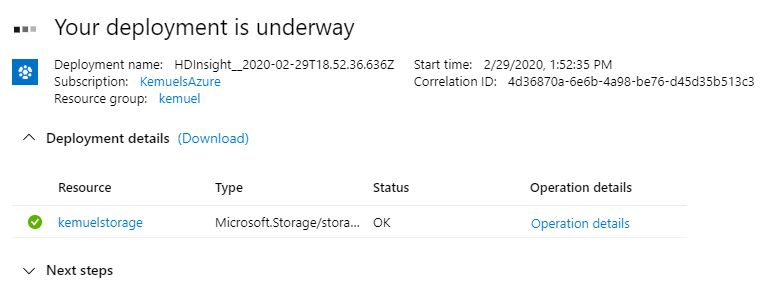
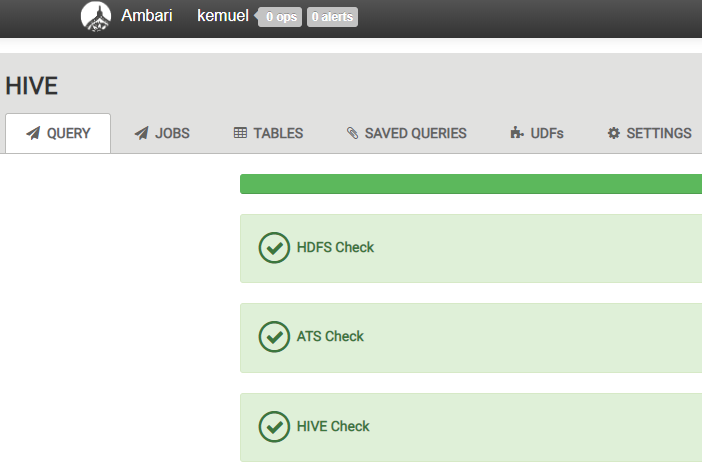
# DATA Transformation Documentation

1. Hadoop 2.7 HDInsight Cluster was re-deployed in Microsoft Azure using Virtual Machine configurations as follows
   1. Head (node size D12V2, 4 cores) with 2 nodes
   2. Work (node size D12V2 4 cores) with 1 node

* Same resource group and container was chosen as before

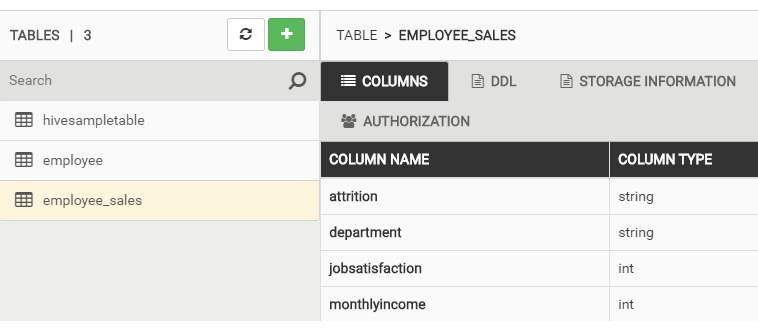
*Fig. 1 - Cluster Deployment in Progress*

1. Hive (Ambari View) was used to load data from Azure storage to the HDInsight cluster, create a new table to be used for data transformation activities and setup the data schema for the new table.



*Fig. 2- Successful System Checks in Hive on Ambari Views*

1. Table View in Ambari was used to verify the new table and schema as shown in Fig. 3.



*Fig.3 - Schema for New Table in Table View*

1. SQL queries were used to verify that the data was loaded properly into the cluster and to perform various transformations of the data, overwriting the same table for each transformation carried out.

**Create New Table**

New table called employee\_sales was created using the following SQL code:

CREATE TABLE employee\_sales

(

Attrition string,

Department string,

JobSatisfaction int,

MonthlyIncome int

);

An SQL query was used to preview the table (note that there were no rows since the table had no data.

SELECT \* FROM employee\_sales LIMIT 10;



*Fig. 4 - New Table Showing Columns Names with No Data*

The *employee* table from previous work was loaded into this new table and only 4 columns were selected (as per the schema above). This demonstrated the use of the SELECT transformation in SQL. The SQL code used to achieve this is:

INSERT OVERWRITE TABLE employee\_sales

SELECT Attrition, Department, JobSatisfaction, MonthlyIncome

FROM employee;

The top 10 rows of the table that now contains data was displayed using the following query again and the results are shown in Fig. 5 below:

SELECT \* FROM employee\_sales LIMIT 10;



*Fig. 5 - Preview of the Top 10 Rows of the Populated Table*

**Mutate**

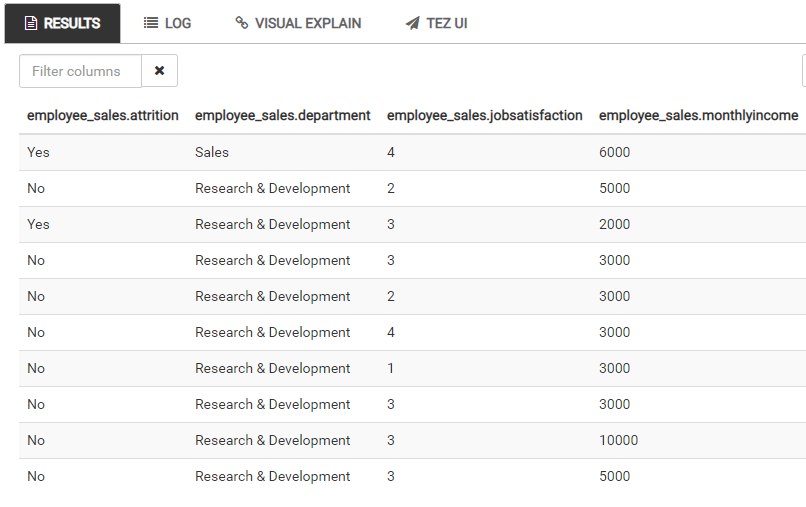
*MonthlyIncome* was rounded to the nearest $1000 using the *ROUND* SQL function as shown below. This is an example of a MUTATE transformation in SQL.

INSERT OVERWRITE TABLE employee\_sales

SELECT Attrition, Department, JobSatisfaction, ROUND(MonthlyIncome, -3) as MonthlyIncome

FROM employee\_sales;

The top 10 rows were then displayed using the *SELECT* SQL command as before. The result is shown in Fig. 6 and confirms that *MonthlyIncome* was rounded to the nearest $1000.



*Fig. 6 - Table showing MonthlyIncome now rounded to the nearest $1000*

**Filtering**

The data was filtered using the SQL query below to show only records for employees in the Sales Department and the results from the Hive interface are shown in Fig. 7 below:

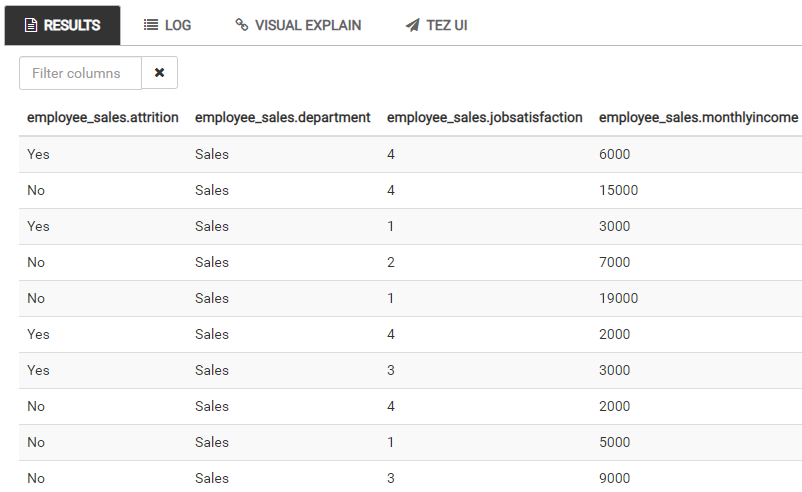
INSERT OVERWRITE TABLE employee\_sales

SELECT \*

FROM employee\_sales

WHERE Department LIKE "%Sales%";

This is an example of filtering data using SQL’s *WHERE* command. *LIKE*, prefixed and suffixed by the % operator, is commonly used when filtering using criteria which is of data type *string,* to ensure the query is capturing data even if records contains leading whitespaces.

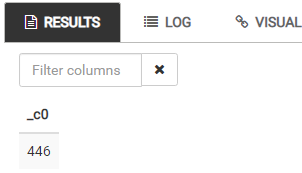


*Fig. 7 - Results from the SQL Filter Query for Filter Data by Department “Sales”*

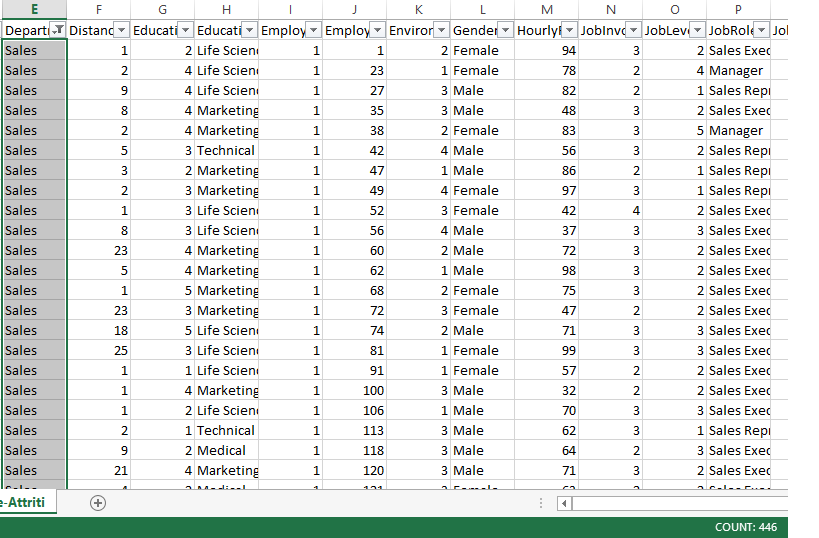
**Count**

To verify the query returned the proper number of rows for the filter, the original CSV file was filtered in Excel to count the number of records for Department “Sales”. The following SQL query *COUNT* was used to count the number of records in the filtered table. The results from the SQL query and the physical CSV dataset are shown in Fig. 8 and Fig. 9 below.

SELECT COUNT(\*) FROM employee\_sales;

**

*Fig. 8 - Result from Hive for Count of Records of the Filtered Table*

**

*Fig. 9 - Excel filter applied to original CSV dataset confirms there are 446 records for Department “Sales”*

**Arrange**

The data was sorted by *JobSatisfaction* in descending order so records with the highest job satisfaction score appear on top. This is an example of the ARRANGE transformation which utilizes the ORDER BY command in SQL.

INSERT OVERWRITE TABLE employee\_sales

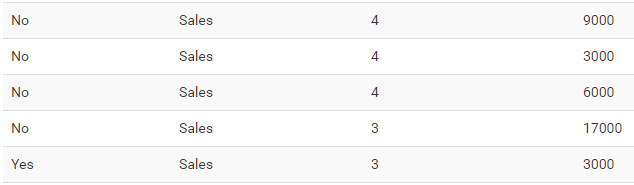
SELECT \*

FROM employee\_sales

ORDER BY JobSatisfaction DESC;

The results returned for the new table are shown in Fig. 10 below (3rd page of the Hive results). To verify the SQL query worked, an SQL query was used to preview a larger number of records as they were more than 100 records with *JobSatisfaction = 4*

SELECT \* FROM employee\_sales LIMIT 150;



*Fig. 10 - Page 3 of the Hive Results for the table sorted in descending order by JobSatisfaction*

The final transformed table *employee\_sales* really shows data on employee attrition (Yes or No) for all employees in the Sales Department with the most satisfied employees at the top. Monthly income is also shown for context.

**Data Source:**

IBM HR Analytics Employee Attrition & Performance (2017). Retrieved Feb 13, 2020 from <https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset/data>