UNIVERSITI MALAYA

WQD7009 Big Data Applications and Analytics

Semester 1 2024/2025

INDIVIDUAL ASSIGNMENT

Matric Number	17204762
Name	HAU JIA QI
Occurrence	2
Lecturer	DR. RIYAZ AHAMED ARIYALURAN HABEEB MOHAMED

1.0 Introduction

This dataset provides detailed insights into greenhouse gas emissions across various economic sectors, measured in CO^2 equivalents per monetary unit (2021 USD). It includes direct and supply chain-related emissions, enabling a comprehensive understanding of the environmental impact of economic activities across industries. The dataset is valuable for carbon footprint analysis, lifecycle assessments, sustainability policy formulation, and crafting low-carbon investment strategies. The original dataset comprises eight columns, offering a structured framework for modelling and analysis.

Link of chosen dataset:
Supply Chain Greenhouse Gas Emission

1.1 Explanation of Important Parameters in Dataset

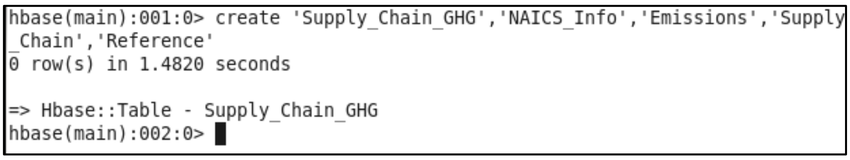
No.	Parameters	Description
1	NAICS Code	The NAICS code corresponds to a specific industry or sector. In this project, this parameter serves as the row key, the unique identifier of my record.
2	NAICS Title	The name of the industry or sector associated with the NAICS code.
3	GHG	The type of greenhouse gas(es) considered.
4	Supply Chain Emission Factors without Margins	The GHG emission factor for each dollar spent in the industry, excluding margin costs.
5	Margins of Supply Chain Emission Factors	The additional emission factor is attributable to margins.
6	Supply Chain Emission Factors with Margins	The total GHG emission factor, including the impact of margins.
7	Reference USEEIO Code	A code referencing the United States Environmentally Extended Input-Output (USEEIO) model, provides a more detailed breakdown or specific classification within the sector.

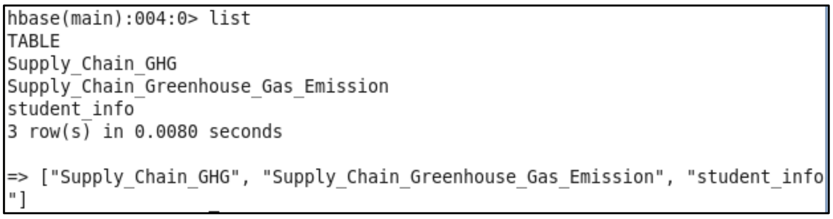
2.0 HBase Queries

2.1 Data Definition Language (DDL) Queries

DDL (Data Definition Language) is used to define, modify, and manage the structure of database objects. These objects include tables, indexes, schemas, and sequences. DDL can create, alter, and delete database structures but not manipulate the data stored within them.

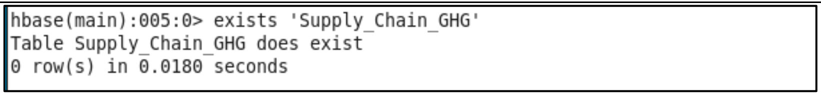
No.	Queries	Output with Description and Explanation

1	create	**Syntax:** create ‘table_name’,’column_family’
		Description: To create a new table in HBase
		Output and explanation:
		
		`hbase(main):001:0> create 'Supply_Chain_GHG','NAICS_Info','Emissions','Supply`
		`_Chain','Reference'`
		`0 row(s) in 1.4820 seconds`
		`=> Hbase::Table - Supply_Chain_GHG`
		`hbase(main):002:0> ▌`
		A table named ‘Supply_Chain_GHG’ is created with three column families that use to categorize each column or parameters - ‘NAICS_Info’, ‘Emissions’, ‘Supply_Chain’ and ‘Reference’.

2	list	**Syntax:** list
		Description: To list all the tables in HBase
		Output and explanation:
		
		`hbase(main):004:0> list`
		`TABLE`
		`Supply_Chain_GHG`
		`Supply_Chain_Greenhouse_Gas_Emission`
		`student_info`
		`3 row(s) in 0.0080 seconds`
		`=> ["Supply_Chain_GHG", "Supply_Chain_Greenhouse_Gas_Emission", "student_info`
		`"]`
		By using this command, all created tables in HBase will be listed. From the output, I can confirm that my table is successfully created.

3	exists	**Syntax:** exists ‘table_name’ **Description:** To verify whether a table exists **Output and explanation:**

<table>
<tr>
<td></td>
<td></td>
<td>



```
hbase(main):005:0> exists 'Supply_Chain_GHG'
Table Supply_Chain_GHG does exist
0 row(s) in 0.0180 seconds
```

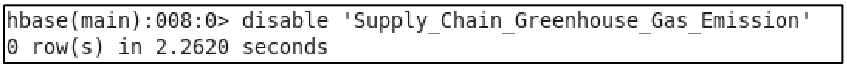
An alternative way to check if my table is successfully created.

</td>
</tr>
<tr>
<td>4</td>
<td>disable</td>
<td>

Syntax: disable ‘table_name’

Description: To disable a table

Output and explanation:



```
hbase(main):008:0> disable 'Supply_Chain_Greenhouse_Gas_Emission'
0 row(s) in 2.2620 seconds
```

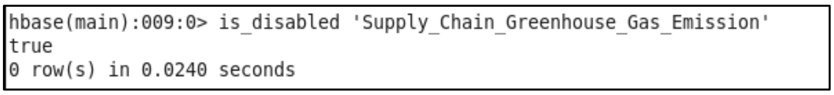
The output of ‘list’ commands contains a wrongly created table. Therefore, it is disabled to ensure no operations occur while performing tasks.

</td>
</tr>
<tr>
<td>5</td>
<td>is_disabled</td>
<td>

Syntax: is_disabled ‘table_name’

Description: To verify whether a table is disabled

Output and explanation:



```
hbase(main):009:0> is_disabled 'Supply_Chain_Greenhouse_Gas_Emission'
true
0 row(s) in 0.0240 seconds
```

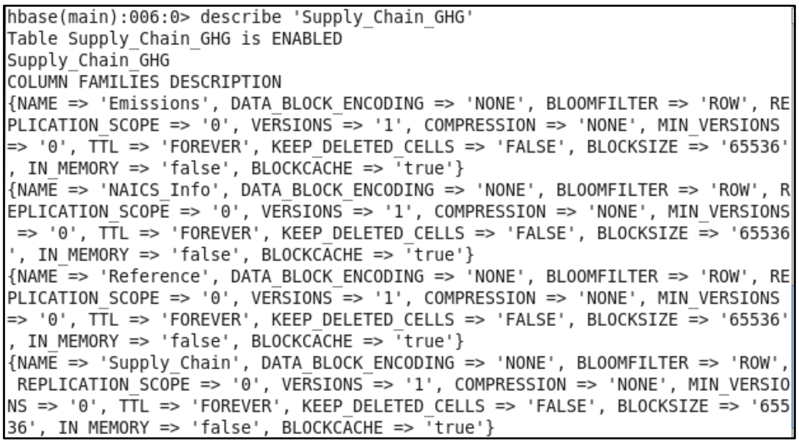
Confirmation on the unwanted table is disabled.

</td>
</tr>
<tr>
<td>6</td>
<td>describe</td>
<td>

Syntax: describe ‘table_name’

Description: To provide description of a table

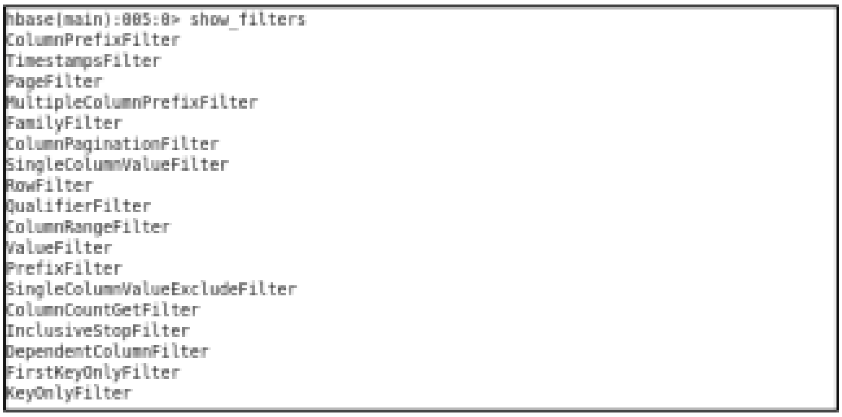
Output and explanation:



```
hbase(main):006:0> describe 'Supply_Chain_GHG'
Table Supply_Chain_GHG is ENABLED
Supply_Chain_GHG
COLUMN FAMILIES DESCRIPTION
{NAME => 'Emissions', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', RE
PLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS
=> '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536'
, IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'NAICS_Info', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', R
EPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS
 => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536
', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'Reference', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', RE
PLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS
=> '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536'
, IN_MEMORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'Supply_Chain', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW',
 REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIO
NS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '655
36', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
```

The description on the created table is displayed.

</td>
</tr>
</table>

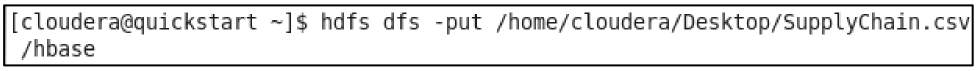
7	show_filters	**Syntax:** show_filters **Description:** To show all the filters in HBase **Output and explanation:**  hbase(main):005:0> show_filters ColumnPrefixFilter TimestampsFilter PageFilter MultipleColumnPrefixFilter FamilyFilter ColumnPaginationFilter SingleColumnValueFilter RowFilter QualifierFilter ColumnRangeFilter ValueFilter PrefixFilter SingleColumnValueExcludeFilter ColumnCountGetFilter InclusiveStopFilter DependentColumnFilter FirstKeyOnlyFilter KeyOnlyFilter All the filters in HBase which are available for the query process are listed.

2.2 Upload and Import Dataset

Before proceeding to the DML queries, I uploaded the dataset CSV file into the Hadoop Distributed File System (HDFS) which later allowed the dataset to be imported into the table created.

<table>
<tr><th>No.</th><th>Commands and Executions</th></tr>
<tr><td>1</td><td>

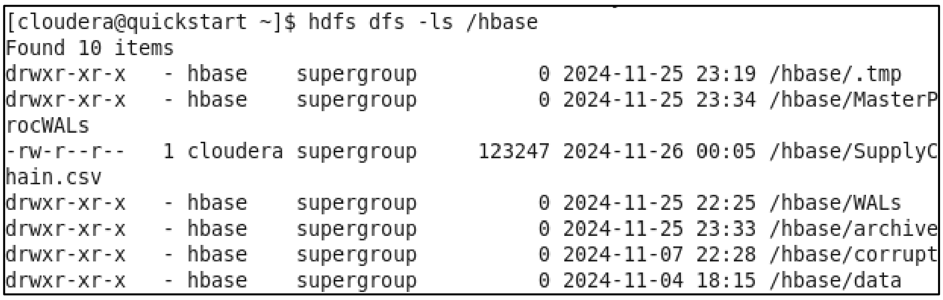
- Upload dataset to HDFS



```
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Desktop/SupplyChain.csv
 /hbase
```

</td></tr>
<tr><td>2</td><td>

- Check if the file is successfully uploaded



```
[cloudera@quickstart ~]$ hdfs dfs -ls /hbase
Found 10 items
drwxr-xr-x   - hbase    supergroup          0 2024-11-25 23:19 /hbase/.tmp
drwxr-xr-x   - hbase    supergroup          0 2024-11-25 23:34 /hbase/MasterP
rocWALs
-rw-r--r--   1 cloudera supergroup     123247 2024-11-26 00:05 /hbase/SupplyC
hain.csv
drwxr-xr-x   - hbase    supergroup          0 2024-11-25 22:25 /hbase/WALs
drwxr-xr-x   - hbase    supergroup          0 2024-11-25 23:33 /hbase/archive
drwxr-xr-x   - hbase    supergroup          0 2024-11-07 22:28 /hbase/corrupt
drwxr-xr-x   - hbase    supergroup          0 2024-11-04 18:15 /hbase/data
```

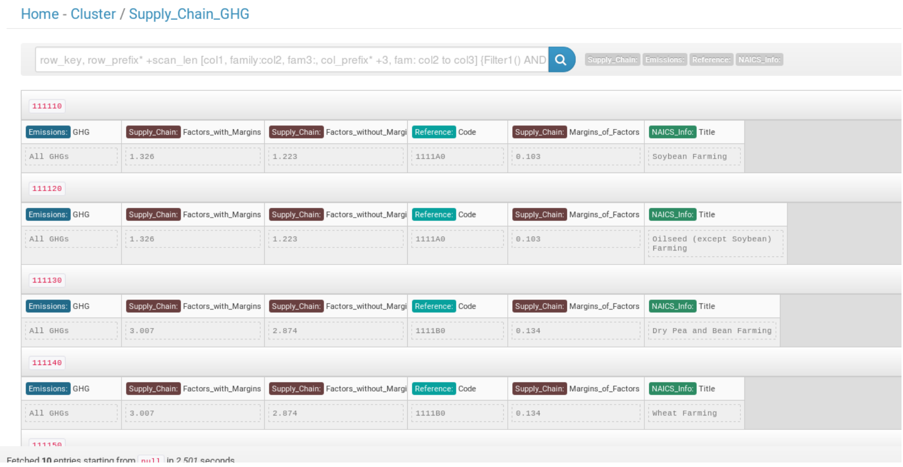
</td></tr>
<tr><td>3</td><td>

- Import the file into HBase table created – Supply_Chain_GHG

```
> hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.separator=',' -
Dimporttsv.columns="HBASE_ROW_KEY,NAICS_Info:Title,Emissions:GHG,Supply
_Chain:Factors_without_Margins,Supply_Chain:Margins_of_Factors,Emission
s:Factors_with_Margins,Reference:Code" Supply_Chain_GHG
/hbase/SupplyChain.csv
```

</td></tr>
<tr><td>4</td><td>

- Check the table in Apache Hue



Home - Cluster / Supply_Chain_GHG

row_key, row_prefix* +scan_len [col1, family:col2, fam3:, col_prefix* +3, fam: col2 to col3] {Filter1 () AND
Supply_Chain:
Emissions:
Reference:
NAICS_Info:

111110

Emissions: GHG	Supply_Chain: Factors_with_Margins	Supply_Chain: Factors_without_Margi	Reference: Code	Supply_Chain: Margins_of_Factors	NAICS_Info: Title
All GHGs	1.326	1.223	1111A0	0.103	Soybean Farming

111120

Emissions: GHG	Supply_Chain: Factors_with_Margins	Supply_Chain: Factors_without_Margi	Reference: Code	Supply_Chain: Margins_of_Factors	NAICS_Info: Title
All GHGs	1.326	1.223	1111A0	0.103	Oilseed (except Soybean) Farming

111130

Emissions: GHG	Supply_Chain: Factors_with_Margins	Supply_Chain: Factors_without_Margi	Reference: Code	Supply_Chain: Margins_of_Factors	NAICS_Info: Title
All GHGs	3.007	2.874	1111B0	0.134	Dry Pea and Bean Farming

111140

Emissions: GHG	Supply_Chain: Factors_with_Margins	Supply_Chain: Factors_without_Margi	Reference: Code	Supply_Chain: Margins_of_Factors	NAICS_Info: Title
All GHGs	3.007	2.874	1111B0	0.134	Wheat Farming

111150

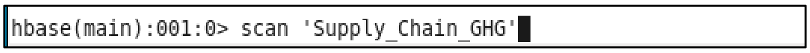
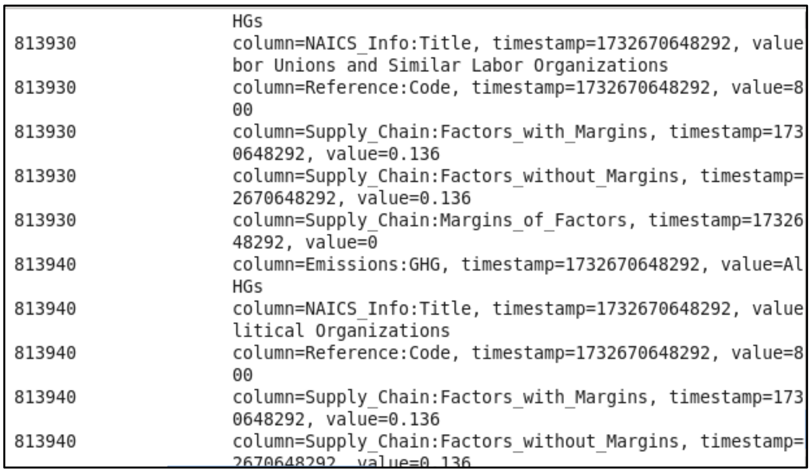
Fetched **10** entries starting from null in 2.501 seconds

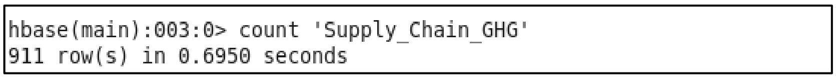
</td></tr>
</table>

2.3 Data Manipulation Language (DML) Queries

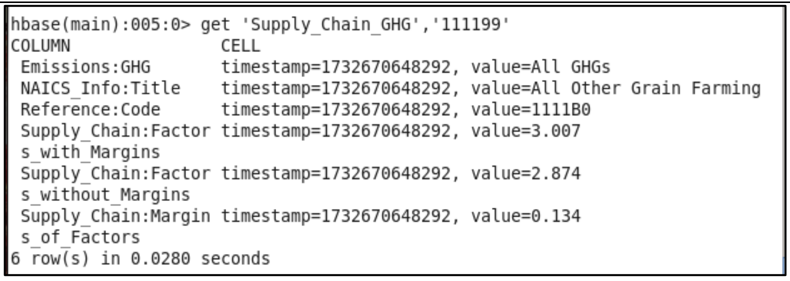
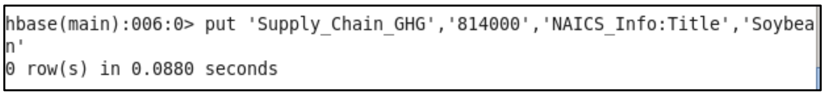
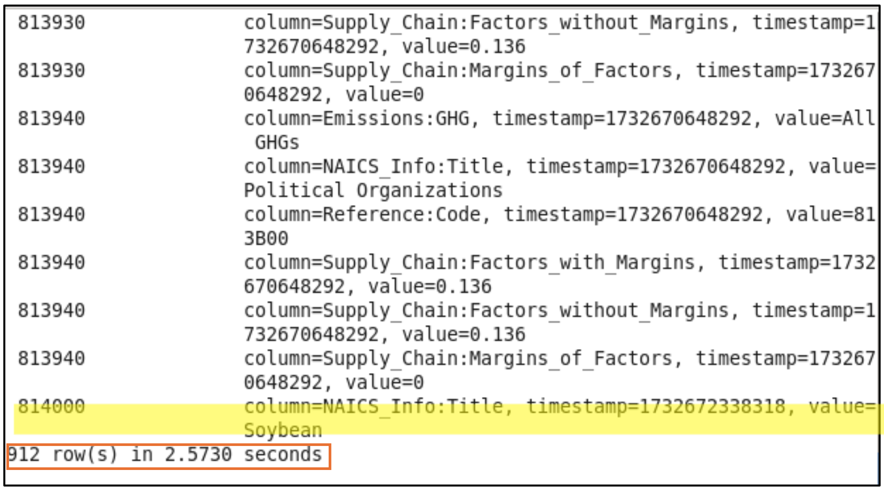
DML (Data Manipulation Language) is a subset of SQL used to manage and manipulate the data stored within database tables. DML focuses on querying, updating, inserting, and deleting the data.

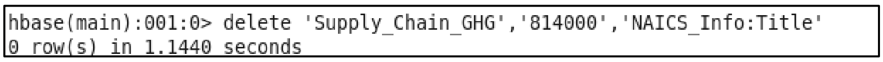
No.	Queries	Output with Description and Explanation

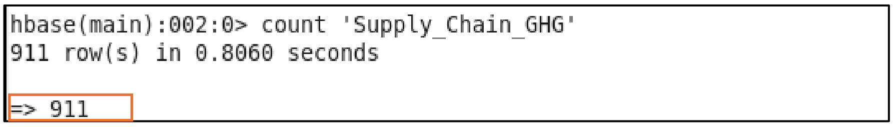
1	scan	**Syntax:** scan ‘table_name’
		Description: To scan and return table data
		Output and explanation:
		
		`hbase(main):001:0> scan 'Supply_Chain_GHG'`
		
		`                     HGs`
		`813930               column=NAICS_Info:Title, timestamp=1732670648292, value`
		`                     bor Unions and Similar Labor Organizations`
		`813930               column=Reference:Code, timestamp=1732670648292, value=8`
		`                     00`
		`813930               column=Supply_Chain:Factors_with_Margins, timestamp=173`
		`                     0648292, value=0.136`
		`813930               column=Supply_Chain:Factors_without_Margins, timestamp=`
		`                     2670648292, value=0.136`
		`813930               column=Supply_Chain:Margins_of_Factors, timestamp=17326`
		`                     48292, value=0`
		`813940               column=Emissions:GHG, timestamp=1732670648292, value=Al`
		`                     HGs`
		`813940               column=NAICS_Info:Title, timestamp=1732670648292, value`
		`                     litical Organizations`
		`813940               column=Reference:Code, timestamp=1732670648292, value=8`
		`                     00`
		`813940               column=Supply_Chain:Factors_with_Margins, timestamp=173`
		`                     0648292, value=0.136`
		`813940               column=Supply_Chain:Factors_without_Margins, timestamp=`
		`                     2670648292, value=0.136`
		The output shows that data is imported into the table.

2	count	**Syntax:** count ‘table_name’
		Description: To count and return number of rows in table
		Output and explanation:
		
		`hbase(main):003:0> count 'Supply_Chain_GHG'`
		`911 row(s) in 0.6950 seconds`
		There is a total of 911 rows in the table.

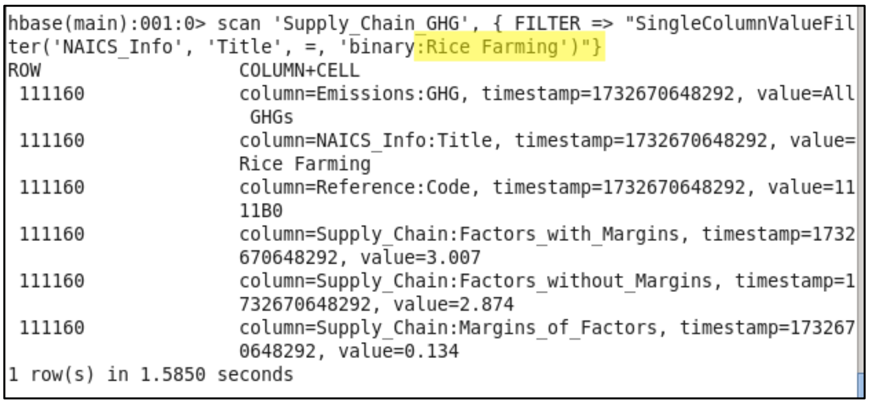
3	get	**Syntax:** get ‘table_name’,’ROW_KEY’ **Description:** To fetch the contents of row or a cell **Output and explanation:**

<table>
<tr><td></td><td></td><td>

<pre>
hbase(main):005:0> get 'Supply_Chain_GHG','111199'
COLUMN   CELL
 Emissions:GHG   timestamp=1732670648292, value=All GHGs
 NAICS_Info:Title   timestamp=1732670648292, value=All Other Grain Farming
 Reference:Code   timestamp=1732670648292, value=1111B0
 Supply_Chain:Factor timestamp=1732670648292, value=3.007
 s_with_Margins
 Supply_Chain:Factor timestamp=1732670648292, value=2.874
 s_without_Margins
 Supply_Chain:Margin timestamp=1732670648292, value=0.134
 s_of_Factors
6 row(s) in 0.0280 seconds
</pre>
The row ‘111199’ is retrieved by using this command. ‘111199’ acts as the row key which helps the system to fetch all the attributes and values for this specific row.
</td></tr>
<tr><td>4</td><td>put</td><td>
<b>Syntax:</b> put ‘table_name’,’ROW_KEY’, ‘column_family:column_name’, ‘new_value’
<b>Description:</b> To insert a cell value at a specified column in a specified row in a table
<b>Output and explanation:</b>

<pre>
hbase(main):006:0> put 'Supply_Chain_GHG','814000','NAICS_Info:Title','Soybea
n'
0 row(s) in 0.0880 seconds
</pre>
A new row ‘814000’ is inserted with a new title ‘Soybean’. The ‘put’ command allows actions like inserting or updating the value of a cell. As this cell does not exist, a new cell is created. If the cell already exists, the cell will be updated with the new value.
</td></tr>
<tr><td>5</td><td>scan
(To check the updated table)</td><td>
<b>Output and explanation:</b>

<pre>
 813930   column=Supply_Chain:Factors_without_Margins, timestamp=1
                      732670648292, value=0.136
 813930   column=Supply_Chain:Margins_of_Factors, timestamp=173267
                      0648292, value=0
 813940   column=Emissions:GHG, timestamp=1732670648292, value=All
                       GHGs
 813940   column=NAICS_Info:Title, timestamp=1732670648292, value=
                      Political Organizations
 813940   column=Reference:Code, timestamp=1732670648292, value=81
                      3B00
 813940   column=Supply_Chain:Factors_with_Margins, timestamp=1732
                      670648292, value=0.136
 813940   column=Supply_Chain:Factors_without_Margins, timestamp=1
                      732670648292, value=0.136
 813940   column=Supply_Chain:Margins_of_Factors, timestamp=173267
                      0648292, value=0
 814000   column=NAICS_Info:Title, timestamp=1732672338318, value=
                      Soybean
912 row(s) in 2.5730 seconds
</pre>
The output shows that the number of rows increased from 911 to 912. The highlighted part also shows the new cell inserted using ‘put’ command.
</td></tr>
</table>

6	delete	**Syntax:** delete ‘table_name’,’ROW_KEY’, ‘column_family:column_name’
		Description: To delete a cell value in a table
		Output and explanation:
		
		`hbase(main):001:0> delete 'Supply_Chain_GHG','814000','NAICS_Info:Title'`
		`0 row(s) in 1.1440 seconds`
		The cell inserted in the previous step is deleted.

7	count	**Output and explanation:**
		
		`hbase(main):002:0> count 'Supply_Chain_GHG'`
		`911 row(s) in 0.8060 seconds`
		
		`=> 911`
		The deletion is confirmed as the number of rows is returned to 911.

This ‘scan’ command along with the **filter** feature optimizes performance and enables targeted data retrieval especially when querying large datasets. Therefore, it is useful in our analysis as we can retrieve records within specific values, ranges or categories.		

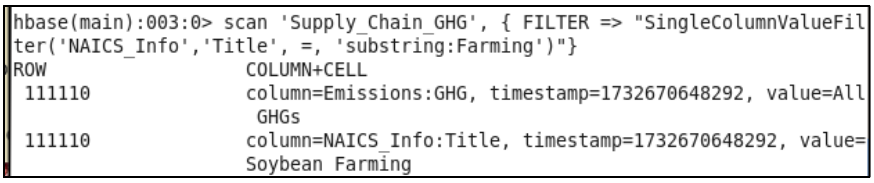
8	Scan with filter	**Syntax:** scan ‘table_name’, { FILTER=> “Filter_type(=, ‘column_family’, ‘column’, =, ‘binary:Value’)”}
		Description: To selectively retrieve data from the table
		Output and explanation:
		
		`hbase(main):001:0> scan 'Supply_Chain_GHG', { FILTER => "SingleColumnValueFil`
		`ter('NAICS_Info', 'Title', =, 'binary:Rice Farming')"}`
		`ROW                   COLUMN+CELL`
		` 111160               column=Emissions:GHG, timestamp=1732670648292, value=All`
		`                       GHGs`
		` 111160               column=NAICS_Info:Title, timestamp=1732670648292, value=`
		`                      Rice Farming`
		` 111160               column=Reference:Code, timestamp=1732670648292, value=11`
		`                      11B0`
		` 111160               column=Supply_Chain:Factors_with_Margins, timestamp=1732`
		`                      670648292, value=3.007`
		` 111160               column=Supply_Chain:Factors_without_Margins, timestamp=1`
		`                      732670648292, value=2.874`
		` 111160               column=Supply_Chain:Margins_of_Factors, timestamp=173267`
		`                      0648292, value=0.134`
		`1 row(s) in 1.5850 seconds`
		This command scans the table and returns all the cells in the same row with the column ‘Title’ that contains ‘Rice Farming’.

9 - 11

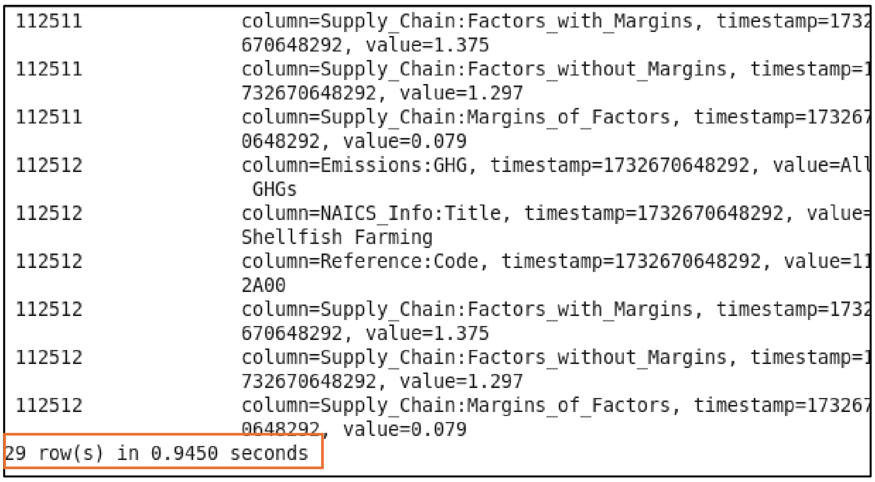
Analysis 1 – **To identify industries that are more frequently involved in supply chain GHG emissions (based on column [NAICS_Info:Title])**

a) scan with filter: Farming industry

Output and explanation:



```
hbase(main):003:0> scan 'Supply_Chain_GHG', { FILTER => "SingleColumnValueFil
ter('NAICS_Info','Title', =, 'substring:Farming')"}
ROW                    COLUMN+CELL
 111110                column=Emissions:GHG, timestamp=1732670648292, value=All
                        GHGs
 111110                column=NAICS_Info:Title, timestamp=1732670648292, value=
                       Soybean Farming
```

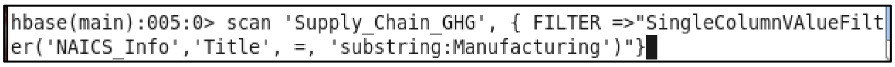


```
 112511                column=Supply_Chain:Factors_with_Margins, timestamp=1732
                       670648292, value=1.375
 112511                column=Supply_Chain:Factors_without_Margins, timestamp=1
                       732670648292, value=1.297
 112511                column=Supply_Chain:Margins_of_Factors, timestamp=173267
                       0648292, value=0.079
 112512                column=Emissions:GHG, timestamp=1732670648292, value=All
                        GHGs
 112512                column=NAICS_Info:Title, timestamp=1732670648292, value=
                       Shellfish Farming
 112512                column=Reference:Code, timestamp=1732670648292, value=11
                       2A00
 112512                column=Supply_Chain:Factors_with_Margins, timestamp=1732
                       670648292, value=1.375
 112512                column=Supply_Chain:Factors_without_Margins, timestamp=1
                       732670648292, value=1.297
 112512                column=Supply_Chain:Margins_of_Factors, timestamp=173267
                       0648292, value=0.079
29 row(s) in 0.9450 seconds
```

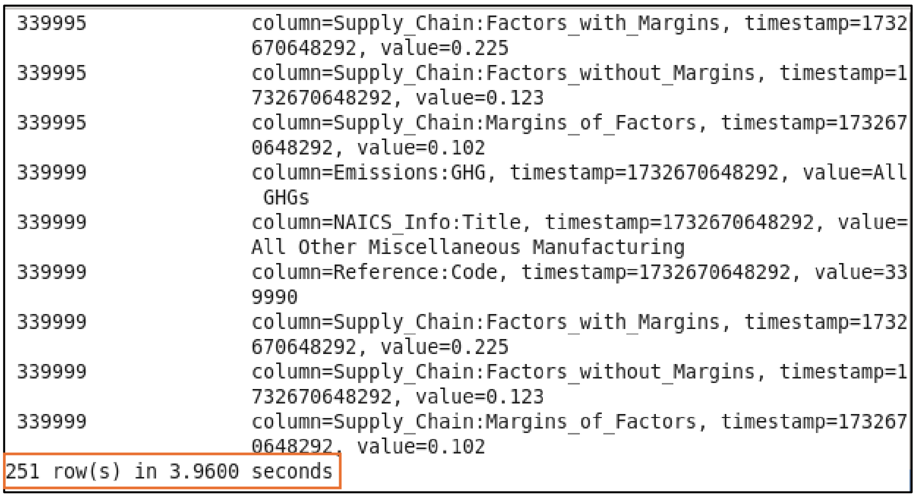
From the output, there are 29 subindustries in the Farming industry.

b) scan with filter: Manufacturing industry

Output and explanation:



```
hbase(main):005:0> scan 'Supply_Chain_GHG', { FILTER =>"SingleColumnVAlueFilt
er('NAICS_Info','Title', =, 'substring:Manufacturing')"}
```

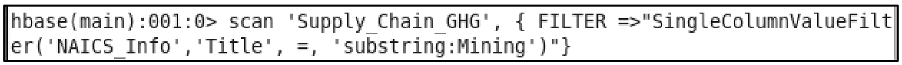


```
 339995                column=Supply_Chain:Factors_with_Margins, timestamp=1732
                       670648292, value=0.225
 339995                column=Supply_Chain:Factors_without_Margins, timestamp=1
                       732670648292, value=0.123
 339995                column=Supply_Chain:Margins_of_Factors, timestamp=173267
                       0648292, value=0.102
 339999                column=Emissions:GHG, timestamp=1732670648292, value=All
                        GHGs
 339999                column=NAICS_Info:Title, timestamp=1732670648292, value=
                       All Other Miscellaneous Manufacturing
 339999                column=Reference:Code, timestamp=1732670648292, value=33
                       9990
 339999                column=Supply_Chain:Factors_with_Margins, timestamp=1732
                       670648292, value=0.225
 339999                column=Supply_Chain:Factors_without_Margins, timestamp=1
                       732670648292, value=0.123
 339999                column=Supply_Chain:Margins_of_Factors, timestamp=173267
                       0648292, value=0.102
251 row(s) in 3.9600 seconds
```

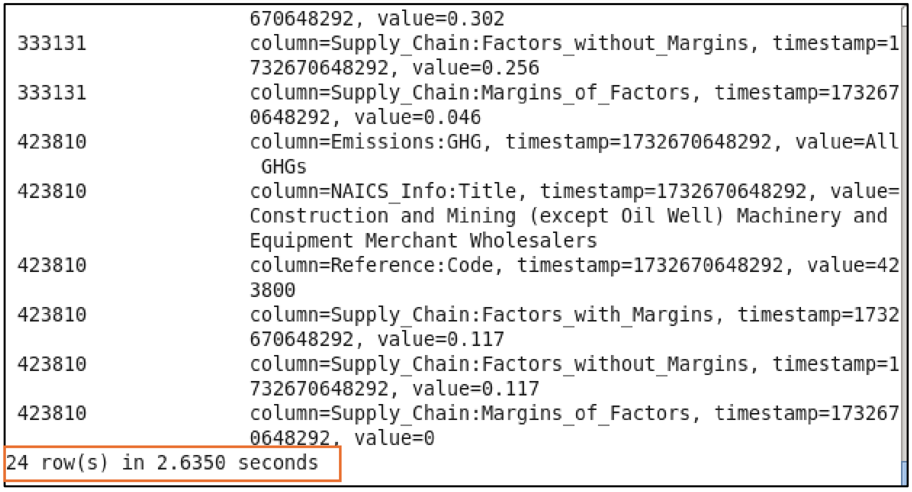
From the output, there are 251 subindustries in the Manufacturing industry.

c) scan with filter: Mining industry

Output and explanation:



```
hbase(main):001:0> scan 'Supply_Chain_GHG', { FILTER =>"SingleColumnValueFilt
er('NAICS_Info','Title', =, 'substring:Mining')"}
```



```
                         670648292, value=0.302
 333131                  column=Supply_Chain:Factors_without_Margins, timestamp=1
                         732670648292, value=0.256
 333131                  column=Supply_Chain:Margins_of_Factors, timestamp=173267
                         0648292, value=0.046
 423810                  column=Emissions:GHG, timestamp=1732670648292, value=All
                          GHGs
 423810                  column=NAICS_Info:Title, timestamp=1732670648292, value=
                         Construction and Mining (except Oil Well) Machinery and
                         Equipment Merchant Wholesalers
 423810                  column=Reference:Code, timestamp=1732670648292, value=42
                         3800
 423810                  column=Supply_Chain:Factors_with_Margins, timestamp=1732
                         670648292, value=0.117
 423810                  column=Supply_Chain:Factors_without_Margins, timestamp=1
                         732670648292, value=0.117
 423810                  column=Supply_Chain:Margins_of_Factors, timestamp=173267
                         0648292, value=0
24 row(s) in 2.6350 seconds
```

From the output, there are 24 subindustries in Mining industry.

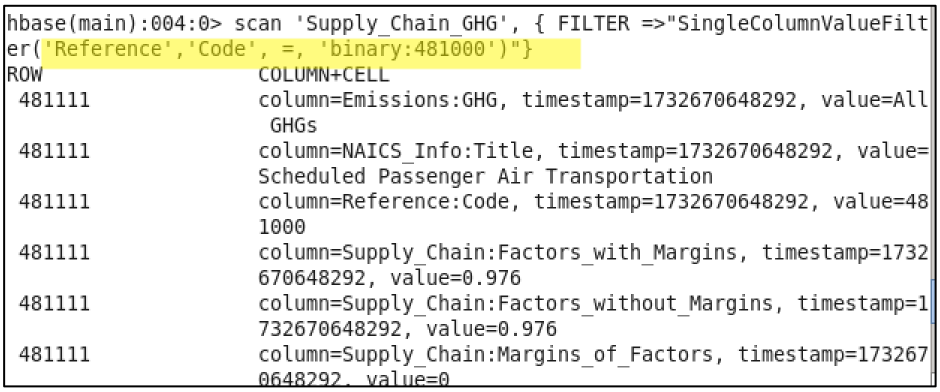
Analysis 1:

A simple comparison is made between three industries, ‘Farming’, ‘Manufacturing’, and ‘Mining’, to determine which one has a higher involvement in supply chain greenhouse gas emissions. By querying the data using ‘scan with filter’, we can determine that ‘Manufacturing’ has a higher involvement among the scanned industries by displaying 251 rows of subindustries, while ‘Mining’ has the lowest count with 24 rows.

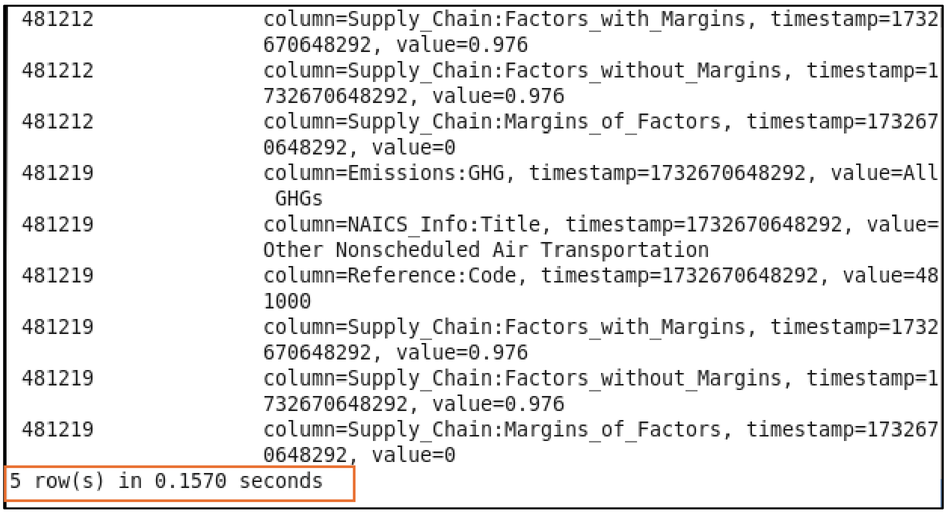
12

Analysis 2 – **To determine the total GHG emissions of certain industries (based on column [Reference:Code])** #The code refers to USEEIO code

Output and explanation:



```
hbase(main):004:0> scan 'Supply_Chain_GHG', { FILTER =>"SingleColumnValueFilt
er('Reference','Code', =, 'binary:481000')"}
ROW                  COLUMN+CELL
 481111              column=Emissions:GHG, timestamp=1732670648292, value=All
                      GHGs
 481111              column=NAICS_Info:Title, timestamp=1732670648292, value=
                     Scheduled Passenger Air Transportation
 481111              column=Reference:Code, timestamp=1732670648292, value=48
                     1000
 481111              column=Supply_Chain:Factors_with_Margins, timestamp=1732
                     670648292, value=0.976
 481111              column=Supply_Chain:Factors_without_Margins, timestamp=1
                     732670648292, value=0.976
 481111              column=Supply_Chain:Margins_of_Factors, timestamp=173267
                     0648292, value=0
```



```
 481212              column=Supply_Chain:Factors_with_Margins, timestamp=1732
                     670648292, value=0.976
 481212              column=Supply_Chain:Factors_without_Margins, timestamp=1
                     732670648292, value=0.976
 481212              column=Supply_Chain:Margins_of_Factors, timestamp=173267
                     0648292, value=0
 481219              column=Emissions:GHG, timestamp=1732670648292, value=All
                      GHGs
 481219              column=NAICS_Info:Title, timestamp=1732670648292, value=
                     Other Nonscheduled Air Transportation
 481219              column=Reference:Code, timestamp=1732670648292, value=48
                     1000
 481219              column=Supply_Chain:Factors_with_Margins, timestamp=1732
                     670648292, value=0.976
 481219              column=Supply_Chain:Factors_without_Margins, timestamp=1
                     732670648292, value=0.976
 481219              column=Supply_Chain:Margins_of_Factors, timestamp=173267
                     0648292, value=0
5 row(s) in 0.1570 seconds
```

There are 5 subindustries under USEEIO code 481000.

Analysis 2:

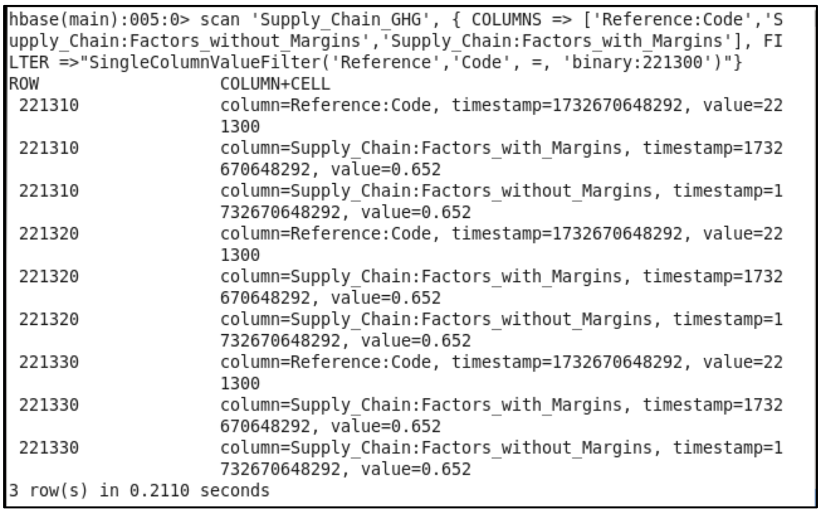
USEEIO code is used to uniquely identify and classify industries, products, or sectors within the USEEIO model. Therefore, by using 'scan with filter', we can obtain all the subindustries under a certain industry or sector based on their unique USEEIO code. With this, analysis like determining the total GHG emissions of certain industries can be done. By filtering out all the subindustries under a sector from the large dataset, we can easily calculate the total gas emission of a certain industry with or without margins.

In this example of analysis, code 481000 refers to Air Transportation sector. The output shows that there are 5 subindustries under the mentioned sector. We can obtain the total supply chain gas emissions for this sector by simply summing up the 5 values under the column which is 4.88.

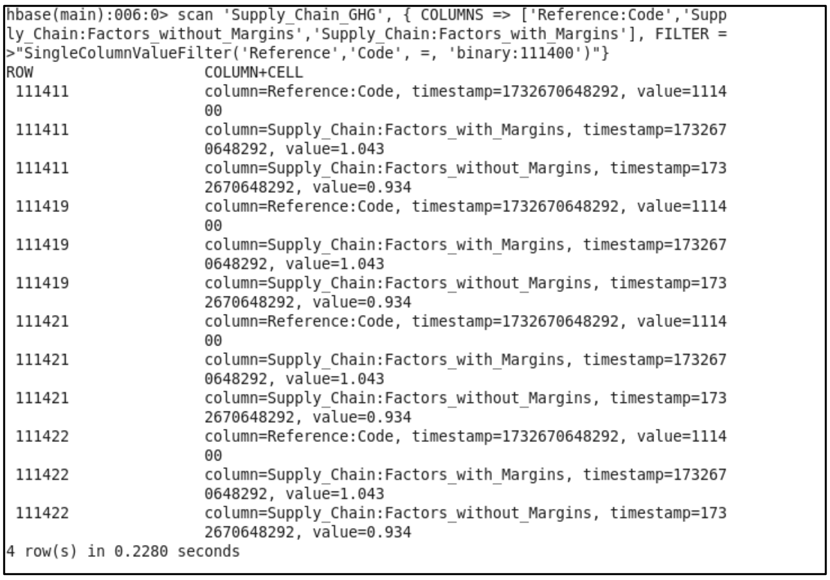
13

Analysis 3 – **To compare emission factors across industries (based on column [Reference:Code], [Supply_Chain:Factors_without_Margins], [Supply_Chain:Factors_with_Margins])**

Output and explanation:



```
hbase(main):005:0> scan 'Supply_Chain_GHG', { COLUMNS => ['Reference:Code','S
upply_Chain:Factors_without_Margins','Supply_Chain:Factors_with_Margins'], FI
LTER =>"SingleColumnValueFilter('Reference','Code', =, 'binary:221300')"}
ROW                   COLUMN+CELL
 221310               column=Reference:Code, timestamp=1732670648292, value=22
                      1300
 221310               column=Supply_Chain:Factors_with_Margins, timestamp=1732
                      670648292, value=0.652
 221310               column=Supply_Chain:Factors_without_Margins, timestamp=1
                      732670648292, value=0.652
 221320               column=Reference:Code, timestamp=1732670648292, value=22
                      1300
 221320               column=Supply_Chain:Factors_with_Margins, timestamp=1732
                      670648292, value=0.652
 221320               column=Supply_Chain:Factors_without_Margins, timestamp=1
                      732670648292, value=0.652
 221330               column=Reference:Code, timestamp=1732670648292, value=22
                      1300
 221330               column=Supply_Chain:Factors_with_Margins, timestamp=1732
                      670648292, value=0.652
 221330               column=Supply_Chain:Factors_without_Margins, timestamp=1
                      732670648292, value=0.652
3 row(s) in 0.2110 seconds
```



```
hbase(main):006:0> scan 'Supply_Chain_GHG', { COLUMNS => ['Reference:Code','Supp
ly_Chain:Factors_without_Margins','Supply_Chain:Factors_with_Margins'], FILTER =
>"SingleColumnValueFilter('Reference','Code', =, 'binary:111400')"}
ROW                   COLUMN+CELL
 111411               column=Reference:Code, timestamp=1732670648292, value=1114
                      00
 111411               column=Supply_Chain:Factors_with_Margins, timestamp=173267
                      0648292, value=1.043
 111411               column=Supply_Chain:Factors_without_Margins, timestamp=173
                      2670648292, value=0.934
 111419               column=Reference:Code, timestamp=1732670648292, value=1114
                      00
 111419               column=Supply_Chain:Factors_with_Margins, timestamp=173267
                      0648292, value=1.043
 111419               column=Supply_Chain:Factors_without_Margins, timestamp=173
                      2670648292, value=0.934
 111421               column=Reference:Code, timestamp=1732670648292, value=1114
                      00
 111421               column=Supply_Chain:Factors_with_Margins, timestamp=173267
                      0648292, value=1.043
 111421               column=Supply_Chain:Factors_without_Margins, timestamp=173
                      2670648292, value=0.934
 111422               column=Reference:Code, timestamp=1732670648292, value=1114
                      00
 111422               column=Supply_Chain:Factors_with_Margins, timestamp=173267
                      0648292, value=1.043
 111422               column=Supply_Chain:Factors_without_Margins, timestamp=173
                      2670648292, value=0.934
4 row(s) in 0.2280 seconds
```

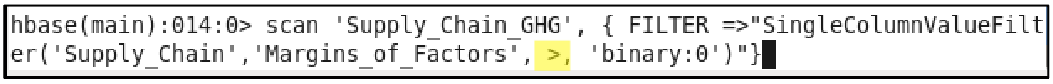
Analysis 3:

This scan allows us to view emission factors for different industries identified by their USEEIO codes and compare the "Factors without Margins" and "Factors with Margins" by calculating the differences, and averages which enable us to identify which industries have higher emissions with or without margins.

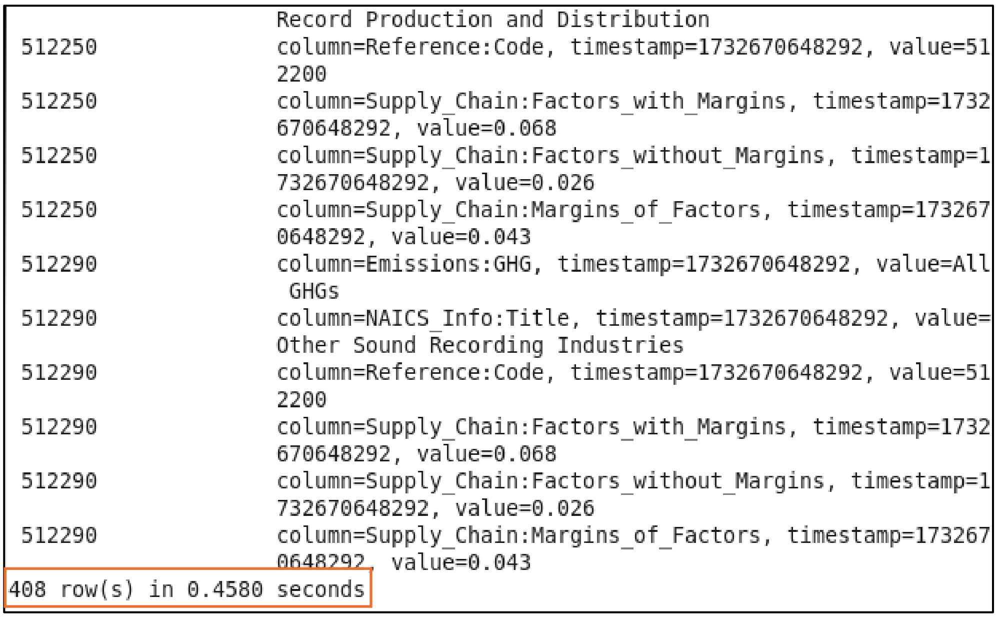
14

Analysis 4 - **To determine industries with and without margins (based on column [Supply_Chain:Margins_of_Factors])**

Output and explanation:



```
hbase(main):014:0> scan 'Supply_Chain_GHG', { FILTER =>"SingleColumnValueFilt
er('Supply_Chain','Margins_of_Factors', >, 'binary:0')"}
```



```
                           Record Production and Distribution
 512250                    column=Reference:Code, timestamp=1732670648292, value=51
                           2200
 512250                    column=Supply_Chain:Factors_with_Margins, timestamp=1732
                           670648292, value=0.068
 512250                    column=Supply_Chain:Factors_without_Margins, timestamp=1
                           732670648292, value=0.026
 512250                    column=Supply_Chain:Margins_of_Factors, timestamp=173267
                           0648292, value=0.043
 512290                    column=Emissions:GHG, timestamp=1732670648292, value=All
                            GHGs
 512290                    column=NAICS_Info:Title, timestamp=1732670648292, value=
                           Other Sound Recording Industries
 512290                    column=Reference:Code, timestamp=1732670648292, value=51
                           2200
 512290                    column=Supply_Chain:Factors_with_Margins, timestamp=1732
                           670648292, value=0.068
 512290                    column=Supply_Chain:Factors_without_Margins, timestamp=1
                           732670648292, value=0.026
 512290                    column=Supply_Chain:Margins_of_Factors, timestamp=173267
                           0648292, value=0.043
408 row(s) in 0.4580 seconds
```

408 out of 911 rows have greater value than 0 under the column 'Margins_of_Factors'.

Analysis 4:

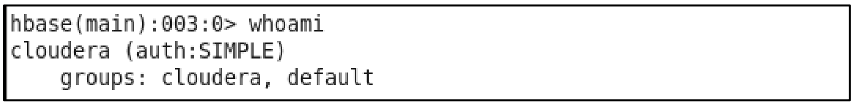
From the dataset, certain industries do not have Margins of Supply Chain Emission Factors. To filter out those with margins and those without margins, we can use 'scan with filter'.

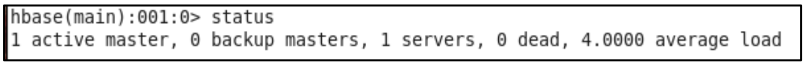
In this example, industries without margins will have a 0 value. So, I use '>' to obtain values that are larger than 0, indicating the industries have margins. The output shows that there are 408 subindustries with margins. As we already know the total rows are 911 in this table with the help of previous commands, hence, the number of subindustries that do not have margins will be 503.

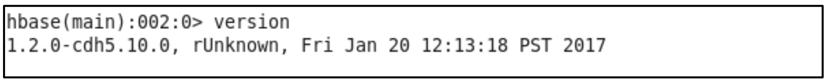
2.4 Shell Commands

HBase shell commands are general administration commands that are used for the overall system.

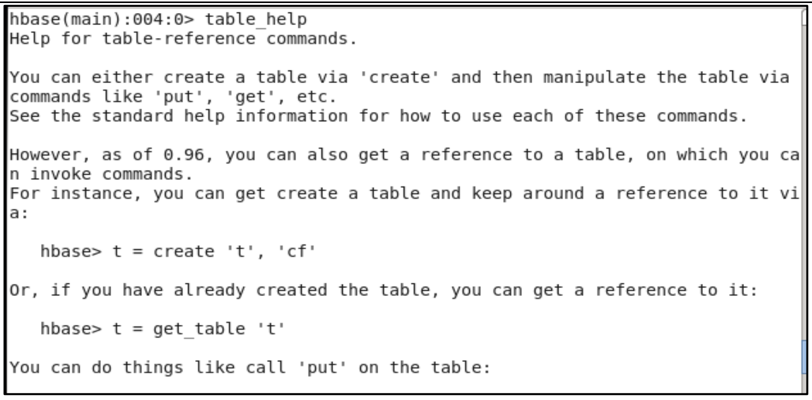
No.	Queries	Output with Description and Explanation

1	whoami	**Syntax:** whoami
		Description: To display current user details
		Output and explanation:
		
		`hbase(main):003:0> whoami`
		`cloudera (auth:SIMPLE)`
		`    groups: cloudera, default`

2	status	**Syntax:** status
		Description: To check status of clusters
		Output and explanation:
		
		`hbase(main):001:0> status`
		`1 active master, 0 backup masters, 1 servers, 0 dead, 4.0000 average load`
		The output indicates that one active master is managing the HBase cluster without any backup master. A single server is responsible for handling the storage in the HBase tables, and there are no unavailable servers in the cluster. The workload managed by the server is 40,000.

3	version	**Syntax:** version
		Description: To check the HBase version
		Output and explanation:
		
		`hbase(main):002:0> version`
		`1.2.0-cdh5.10.0, rUnknown, Fri Jan 20 12:13:18 PST 2017`
		The version is 1.2.0 with CDH 5.10.0 version. 'rUnKnown' shows that the build information is unknown, and the software was built on Friday, 20th of January,2017 at 12:13:18 PST.

4	table_help	**Syntax:** table_help **Description:** To provide references on table-related commands and syntax **Output and explanation:**

<table>
<tr>
<td></td>
<td></td>
<td>

<pre>
hbase(main):004:0> table_help
Help for table-reference commands.

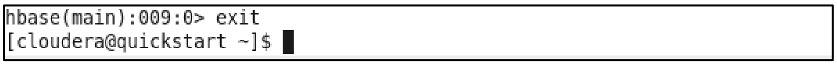
You can either create a table via 'create' and then manipulate the table via
commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you ca
n invoke commands.
For instance, you can get create a table and keep around a reference to it vi
a:

   hbase> t = create 't', 'cf'

Or, if you have already created the table, you can get a reference to it:

   hbase> t = get_table 't'

You can do things like call 'put' on the table:
</pre>
</td>
</tr>
<tr>
<td>5</td>
<td>exit</td>
<td>
<b>Syntax:</b> exit
<b>Description:</b> To quit HBase shell
<b>Output and explanation:</b>

<pre>
hbase(main):009:0> exit
[cloudera@quickstart ~]$ █
</pre>
User is returned to the original base terminal.
</td>
</tr>
</table>

3.0 Conclusion

HBase queries facilitate the analysis process by enabling efficient and scalable data retrieval, making it easier to query large datasets.

In this project, an example of how the dataset can inform policies and strategies for sustainability practices is by identifying high-GHG emission industries. Therefore, authorities can prioritise regulation and sustainability programs for those industries with high emissions. For instance, incentivise renewable energy or efficiency improvements in manufacturing. Additionally, by querying ‘USEEIO Code’ or ‘NAICS Title’, we can categorise the industries and analyse their emissions based on each category. In this case, industry-specific sustainability strategies can be designed. Meanwhile, we can assess the marginal emission impacts by obtaining the difference between "Supply Chain Emission Factors with Margins" and "Supply Chain Emission Factors without Margins". Hence, policies can target margin-related emission contributors, such as by optimising supply chain processes or reducing overproduction. Many more analyses can be done using this dataset with the help of the HBase query.

References

Cik, O. (2018, June 7). *Import CSV data into HBase*. BIG DATA PROGRAMMERS. https://bigdataprogrammers.com/import-csv-data-into-hbase/

Cloudduggu. (n.d.). https://www.cloudduggu.com/hbase/ddl-commands/

Load data into HBase table. (n.d.). https://docs.cloudera.com/cdsw/1.10.5/import-data/topics/cdsw-load-data-into-hbase-table.html

Perform scans using HBase Shell. (n.d.). https://docs.cloudera.com/runtime/7.2.18/managing-hbase/topics/hbase-perform-scans-using-hbase-shell.html

Sankar, H. H. (2023, July 27). *Hbase commands*. Scaler Topics. https://www.scaler.com/topics/hadoop/hbase-commands/