Week 5 Assignment

Guruprasad Velikadu Krishnamoorthy

College of Science and Technology, Bellevue University

DSC650-T301: Big Data

Professor. Nasheb Ismaily

December 19, 2023

1. **<u>SparkSQL with Scala:</u>**

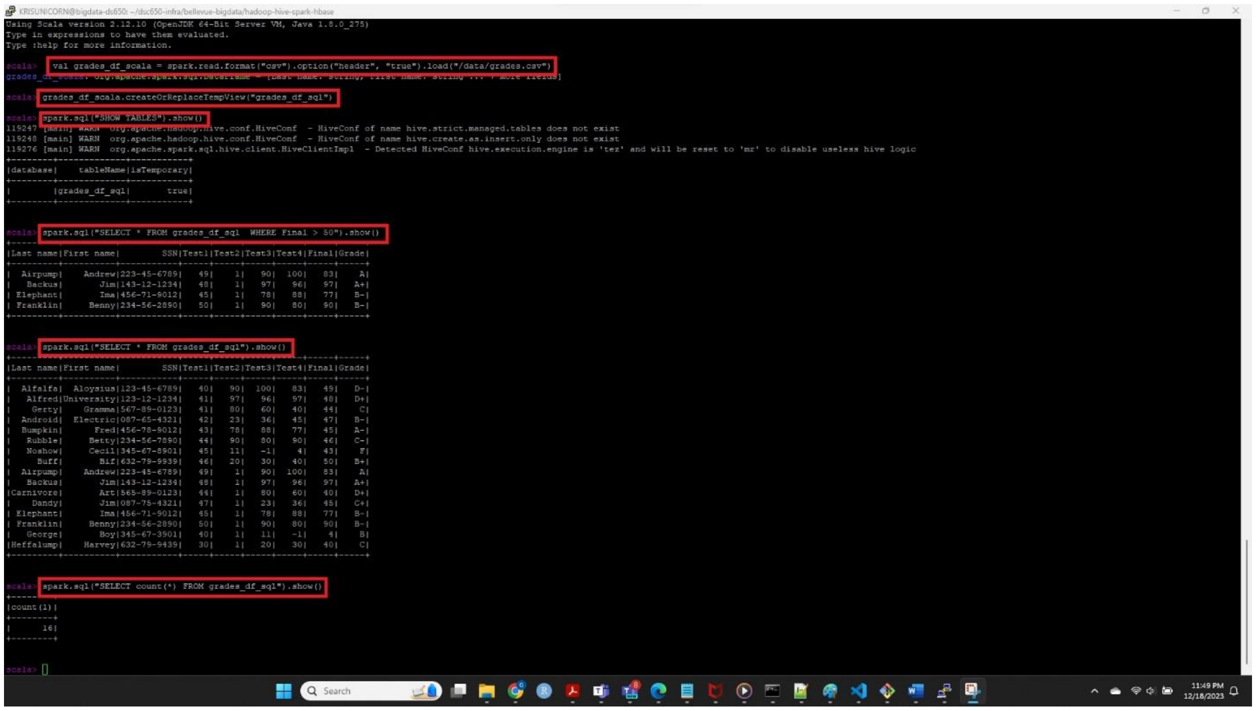
   val grades_df_scala = spark.read.format("csv").option("header", "true").load("/data/grades.csv")

   grades_df_scala.createOrReplaceTempView("grades_df_sql")

   spark.sql("SHOW TABLES").show()

   spark.sql("SELECT * FROM grades_df_sql WHERE Final > 50").show()

   spark.sql("SELECT * FROM grades_df_sql").show()



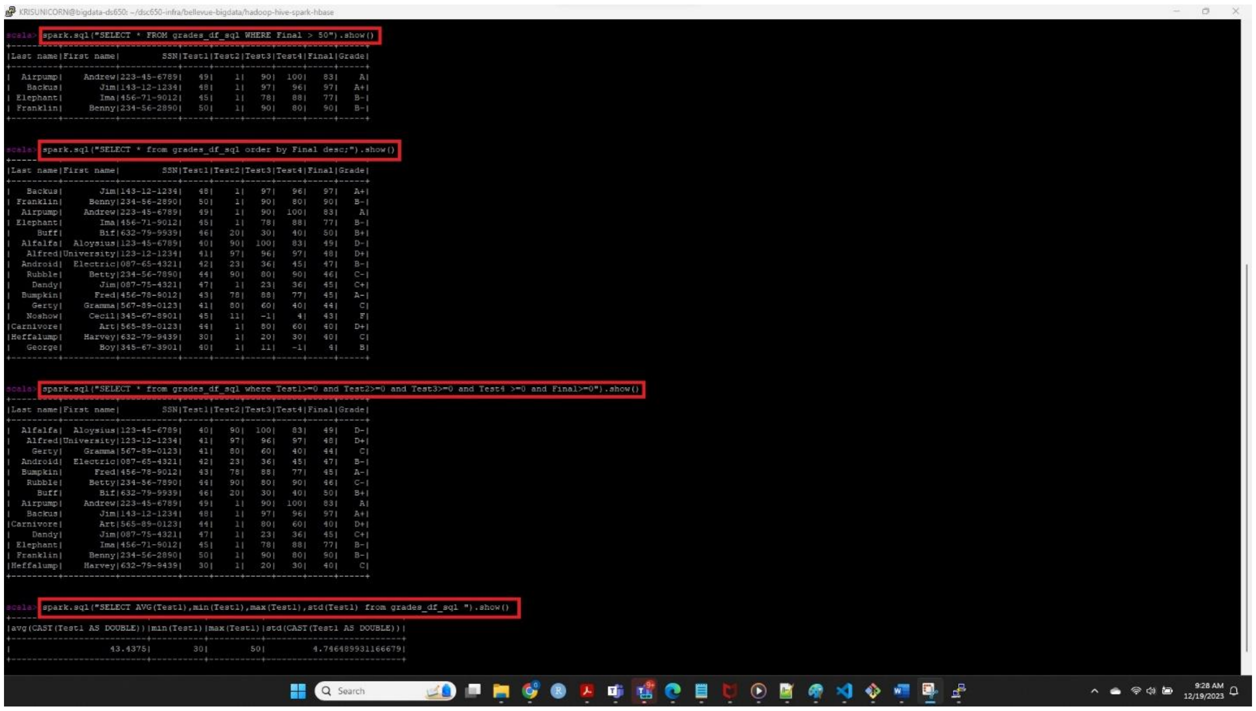
```
KRISUNICORN@bigdata-ds650: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_275)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val grades_df_scala = spark.read.format("csv").option("header", "true").load("/data/grades.csv")
grades_df_scala: org.apache.spark.sql.DataFrame = [Last name: string, First name: string ... 7 more fields]

scala> grades_df_scala.createOrReplaceTempView("grades_df_sql")

scala> spark.sql("SHOW TABLES").show()
119247 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
119248 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
119276 [main] WARN  org.apache.spark.sql.hive.client.HiveClientImpl  - Detected HiveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable useless hive logic
+--------+-------------+-----------+
|database|    tableName|isTemporary|
+--------+-------------+-----------+
|        |grades_df_sql|       true|
+--------+-------------+-----------+


scala> spark.sql("SELECT * FROM grades_df_sql  WHERE Final > 50").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+


scala> spark.sql("SELECT * FROM grades_df_sql").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|   George|       Boy|345-67-3901|   40|    1|   11|   -1|    4|    B|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+


scala> spark.sql("SELECT count(*) FROM grades_df_sql").show()
+--------+
|count(1)|
+--------+
|      16|
+--------+


scala> 
```

Additional SparkSQL commands used:

1.1. spark.sql("SELECT * from grades_df_sql order by Final desc;").show()
Significance: This command lists the in the Descending order of the values of column "Final"

1.2. spark.sql("SELECT * from grades_df_sql where Test1>=0 and Test2>=0 and Test3>=0 and Test4 >=0 and Final>=0").show()
Significance: This returns the records where the Test1,Test2, Test3 ,Test4 and Final are only positive values.

1.3. spark.sql("SELECT AVG(Test1),min(Test1),max(Test1),std(Test1) from grades_df_sql ").show()
Significance: Returns the statistical quantities of Test1 values.



```
KRISUNICORN@bigdata-ds650: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

scala> spark.sql("SELECT * FROM grades_df_sql WHERE Final > 50").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+


scala> spark.sql("SELECT * from grades_df_sql order by Final desc;").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
|   George|       Boy|345-67-3901|   40|    1|   11|   -1|    4|    B|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+


scala> spark.sql("SELECT * from grades_df_sql where Test1>=0 and Test2>=0 and Test3>=0 and Test4 >=0 and Final>=0").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+


scala> spark.sql("SELECT AVG(Test1),min(Test1),max(Test1),std(Test1) from grades_df_sql ").show()
+--------------------------+----------+----------+--------------------------+
|avg(CAST(Test1 AS DOUBLE))|min(Test1)|max(Test1)|std(CAST(Test1 AS DOUBLE))|
+--------------------------+----------+----------+--------------------------+
|                   43.4375|        30|        50|         4.746489931166679|
+--------------------------+----------+----------+--------------------------+
```

2. **<u>SparkSQL with Python:</u>**

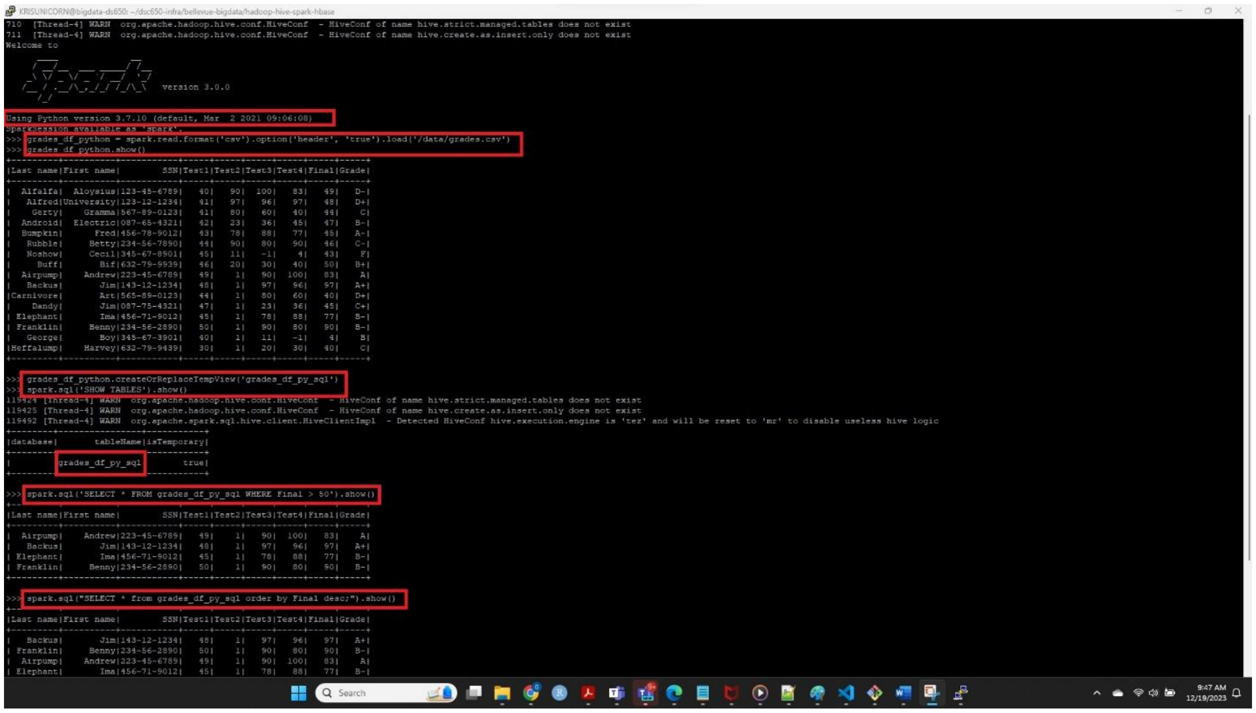
grades_df_python = spark.read.format('csv').option('header', 'true').load('/data/grades.csv')

grades_df_python.show()

grades_df_python.createOrReplaceTempView('grades_df_py_sql')

spark.sql('SHOW TABLES').show()

spark.sql('SELECT * FROM grades_df_py_sql WHERE Final > 50').show()



```
KRISUNICORN@bigdata-ds650: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
710  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
711  [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.0
      /_/

Using Python version 3.7.10 (default, Mar  2 2021 09:06:08)
SparkSession available as 'spark'.
>>> grades_df_python = spark.read.format('csv').option('header', 'true').load('/data/grades.csv')
>>> grades_df_python.show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|   George|       Boy|345-67-3901|   40|    1|   11|   -1|    4|    B|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+

>>> grades_df_python.createOrReplaceTempView('grades_df_py_sql')
>>> spark.sql('SHOW TABLES').show()
119424 [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
119425 [Thread-4] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
119492 [Thread-4] WARN  org.apache.spark.sql.hive.client.HiveClientImpl  - Detected HiveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable useless hive logic
+--------+----------------+-----------+
|database|       tableName|isTemporary|
+--------+----------------+-----------+
|        |grades_df_py_sql|       true|
+--------+----------------+-----------+

>>> spark.sql('SELECT * FROM grades_df_py_sql WHERE Final > 50').show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+

>>> spark.sql("SELECT * from grades_df_py_sql order by Final desc;").show()
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
```

Additional SparkSQL commands used:

2.1. spark.sql("SELECT * from grades_df_py_sql order by Final desc;").show()
Significance: This command lists the in the Descending order of the values of column "Final"

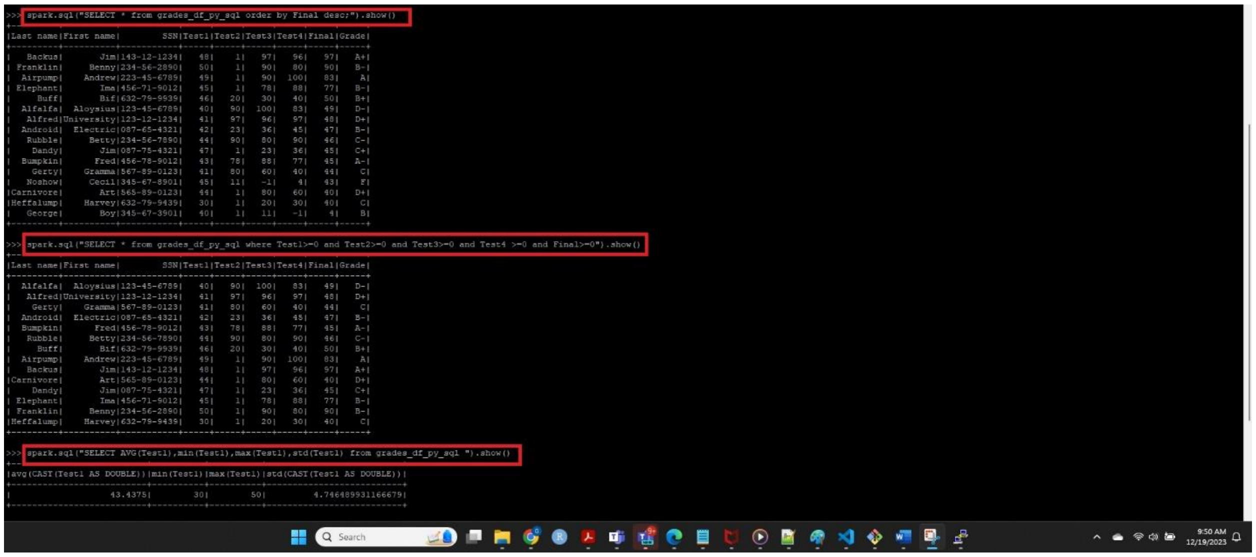
2.2. spark.sql("SELECT * from grades_df_py_sql where Test1>=0 and Test2>=0 and Test3>=0 and Test4 >=0 and Final>=0").show()
Significance: This returns the records where the Test1,Test2, Test3 ,Test4 and Final are only positive values.

2.3. spark.sql("SELECT AVG(Test1),min(Test1),max(Test1),std(Test1) from grades_df_py_sql ").show()
Significance: Returns the statistical quantities of Test1 values.

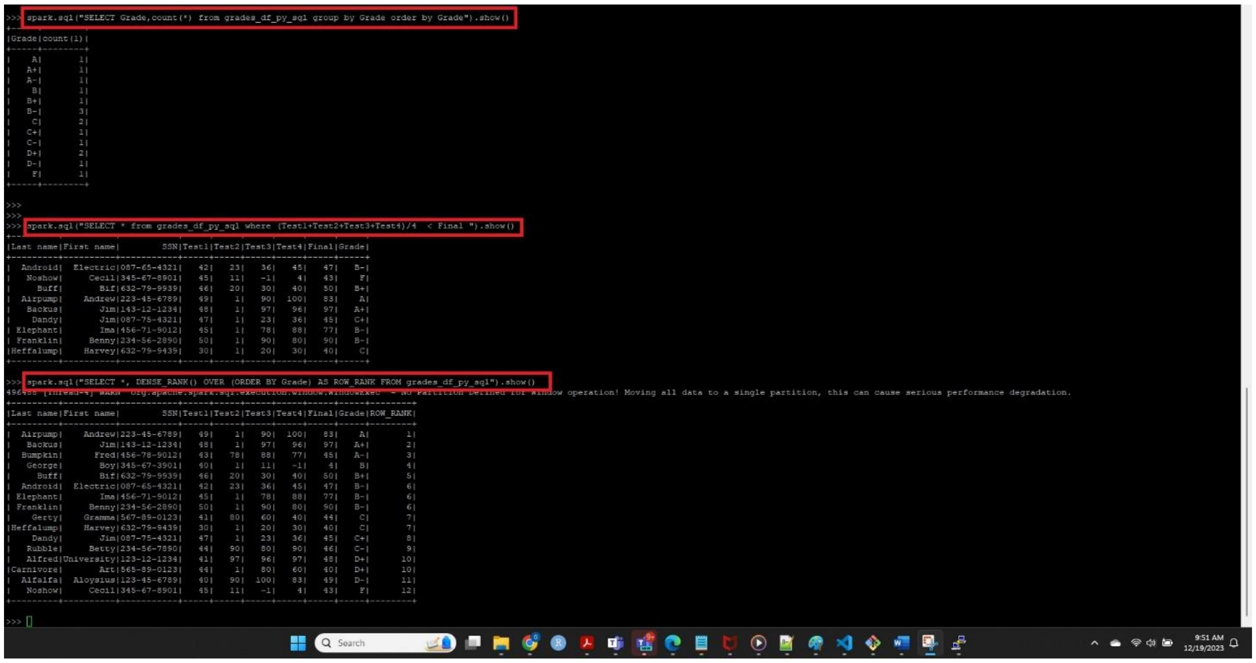
2.4. spark.sql("SELECT Grade,count(*) from grades_df_py_sql group by Grade order by Grade").show()
Significance: Returns the count of the number of students grouped by the Grade.

2.5. spark.sql("SELECT * from grades_df_py_sql where (Test1+Test2+Test3+Test4)/4 < Final ").show()
Significance: Returns the records where the average of Test1-4 scores is lesser than the Final scores.

2.6. spark.sql("SELECT *, DENSE_RANK() OVER (ORDER BY Grade) AS ROW_RANK FROM grades_df_py_sql").show()
Significance: This assigns Rank to the students ordered based on the Grades.



```
>>> spark.sql("SELECT * from grades_df_py_sql order by Final desc;").show()
+--
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
|   George|       Boy|345-67-3901|   40|    1|   11|   -1|    4|    B|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+

>>> spark.sql("SELECT * from grades_df_py_sql where Test1>=0 and Test2>=0 and Test3>=0 and Test4 >=0 and Final>=0").show()
+--
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+

>>> spark.sql("SELECT AVG(Test1),min(Test1),max(Test1),std(Test1) from grades_df_py_sql ").show()
+--
|avg(CAST(Test1 AS DOUBLE))|min(Test1)|max(Test1)|std(CAST(Test1 AS DOUBLE))|
+--------------------------+----------+----------+--------------------------+
|                   43.4375|        30|        50|         4.746489931166679|
+--------------------------+----------+----------+--------------------------+
```



```
>>> spark.sql("SELECT Grade,count(*) from grades_df_py_sql group by Grade order by Grade").show()
+--
|Grade|count(1)|
+-----+--------+
|    A|       1|
|   A+|       1|
|   A-|       1|
|    B|       1|
|   B+|       1|
|   B-|       3|
|    C|       2|
|   C+|       1|
|   C-|       1|
|   D+|       2|
|   D-|       1|
|    F|       1|
+-----+--------+

>>>
>>>
>>> spark.sql("SELECT * from grades_df_py_sql where (Test1+Test2+Test3+Test4)/4  < Final ").show()
+--
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+

>>> spark.sql("SELECT *, DENSE_RANK() OVER (ORDER BY Grade) AS ROW_RANK FROM grades_df_py_sql").show()
496[illegible] [Thread-4] WARN  org.apache.spark.sql.execution.window.WindowExec  - No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+--------+
|Last name|First name|        SSN|Test1|Test2|Test3|Test4|Final|Grade|ROW_RANK|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+--------+
|  Airpump|    Andrew|223-45-6789|   49|    1|   90|  100|   83|    A|       1|
|   Backus|       Jim|143-12-1234|   48|    1|   97|   96|   97|   A+|       2|
|  Bumpkin|      Fred|456-78-9012|   43|   78|   88|   77|   45|   A-|       3|
|   George|       Boy|345-67-3901|   40|    1|   11|   -1|    4|    B|       4|
|     Buff|       Bif|632-79-9939|   46|   20|   30|   40|   50|   B+|       5|
|  Android|  Electric|087-65-4321|   42|   23|   36|   45|   47|   B-|       6|
| Elephant|       Ima|456-71-9012|   45|    1|   78|   88|   77|   B-|       6|
| Franklin|     Benny|234-56-2890|   50|    1|   90|   80|   90|   B-|       6|
|    Gerty|    Gramma|567-89-0123|   41|   80|   60|   40|   44|    C|       7|
|Heffalump|    Harvey|632-79-9439|   30|    1|   20|   30|   40|    C|       7|
|    Dandy|       Jim|087-75-4321|   47|    1|   23|   36|   45|   C+|       8|
|   Rubble|     Betty|234-56-7890|   44|   90|   80|   90|   46|   C-|       9|
|   Alfred|University|123-12-1234|   41|   97|   96|   97|   48|   D+|      10|
|Carnivore|       Art|565-89-0123|   44|    1|   80|   60|   40|   D+|      10|
|  Alfalfa|  Aloysius|123-45-6789|   40|   90|  100|   83|   49|   D-|      11|
|   Noshow|     Cecil|345-67-8901|   45|   11|   -1|    4|   43|    F|      12|
+---------+----------+-----------+-----+-----+-----+-----+-----+-----+--------+

>>> 
```

3. **<u>SparkSQL</u> with Custom Data Set:**
   For this exercise, the Hollywood movies dataset used in the prior Hive assignment is reused.

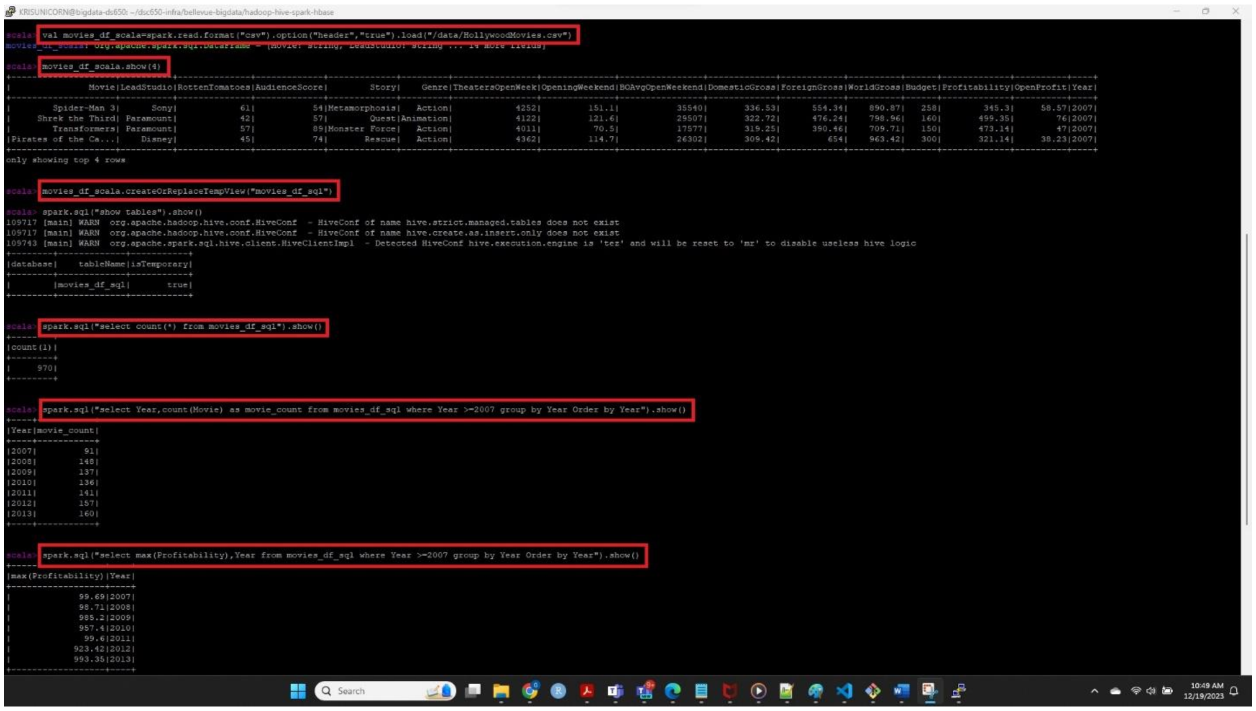
3.1. The description of some of the commonly used columns is explained below:

- ***Movie:*** Name of the movie
- ***LeadStudio:*** The Studio that produced the movie.
- ***RottenTomatoes:*** Represents the score rated by Rotten Tomatoes critics.
- ***AudienceScore***: Represents the score rated by Audience critics.
- ***Genre***: Represents the Genre of the movie
- ***TheatersOpenWeek***: Number of theatres where the movie was released in the opening week in the US and Canada. This is an integer value.

- ***OpeningWeekend***: Box Office collection in the opening week in the US and Canada in USD. This is a floating number value.
- ***DomesticGross:*** *Total domestic collection in the US and Canada represented in USD.*
- ***ForeignGross:*** *Total collection from countries other than the US and Canada represented in USD.*
- ***WorldGross:*** Sum of Domestic and Foreign Gross collection represented in USD.
- ***Budget:*** Total movie budget represented in USD.
- ***Profitability:*** Total profits made from the movie represented in Percentage.
- ***Year***: The year in which the movie was released.

3.2. Commands to copy the dataset into HDFS, start Spark shell and on **Scala**, and load the data into Data frame.

```
# Copying file to HDFS
hdfs dfs -put /data/HollywoodMovies.csv /data/HollywoodMovies.csv
hdfs dfs -ls  /data/HollywoodMovies.csv

# Starting Spark shell on Scala
spark-shell

# Creating a Dataframe using Spark Scala and printing sample rows:
val
movies_df_scala=spark.read.format("csv").option("header","true").load("/data/Hollywo
odMovies.csv")
movies_df_scala.show(4)

# Creating a View on the Dataframe to use Spark Sql:
movies_df_scala.createOrReplaceTempView("movies_df_sql")
spark.sql("show tables").show()
```



```
KRISUNICORN@bigdata-ds650: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase

scala> val movies_df_scala=spark.read.format("csv").option("header","true").load("/data/HollywoodMovies.csv")
movies_df_scala: org.apache.spark.sql.DataFrame = [Movie: string, LeadStudio: string ... 14 more fields]

scala> movies_df_scala.show(4)
+--------------------+----------+--------------+-------------+-------------+---------+----------------+--------------+----------------+-------------+------------+----------+------+-------------+----------+----+
|               Movie|LeadStudio|RottenTomatoes|AudienceScore|        Story|    Genre|TheatersOpenWeek|OpeningWeekend|BOAvgOpenWeekend|DomesticGross|ForeignGross|WorldGross|Budget|Profitability|OpenProfit|Year|
+--------------------+----------+--------------+-------------+-------------+---------+----------------+--------------+----------------+-------------+------------+----------+------+-------------+----------+----+
|        Spider-Man 3|      Sony|            61|           54|Metamorphosis|   Action|            4252|         151.1|           35540|       336.53|      554.34|    890.87|   258|        345.3|     58.57|2007|
|     Shrek the Third| Paramount|            42|           57|        Quest|Animation|            4122|         121.6|           29507|       322.72|      476.24|    798.96|   160|       499.35|        76|2007|
|        Transformers| Paramount|            57|           89|Monster Force|   Action|            4011|          70.5|           17577|       319.25|      390.46|    709.71|   150|       473.14|        47|2007|
|Pirates of the Ca...|    Disney|            45|           74|       Rescue|   Action|            4362|         114.7|           26302|       309.42|         654|    963.42|   300|       321.14|     38.23|2007|
+--------------------+----------+--------------+-------------+-------------+---------+----------------+--------------+----------------+-------------+------------+----------+------+-------------+----------+----+
only showing top 4 rows


scala> movies_df_scala.createOrReplaceTempView("movies_df_sql")

scala> spark.sql("show tables").show()
109717 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.strict.managed.tables does not exist
109717 [main] WARN  org.apache.hadoop.hive.conf.HiveConf  - HiveConf of name hive.create.as.insert.only does not exist
109743 [main] WARN  org.apache.spark.sql.hive.client.HiveClientImpl  - Detected HiveConf hive.execution.engine is 'tez' and will be reset to 'mr' to disable useless hive logic
+--------+-------------+-----------+
|database|    tableName|isTemporary|
+--------+-------------+-----------+
|        |movies_df_sql|       true|
+--------+-------------+-----------+


scala> spark.sql("select count(*) from movies_df_sql").show()
+--------+
|count(1)|
+--------+
|     970|
+--------+


scala> spark.sql("select Year,count(Movie) as movie_count from movies_df_sql where Year >=2007 group by Year Order by Year").show()
+----+-----------+
|Year|movie_count|
+----+-----------+
|2007|         91|
|2008|        148|
|2009|        137|
|2010|        136|
|2011|        141|
|2012|        157|
|2013|        160|
+----+-----------+


scala> spark.sql("select max(Profitability),Year from movies_df_sql where Year >=2007 group by Year Order by Year").show()
+------------------+----+
|max(Profitability)|Year|
+------------------+----+
|             99.69|2007|
|             98.71|2008|
|             985.2|2009|
|             957.4|2010|
|              99.6|2011|
|            923.42|2012|
|            993.35|2013|
+------------------+----+
```

3.3. Additional Scala Sql Commands used:

3.3.1. spark.sql("select count(*) from movies_df_sql").show()
Significance: Returns the count of number of records in the Dataframe.

3.3.2. spark.sql("select Year,count(Movie) as movie_count from movies_df_sql where Year >=2007 group by Year Order by Year").show()
Significance: The query returns the count of the number of movies from each year in the dataframe.

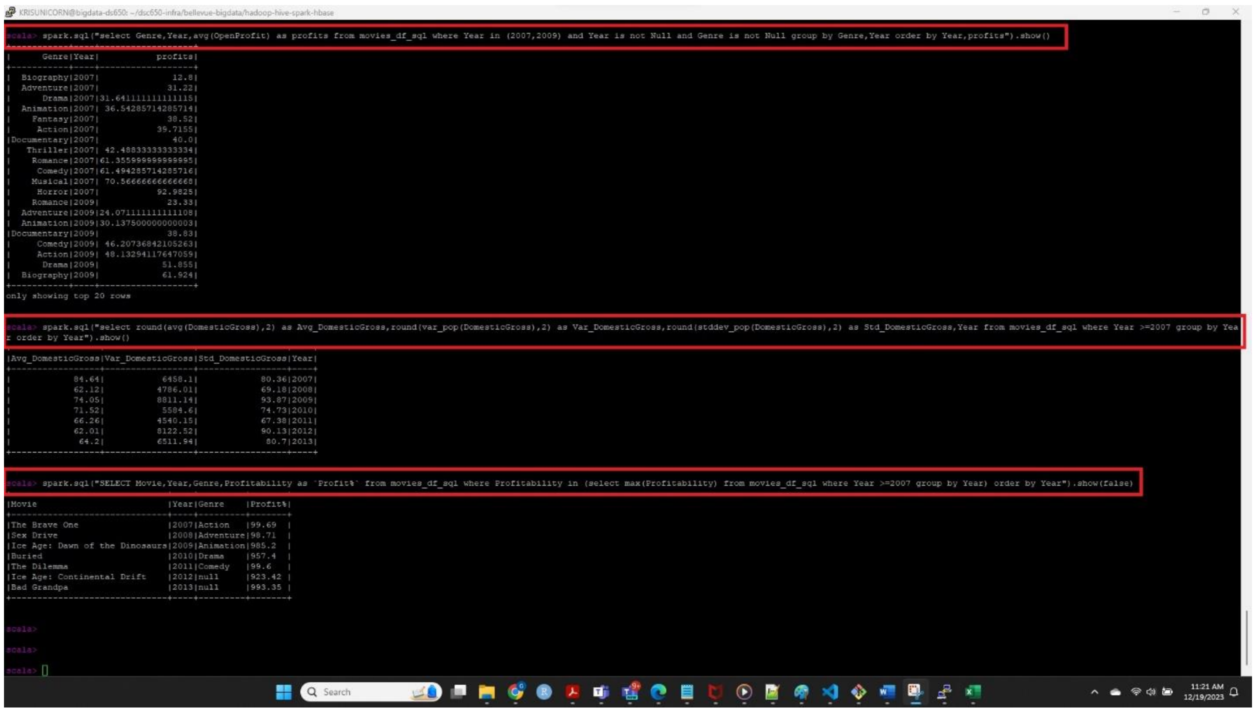
3.3.3. spark.sql("select max(Profitability),Year from movies_df_sql where Year >=2007 group by Year Order by Year").show()
Significance: Returns the maximum profitability achieved by a movie over the period of 2007-2013.

3.3.4. spark.sql("select Genre,Year,avg(OpenProfit) as profits from movies_df_sql where Year in (2007,2009) and Year is not Null and Genre is not Null group by Genre,Year order by Year,profits").show()
Significance: Returns the average Profits made in the years 2007 and 2009 by Genre, the results are ordered in ascending order of Year and average profit.

3.3.5. spark.sql("select round(avg(DomesticGross),2) as Avg_DomesticGross,round(var_pop(DomesticGross),2) as Var_DomesticGross,round(stddev_pop(DomesticGross),2) as Std_DomesticGross,Year from movies_df_sql where Year >=2007 group by Year order by Year").show()

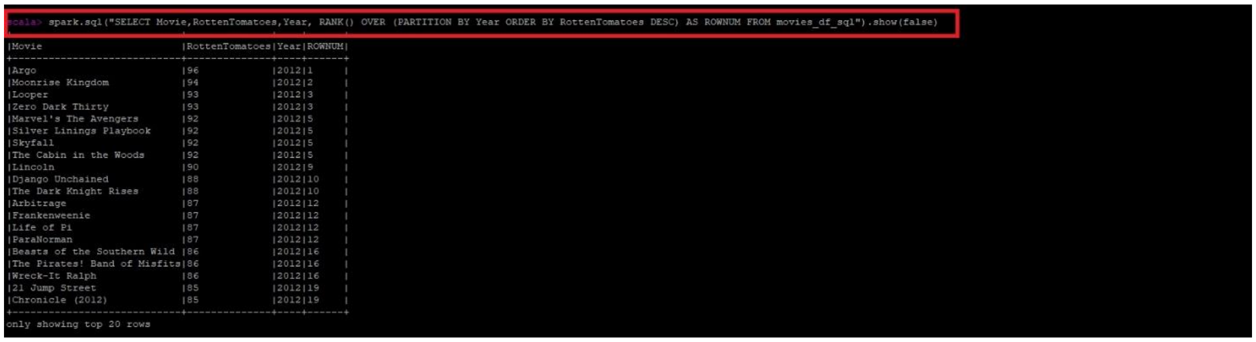
Significance: Returns the statistical quantities such as Mean, Variance, and standard deviation of Domestic Gross , arranged by Year.

3.3.6. spark.sql("SELECT Movie,Year,Genre,Profitability as `Profit%` from movies_df_sql where Profitability in (select max(Profitability) from movies_df_sql where Year >=2007 group by Year) order by Year").show(false)
Significance: Using Subqueries, the SQL returns the movie name and its Genre with the highest profit % for each year.



```
KRISUNICORN@bigdata-ds650: ~/dsc650-infra/bellevue-bigdata/hadoop-hive-spark-hbase
scala> spark.sql("select Genre,Year,avg(OpenProfit) as profits from movies_df_sql where Year in (2007,2009) and Year is not Null and Genre is not Null group by Genre,Year order by Year,profits").show()
+-----------+----+------------------+
|      Genre|Year|           profits|
+-----------+----+------------------+
|  Biography|2007|              12.8|
|  Adventure|2007|             31.22|
|      Drama|2007|31.641111111111115|
|  Animation|2007| 36.54285714285714|
|    Fantasy|2007|             38.52|
|     Action|2007|           39.7155|
|Documentary|2007|              40.0|
|   Thriller|2007| 42.48833333333334|
|    Romance|2007|61.355999999999995|
|     Comedy|2007|61.494285714285716|
|    Musical|2007| 70.56666666666668|
|     Horror|2007|           92.9825|
|    Romance|2009|             23.33|
|  Adventure|2009|24.071111111111108|
|  Animation|2009|30.137500000000003|
|Documentary|2009|             38.83|
|     Comedy|2009| 46.20736842105263|
|     Action|2009| 48.13294117647059|
|      Drama|2009|            51.855|
|  Biography|2009|            61.924|
+-----------+----+------------------+
only showing top 20 rows

scala> spark.sql("select round(avg(DomesticGross),2) as Avg_DomesticGross,round(var_pop(DomesticGross),2) as Var_DomesticGross,round(stddev_pop(DomesticGross),2) as Std_DomesticGross,Year from movies_df_sql where Year >=2007 group by Yea
r order by Year").show()
+-----------------+-----------------+-----------------+----+
|Avg_DomesticGross|Var_DomesticGross|Std_DomesticGross|Year|
+-----------------+-----------------+-----------------+----+
|            84.64|           6458.1|            80.36|2007|
|            62.12|          4786.01|            69.18|2008|
|            74.05|          8811.14|            93.87|2009|
|            71.52|           5584.6|            74.73|2010|
|            66.26|          4540.15|            67.38|2011|
|            62.01|          8122.52|            90.13|2012|
|             64.2|          6511.94|             80.7|2013|
+-----------------+-----------------+-----------------+----+

scala> spark.sql("SELECT Movie,Year,Genre,Profitability as `Profit%` from movies_df_sql where Profitability in (select max(Profitability) from movies_df_sql where Year >=2007 group by Year) order by Year").show(false)
+-----------------------------+----+---------+-------+
|Movie                        |Year|Genre    |Profit%|
+-----------------------------+----+---------+-------+
|The Brave One                |2007|Action   |99.69  |
|Sex Drive                    |2008|Adventure|98.71  |
|Ice Age: Dawn of the Dinosaurs|2009|Animation|985.2  |
|Buried                       |2010|Drama    |957.4  |
|The Dilemma                  |2011|Comedy   |99.6   |
|Ice Age: Continental Drift   |2012|null     |923.42 |
|Bad Grandpa                  |2013|null     |993.35 |
+-----------------------------+----+---------+-------+


scala>

scala>

scala> 
```

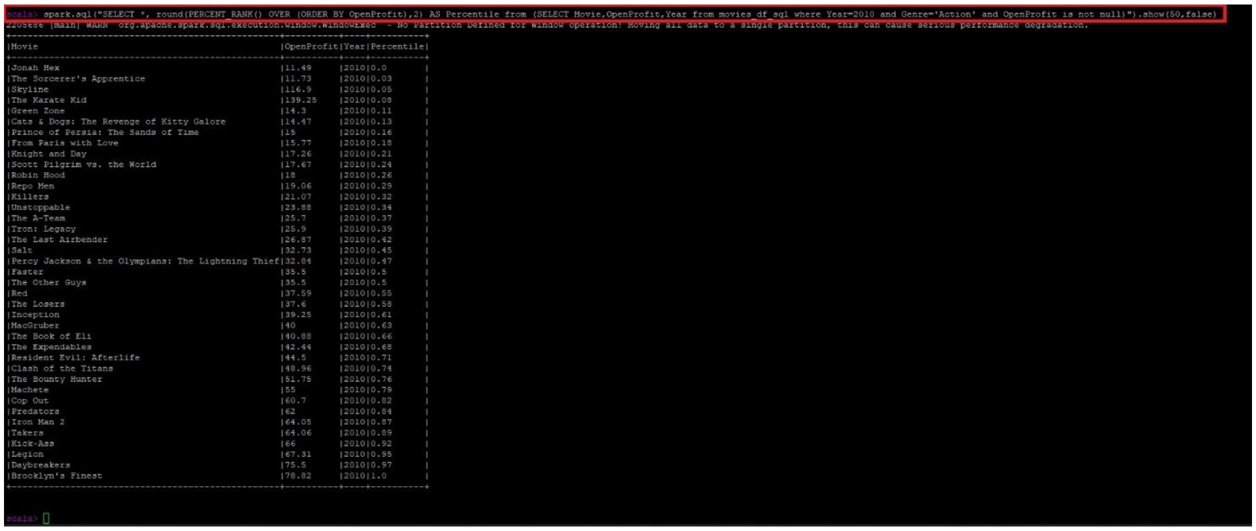
3.3.7. spark.sql("SELECT Movie,RottenTomatoes,Year, RANK() OVER (PARTITION BY Year ORDER BY RottenTomatoes DESC) AS ROWNUM FROM movies_df_sql").show(false)
Significance: Ranks the Movies based on the Rotten Tomatoes rating for each year.



```
scala> spark.sql("SELECT Movie,RottenTomatoes,Year, RANK() OVER (PARTITION BY Year ORDER BY RottenTomatoes DESC) AS ROWNUM FROM movies_df_sql").show(false)
+-----------------------------+--------------+----+------+
|Movie                        |RottenTomatoes|Year|ROWNUM|
+-----------------------------+--------------+----+------+
|Argo                         |96            |2012|1     |
|Moonrise Kingdom             |94            |2012|2     |
|Looper                       |93            |2012|3     |
|Zero Dark Thirty             |93            |2012|3     |
|Marvel's The Avengers        |92            |2012|5     |
|Silver Linings Playbook      |92            |2012|5     |
|Skyfall                      |92            |2012|5     |
|The Cabin in the Woods       |92            |2012|5     |
|Lincoln                      |90            |2012|9     |
|Django Unchained             |88            |2012|10    |
|The Dark Knight Rises        |88            |2012|10    |
|Arbitrage                    |87            |2012|12    |
|Frankenweenie                |87            |2012|12    |
|Life of Pi                   |87            |2012|12    |
|ParaNorman                   |87            |2012|12    |
|Beasts of the Southern Wild  |86            |2012|16    |
|The Pirates! Band of Misfits|86            |2012|16    |
|Wreck-It Ralph               |86            |2012|16    |
|21 Jump Street               |85            |2012|19    |
|Chronicle (2012)             |85            |2012|19    |
+-----------------------------+--------------+----+------+
only showing top 20 rows
```

3.3.8. spark.sql("SELECT *, round(PERCENT_RANK() OVER (ORDER BY OpenProfit),2) AS Percentile from (SELECT Movie,OpenProfit,Year from movies_df_sql where Year=2010 and Genre='Action' and OpenProfit is not null)").show(50, false)

Significance: Returns the percentile value of Action movies of 2010, based on the Profits made.



```
scala> spark.sql("SELECT *, round(PERCENT_RANK() OVER (ORDER BY OpenProfit),2) AS Percentile from (SELECT Movie,OpenProfit,Year from movies_df_sql where Year=2010 and Genre='Action' and OpenProfit is not null)").show(50,false)
2205686 [main] WARN  org.apache.spark.sql.execution.window.WindowExec  - No Partition Defined for Window operation! Moving all data to a single partition, this can cause serious performance degradation.
+--------------------------------------------------+----------+----+----------+
|Movie                                             |OpenProfit|Year|Percentile|
+--------------------------------------------------+----------+----+----------+
|Jonah Hex                                         |11.49     |2010|0.0       |
|The Sorcerer's Apprentice                         |11.73     |2010|0.03      |
|Skyline                                           |116.9     |2010|0.05      |
|The Karate Kid                                    |139.25    |2010|0.08      |
|Green Zone                                        |14.3      |2010|0.11      |
|Cats & Dogs: The Revenge of Kitty Galore          |14.47     |2010|0.13      |
|Prince of Persia: The Sands of Time               |15        |2010|0.16      |
|From Paris with Love                              |15.77     |2010|0.18      |
|Knight and Day                                    |17.26     |2010|0.21      |
|Scott Pilgrim vs. the World                       |17.67     |2010|0.24      |
|Robin Hood                                        |18        |2010|0.26      |
|Repo Men                                          |19.06     |2010|0.29      |
|Killers                                           |21.07     |2010|0.32      |
|Unstoppable                                       |23.88     |2010|0.34      |
|The A-Team                                        |25.7      |2010|0.37      |
|Tron: Legacy                                      |25.9      |2010|0.39      |
|The Last Airbender                                |26.87     |2010|0.42      |
|Salt                                              |32.73     |2010|0.45      |
|Percy Jackson & the Olympians: The Lightning Thief|32.84     |2010|0.47      |
|Faster                                            |35.5      |2010|0.5       |
|The Other Guys                                    |35.5      |2010|0.5       |
|Red                                               |37.59     |2010|0.55      |
|The Losers                                        |37.6      |2010|0.58      |
|Inception                                         |39.25     |2010|0.61      |
|MacGruber                                         |40        |2010|0.63      |
|The Book of Eli                                   |40.88     |2010|0.66      |
|The Expendables                                   |42.44     |2010|0.68      |
|Resident Evil: Afterlife                          |44.5      |2010|0.71      |
|Clash of the Titans                               |48.96     |2010|0.74      |
|The Bounty Hunter                                 |51.75     |2010|0.76      |
|Machete                                           |55        |2010|0.79      |
|Cop Out                                           |60.7      |2010|0.82      |
|Predators                                         |62        |2010|0.84      |
|Iron Man 2                                        |64.05     |2010|0.87      |
|Takers                                            |64.06     |2010|0.89      |
|Kick-Ass                                          |66        |2010|0.92      |
|Legion                                            |67.31     |2010|0.95      |
|Daybreakers                                       |75.5      |2010|0.97      |
|Brooklyn's Finest                                 |78.82     |2010|1.0       |
+--------------------------------------------------+----------+----+----------+


scala> 
```