram":["is"],"estfrequency":1749.0}        |
| {"ngram":["for"],"estfrequency":1646.0}       |
| {"ngram":["was"],"estfrequency":1646.0}       |
| {"ngram":["all"],"estfrequency":1527.0}       |
| {"ngram":["this"],"estfrequency":1443.0}      |
| {"ngram":["at"],"estfrequency":1334.0}        |
+-----------------------------------------------+
```

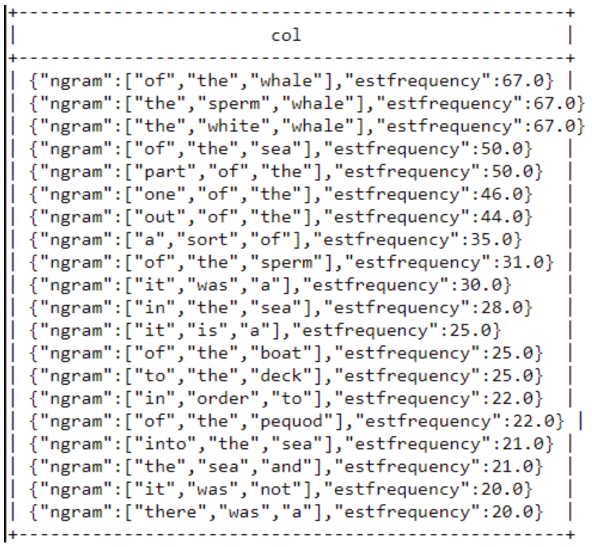
7. Find the 20 most occurring bigrams (two) words in the text. https://en.wikipedia.org/wiki/Bigram. In Hive we can explode, format the output of, the 'ngrams' by reading in each line of text from the file and applying the value of how many, in this case 2) You should be able to produce this output:



```
+----------------------------------------------------+
|                        col                         |
+----------------------------------------------------+
| {"ngram":["of","the"],"estfrequency":1796.0}       |
| {"ngram":["in","the"],"estfrequency":1144.0}       |
| {"ngram":["to","the"],"estfrequency":708.0}        |
| {"ngram":["from","the"],"estfrequency":408.0}      |
| {"ngram":["of","his"],"estfrequency":359.0}        |
| {"ngram":["and","the"],"estfrequency":351.0}       |
| {"ngram":["on","the"],"estfrequency":342.0}        |
| {"ngram":["to","be"],"estfrequency":322.0}         |
| {"ngram":["of","a"],"estfrequency":319.0}          |
| {"ngram":["at","the"],"estfrequency":317.0}        |
| {"ngram":["by","the"],"estfrequency":307.0}        |
| {"ngram":["with","the"],"estfrequency":302.0}      |
| {"ngram":["for","the"],"estfrequency":297.0}       |
| {"ngram":["the","whale"],"estfrequency":292.0}     |
| {"ngram":["it","was"],"estfrequency":278.0}        |
| {"ngram":["it","is"],"estfrequency":269.0}         |
| {"ngram":["in","his"],"estfrequency":255.0}        |
| {"ngram":["in","a"],"estfrequency":254.0}          |
| {"ngram":["with","a"],"estfrequency":237.0}        |
| {"ngram":["into","the"],"estfrequency":236.0}      |
+----------------------------------------------------+
```

8. Find the 20 most occurring trigrams (three) words in the text. https://en.wikipedia.org/wiki/Trigram. In Hive we can explode, format the output of, the ‘ngrams’ by reading in each line of text from the file and applying the value of how many, in this case 2)

   You should be able to produce this output:



```
+----------------------------------------------------+
|                        col                         |
+----------------------------------------------------+
| {"ngram":["of","the","whale"],"estfrequency":67.0} |
| {"ngram":["the","sperm","whale"],"estfrequency":67.0}
| {"ngram":["the","white","whale"],"estfrequency":67.0}
| {"ngram":["of","the","sea"],"estfrequency":50.0}   |
| {"ngram":["part","of","the"],"estfrequency":50.0}  |
| {"ngram":["one","of","the"],"estfrequency":46.0}   |
| {"ngram":["out","of","the"],"estfrequency":44.0}   |
| {"ngram":["a","sort","of"],"estfrequency":35.0}    |
| {"ngram":["of","the","sperm"],"estfrequency":31.0} |
| {"ngram":["it","was","a"],"estfrequency":30.0}     |
| {"ngram":["in","the","sea"],"estfrequency":28.0}   |
| {"ngram":["it","is","a"],"estfrequency":25.0}      |
| {"ngram":["of","the","boat"],"estfrequency":25.0}  |
| {"ngram":["to","the","deck"],"estfrequency":25.0}  |
| {"ngram":["in","order","to"],"estfrequency":22.0}  |
| {"ngram":["of","the","pequod"],"estfrequency":22.0} |
| {"ngram":["into","the","sea"],"estfrequency":21.0} |
| {"ngram":["the","sea","and"],"estfrequency":21.0}  |
| {"ngram":["it","was","not"],"estfrequency":20.0}   |
| {"ngram":["there","was","a"],"estfrequency":20.0}  |
+----------------------------------------------------+
```

9. In the first two attempts we have identified small words often known as “stop words”, and ones we often want to ignore. Using the context_ngram we can search for phrases like “sperm whale” or “white whale” or just “white” if you want.

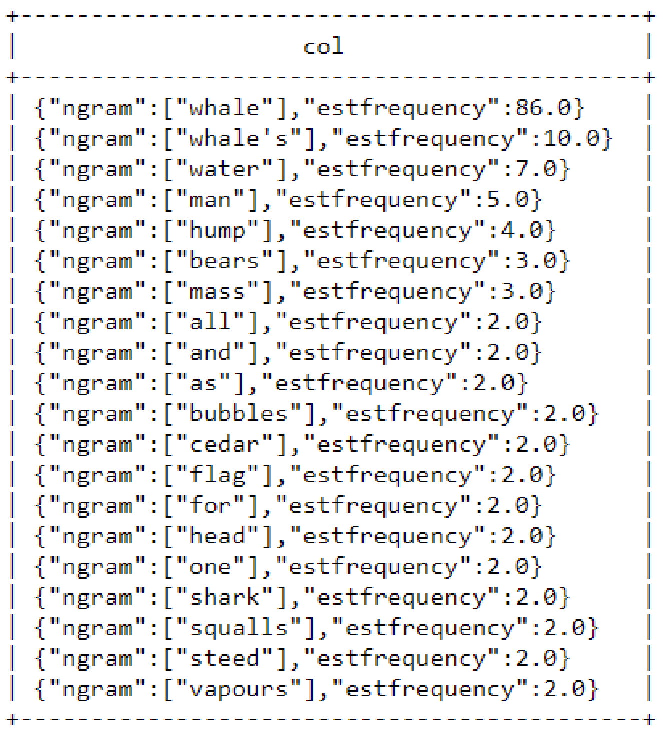
White whale:

```
+------------------------------------------+
|                    col                   |
+------------------------------------------+
| {"ngram":["had"],"estfrequency":7.0}     |
| {"ngram":["a"],"estfrequency":3.0}       |
| {"ngram":["the"],"estfrequency":3.0}     |
| {"ngram":["as"],"estfrequency":2.0}      |
| {"ngram":["did"],"estfrequency":2.0}     |
| {"ngram":["he"],"estfrequency":2.0}      |
| {"ngram":["is"],"estfrequency":2.0}      |
| {"ngram":["must"],"estfrequency":2.0}    |
| {"ngram":["on"],"estfrequency":2.0}      |
| {"ngram":["shirr"],"estfrequency":2.0}   |
| {"ngram":["that"],"estfrequency":2.0}    |
| {"ngram":["was"],"estfrequency":2.0}     |
| {"ngram":["and"],"estfrequency":1.0}     |
| {"ngram":["are"],"estfrequency":1.0}     |
| {"ngram":["be"],"estfrequency":1.0}      |
| {"ngram":["by"],"estfrequency":1.0}      |
| {"ngram":["dashed"],"estfrequency":1.0}  |
| {"ngram":["from"],"estfrequency":1.0}    |
| {"ngram":["so"],"estfrequency":1.0}      |
| {"ngram":["spouts"],"estfrequency":1.0}  |
+------------------------------------------+
```

Sperm whale:

```
+-------------------------------------------+
|                    col                    |
+-------------------------------------------+
| {"ngram":["is"],"estfrequency":10.0}      |
| {"ngram":["and"],"estfrequency":6.0}      |
| {"ngram":["was"],"estfrequency":5.0}      |
| {"ngram":["fishery"],"estfrequency":4.0}  |
| {"ngram":["has"],"estfrequency":4.0}      |
| {"ngram":["in"],"estfrequency":3.0}       |
| {"ngram":["of"],"estfrequency":3.0}       |
| {"ngram":["only"],"estfrequency":3.0}     |
| {"ngram":["will"],"estfrequency":3.0}     |
| {"ngram":["before"],"estfrequency":2.0}   |
| {"ngram":["but"],"estfrequency":2.0}      |
| {"ngram":["for"],"estfrequency":2.0}      |
| {"ngram":["now"],"estfrequency":2.0}      |
| {"ngram":["presents"],"estfrequency":2.0} |
| {"ngram":["to"],"estfrequency":2.0}       |
| {"ngram":["which"],"estfrequency":2.0}    |
| {"ngram":["after"],"estfrequency":1.0}    |
| {"ngram":["compared"],"estfrequency":1.0} |
| {"ngram":["inserted"],"estfrequency":1.0} |
| {"ngram":["on"],"estfrequency":1.0}       |
+-------------------------------------------+
```

White:



```
+-----------------------------------------+
|                   col                   |
+-----------------------------------------+
| {"ngram":["whale"],"estfrequency":86.0}   |
| {"ngram":["whale's"],"estfrequency":10.0} |
| {"ngram":["water"],"estfrequency":7.0}    |
| {"ngram":["man"],"estfrequency":5.0}      |
| {"ngram":["hump"],"estfrequency":4.0}     |
| {"ngram":["bears"],"estfrequency":3.0}    |
| {"ngram":["mass"],"estfrequency":3.0}     |
| {"ngram":["all"],"estfrequency":2.0}      |
| {"ngram":["and"],"estfrequency":2.0}      |
| {"ngram":["as"],"estfrequency":2.0}       |
| {"ngram":["bubbles"],"estfrequency":2.0}  |
| {"ngram":["cedar"],"estfrequency":2.0}    |
| {"ngram":["flag"],"estfrequency":2.0}     |
| {"ngram":["for"],"estfrequency":2.0}      |
| {"ngram":["head"],"estfrequency":2.0}     |
| {"ngram":["one"],"estfrequency":2.0}      |
| {"ngram":["shark"],"estfrequency":2.0}    |
| {"ngram":["squalls"],"estfrequency":2.0}  |
| {"ngram":["steed"],"estfrequency":2.0}    |
| {"ngram":["vapours"],"estfrequency":2.0}  |
+-----------------------------------------+
```

-------------Bonus-----------

There are two additional text files within the HDFS file system. Using similar techniques that you used to work with Moby Dick above apply your understanding to complete the following with these files.

- **sotu-2014-obama.txt**
  State of the Union address 2014

For State of the union text
Find the word the follows the word 'american'

- **FOMC20080916meeting.txt**
  Feds emergency meeting after 2008 financial crisis (un-classified after 5 years)

Feds Meeting minute
Find the most used adjective used to describe the word market.

Review

In this last lab and Mini-Project we have learned about working with tables especially joining them to project answer to specific questions. Remember knowing the data that you are working with is the first step in being successfully in Hive and Hadop.