

Assignment - Data Aggregation Activity (10%)

- 1) Create two tables, one showing the number of salespeople with attrition and the other showing the number of salespeople with non-attrition.

Salespeople with attrition:



The screenshot shows the Hive View 2.0 interface. On the left is a sidebar with services like HDFS, YARN, and MapReduce2. The main area has tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. A 'Worksheet1' tab is active. The QUERY tab shows the following SQL code:

```
1 CREATE TABLE attrition_yes
2 (
3     Attrition string,
4     Department string,
5     MonthlyIncome int
6 );
```

The right side shows a 'default' database with two tables: 'employee' and 'hivesamptable'. Below the code editor are buttons for Execute, Save As, Insert UDF, and Visual Explain.



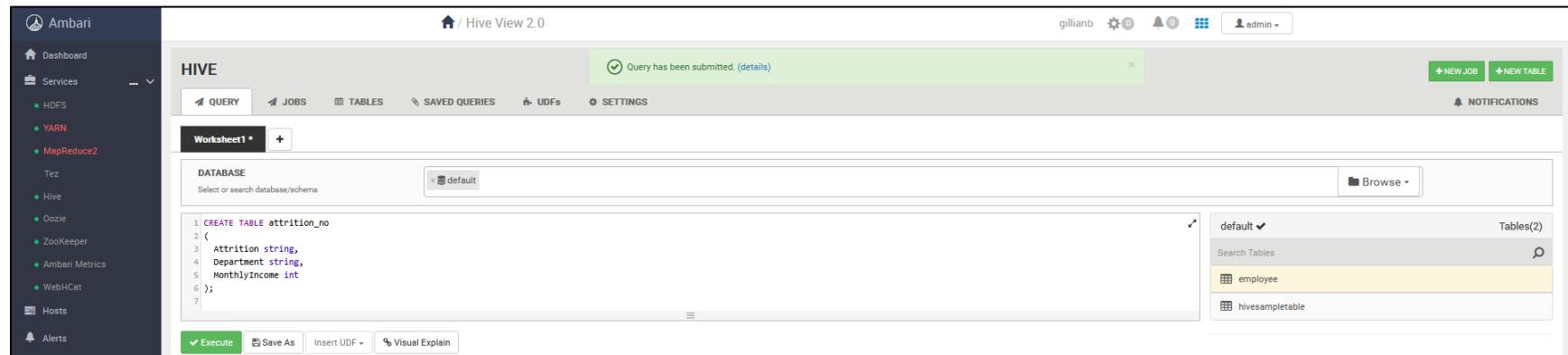
The screenshot shows the same Hive View 2.0 interface after the table 'attrition_yes' has been created. The 'attrition_yes' table now appears in the 'default' database. The QUERY tab shows the following SQL code:

```
1 INSERT OVERWRITE TABLE attrition_yes
2 SELECT Attrition, Department, ROUND(MonthlyIncome,-3)
3 FROM employee
4 WHERE Attrition LIKE "%Yes%" AND Department LIKE "%Sales%";
```

The right side shows the 'attrition_yes' table listed under 'default'. Below the code editor are buttons for Execute, Save As, Insert UDF, and Visual Explain.

I have assumed this activity picks up after the Data Transform Activity and have therefore rounded the Monthly Income data to the nearest thousandth.

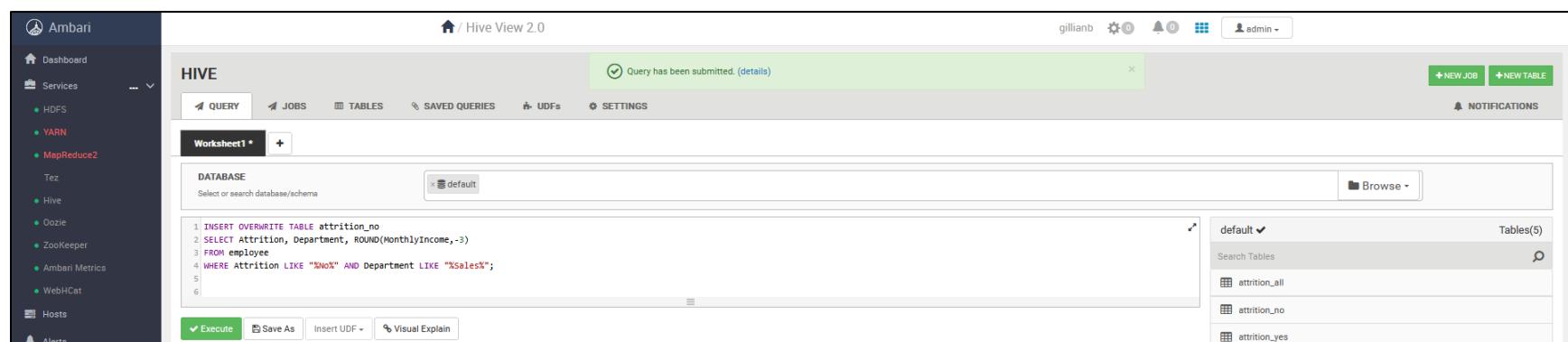
Salespeople with non-attrition:



The screenshot shows the Ambari Hive View 2.0 interface. On the left is a sidebar with various Hadoop services like HDFS, YARN, and MapReduce2. The main area has tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. A green banner at the top says "Query has been submitted. (details)". The QUERY tab is selected, showing a worksheet titled "Worksheet1". The database is set to "default". The query editor contains the following SQL:

```
1 CREATE TABLE attrition_no
2 (
3     Attrition string,
4     Department string,
5     MonthlyIncome int
6 );
7
```

Below the editor are buttons for Execute, Save As, Insert UDF, and Visual Explain. To the right is a sidebar for "Tables(2)" with "employee" and "hivesamptable" listed. A "Browse" button is also present.



This screenshot shows the same Ambari Hive View 2.0 interface after the query has been executed. The green success banner is still visible. The query editor now contains a different set of SQL statements:

```
1 INSERT OVERWRITE TABLE attrition_no
2 SELECT Attrition, Department, ROUND(MonthlyIncome,-3)
3 FROM employee
4 WHERE Attrition LIKE "%No%" AND Department LIKE "%Sales%";
```

The "Execute" button is highlighted in blue. The sidebar on the right shows "Tables(5)" with "attrition_all", "attrition_no", and "attrition_yes" listed under "Search Tables".

I have assumed this activity picks up after the Data Transform Activity and have therefore rounded the Monthly Income data to the nearest thousandth.

- 2) Create three statistics tables showing the monthly income for all salespeople, those with attrition and those without attrition.

All salespeople:

The screenshot shows the Ambari Hive View 2.0 interface. On the left is a sidebar with various Hadoop services like HDFS, YARN, and MapReduce. The main area is titled 'HIVE' and contains a 'Worksheet1' tab with a query editor. The query is:

```
1 SELECT Department, AVG(ROUND(MonthlyIncome,-3)) AS average_monthly_income, MIN(ROUND(MonthlyIncome,-3)) AS min_monthly_income, MAX(ROUND(MonthlyIncome,-3)) AS max_monthly_income
2 FROM employee
3 WHERE Department LIKE "%Sales%"
4 GROUP BY Department;
5
```

Below the query are buttons for 'Execute', 'Save As', 'Insert UDF', and 'Visual Explain'. The results section shows a table with one row for the Sales department:

| department | average_monthly_income | min_monthly_income | max_monthly_income |
|------------|------------------------|--------------------|--------------------|
| Sales | 6959.641255605381 | 1000 | 20000 |

To the right of the results is a sidebar titled 'default' showing a list of tables: attrition_all, attrition_no, attrition_yes, employee (which is selected), and hivesamplable.

I have assumed this activity picks up after the Data Transform Activity and have therefore rounded the Monthly Income data to the nearest thousandth.

Those with attrition:

The screenshot shows the Ambari Hive View 2.0 interface. On the left is a sidebar with various cluster services like HDFS, YARN, and MapReduce2. The main area has tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. A 'Worksheet1' tab is active. The QUERY tab contains a database dropdown set to 'default' and a code editor with the following SQL query:

```
1 SELECT Department, AVG(MonthlyIncome) AS average_monthly_income, MIN(MonthlyIncome) AS min_monthly_income, MAX(MonthlyIncome) AS max_monthly_income
2 FROM attrition_yes
3 GROUP BY Department;
4
```

Below the code editor are buttons for Execute, Save As, Insert UDF, and Visual Explain. The RESULTS tab shows a table with four columns: department, average_monthly_income, min_monthly_income, and max_monthly_income. One row is present for the 'Sales' department. To the right is a sidebar titled 'default' showing a list of tables: attrition_all, attrition_no, attrition_yes, employee, and hivesamplable. The 'employee' table is highlighted.

Those without attrition:

This screenshot is identical to the previous one, except the database dropdown in the QUERY tab is now set to 'attrition_no'. The rest of the interface, including the query code, results table, and sidebar table list, remains the same.

- 3) Create two tables (one for salespeople with attrition and one for salespeople without attrition) showing the monthly income and the count of salespeople making that amount of income.

Salespeople with attrition:

The screenshot shows the Ambari Hive View 2.0 interface. On the left is a sidebar with various service links like HDFS, YARN, MapReduce2, Tez, Hive, Oozie, ZooKeeper, Ambari Metrics, and WebHCat. The main area has tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. A query window titled 'Worksheet1' contains the following SQL code:

```
1 SELECT MonthlyIncome AS monthlyincome, COUNT(MonthlyIncome) AS count
2 FROM attrition_yes
3 GROUP BY MonthlyIncome;
4
```

The 'RESULTS' tab displays the query output as a table:

| monthlyincome | count |
|---------------|-------|
| 1000 | 4 |
| 6000 | 10 |
| 8000 | 5 |
| 9000 | 3 |
| 10000 | 10 |
| 13000 | 2 |
| 19000 | 1 |
| 20000 | 1 |
| 2500 | 15 |
| 3000 | 13 |
| 4000 | 3 |
| 5000 | 15 |
| 7000 | 6 |
| 11000 | 2 |
| 14000 | 2 |

To the right of the results is a sidebar titled 'default' showing a list of tables: attrition_all, attrition_no, attrition_yes, employee, and hivesamplable. The 'employee' table is highlighted in yellow.

Salespeople without attrition:

The screenshot shows the Ambari Hive View 2.0 interface. On the left is a sidebar with the Ambari logo and various cluster management links like Dashboard, Services, Hosts, and Cluster Admin. The main area has tabs for QUERY, JOBS, TABLES, SAVED QUERIES, UDFs, and SETTINGS. The QUERY tab is active, showing a worksheet titled "Worksheet1". The query entered is:

```
1 SELECT monthlyincome AS monthlyincome, COUNT(monthlyincome) AS count
2 FROM attrition_no
3 GROUP BY MonthlyIncome;
4
```

The RESULTS tab is selected, displaying the following table:

| monthlyincome | count |
|---------------|-------|
| 2000 | 18 |
| 3000 | 24 |
| 4000 | 36 |
| 5000 | 77 |
| 7000 | 35 |
| 11000 | 9 |
| 14000 | 5 |
| 16000 | 5 |
| 18000 | 6 |
| 1000 | 2 |
| 6000 | 39 |
| 8000 | 25 |
| 9000 | 20 |
| 10000 | 24 |
| 12000 | 4 |
| 13000 | 6 |
| 15000 | 2 |
| 17000 | 9 |
| 19000 | 4 |
| 20000 | 4 |

To the right of the results table is a sidebar titled "default" showing a list of tables: attrition_all, attrition_no, attrition_yes, employee, and hivesampletable. The "employee" table is highlighted.