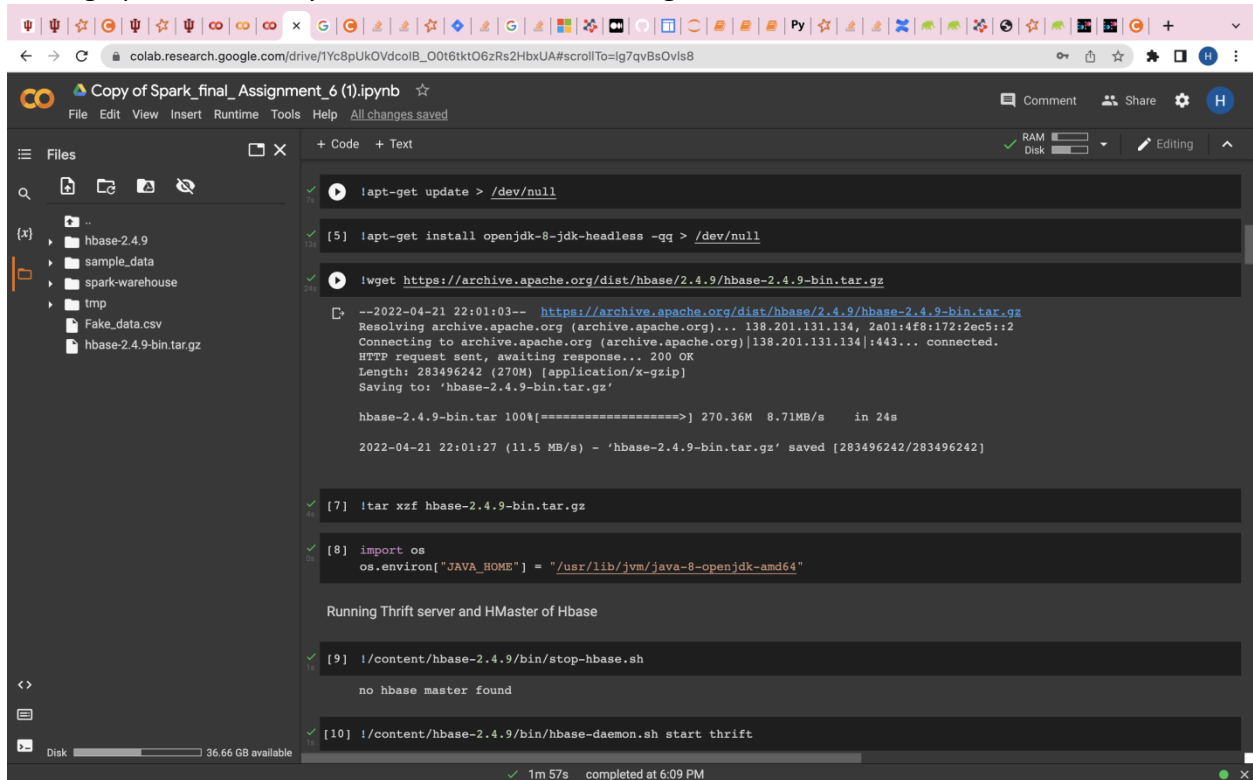


## Setting up Java Home system variable and installing JDK and HBase



```
!apt-get update > /dev/null

[5] !apt-get install openjdk-8-jdk-headless -qq > /dev/null

!wget https://archive.apache.org/dist/hbase/2.4.9/hbase-2.4.9-bin.tar.gz

--2022-04-21 22:01:03-- https://archive.apache.org/dist/hbase/2.4.9/hbase-2.4.9-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 138.201.131.134, 2a01:4f8:172:2ec5::2
Connecting to archive.apache.org (archive.apache.org)|138.201.131.134|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 283496242 (270M) [application/x-gzip]
Saving to: 'hbase-2.4.9-bin.tar.gz'

hbase-2.4.9-bin.tar 100%[=====] 270.36M  8.71MB/s   in 24s

2022-04-21 22:01:27 (11.5 MB/s) - 'hbase-2.4.9-bin.tar.gz' saved [283496242/283496242]

[7] !tar xzf hbase-2.4.9-bin.tar.gz

[8] import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

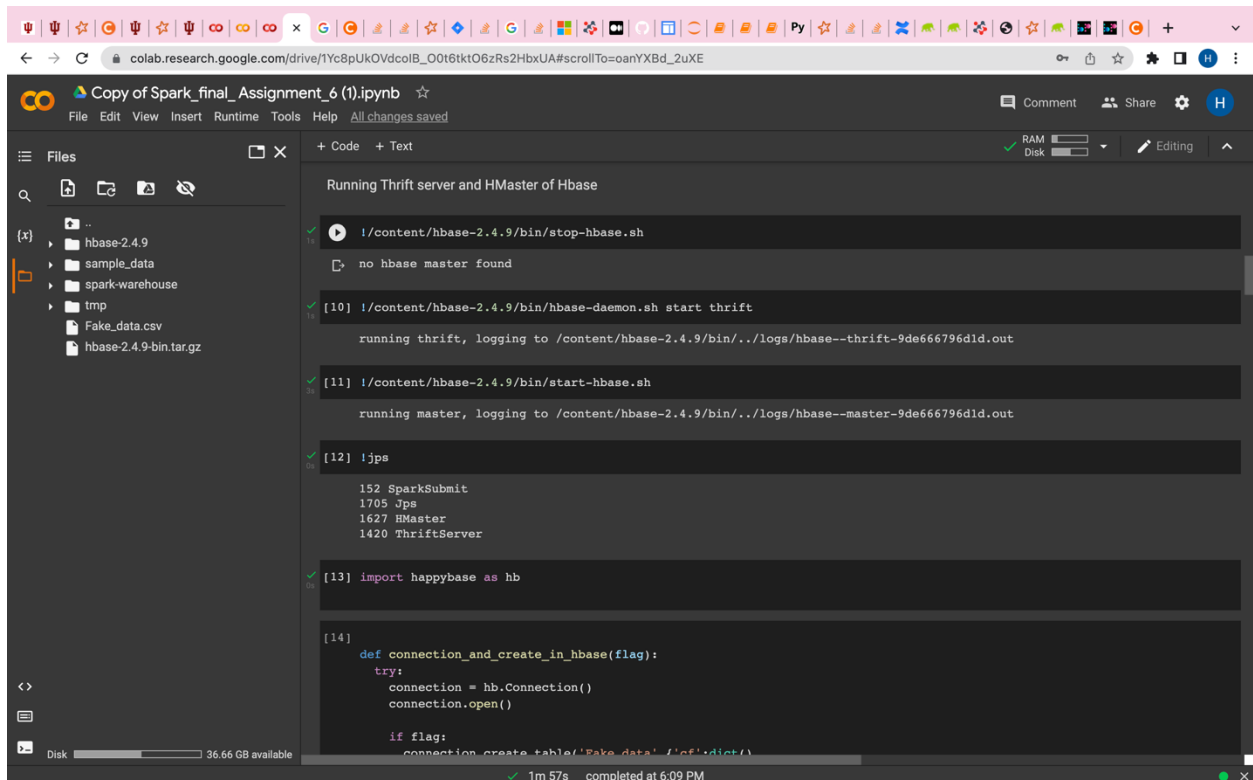
Running Thrift server and HMaster of Hbase

[9] !/content/hbase-2.4.9/bin/stop-hbase.sh

no hbase master found

[10] !/content/hbase-2.4.9/bin/hbase-daemon.sh start thrift
```

Running Thrift server and HMaster of Hbase. Checking the daemons which are running and installing Happybase



```
Running Thrift server and HMaster of Hbase

!./content/hbase-2.4.9/bin/stop-hbase.sh

no hbase master found

[10] !/content/hbase-2.4.9/bin/hbase-daemon.sh start thrift

running thrift, logging to /content/hbase-2.4.9/bin/../logs/hbase--thrift-9de666796d1d.out

[11] !/content/hbase-2.4.9/bin/start-hbase.sh

running master, logging to /content/hbase-2.4.9/bin/../logs/hbase--master-9de666796d1d.out

[12] !jps

152 SparkSubmit
1705 Jps
1627 HMaster
1420 ThriftServer

[13] import happybase as hb

[14] def connection_and_create_in_hbase(flag):
    try:
        connection = hb.Connection()
        connection.open()

        if flag:
            connection.create_table('Fake_data', {'cf':dict()})
```

## Connecting to Hbase and Creating a Table named Fake\_data with column family schema

[illegible]

## Putting the CSV of Fake data into Hbase Table

```
def store_csv_into_hbase():
    try:
        connection = connection_and_create_in_hbase(True)

        table = connection.table('Fake_data')

        csvreader = csv.reader(open('/content/Fake_data.csv'))

        row_count = 0

        for row in csvreader:

            row_count += 1
            if row_count == 1:
                pass
            elif row_count == 10001:
                break
            else:
                table.put(row[0], { "cf:Birth_Country": str(row[1]), "cf:Email": row[2], "cf:First_Name": row[3], "cf:Income"

    except Exception as e:
        print(e)
    finally:
        connection.close()
```

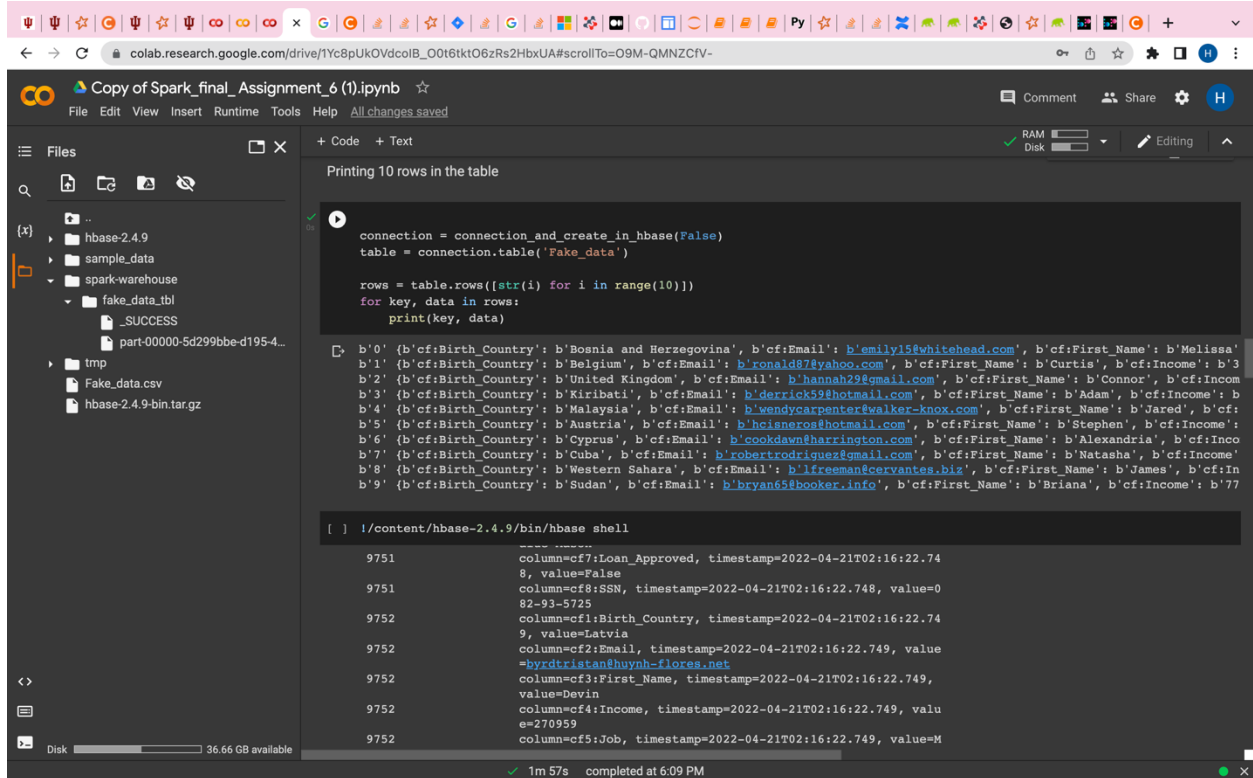
store\_csv\_into\_hbase()

Printing 10 rows in the table

[22]

1m 57s completed at 6:09 PM

Printing a few rows inserted into the Hbase Table using HappyBase and scanning all the rows of that Table through Hbase Shell



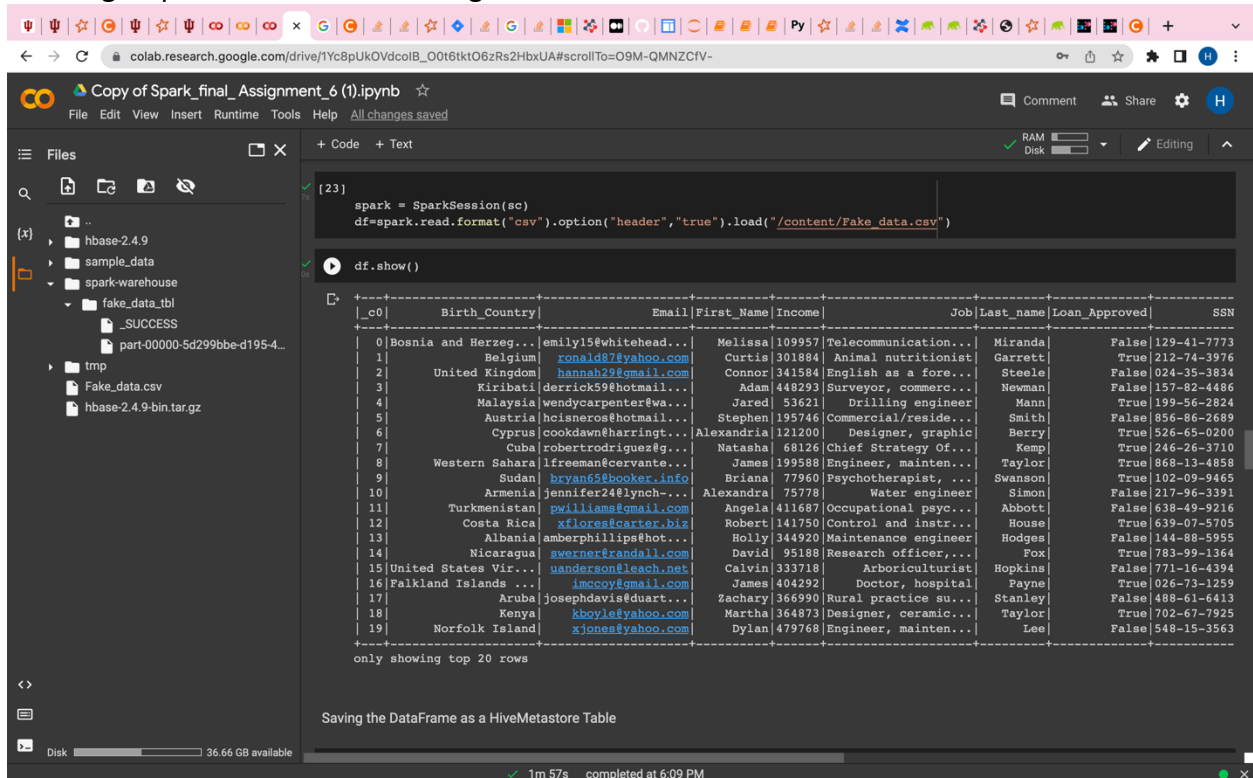
The screenshot shows a Google Colab notebook titled "Copy of Spark\_final\_Assignment\_6 (1).ipynb". The left sidebar displays the file explorer with folders like "hbase-2.4.9", "sample\_data", "spark-warehouse", and "fake\_data\_tbl". The main code area contains the following Python code:

```
connection = connection_and_create_in_hbase(False)
table = connection.table('Fake_data')

rows = table.rows([str(i) for i in range(10)])
for key, data in rows:
    print(key, data)
```

The output shows 10 rows of data, each represented as a tuple of (key, data). The keys are 'b'0' through 'b'9'. The data is a dictionary with fields: 'cf:Birth\_Country', 'cf:Email', 'cf:First\_Name', 'cf:Income', 'cf:Job', 'cf:Last\_Name', 'cf:Loan\_Approved', and 'cf:SSN'. Below the Python code, there is a terminal window showing the output of the command `!content/hbase-2.4.9/bin/hbase shell`. The terminal output shows the HBase shell commands and the resulting data for the first 10 rows.

Creating a Spark context and loading data in a Dataframe



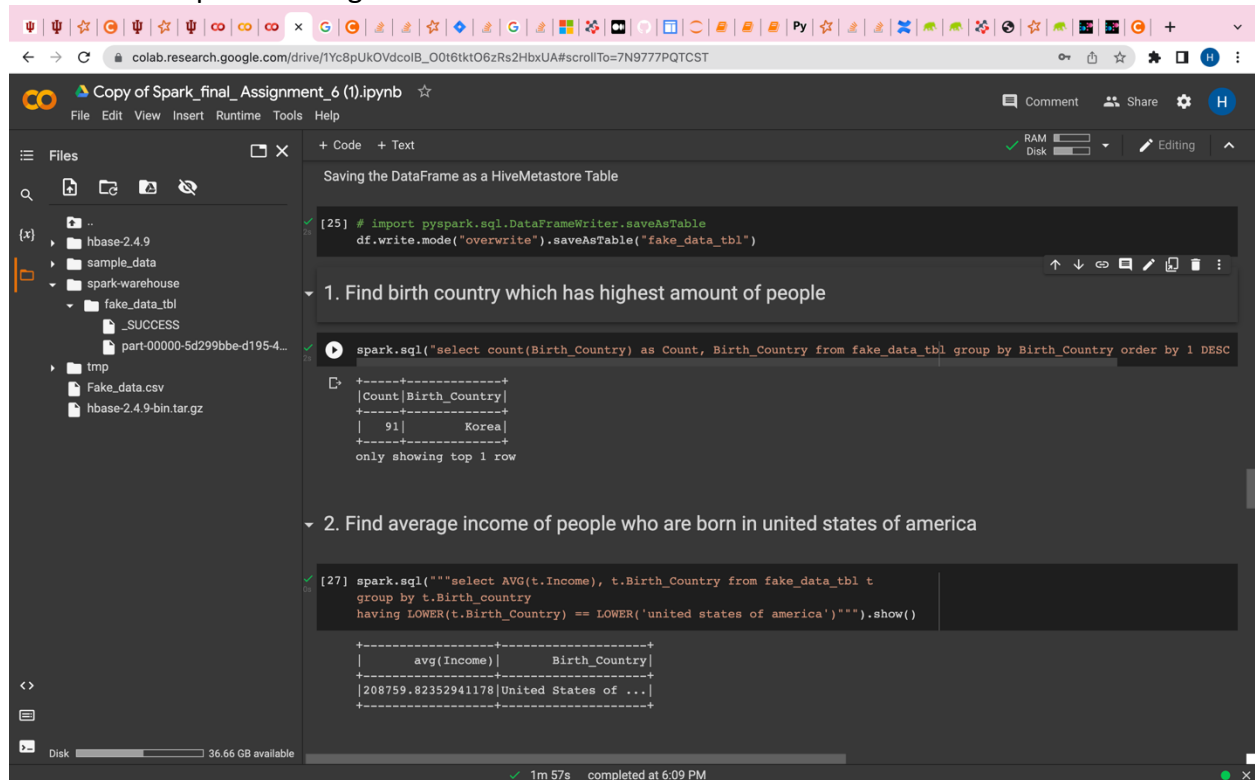
The screenshot shows a Google Colab notebook titled "Copy of Spark\_final\_Assignment\_6 (1).ipynb". The left sidebar displays the file explorer with folders like "hbase-2.4.9", "sample\_data", "spark-warehouse", and "fake\_data\_tbl". The main code area contains the following Python code:

```
spark = SparkSession(sc)
df=spark.read.format("csv").option("header","true").load("/content/Fake_data.csv")

df.show()
```

The output shows a DataFrame with 20 rows of data. The columns are: \_c0, Birth\_Country, Email, First\_Name, Income, Job, Last\_name, Loan\_Approved, and SSN. The data is displayed in a table format. Below the DataFrame, there is a terminal window showing the output of the command `!content/hbase-2.4.9/bin/hbase shell`. The terminal output shows the HBase shell commands and the resulting data for the first 10 rows.

Querying the Hbase Table is difficult so I had to save the dataframe into HiveMetastore Table which can be queried using SQL



The screenshot shows a Google Colab notebook titled "Copy of Spark\_final\_Assignment\_6 (1).ipynb". The left sidebar displays a file explorer with a directory structure including "hbase-2.4.9", "sample\_data", "spark-warehouse", "fake\_data\_tbl", and "tmp". The main code area contains the following steps:

```
[25] # import pyspark.sql.DataFrameWriter.saveAsTable
df.write.mode("overwrite").saveAsTable("fake_data_tbl")
```

1. Find birth country which has highest amount of people

```
spark.sql("select count(Birth_Country) as Count, Birth_Country from fake_data_tbl group by Birth_Country order by 1 DESC")
```

Count	Birth_Country
91	Korea

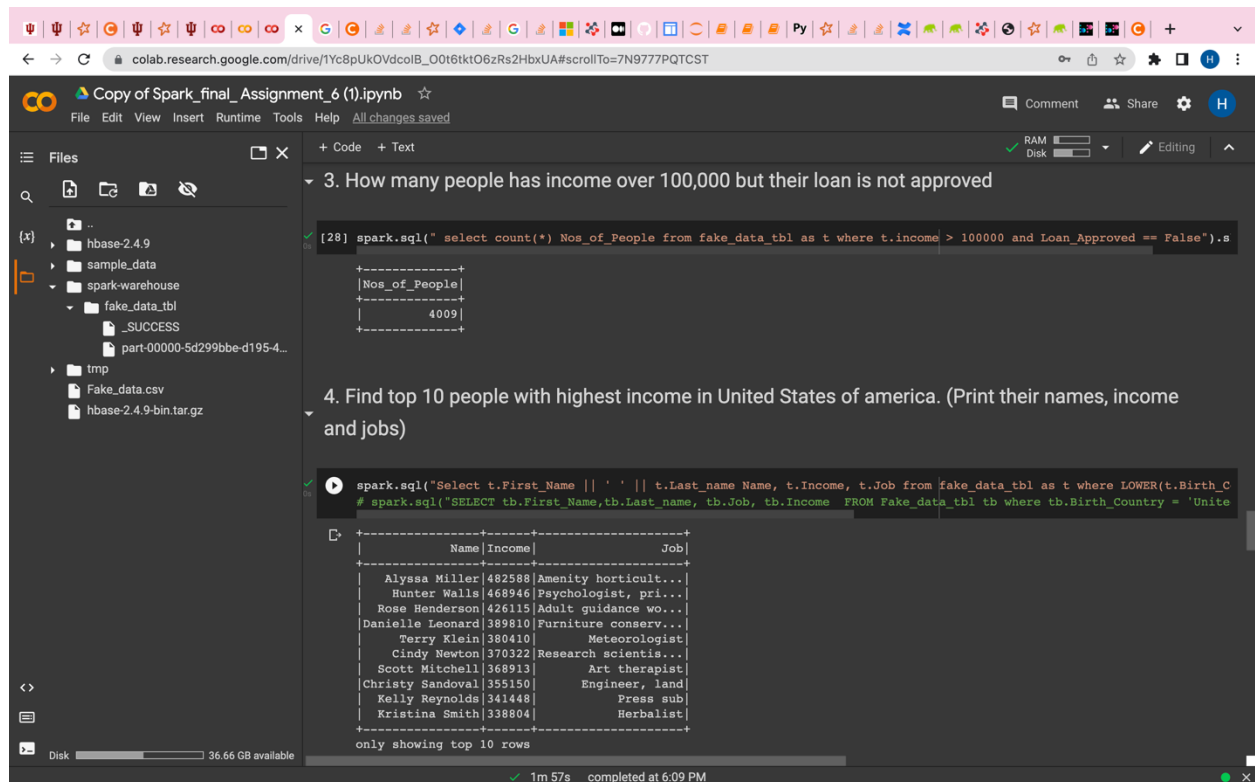
only showing top 1 row

2. Find average income of people who are born in united states of america

```
spark.sql("""select AVG(t.Income), t.Birth_Country from fake_data_tbl t
group by t.Birth_country
having LOWER(t.Birth_Country) == LOWER('united states of america')""").show()
```

avg(Income)	Birth_Country
208759.82352941178	United States of ...

1m 57s completed at 6:09 PM



The screenshot continues the Google Colab notebook with the following steps:

3. How many people has income over 100,000 but their loan is not approved

```
spark.sql(" select count(*) Nos_of_People from fake_data_tbl as t where t.income > 100000 and Loan_Approved == False").s
```

Nos_of_People
4009

4. Find top 10 people with highest income in United States of america. (Print their names, income and jobs)

```
spark.sql("Select t.First_Name || ' ' || t.Last_name Name, t.Income, t.Job from fake_data_tbl as t where LOWER(t.Birth_C
# spark.sql('SELECT tb.First_Name, tb.Last_name, tb.Job, tb.Income FROM Fake_data_tbl tb where tb.Birth_Country = 'Unite
```

Name	Income	Job
Alyssa Miller	482588	Amenity horticult...
Hunter Walls	468946	Psychologist, pri...
Rose Henderson	426115	Adult guidance wo...
Danielle Leonard	389810	Furniture conserv...
Terry Klein	380410	Meteorologist
Cindy Newton	370322	Research scientis...
Scott Mitchell	368913	Art therapist
Christy Sandoval	355150	Engineer, land
Kelly Reynolds	341448	Press sub
Kristina Smith	338804	Herbalist

only showing top 10 rows

1m 57s completed at 6:09 PM

```
[31] spark.sql("SELECT count(*) FROM fake_data_tbl tb ").show()

+-----+
|count(1)|
+-----+
|  10000 |
+-----+

5. How many number of distinct jobs are there?

spark.sql("SELECT count(DISTINCT(tb.Job)) FROM fake_data_tbl tb").show()

+-----+
|count(DISTINCT Job)|
+-----+
|           639 |
+-----+

6. How many writers earn less than 100,000?

[33] spark.sql("SELECT count(*) FROM fake_data_tbl tb where tb.Income <100000 and tb.Job = 'Writer' ").show()

+-----+
|count(1)|
+-----+
|         5 |
+-----+
```

## References:

<https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrameWriter.saveAsTable.html#pyspark.sql.DataFrameWriter.saveAsTable>

<https://spark.apache.org/docs/2.2.0/sql-programming-guide.html#:~:text=Spark%20SQL%20is%20a%20Spark,information%20to%20perform%20extra%20optimizations>

<https://happybase.readthedocs.io/en/latest/user.html>

<https://happybase.readthedocs.io/en/latest/api.html>

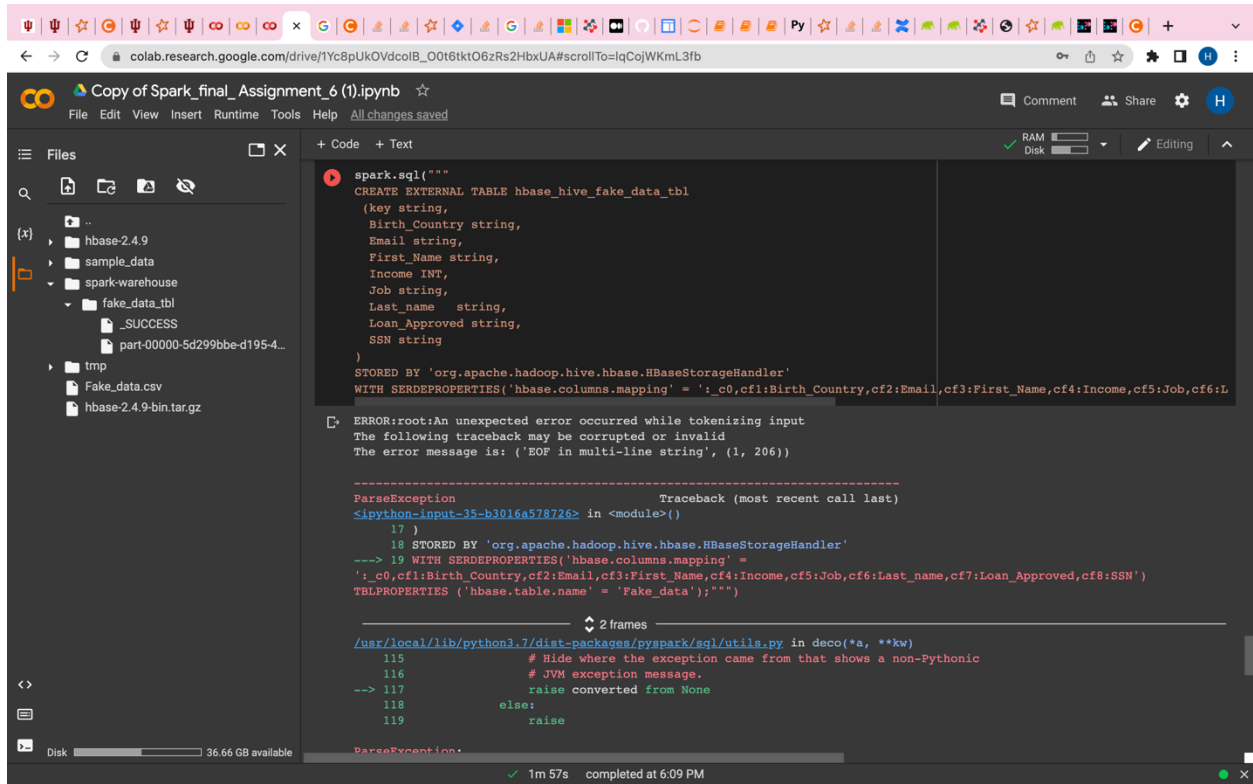
<https://sparkbyexamples.com/apache-hbase-tutorial/>

<https://www.youtube.com/watch?v=CsYaTIUvez0>

<https://www.rittmanmead.com/blog/2015/05/loading-updating-and-deleting-from-hbase-tables-using-hiveql-and-python/>

## Appendix:

I tried creating an External Table in Hive which would be associated with Hbase Table. However, I was facing issues with the StorageHandler class, which I wasn't able to resolve.



The screenshot shows a Google Colab notebook titled "Copy of Spark\_final\_Assignment\_6 (1).ipynb". The left sidebar displays a file explorer with a directory structure including "hbase-2.4.9", "sample\_data", "spark-warehouse", "fake\_data\_tbl", "\_SUCCESS", "part-00000-5d299bbe-d195-4...", "tmp", "Fake\_data.csv", and "hbase-2.4.9-bin.tar.gz". The main area contains a Spark SQL query and a Python traceback error.

```
spark.sql("""
CREATE EXTERNAL TABLE hbase_hive_fake_data_tbl
(
  key string,
  Birth_Country string,
  Email string,
  First_Name string,
  Income INT,
  Job string,
  Last_name string,
  Loan_Approved string,
  SSN string
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES('hbase.columns.mapping' = ':_c0,cf1:Birth_Country,cf2:Email,cf3:First_Name,cf4:Income,cf5:Job,cf6:Last_name,cf7:Loan_Approved,cf8:SSN')
TBLPROPERTIES ('hbase.table.name' = 'Fake_data');""")
```

ERROR:root:An unexpected error occurred while tokenizing input  
The following traceback may be corrupted or invalid  
The error message is: ('EOF in multi-line string', (1, 206))

```
-----
ParseException                                Traceback (most recent call last)
<ipython-input-35-b3016a578726> in <module>()
    17 )
    18 STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
--> 19 WITH SERDEPROPERTIES('hbase.columns.mapping' =
':_c0,cf1:Birth_Country,cf2:Email,cf3:First_Name,cf4:Income,cf5:Job,cf6:Last_name,cf7:Loan_Approved,cf8:SSN')
TBLPROPERTIES ('hbase.table.name' = 'Fake_data');""")

2 frames
/usr/local/lib/python3.7/dist-packages/pyspark/sql/utils.py in deco(*a, **kw)
    115 # Hide where the exception came from that shows a non-Pythonic
    116 # JVM exception message.
--> 117 raise converted from None
    118 else:
    119     raise
ParseException:
```

At the bottom, a status bar indicates "1m 57s completed at 6:09 PM".