Big Data Analytics Project – Project 3
Comparison of Performance between Apache-Hive, Apache-Drill & Spark SQL

Name: Muhammad Ibrahim
ERP: 15976

Machine Configuration

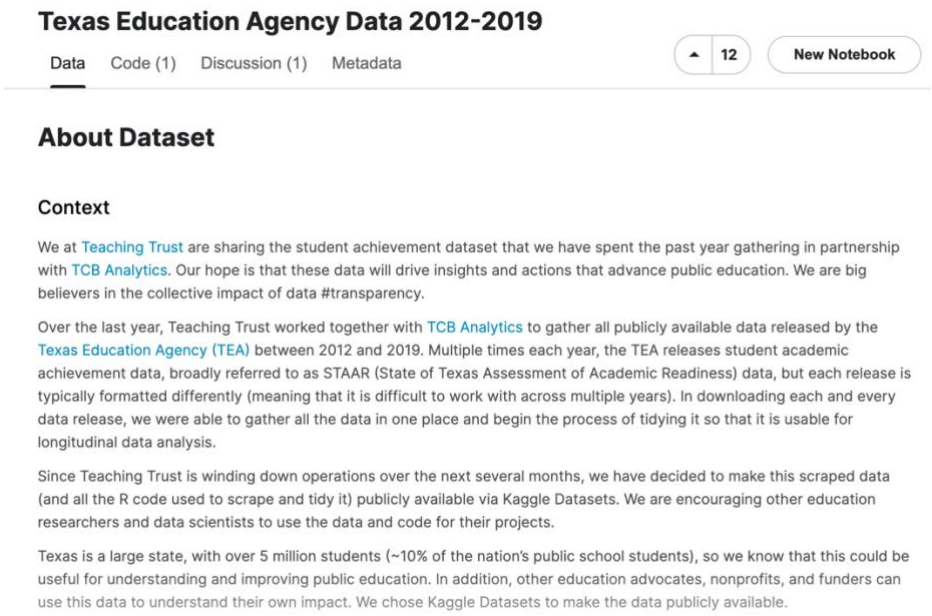
- MacBook Pro 2017
- Core i5
- 256 Flash Drive
- 16 GB RAM

Project Video Link

FInal Video.mov

Data Set Url: https://www.kaggle.com/datasets/emilyparrott/texas-education-agency-data-20122019

Data Set Size: 2.39 GB



Texas Education Agency Data 2012-2019

Data Code (1) Discussion (1) Metadata ▲ 12 New Notebook

About Dataset

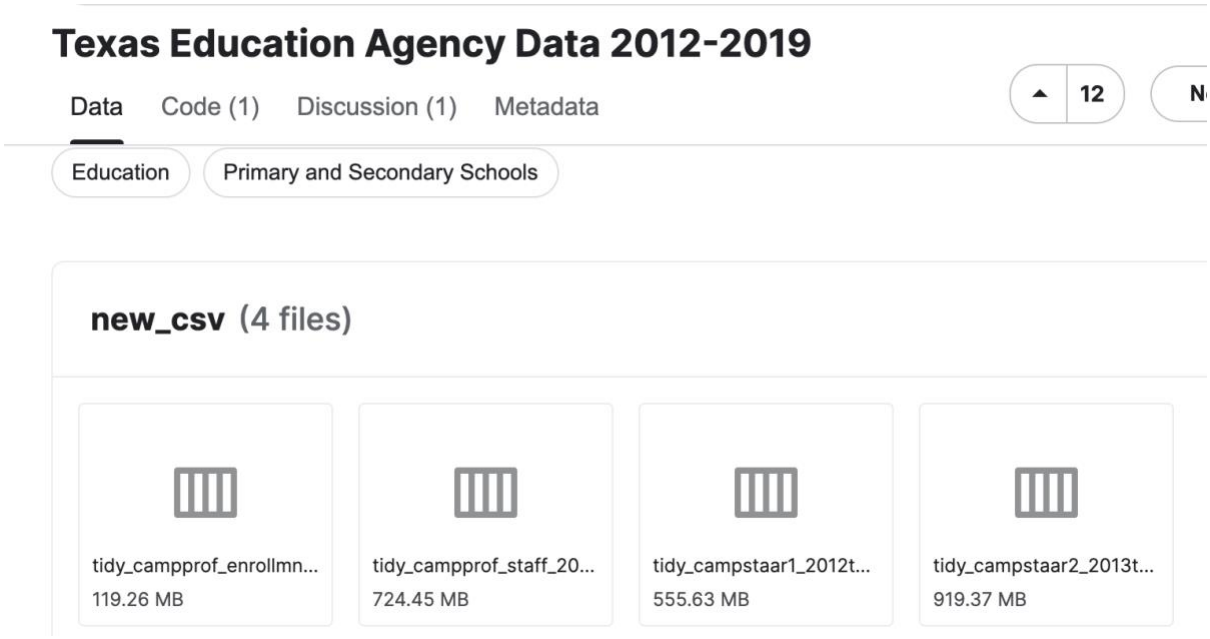
Context

We at Teaching Trust are sharing the student achievement dataset that we have spent the past year gathering in partnership with TCB Analytics. Our hope is that these data will drive insights and actions that advance public education. We are big believers in the collective impact of data #transparency.

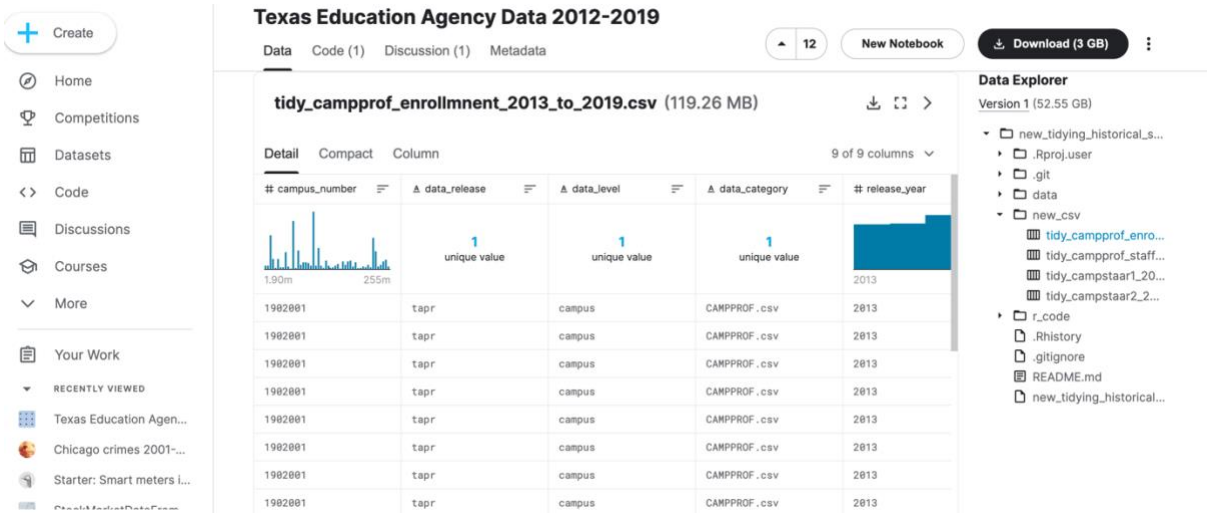
Over the last year, Teaching Trust worked together with TCB Analytics to gather all publicly available data released by the Texas Education Agency (TEA) between 2012 and 2019. Multiple times each year, the TEA releases student academic achievement data, broadly referred to as STAAR (State of Texas Assessment of Academic Readiness) data, but each release is typically formatted differently (meaning that it is difficult to work with across multiple years). In downloading each and every data release, we were able to gather all the data in one place and begin the process of tidying it so that it is usable for longitudinal data analysis.

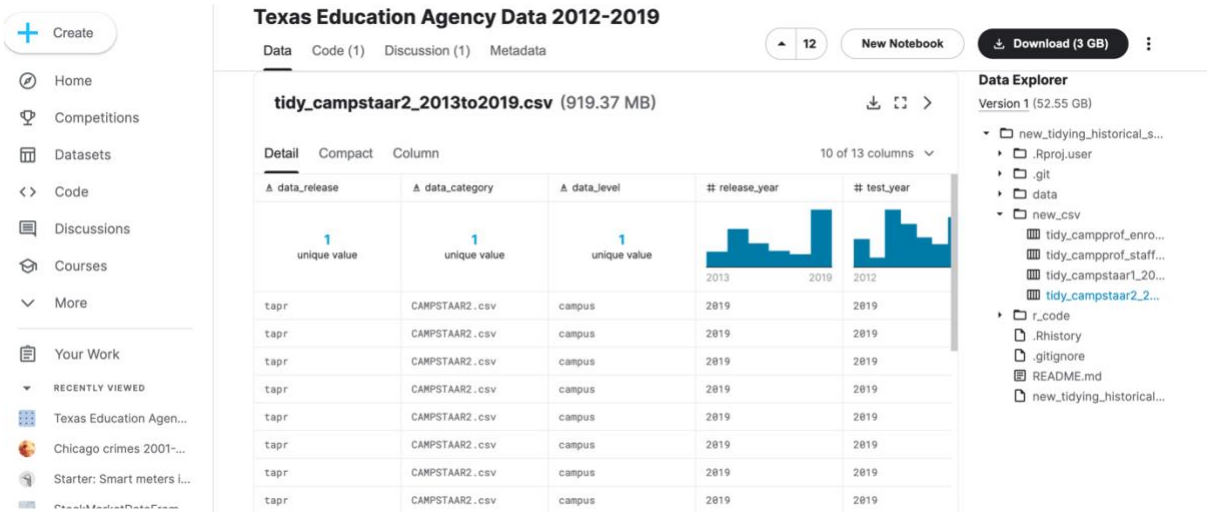
Since Teaching Trust is winding down operations over the next several months, we have decided to make this scraped data (and all the R code used to scrape and tidy it) publicly available via Kaggle Datasets. We are encouraging other education researchers and data scientists to use the data and code for their projects.

Texas is a large state, with over 5 million students (~10% of the nation's public school students), so we know that this could be useful for understanding and improving public education. In addition, other education advocates, nonprofits, and funders can use this data to understand their own impact. We chose Kaggle Datasets to make the data publicly available.



Data Definition i.e. Columns

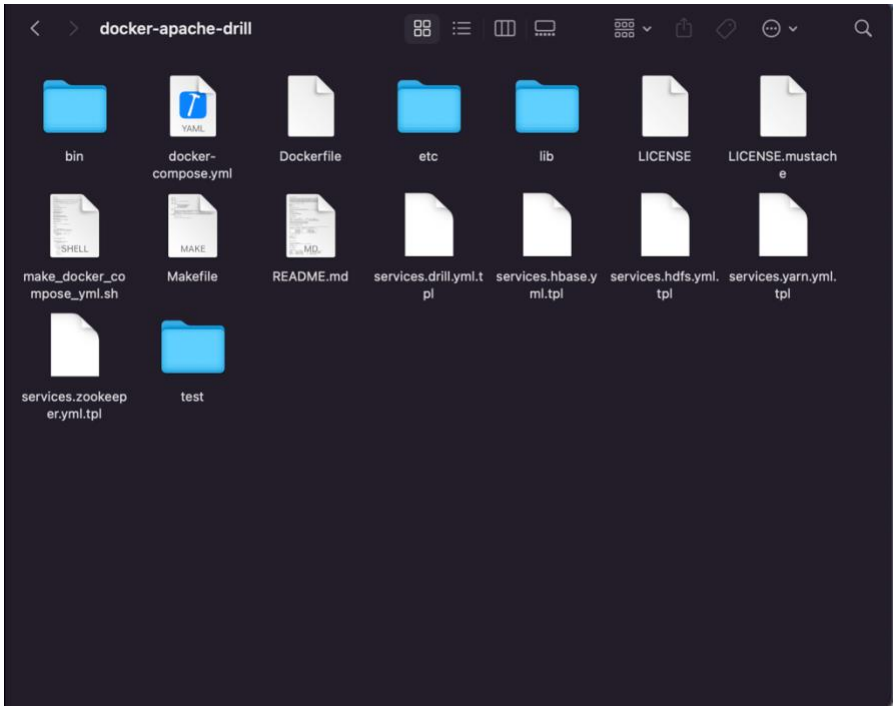




Project Steps Details

1. Apache Drill

- Clone repository and go inside repository directory
  - git clone https://github.com/smizy/docker-apache-drill.git



- Create Virtual network on docker
  - docker network create network
  - The repo includes shell script run it and save results to create docker-compose file
  - ./make_docker_compose.yml sh hdfs drill > docker-compose yml

- Compose File after running shell Script

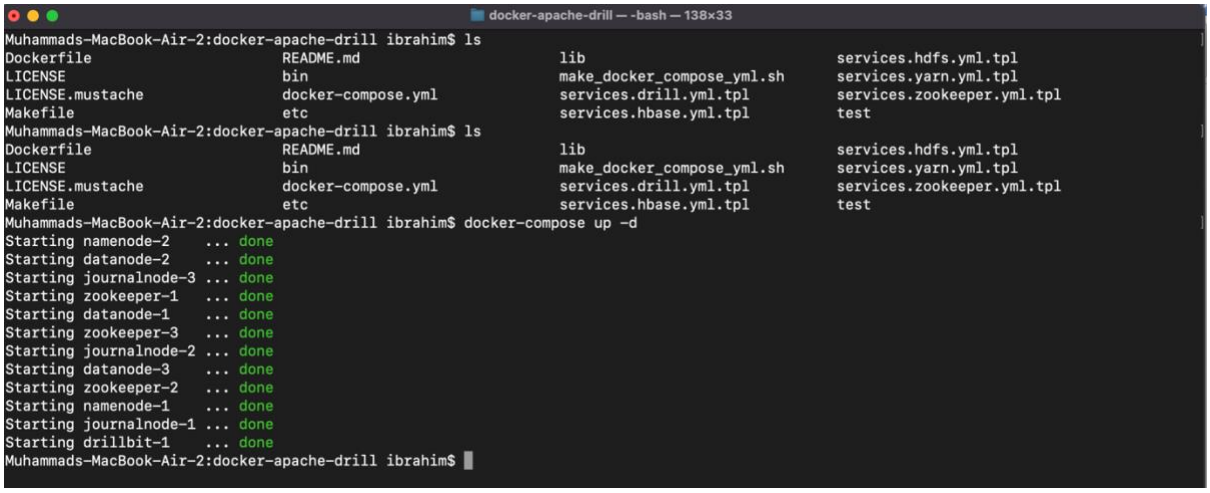
docker-compose.yml

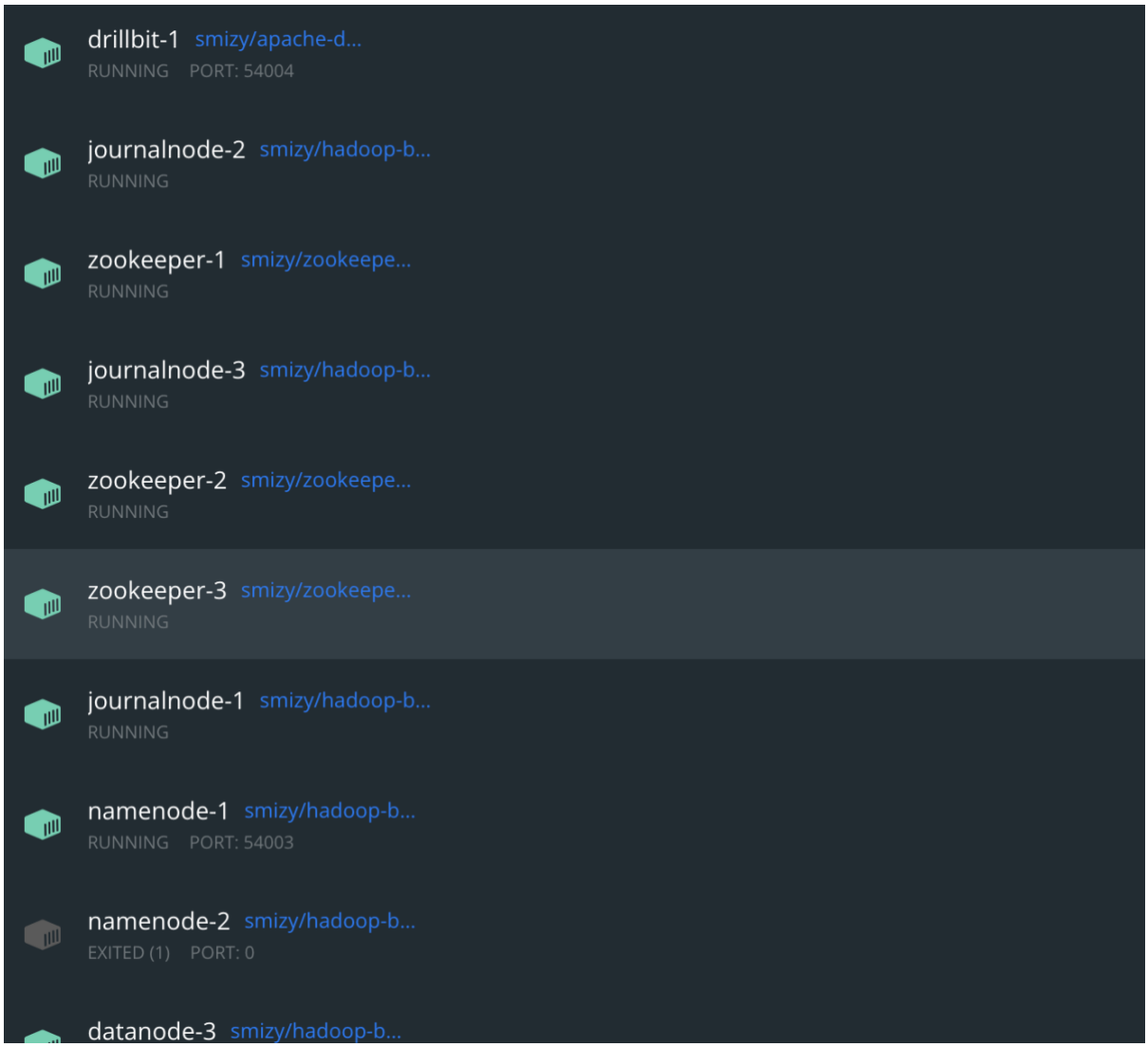
docker-compose

docker-compose 〉 No Selection

```
146   datanode-3:
147     container_name: datanode-3
148     networks: ["vnet"]
149     hostname: datanode-3.vnet
150     image: smizy/hadoop-base:2.7.7-alpine
151     expose: ["50010", "50020", "50075"]
152     environment:
153       - SERVICE_50010_NAME=datanode
154       - SERVICE_50020_IGNORE=true
155       - SERVICE_50075_IGNORE=true
156       -
              HADOOP_ZOOKEEPER_QUORUM=zookeeper-1.vnet:2181,zookeeper-2
              .vnet:2181,zookeeper-3.vnet:2181
157       - HADOOP_HEAPSIZE=1000
158
159     entrypoint: entrypoint.sh
160     command: datanode
161
162   drillbit-1:
163     container_name: drillbit-1
164     networks: ["vnet"]
165     hostname: drillbit-1.vnet
166     image: smizy/apache-drill:1.16.0-alpine
167     ports:
168       - 8047
169     depends_on: ["zookeeper-1"]
170     environment:
171       - SERVICE_8047_NAME=drillbit
172       - DRILL_HEAP=512M
173       - DRILL_MAX_DIRECT_MEMORY=1G
174       -
              DRILL_ZOOKEEPER_QUORUM=zookeeper-1.vnet:2181,zookeeper-2
              .vnet:2181,zookeeper-3.vnet:2181
175
176
```

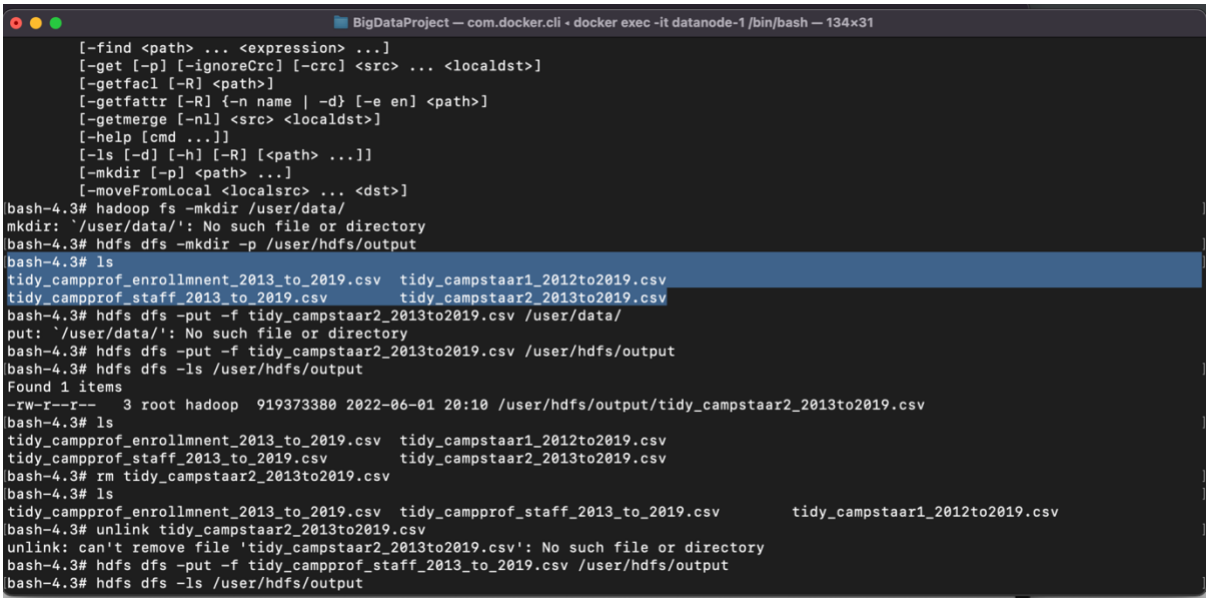
- Now run docker-compose up -d to download and run container for name-node, data-node and Apache Drill on given ports


docker-apache-drill — -bash — 138×33
Muhammads-MacBook-Air-2:docker-apache-drill ibrahim$ ls
Dockerfile README.md lib services.hdfs.yml.tpl
LICENSE bin make_docker_compose_yml.sh services.yarn.yml.tpl
LICENSE.mustache docker-compose.yml services.drill.yml.tpl services.zookeeper.yml.tpl
Makefile etc services.hbase.yml.tpl test
Muhammads-MacBook-Air-2:docker-apache-drill ibrahim$ ls
Dockerfile README.md lib services.hdfs.yml.tpl
LICENSE bin make_docker_compose_yml.sh services.yarn.yml.tpl
LICENSE.mustache docker-compose.yml services.drill.yml.tpl services.zookeeper.yml.tpl
Makefile etc services.hbase.yml.tpl test
Muhammads-MacBook-Air-2:docker-apache-drill ibrahim$ docker-compose up -d
Starting namenode-2 ... done
Starting datanode-2 ... done
Starting journalnode-3 ... done
Starting zookeeper-1 ... done
Starting datanode-1 ... done
Starting zookeeper-3 ... done
Starting journalnode-2 ... done
Starting datanode-3 ... done
Starting zookeeper-2 ... done
Starting namenode-1 ... done
Starting journalnode-1 ... done
Starting drillbit-1 ... done
Muhammads-MacBook-Air-2:docker-apache-drill ibrahim$


drillbit-1 smizy/apache-d...
RUNNING PORT: 54004
journalnode-2 smizy/hadoop-b...
RUNNING
zookeeper-1 smizy/zookeepe...
RUNNING
journalnode-3 smizy/hadoop-b...
RUNNING
zookeeper-2 smizy/zookeepe...
RUNNING
zookeeper-3 smizy/zookeepe...
RUNNING
journalnode-1 smizy/hadoop-b...
RUNNING
namenode-1 smizy/hadoop-b...
RUNNING PORT: 54003
namenode-2 smizy/hadoop-b...
EXITED (1) PORT: 0
datanode-3 smizy/hadoop-b...

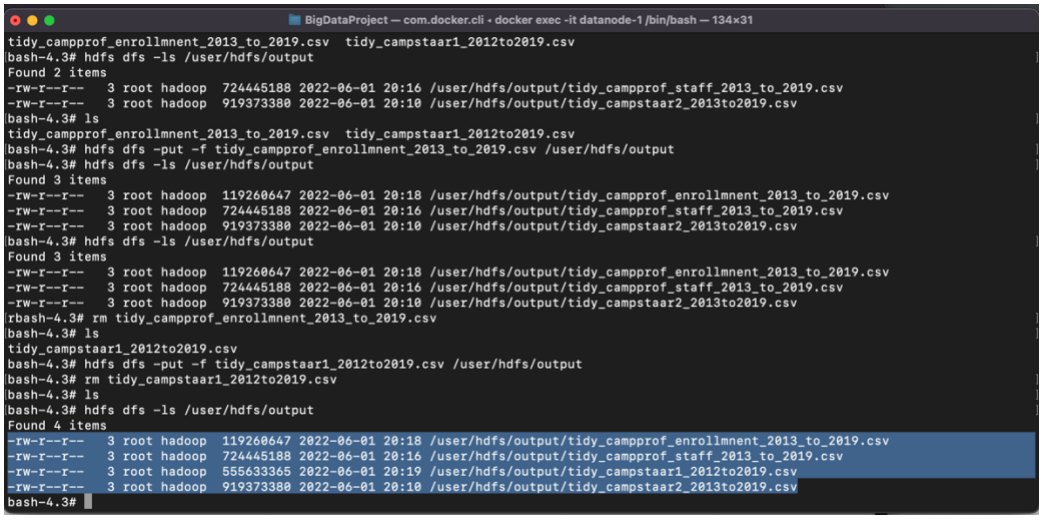
- Copy Data into datanode container using
  - docker cp tidy_campstaar2_2013to2019.csv datanode-1:/home/data
  - docker cp tidy_campstaar1_2012to2019.csv datanode-1:/share
  - docker cp tidy_campstaar2_2013to2019.csv spark-master:/home
  - docker cp tidy_campprof_enrollmnent_2013_to_2019.csv spark-master:/home

Data in Container



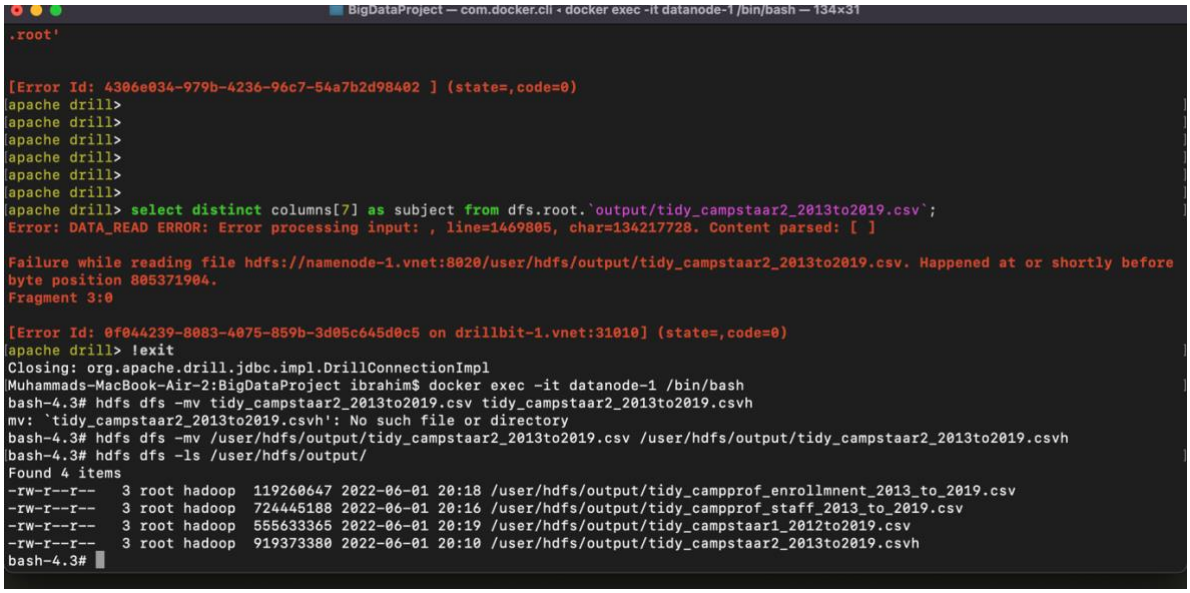
- Copy Data in HDFS using
  - hdfs dfs -put -f tidy_campprof_enrollmnent_2013_to_2019.csv /user/hdfs/output
  - hdfs dfs -put -f tidy_campstaar1_2012to2019.csv /user/hdfs/output
  - hdfs dfs -put -f tidy_campprof_staff_2013_to_2019.csv /user/hdfs/output
  - hdfs dfs -put -f tidy_campstaar2_2013to2019.csv /user/hdfs/output

Confirm Data transfer using ls command



Apache drill is capable of querying text and CSV files, so I will query directly from CSV, changing extension to CSVH will allow us to use CSV Column names directly otherwise we have to use Column Index i.e. columns[2]

- Change CSV extension to CSVH using
  - hdfs dfs mv tidy_campstaar2_2013to2019.csv tidy_campstaar2_2013to2019.csvh



Change Apache Drill configuration to read data from HDFS root directory, open **Error! Hyperlink reference not valid.** i.e. http://192.168.1.5:54004/storage/dfs


```
{
 "type": "file",
 "enabled": true,
 "connection": "hdfs://namenode-1.vnet:8020/",
 "config": null,
 "workspaces": {
  "root": {
   "location": "/user/hdfs",
   "writable": false,
   "defaultInputFormat": null
  },
  "tmp": {
   "location": "/tmp",
   "writable": true,
   "defaultInputFormat": null
  }
 }
```


- Open Drill Command Line using

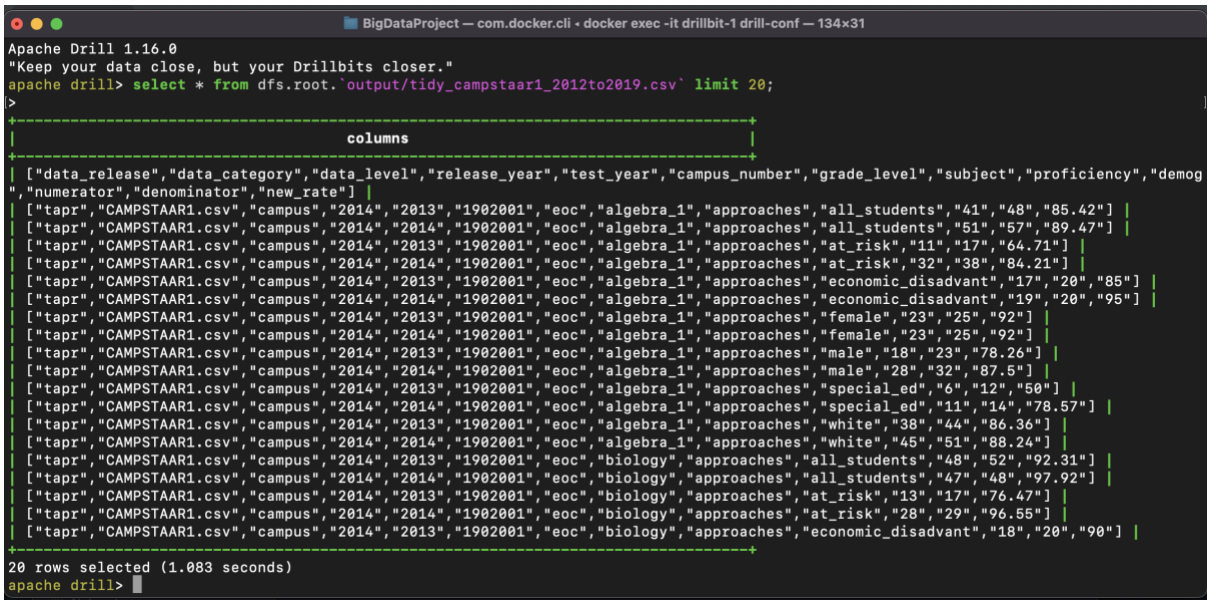
- docker exec -it drillbit-1 drill-conf

```
Muhammads-MacBook-Air-2:docker-apache-drill ibrahim$ docker exec -it drillbit-1 drill-conf
Apache Drill 1.16.0
"Say hello to my little Drill."
apache drill>
```

Queries in Apache Drill

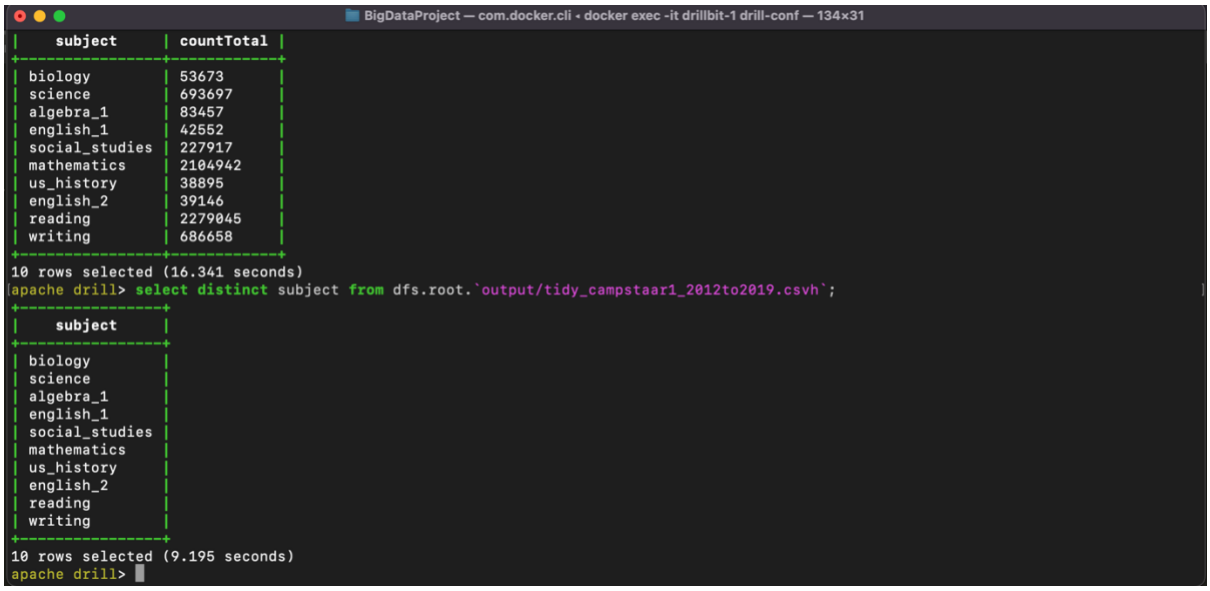
-> Query Data in drill using csv in HDFS root path

select * **from** dfs.root.`output/tidy_campstaar1_2012to2019.csv`;



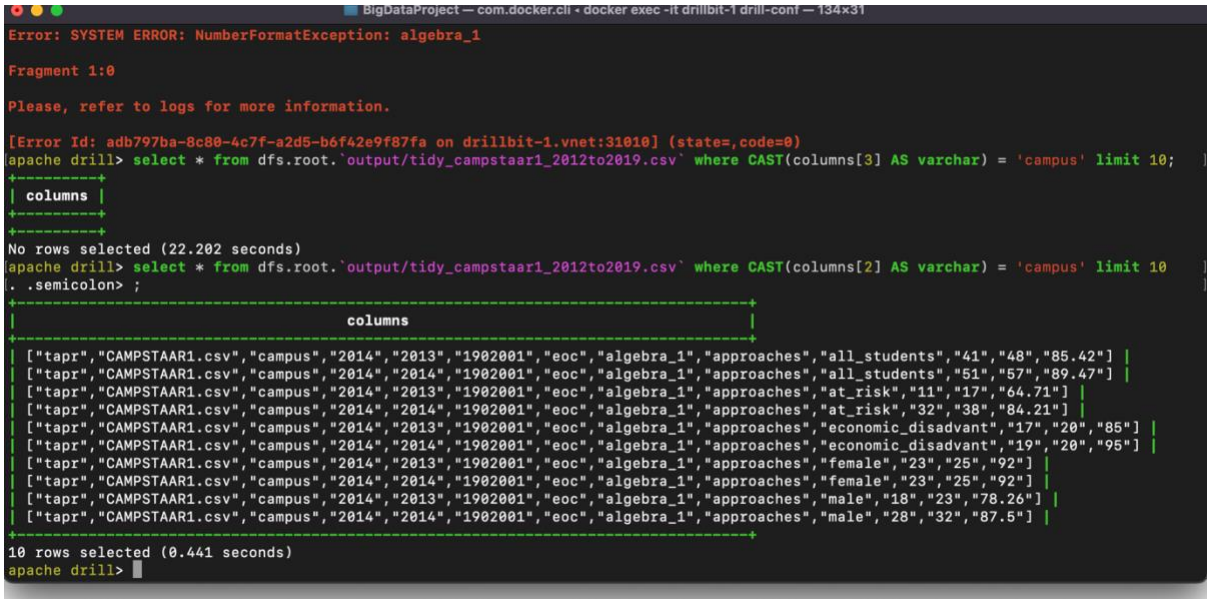
Distinct Query:

select distinct subject from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`;



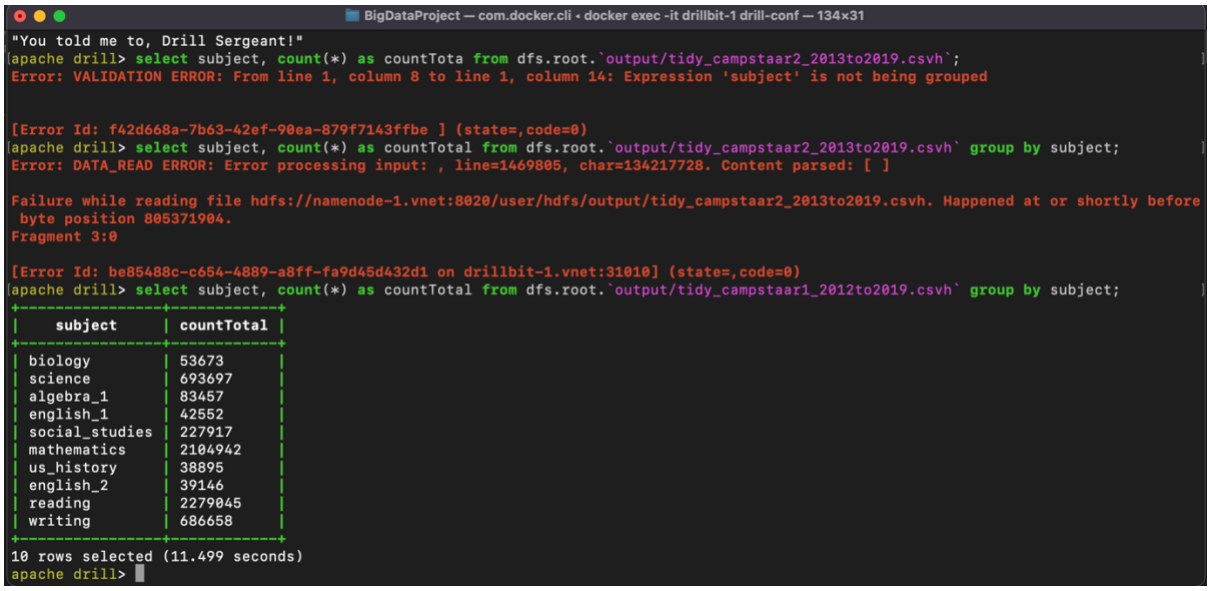
Where Clause:

select * from dfs.root.`output/tidy_campstaar1_2012to2019.csv` where CAST(columns[2] AS varchar) = 'campus' limit 10



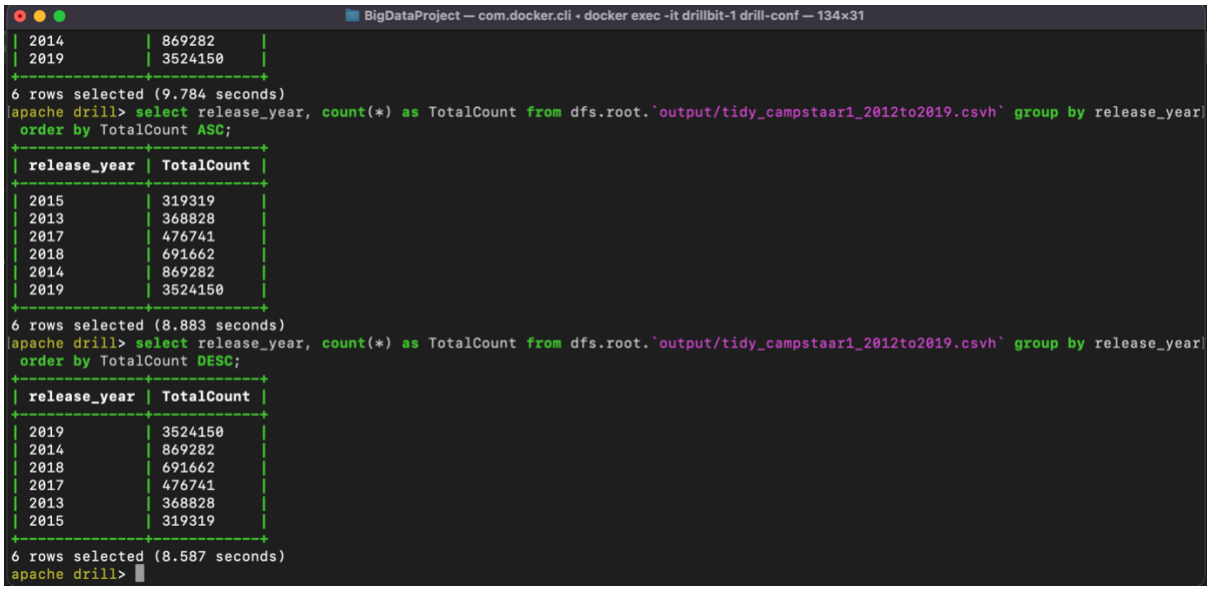
Count, Group by subjects

select subject, count(*) as countTotal from dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by subject;



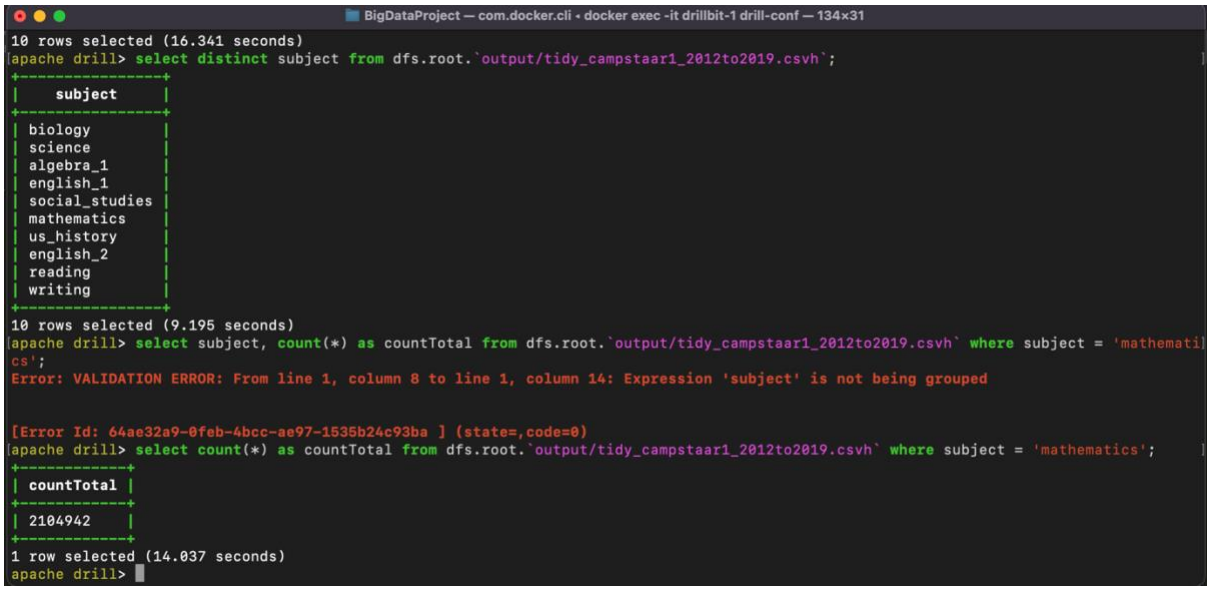
Order by DESC query – Year Wise Data

select release_year, count(*) as TotalCount from dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by release_year order by TotalCount DESC;



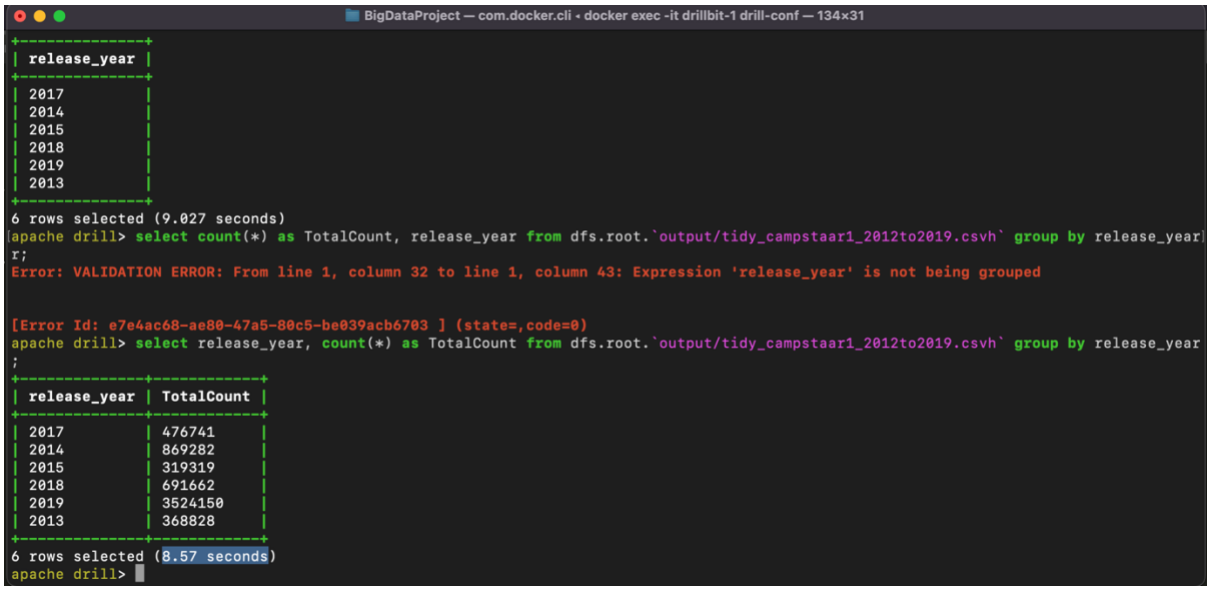
Total Star Students Learning Mathematics -- Count, Where Mathematics, Result -> 2104942

select count(*) as countTotal from dfs.root.`output/tidy_campstaar1_2012to2019.csvh` where subject = 'mathematics'



Check which years data we have

select distinct release_year from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`;



Following Queries were executed on Apache Drill to note performance

1. Distinct Subject

```
select distinct subject from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh`;
```

9.195 seconds

2. Count, Group by subjects

```
select subject, count(*) as countTotal from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by subject;
```

11.499 seconds

3. Total Star Students Learning Mathematics -- Count, Where Mathematics, Result -> 2104942

```
select count(*) as countTotal from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh` where subject = 'mathematics'
```

14.037 seconds

4. Check which years data we have

```
select distinct release_year from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh`;
```

9.027 seconds

5. Total Count of data year wise

```
select release_year, count(*) as TotalCount from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by release_year;
```

8.57 seconds

6. which year have most data

```
select release_year, count(*) as TotalCount from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by release_year order by
TotalCount DESC;
```

8.587 seconds

7. Look at demographics of Star students

```
select distinct demog from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh`;
```

7.234 seconds

8. Count female students -> 672545

```
select count(*) from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`
where demog = 'female';
```

6.655 seconds

9. Count female students learning maths -> 221017

```
select count(*) from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`
where demog = 'female' and subject = 'mathematics';
```

8.72 seconds

10. Count of male students

```
select count(*) from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`
where demog = 'male';
```

6.929 seconds

11. Count of male learning biology -> 6403

```
select count(*) from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`
where demog = 'male' and subject = 'biology'
```

9.033 seconds

12. Count of asian students in year 2019 -> 63611

```
select count(*) from dfs.root.`output/tidy_campstaar1_2012to2019.csvh`
where demog = 'asian' and release_year = '2019';
```

8.036 seconds

13. select which subject is mostly learned

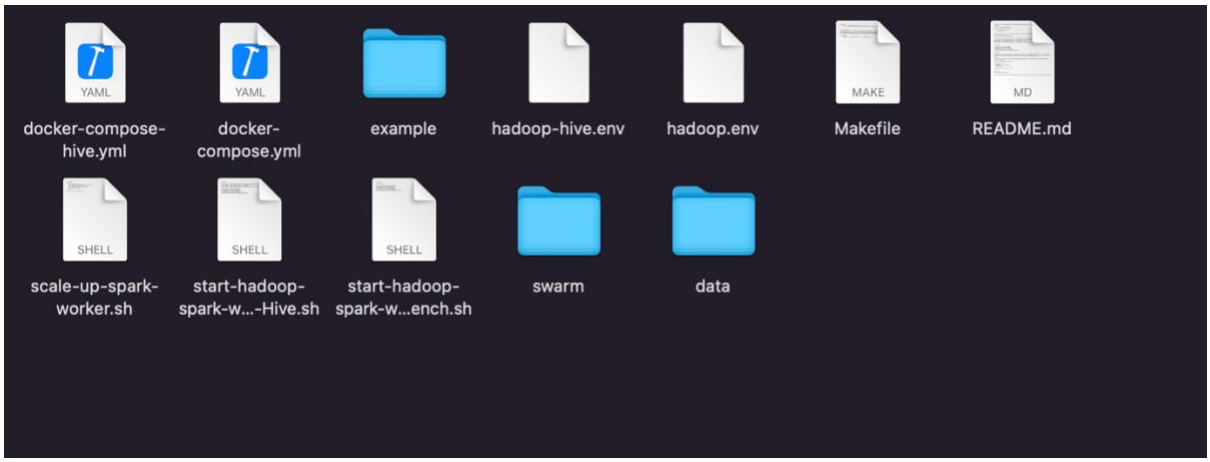
```
select subject, count(*) as TotalCount from
dfs.root.`output/tidy_campstaar1_2012to2019.csvh` group by subject order by
TotalCount DESC
```

7.505 seconds

Project Steps Details

Apache Spark & Hive

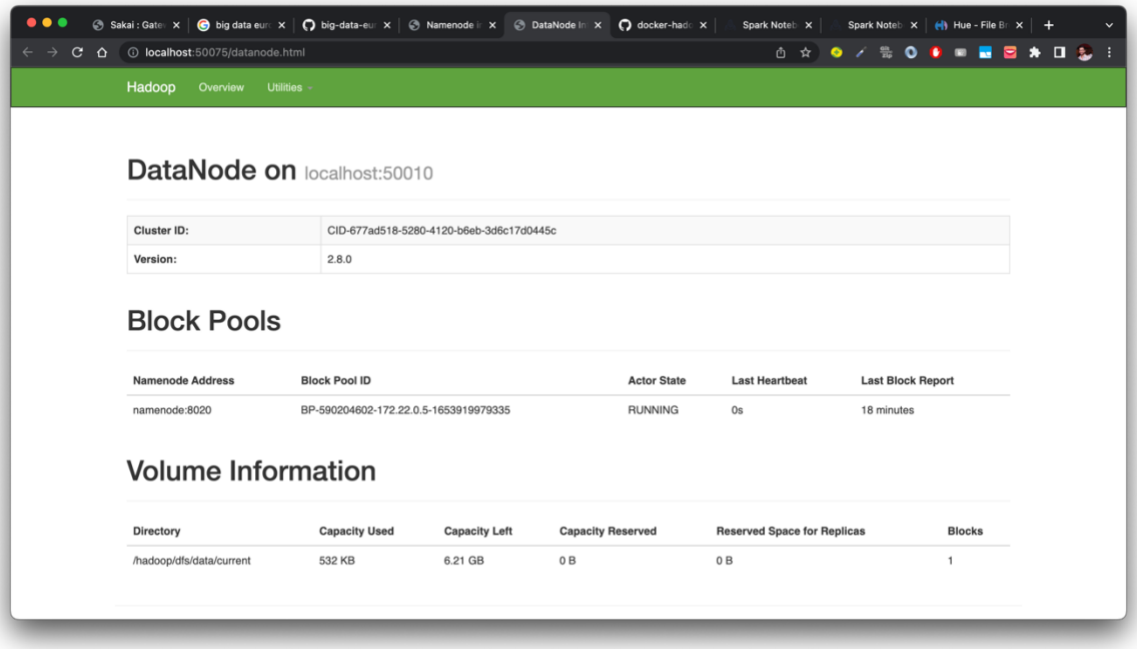
- Clone repository and go inside repository directory
- It includes Spark and Hive
    - git clone https://github.com/big-data-europe/docker-hadoop-spark-workbench.git



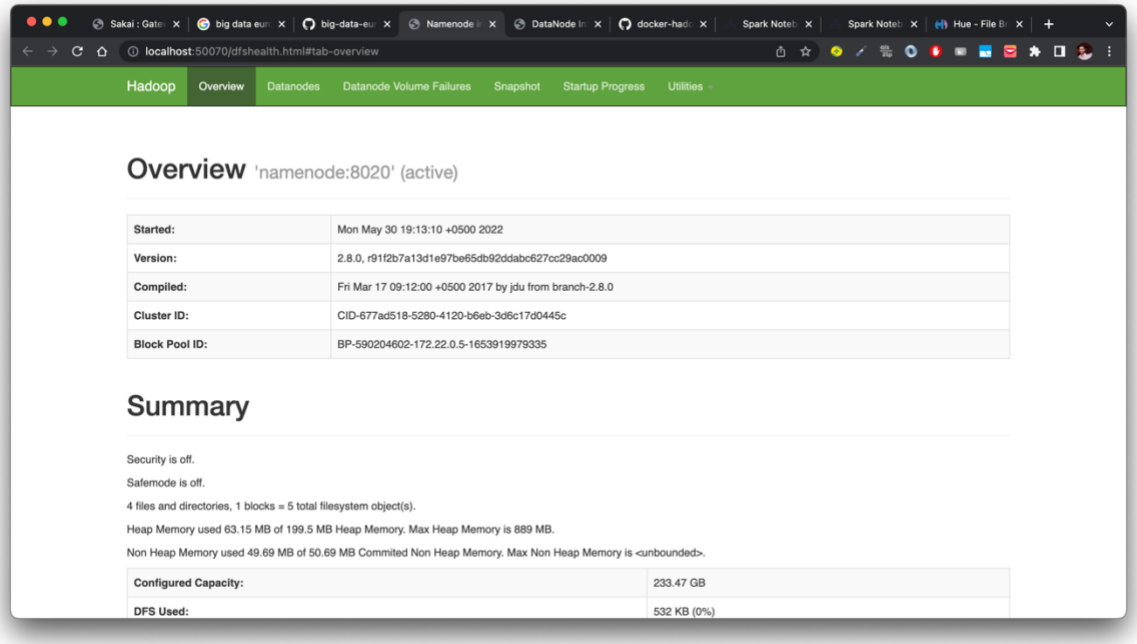
- Run Following one by one
    - docker-compose up -d
    - docker-compose -f docker-compose-hive.yml up -d namenode hive-metastore-postgresql
    - docker-compose -f docker-compose-hive.yml up -d datanode hive-metastore
    - docker-compose -f docker-compose-hive.yml up -d hive-server
    - docker-compose -f docker-compose-hive.yml up -d spark-master spark-worker spark-notebook hue

```
docker-hadoop-spark-workbench — -bash — 158×33
4654ee5f6659: Pull complete
88a820555dbf: Pull complete
14ead1715153: Pull complete
df32fa293aee: Pull complete
f0c9fa3e4fa9: Pull complete
ce0284e53103: Pull complete
cbdb8c564fcc: Pull complete
becd891f4d83: Pull complete
4305be6507a2: Pull complete
cd02abe11589: Pull complete
55b03a36febf: Pull complete
503ca7cfdec1: Pull complete
077f5d65a2f0: Pull complete
16b679fd89e6: Pull complete
787b2d43064e: Pull complete
7decc2d090fb: Pull complete
914d86a00eba: Pull complete
a8ec9c576cb2: Pull complete
4714408faa44: Pull complete
ec27fa202f55: Pull complete
5687b41783e7: Pull complete
538fcca1f5af: Pull complete
8b1a2bff446c: Pull complete
cf63e94ebc3b: Pull complete
Digest: sha256:95b2b8b2e622b474851cb362934592fc22e520f608e891f9aa4be45032b23ccd
Status: Downloaded newer image for bde2020/hdfs-filebrowser:3.11
Creating spark-notebook                    ... done
Creating spark-master                      ... done
Creating docker-hadoop-spark-workbench_hue_1 ... done
Creating namenode                          ... done
Creating docker-hadoop-spark-workbench_datanode_1     ... done
Creating docker-hadoop-spark-workbench_spark-worker_1 ... done
172-15-3-86:docker-hadoop-spark-workbench ibrahim$
```

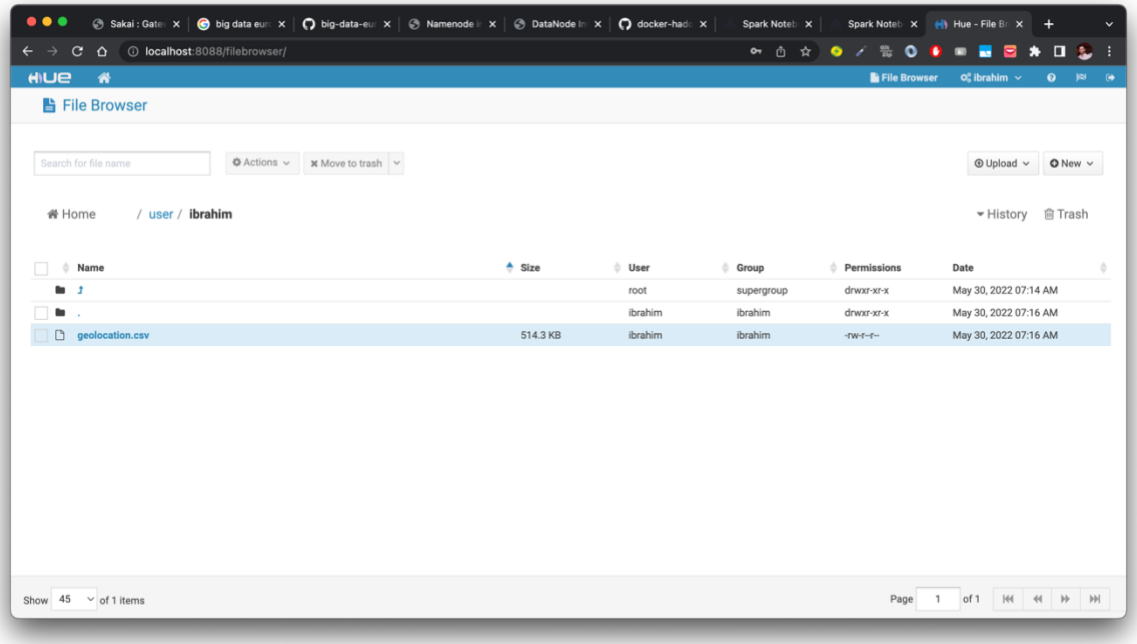
Data Node for Spark & Hive



Name Node

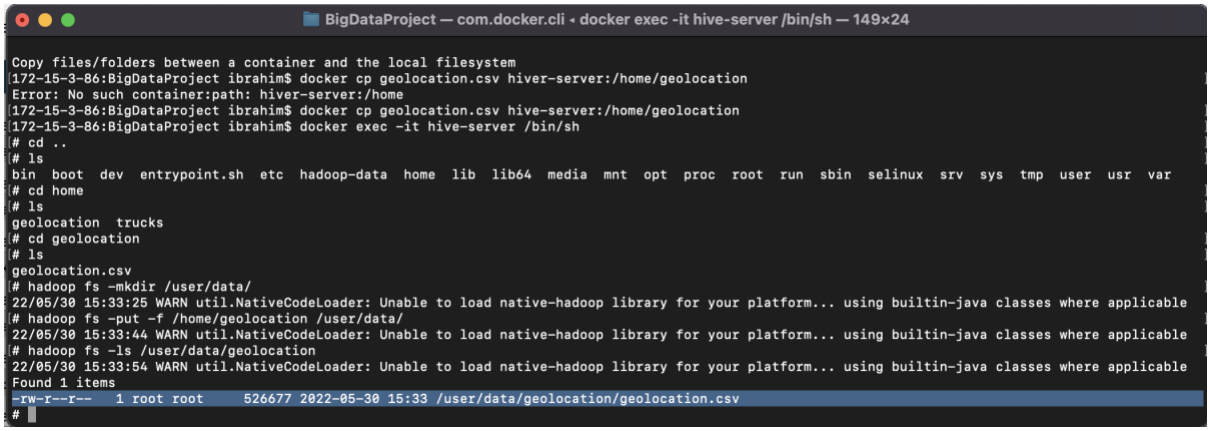


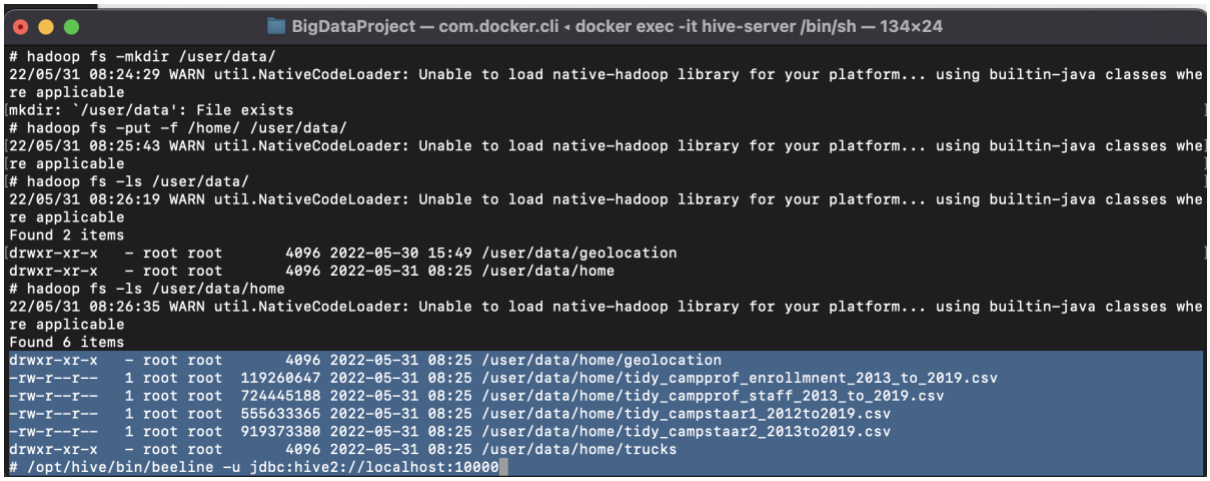
HDFS Browser

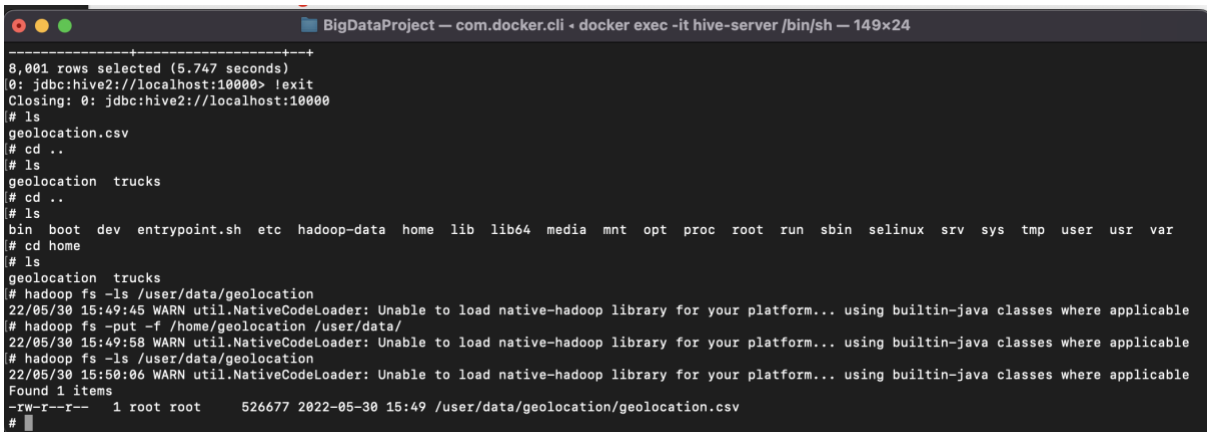


Copy Data Set to Container and then HDFS for Hive & Spark

docker cp tidy_campstaar2_2013to2019.csv spark-master:/home
docker cp tidy_campstaar2_2013to2019.csv hive-server:/home



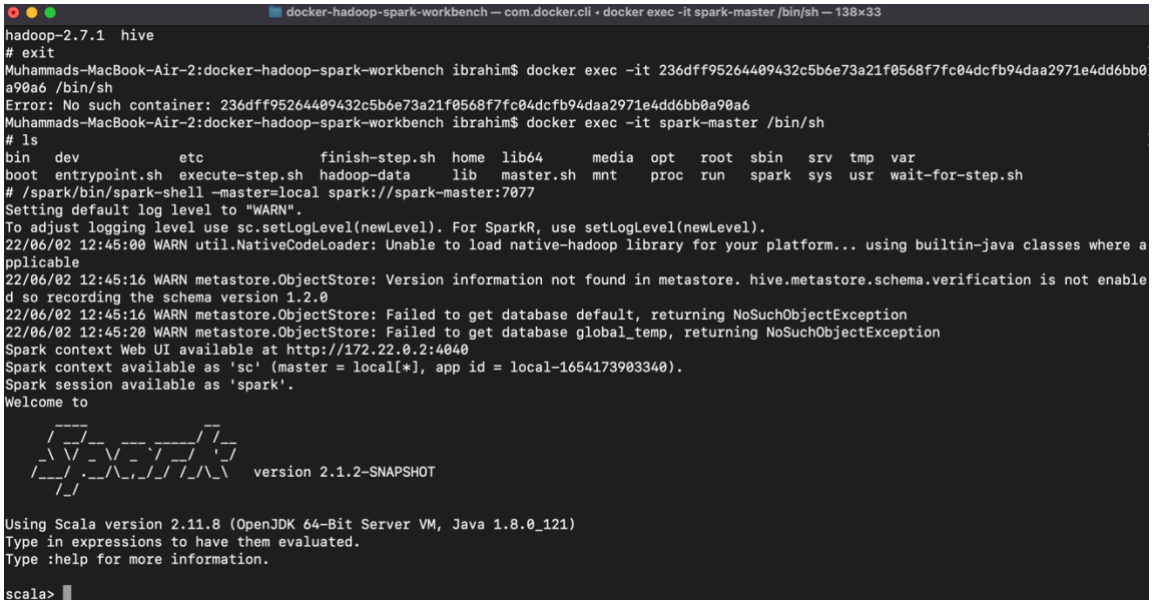


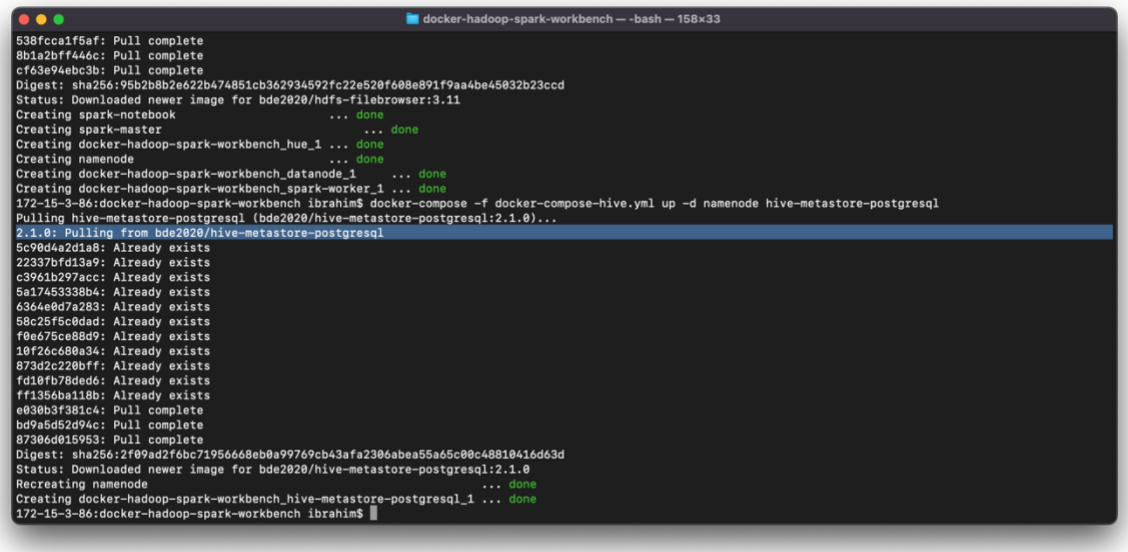


Start Spark & Hive

- docker exec -it hive-server /bin/sh
  - /opt/hive/bin/beeline -u jdbc:hive2://localhost:10000

- docker exec -it spark-master /bin/sh
  - /spark/bin/spark-shell –master=local spark://spark-master:7077





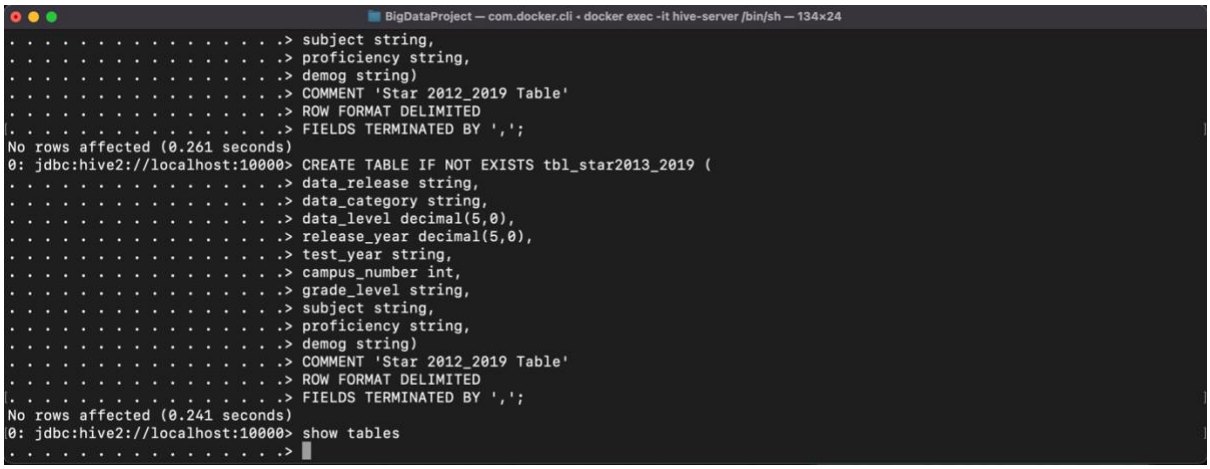
Running Queries on Hive

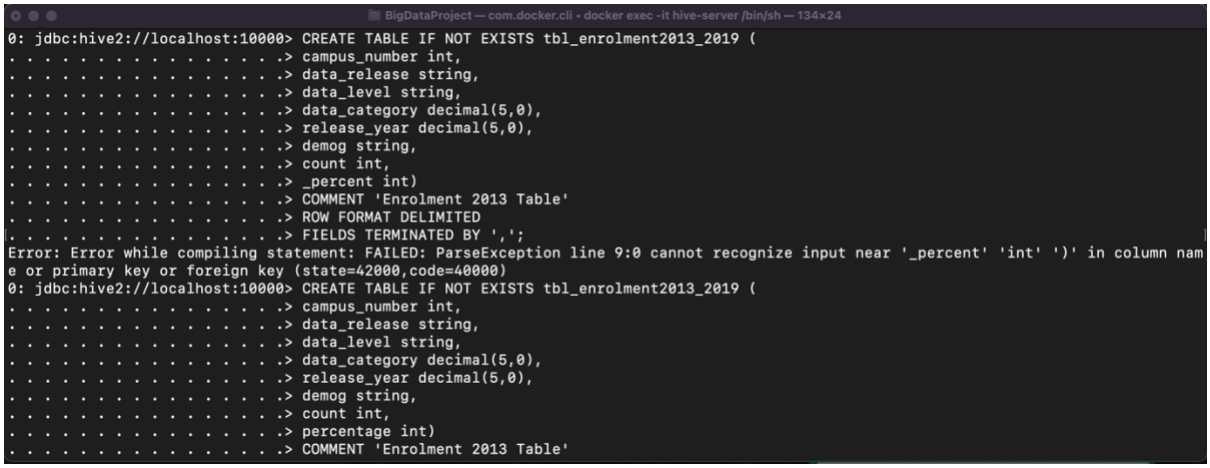
Creating Tables by reading CSV from HDFS

LOAD DATA INPATH '/user/data/home/tidy_campprof_enrollmnent_2013_to_2019.csv' INTO TABLE tbl_enrolment2013_2019;

LOAD DATA INPATH '/user/data/home/tidy_campstaar1_2012to2019.csv' INTO TABLE tbl_star2012_2019;

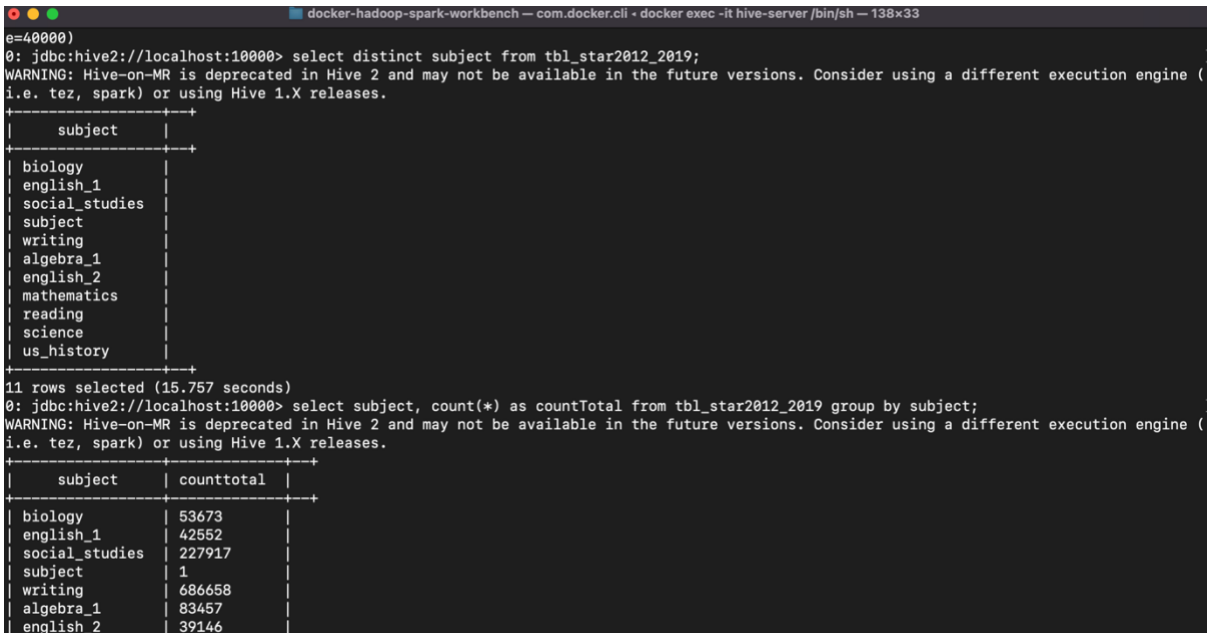
LOAD DATA INPATH '/user/data/home/tidy_campstaar2_2013to2019.csv' INTO TABLE tbl_star2013_2019





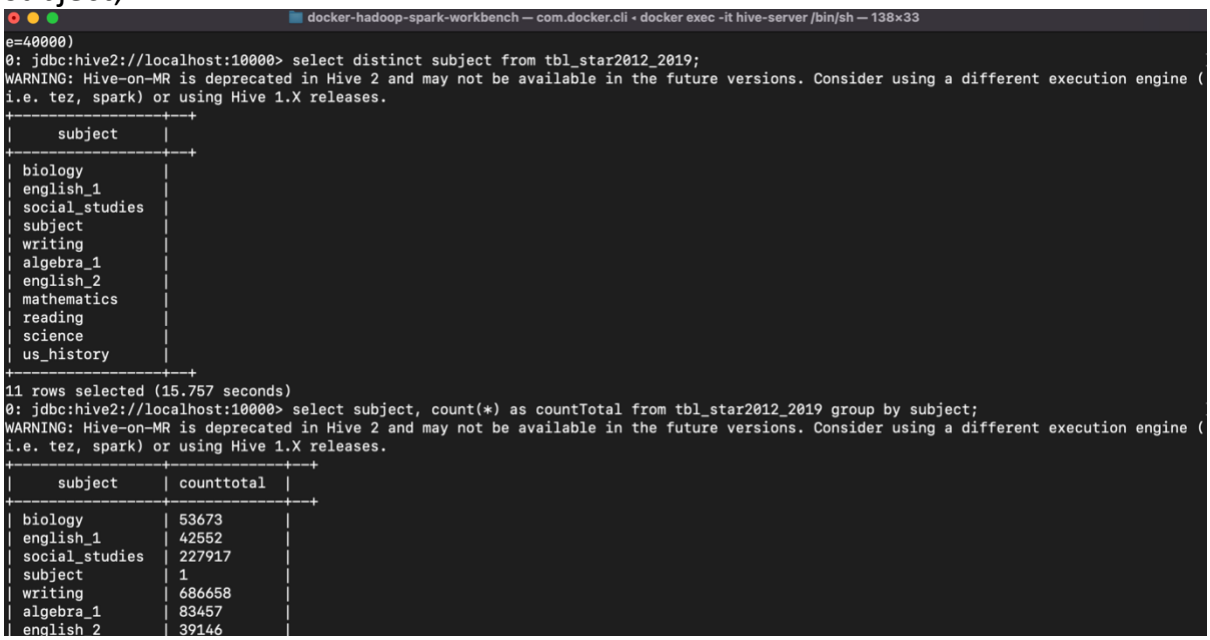
Distinct Query

select distinct subject from tbl_star2012_2019;

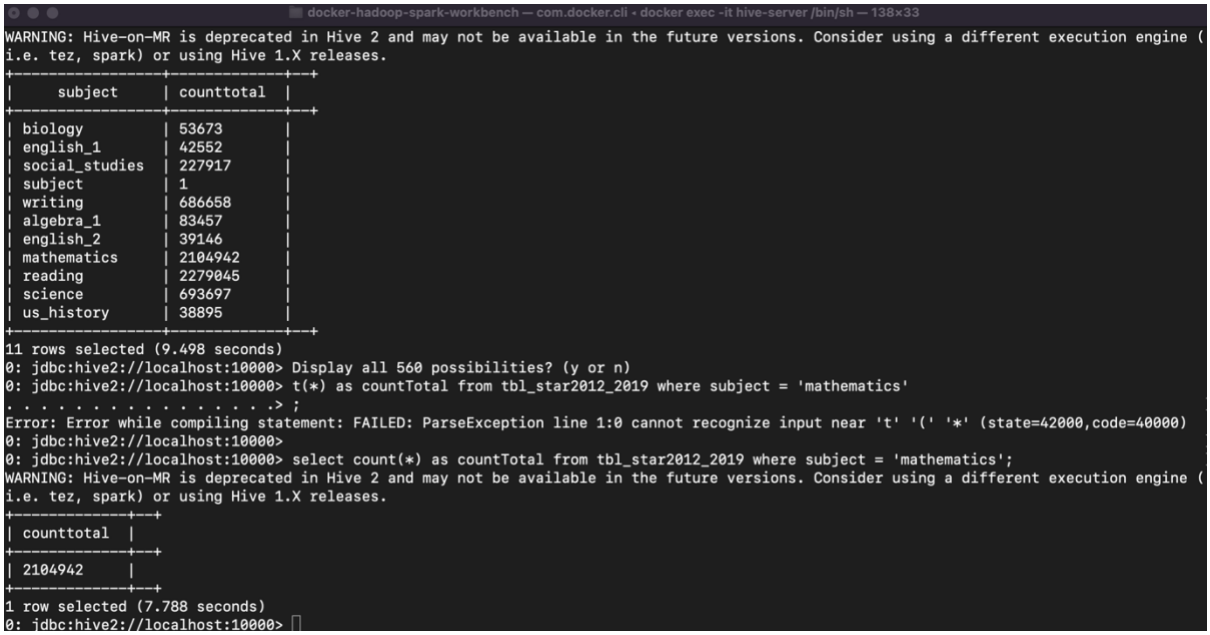


Group by & Count (*)

select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;

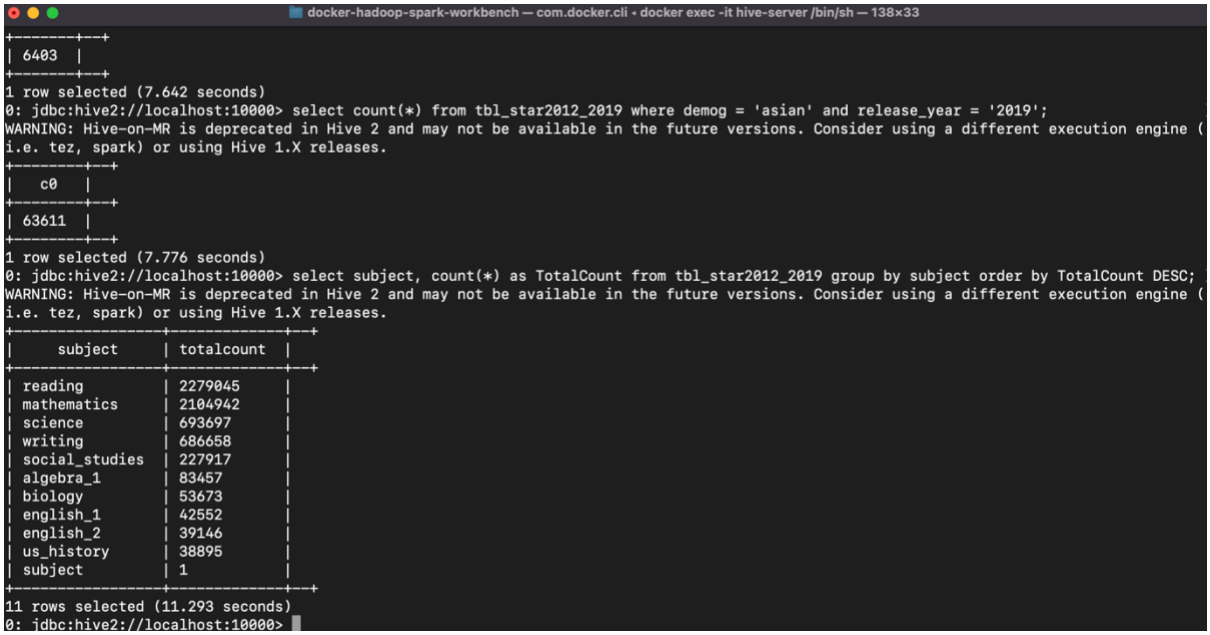


Where Clause



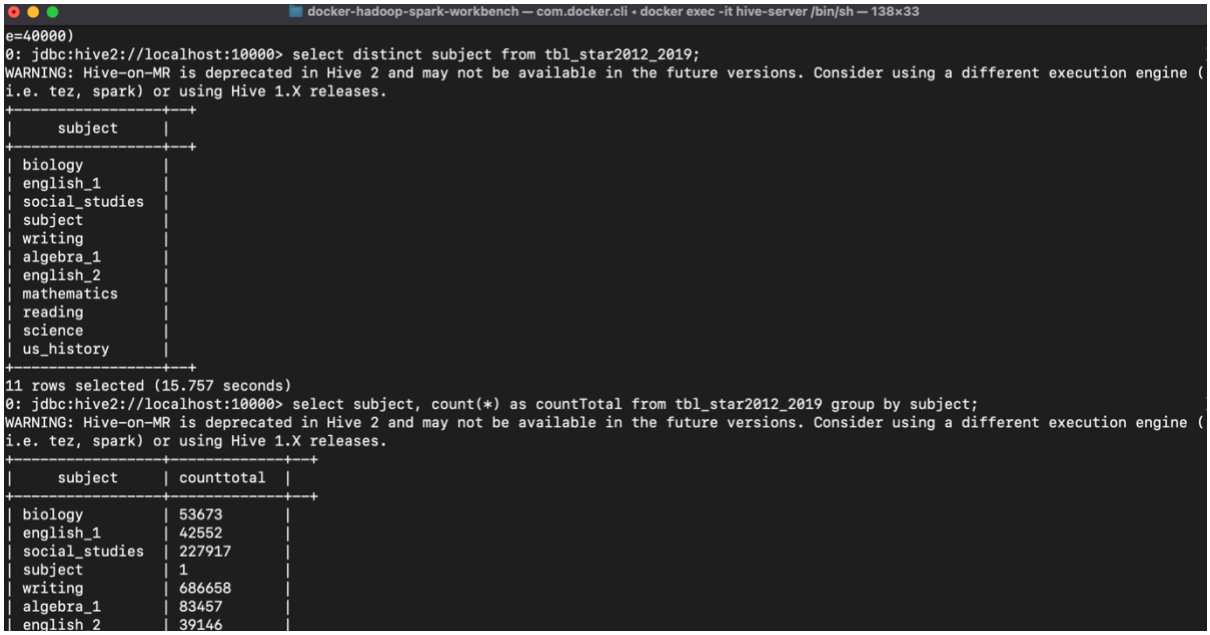
Order by

select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year order by TotalCount DESC;



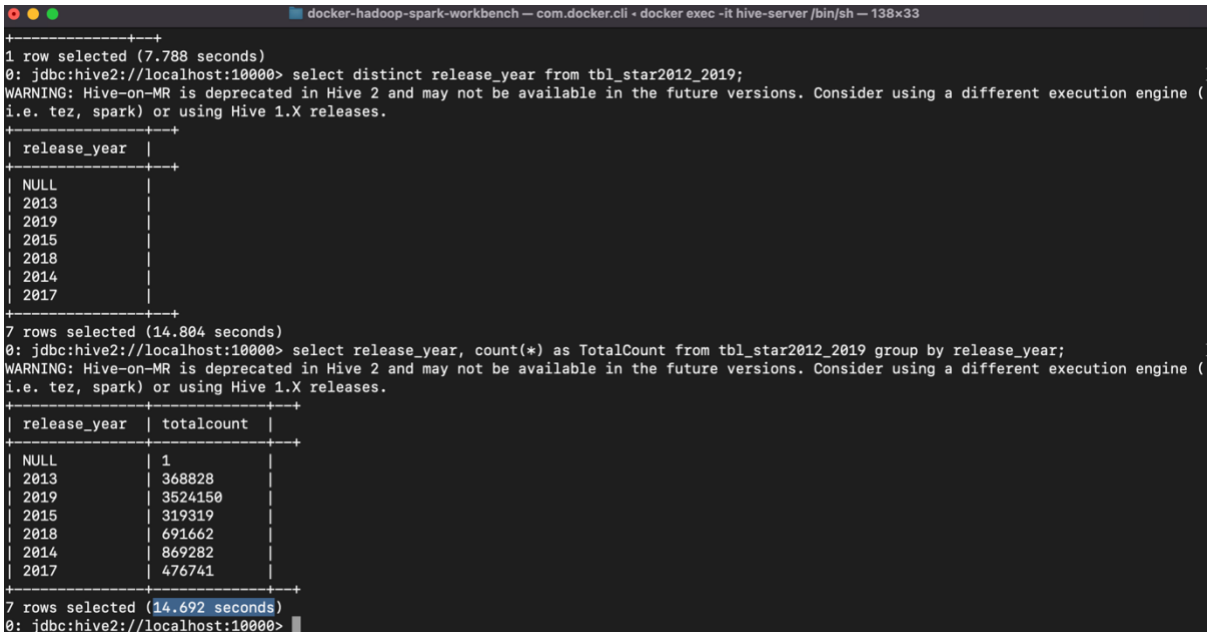
Checking Records for Each Subject

select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;



Looking at how many records belongs to each year

select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year;



Following Queries are executed to note performance of Apache Hive

1. Distinct Subject

```
select distinct subject from tbl_star2012_2019;
```

15.757 seconds

2. Count, Group by subjects

```
select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;
```

9.498 seconds

3. Total Star Students Learning Mathematics -- Count, Where Mathematics, Result -> 2104942

```
select count(*) as countTotal from tbl_star2012_2019 where subject = 'mathematics'
```

7.788 seconds

4. Check which years data we have

```
select distinct release_year from tbl_star2012_2019;
```

14.804 seconds

5. Total Count of data year wise

```
select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year;
```

14.692 seconds

6. which year have most data

```
select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year order by TotalCount DESC;
```

16.159 seconds

7. Look at demographics of Star students

```
select distinct demog from tbl_star2012_2019;
```

8.727 seconds

8. Count female students -> 672545

```
select count(*) from tbl_star2012_2019 where demog = 'female';
```

6.689 seconds

9. Count female students learning maths -> 221017

```
select count(*) from tbl_star2012_2019 where demog = 'female' and subject = 'mathematics';
```

8.935 seconds

10. Count of male students

```
select count(*) from tbl_star2012_2019 where demog = 'male';
```

6.953

11. Count of male learning biology -> 6403

```
select count(*) from tbl_star2012_2019 where demog = 'male' and subject = 'biology'
```

7.642 seconds

12. Count of asian students in year 2019 -> 63611

```
select count(*) from tbl_star2012_2019 where demog = 'asian' and release_year = '2019';
```

7.776 seconds

13. select which subject is mostly learned

```
select subject, count(*) as TotalCount from tbl_star2012_2019 group by subject order by TotalCount DESC
```

11.293 seconds

Query on Apache Spark

Config required to write queries

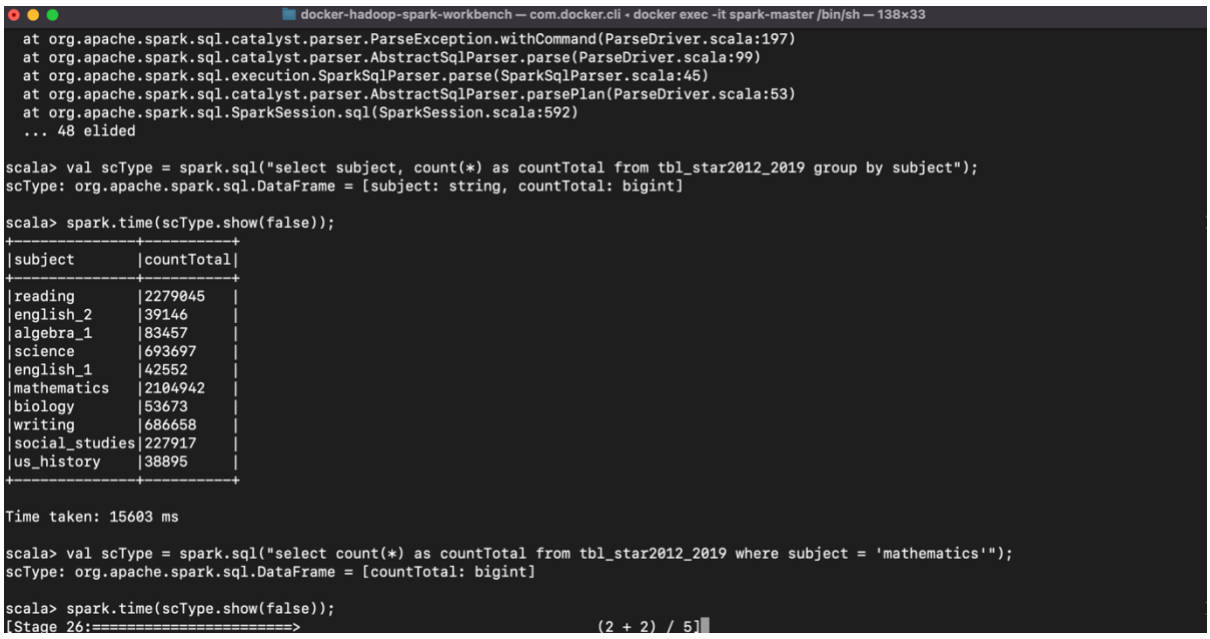
val df =
spark.read.text("hdfs://namenode:8020/user/home/tidy_campstaar1_2012to2019.csv")

df.createOrReplaceTempView("tidy_campstaar1_2012to2019)

spark.time(spark.sql("select distinct subject from tidy_campstaar1_2012to2019"))

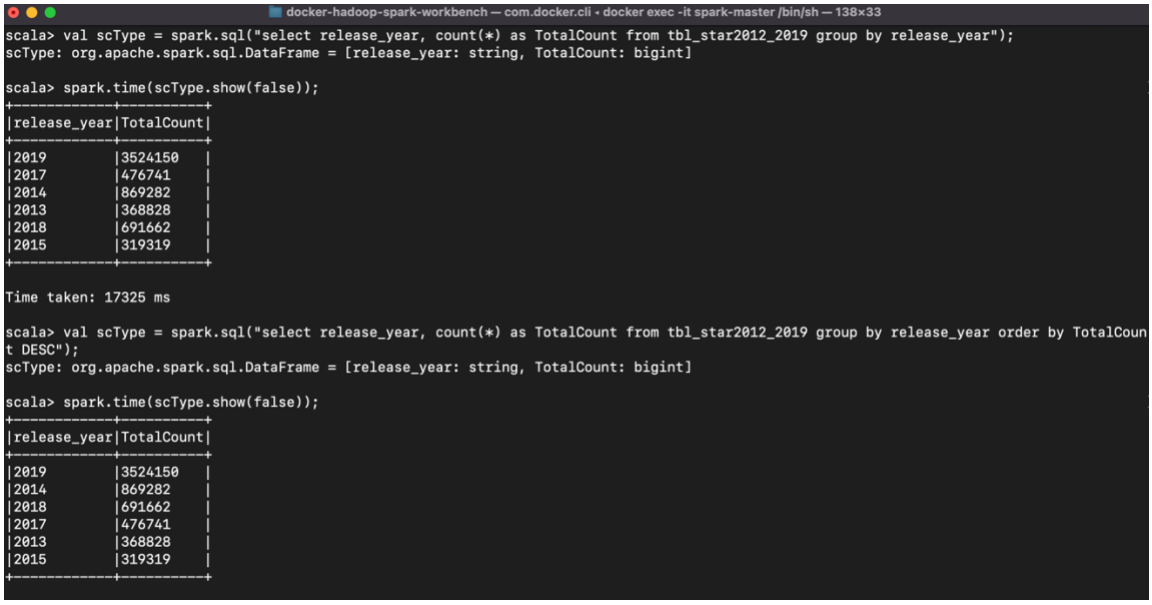
Count, Group by subjects

select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;



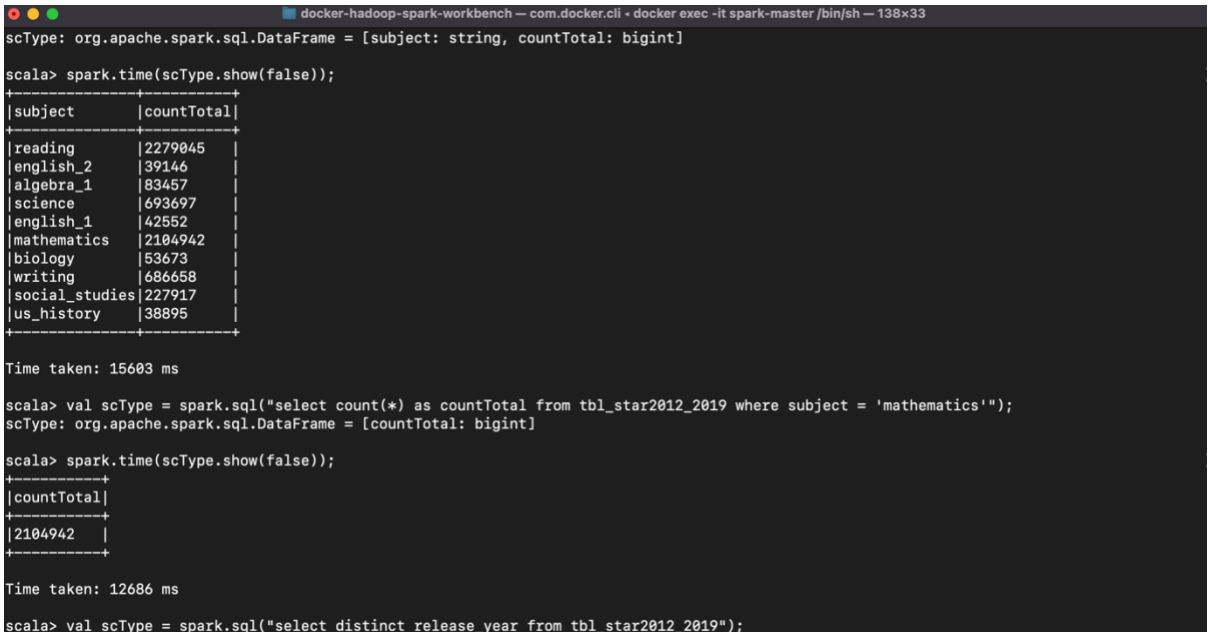
Total Count of data year wise

select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year;



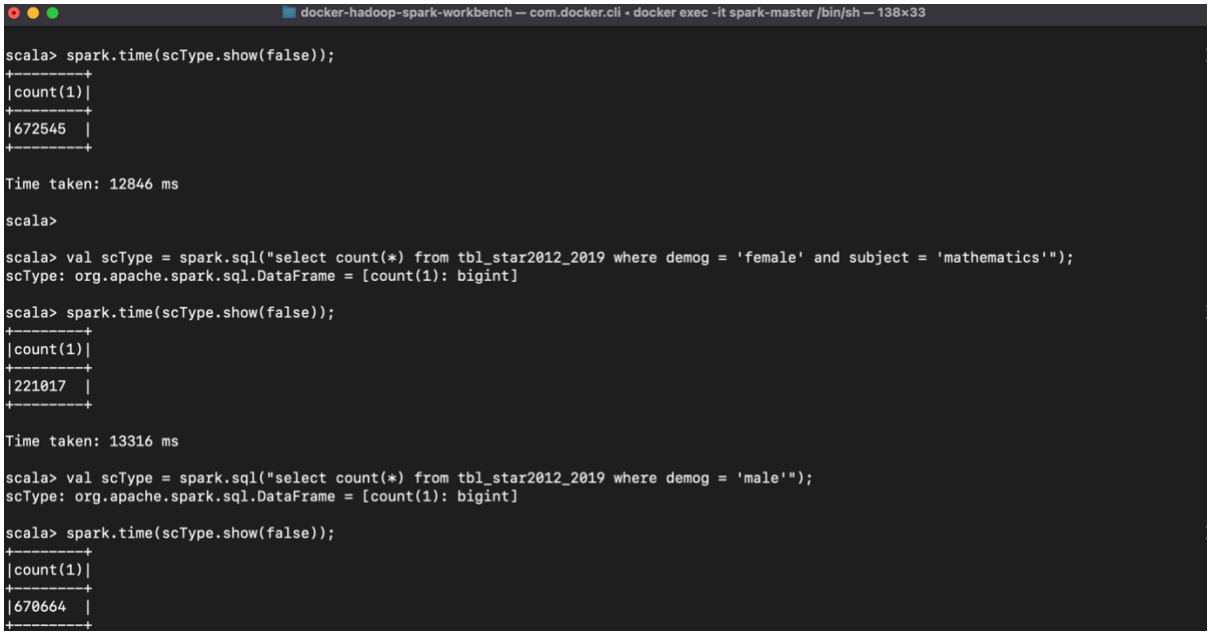
Total Star Students Learning Mathematics -- Count, Where Mathematics, Result -> 2104942

select count(*) as countTotal from tbl_star2012_2019 where subject = 'mathematics'



Count female students learning maths -> 221017

select count(*) from tbl_star2012_2019 where demog = 'female' and subject = 'mathematics';



Following Queries were Executed on Spark

1. Distinct Subject

select distinct subject from tbl_star2012_2019

35.538 ms

2. Count, Group by subjects

select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;

15.603 ms

3. Total Star Students Learning Mathematics -- Count, Where Mathematics, Result -> 21.04942

select count(*) as countTotal from tbl_star2012_2019 where subject = 'mathematics'

12.686 ms

4. Check which years data we have

select distinct release_year from tbl_star2012_2019

13.914 ms

5. Total Count of data year wise

select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year;

17.325 ms

6. which year have most data

```
select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year order by TotalCount DESC;
```

15.701 ms

7. Look at demographics of Star students

```
select distinct demog from tbl_star2012_2019;
```

14.060 ms

8. Count female students -> 672545

```
select count(*) from tbl_star2012_2019 where demog = 'female';
```

12.846 ms

9. Count female students learning maths -> 221017

```
select count(*) from tbl_star2012_2019 where demog = 'female' and subject = 'mathematics';
```

13.316 ms

10. Count of male students

```
select count(*) from tbl_star2012_2019 where demog = 'male';
```

13.167 ms

11. Count of male learning biology -> 6403

```
select count(*) from tbl_star2012_2019 where demog = 'male' and subject = 'biology'
```

17.000 ms

12. Count of asian students in year 2019 -> 63611

```
select count(*) from tbl_star2012_2019 where demog = 'asian' and release_year = '2019';
```

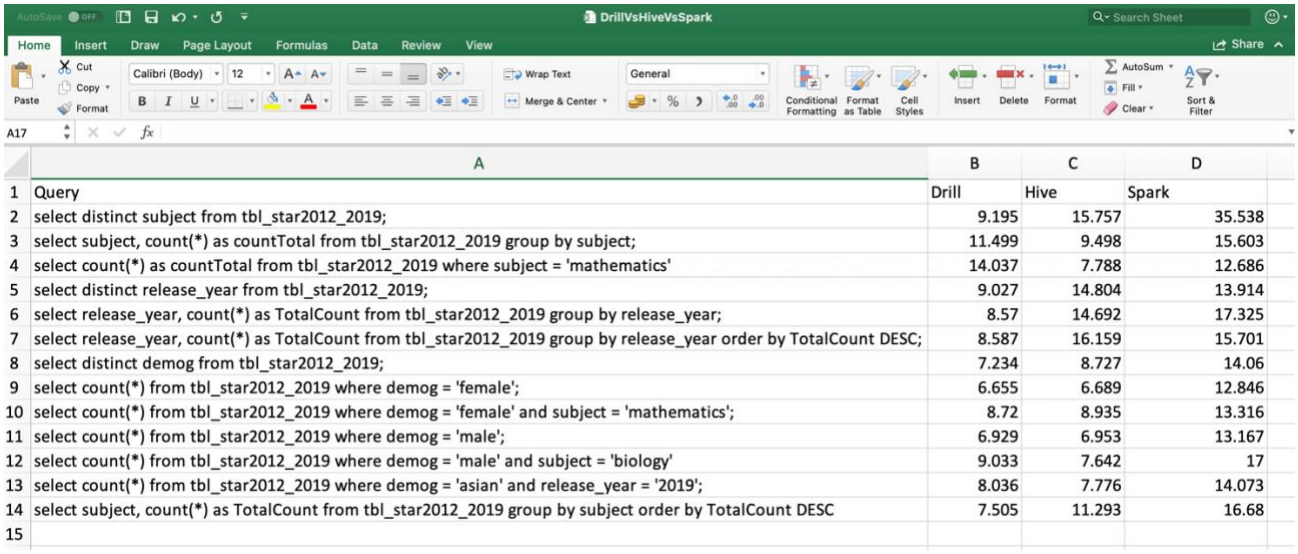
14.073 ms

13. select which subject is mostly learned

```
select subject, count(*) as TotalCount from tbl_star2012_2019 group by subject order by TotalCount DESC
```

16.680 ms

Following CSV file was generated for Performance Comparison between above Three



	A	B	C	D
1	Query	Drill	Hive	Spark
2	select distinct subject from tbl_star2012_2019;	9.195	15.757	35.538
3	select subject, count(*) as countTotal from tbl_star2012_2019 group by subject;	11.499	9.498	15.603
4	select count(*) as countTotal from tbl_star2012_2019 where subject = 'mathematics'	14.037	7.788	12.686
5	select distinct release_year from tbl_star2012_2019;	9.027	14.804	13.914
6	select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year;	8.57	14.692	17.325
7	select release_year, count(*) as TotalCount from tbl_star2012_2019 group by release_year order by TotalCount DESC;	8.587	16.159	15.701
8	select distinct demog from tbl_star2012_2019;	7.234	8.727	14.06
9	select count(*) from tbl_star2012_2019 where demog = 'female';	6.655	6.689	12.846
10	select count(*) from tbl_star2012_2019 where demog = 'female' and subject = 'mathematics';	8.72	8.935	13.316
11	select count(*) from tbl_star2012_2019 where demog = 'male';	6.929	6.953	13.167
12	select count(*) from tbl_star2012_2019 where demog = 'male' and subject = 'biology'	9.033	7.642	17
13	select count(*) from tbl_star2012_2019 where demog = 'asian' and release_year = '2019';	8.036	7.776	14.073
14	select subject, count(*) as TotalCount from tbl_star2012_2019 group by subject order by TotalCount DESC	7.505	11.293	16.68
15				

Docker Compose File for Apache Hive and Spark

```yaml
version: '2'
services:
  namenode:
    image: bde2020/hadoop-namenode:1.1.0-hadoop2.8-java8
    container_name: namenode
    volumes:
      - ./data/namenode:/hadoop/dfs/name
    environment:
      - CLUSTER_NAME=test
    env_file:
      - ./hadoop-hive.env
    ports:
      - 50070:50070
  datanode:
    image: bde2020/hadoop-datanode:1.1.0-hadoop2.8-java8
    depends_on:
      - namenode
    volumes:
      - ./data/datanode:/hadoop/dfs/data
    env_file:
      - ./hadoop-hive.env
    ports:
      - 50075:50075
  hive-server:
    image: bde2020/hive:2.1.0-postgresql-metastore
    container_name: hive-server
    env_file:
      - ./hadoop-hive.env
    environment:
      - "HIVE_CORE_CONF_javax_jdo_option_ConnectionURL=jdbc:postgresql://hive-metastore/metastore"
    ports:
      - "10000:10000"
  hive-metastore:
    image: bde2020/hive:2.1.0-postgresql-metastore
    container_name: hive-metastore
    env_file:
      - ./hadoop-hive.env
    command: /opt/hive/bin/hive --service metastore
  hive-metastore-postgresql:
    image: bde2020/hive-metastore-postgresql:2.1.0
  spark-master:
    image: bde2020/spark-master:2.1.0-hadoop2.8-hive-java8
    container_name: spark-master
    ports:
      - 8080:8080
      - 7077:7077
    env_file:
```

Docker Compose for Apache Drill

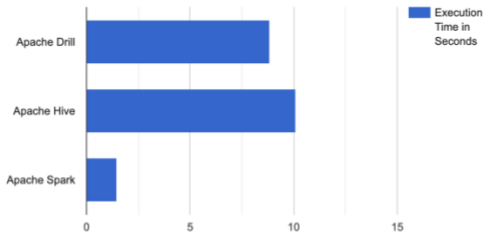
```
namenode-1:
  container_name: namenode-1
  networks: ["vnet"]
  hostname: namenode-1.vnet
  image: smizy/hadoop-base:2.7.7-alpine
  expose: ["8020"]
  ports:  ["50070"]
  environment:
    - SERVICE_8020_NAME=namenode
    - SERVICE_50070_IGNORE=true
    - HADOOP_ZOOKEEPER_QUORUM=zookeeper-1.vnet:2181,zookeeper-2.vnet:2181,zookeeper-3.vnet:2181
    - HADOOP_HEAPSIZE=1000

  entrypoint: entrypoint.sh
  command: namenode-1

namenode-2:
  container_name: namenode-2
  networks: ["vnet"]
  hostname: namenode-2.vnet
  image: smizy/hadoop-base:2.7.7-alpine
  expose: ["8020"]
  ports:  ["50070"]
  environment:
    - SERVICE_8020_NAME=namenode
    - SERVICE_50070_IGNORE=true
    - HADOOP_ZOOKEEPER_QUORUM=zookeeper-1.vnet:2181,zookeeper-2.vnet:2181,zookeeper-3.vnet:2181
    - HADOOP_HEAPSIZE=1000

  entrypoint: entrypoint.sh
  command: namenode-2

datanode-1:
  container_name: datanode-1
  networks: ["vnet"]
  hostname: datanode-1.vnet
  image: smizy/hadoop-base:2.7.7-alpine
  expose: ["50010", "50020", "50075"]
  environment:
    - SERVICE_50010_NAME=datanode
    - SERVICE_50020_IGNORE=true
    - SERVICE_50075_IGNORE=true
    - HADOOP_ZOOKEEPER_QUORUM=zookeeper-1.vnet:2181,zookeeper-2.vnet:2181,zookeeper-3.vnet:2181
    - HADOOP_HEAPSIZE=1000

  entrypoint: entrypoint.sh
  command: datanode
```

Project Front End using Python & Flask

Performance Comparison Between Apache Hive, Apache Drill and Spark

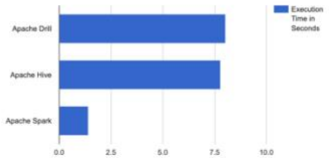
Total Queries Executed: 13

Query: Average Query Time of All Queries on Particular tools

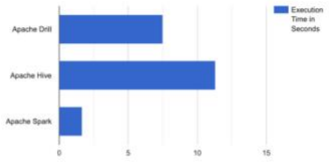


Query: select distinct subject from tbl_star2012_2019;

Query: select count(*) from tbl_star2012_2019 where demog = 'asian' and release_year = '2019';

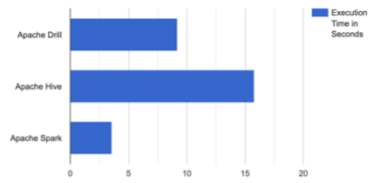


Query: select subject count(*) as TotalCount from tbl_star2012_2019 group by subject order by TotalCount DESC



@ Project Made By M.Ibrahim Arain - ERP: 15976 – IBA Karachi

Query: select distinct subject from tbl_star2012_2019;



Query: select subject count(*) as countTotal from tbl_star2012_2019 group by subject;