APACHE HIVE (Handout)

Group 5:
(Javeria Raja, Lidia Makishti, Raphael Steinborn, Saqib Sarwar, Vineet Racharla)

This guide will help us set up Apache Hive along with Hue using Docker containers.

Apache Hive is a distributed data warehouse system for analyzing large datasets using SQL, while Hue is a web-based interface for interacting with Apache Hadoop ecosystem components.

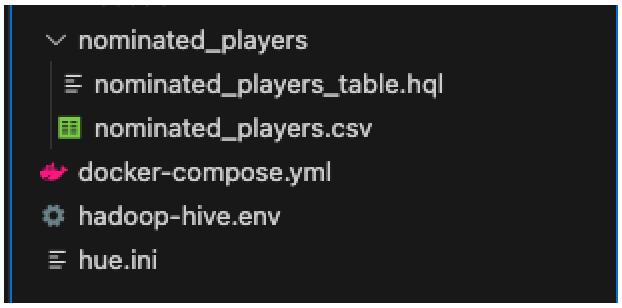
Before we begin, we need to have Docker installed and opened on our machines.

Step 1: Download Hive.zip:

Download the Hive.zip file from moodle and extract the files.

Step 2: Directory Structure:

Now we should see the following files inside this folder.



```
nominated_players
    nominated_players_table.hql
    nominated_players.csv
docker-compose.yml
hadoop-hive.env
hue.ini
```

1. Docker-compose.yml:
   - It is written in YAML format and contains the networks, volumes, and services required for our Docker setup.

2. Hadoop-hive.env:

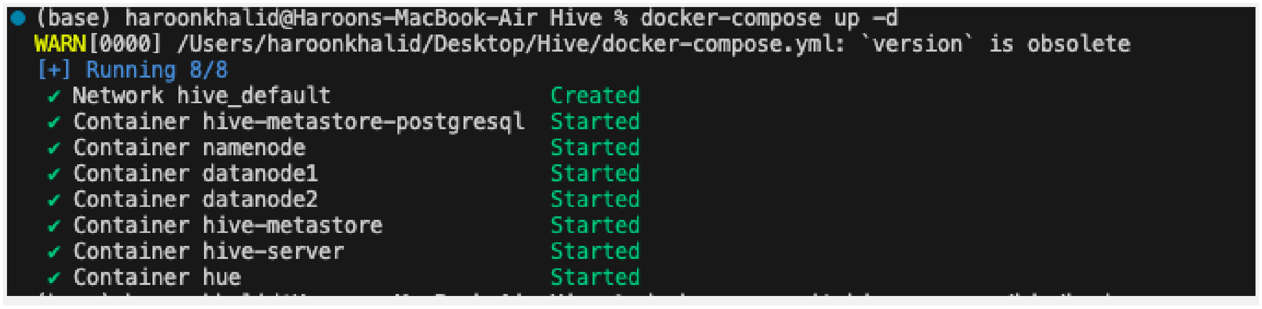
- This file is used to set environment variables(such as memory allocation, file paths, database connection strings, and other configuration options) specific to the Hadoop and Hive components.

3. Hue.ini:
    - This file contains configuration settings for the Hue application
4. Nominated_players_table.hql
    - This file contains the necessary queries to create our database table. It is written in Hql language.
5. Nominated_players.csv
    - This files contains the data of the players

Step 3: Create & Start all services:

Open the terminal inside our Hive directory and run the single docker compose command to create and start all services.

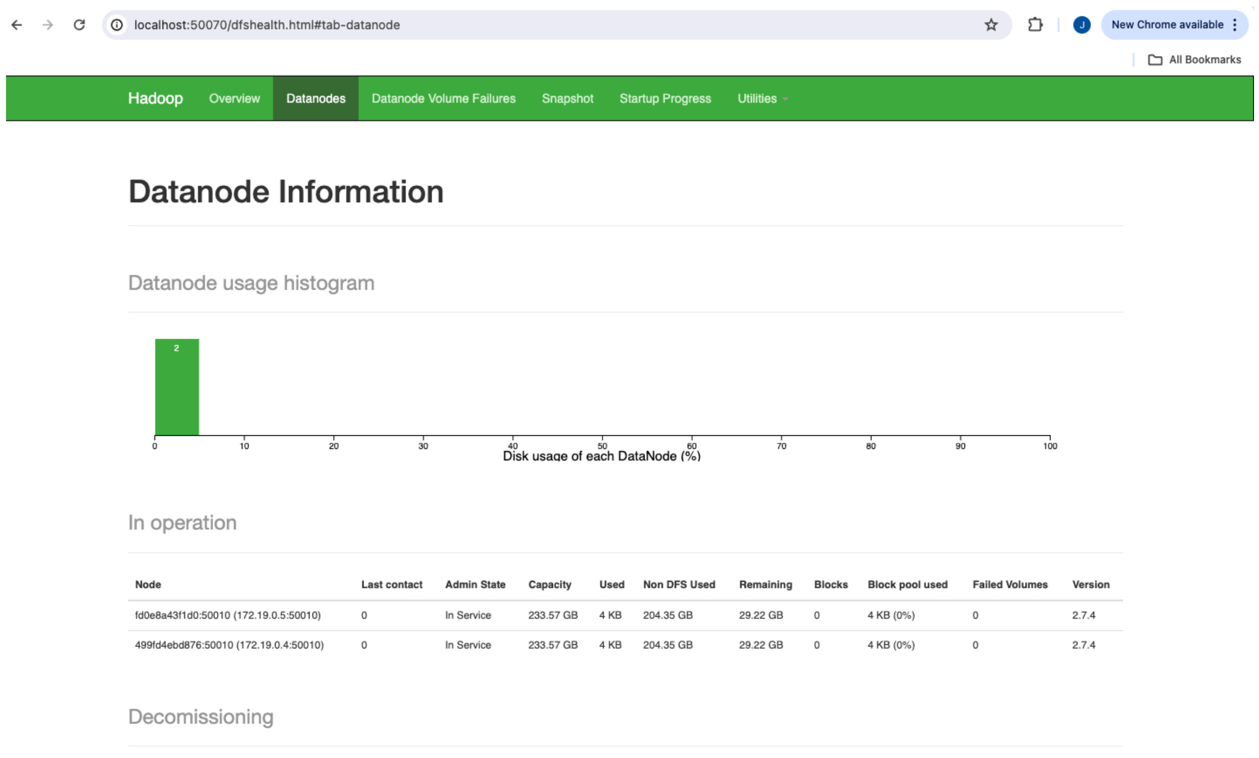
```
$ docker-compose up -d
```



```
● (base) haroonkhalid@Haroons-MacBook-Air Hive % docker-compose up -d
  WARN[0000] /Users/haroonkhalid/Desktop/Hive/docker-compose.yml: `version` is obsolete
  [+] Running 8/8
   ✔ Network hive_default                  Created
   ✔ Container hive-metastore-postgresql   Started
   ✔ Container namenode                    Started
   ✔ Container datanode1                   Started
   ✔ Container datanode2                   Started
   ✔ Container hive-metastore              Started
   ✔ Container hive-server                 Started
   ✔ Container hue                         Started
```

Step 4: Verify that the namenode and both datanodes have started.

The namenode would be running on localhost:50070. You can view the nodes in the datanodes section.



Datanode Information

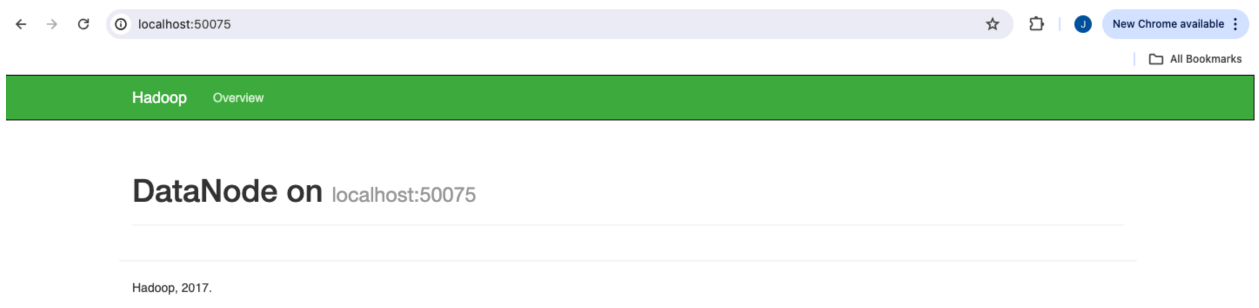
Datanode usage histogram

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
fd0e8a43f1d0:50010 (172.19.0.5:50010)	0	In Service	233.57 GB	4 KB	204.35 GB	29.22 GB	0	4 KB (0%)	0	2.7.4
499fd4ebd876:50010 (172.19.0.4:50010)	0	In Service	233.57 GB	4 KB	204.35 GB	29.22 GB	0	4 KB (0%)	0	2.7.4

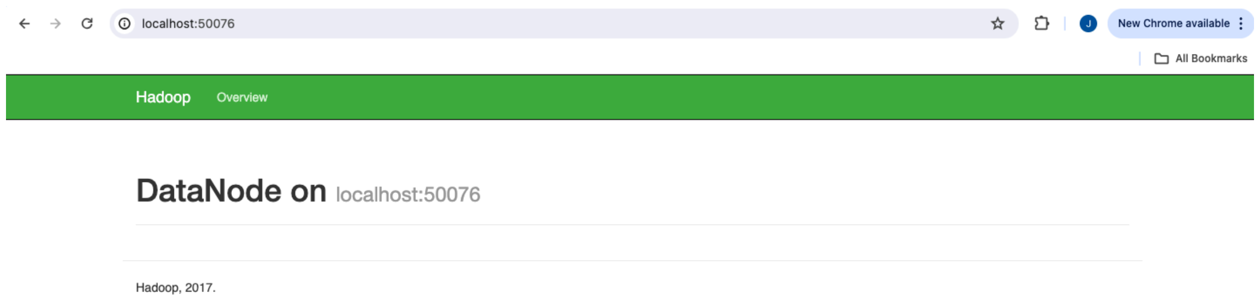
Decomissioning

the datanodes would be running on localhost:50075 and localhost:50076 respectively



DataNode on localhost:50075

Hadoop, 2017.

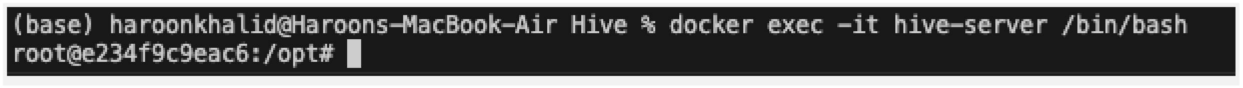


DataNode on localhost:50076

Hadoop, 2017.

Step 5: Log on to the Hive server using the following command.

```
$ docker exec -it hive-server /bin/bash
```



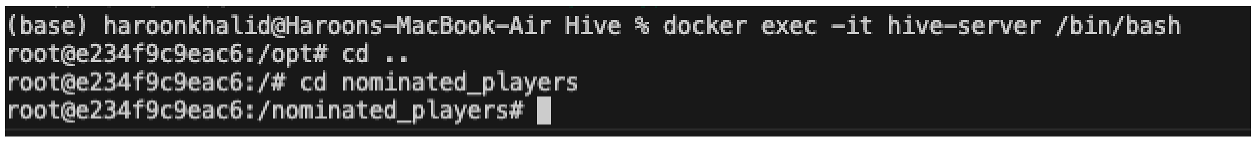
```
(base) haroonkhalid@Haroons-MacBook-Air Hive % docker exec -it hive-server /bin/bash
root@e234f9c9eac6:/opt# 
```

Step 6: Navigate to the Hive server container's directory and execute the Hql file

To navigate to the hive server directory, run the following commands:

```
root@e234f9c9eac6:/opt# cd ..
```

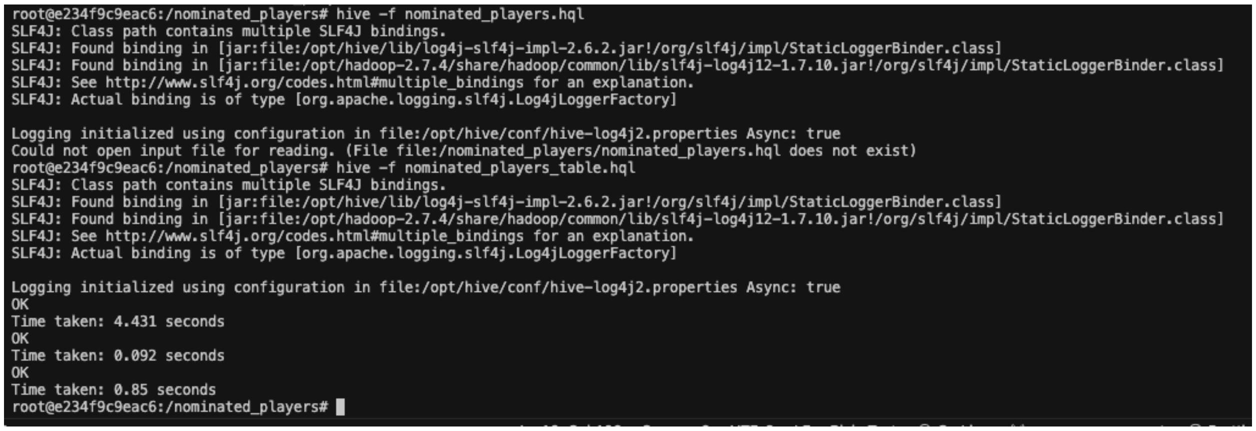
```
root@e234f9c9eac6:/# cd nominated_players
```



```
(base) haroonkhalid@Haroons-MacBook-Air Hive % docker exec -it hive-server /bin/bash
root@e234f9c9eac6:/opt# cd ..
root@e234f9c9eac6:/# cd nominated_players
root@e234f9c9eac6:/nominated_players# 
```

Now, execute the following command to run the nominated_players.hql file in this directory and create the database and the table structure

```
root@e234f9c9eac6:/nominated_players# hive -f
nominated_players_table.hql
```



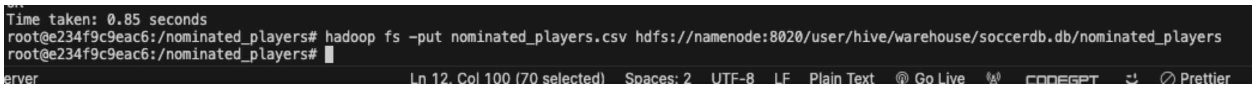
```
root@e234f9c9eac6:/nominated_players# hive -f nominated_players.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Could not open input file for reading. (File file:/nominated_players/nominated_players.hql does not exist)
root@e234f9c9eac6:/nominated_players# hive -f nominated_players_table.hql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
OK
Time taken: 4.431 seconds
OK
Time taken: 0.092 seconds
OK
Time taken: 0.85 seconds
root@e234f9c9eac6:/nominated_players# █
```

After creating the table, we can now executed the following command to load data from our csv file into the table.

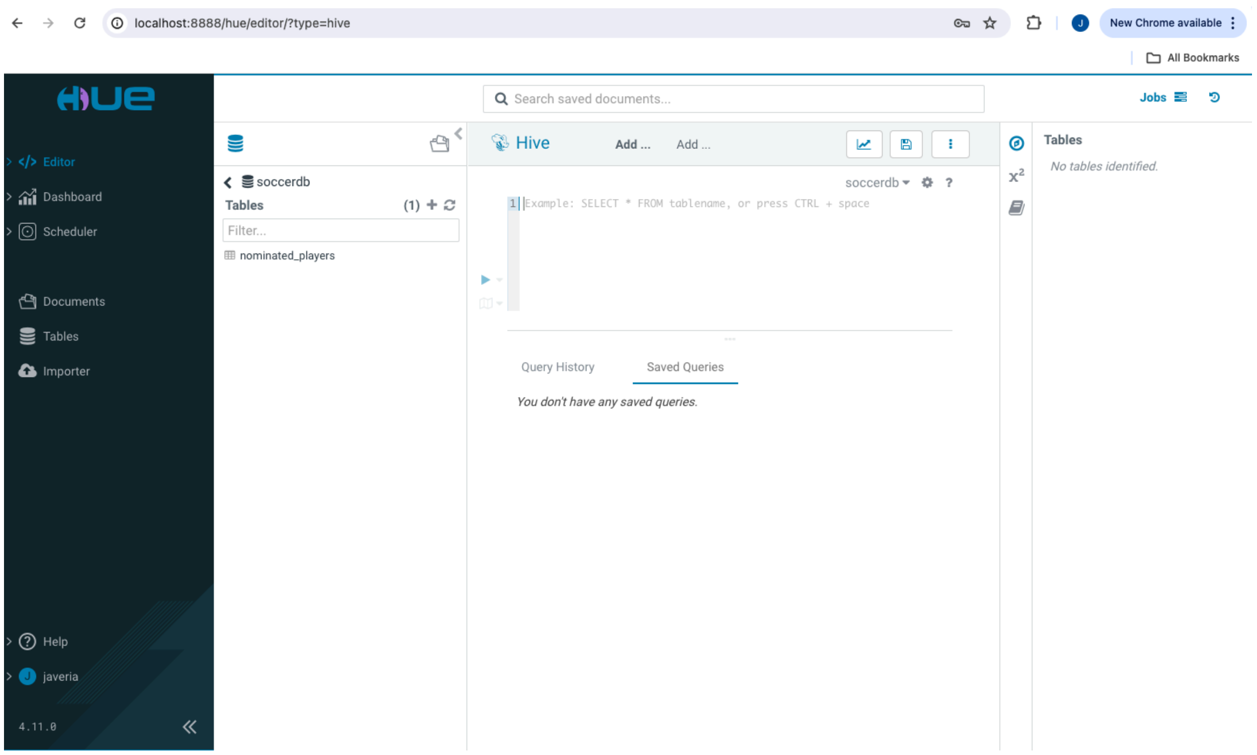
```
root@e234f9c9eac6:/nominated_players# hadoop fs -put
nominated_players.csv
hdfs://namenode:8020/user/hive/warehouse/soccerdb.db/nominated_p
layers
```



```
Time taken: 0.85 seconds
root@e234f9c9eac6:/nominated_players# hadoop fs -put nominated_players.csv hdfs://namenode:8020/user/hive/warehouse/soccerdb.db/nominated_players
root@e234f9c9eac6:/nominated_players# █
```

Step 7: Access Hue:

Access the Hue interface by navigating to `http://localhost:8888` in your web browser.



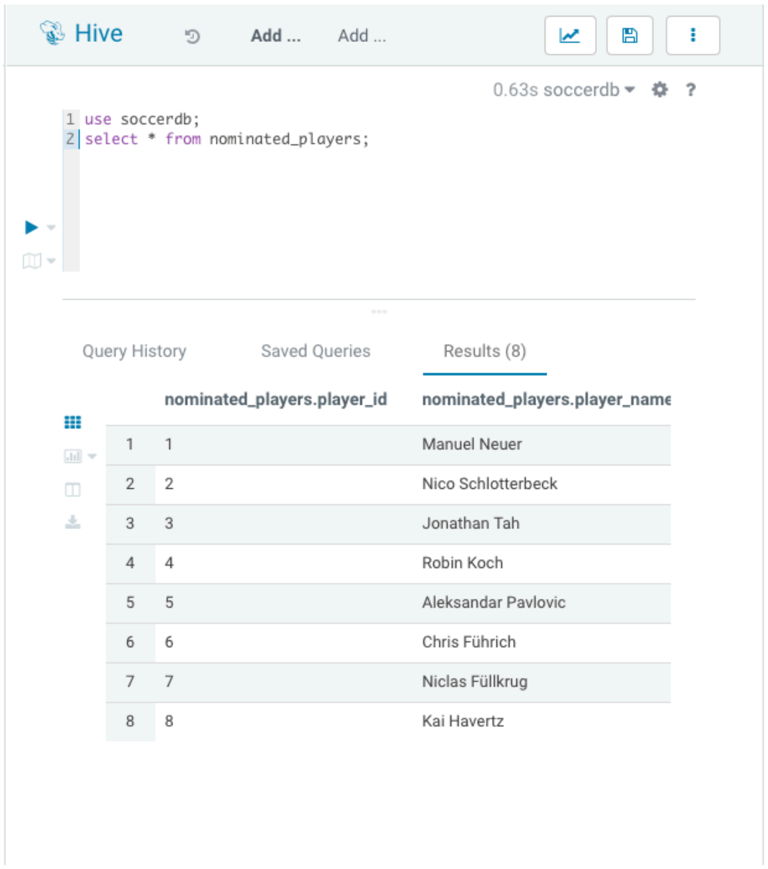
Now let's run the following commands to view our players' data!

```
use soccerdb;

select * from nominated_players;
```

It might take a while!

Now, we can see our data inside our table.



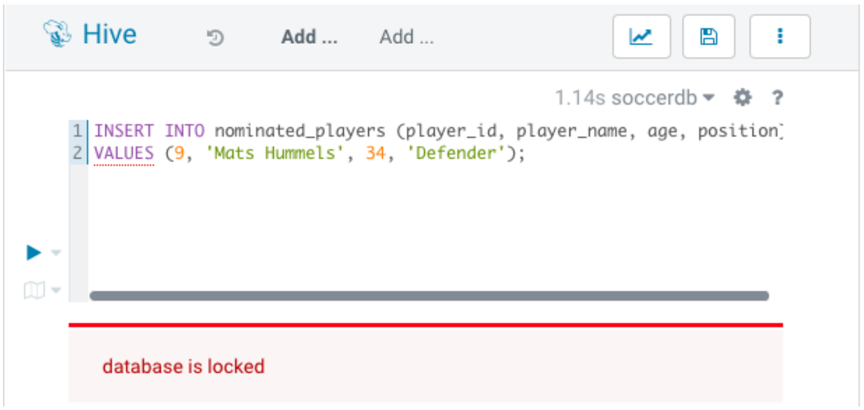
```
use soccerdb;
select * from nominated_players;
```

	nominated_players.player_id	nominated_players.player_name
1	1	Manuel Neuer
2	2	Nico Schlotterbeck
3	3	Jonathan Tah
4	4	Robin Koch
5	5	Aleksandar Pavlovic
6	6	Chris Führich
7	7	Niclas Füllkrug
8	8	Kai Havertz

We can also use the following command to insert data into our table.

```
INSERT INTO nominated_players (player_id, player_name, age,
position)

VALUES (9, 'Mats Hummels', 34, 'Defender');
```

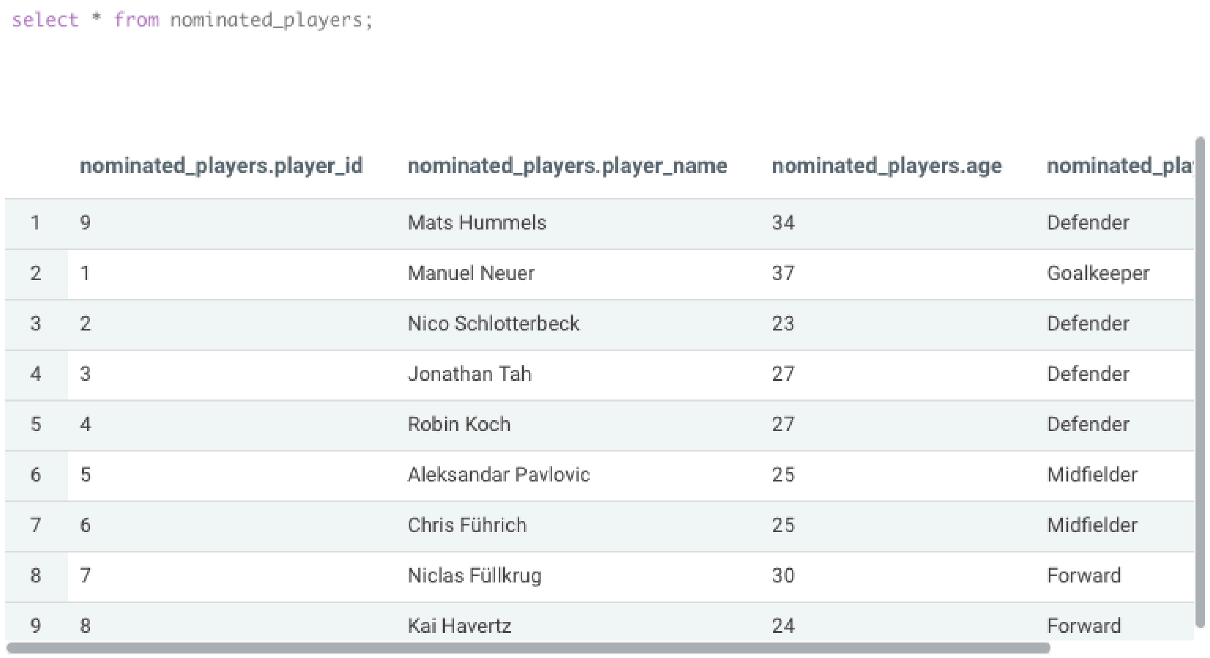


```
INSERT INTO nominated_players (player_id, player_name, age, position)
VALUES (9, 'Mats Hummels', 34, 'Defender');
```

database is locked

Now we can again use the select command to view our inserted data.

```
select * from nominated_players;
```



```
select * from nominated_players;
```

	nominated_players.player_id	nominated_players.player_name	nominated_players.age	nominated_pla
1	9	Mats Hummels	34	Defender
2	1	Manuel Neuer	37	Goalkeeper
3	2	Nico Schlotterbeck	23	Defender
4	3	Jonathan Tah	27	Defender
5	4	Robin Koch	27	Defender
6	5	Aleksandar Pavlovic	25	Midfielder
7	6	Chris Führich	25	Midfielder
8	7	Niclas Füllkrug	30	Forward
9	8	Kai Havertz	24	Forward

Thank you!

References:

https://docs.gethue.com/user/querying/

https://hshirodkar.medium.com/apache-hive-on-docker-4d7280ac6f8e

https://hub.docker.com/r/gethue/hue#!