```
import sys
from pyspark import SparkContext
from pyspark.sql import SQLContext
from pyspark.sql  import SparkSession
from pyspark.sql.types import *
from pyspark.sql.types import IntegerType




sc =SparkContext()
sqlContext = SQLContext(sc)
spark = SparkSession.builder.getOrCreate()

df = spark.read.format("csv").option("header", "true").load("C://Users//kurkumar//Desktop//EMP_BONUS.csv")

df.show()
colCasting = df.withColumn("sal", df["salary"].cast(IntegerType()))
colAddition =colCasting.withColumn("bonus",colCasting.sal/10).drop("salary")



colAddition.write.format('jdbc').options(url="jdbc:postgresql://localhost:5432/postgres",driver='org.postgresql.Driv
r',dbtable="employee",user="postgres",\
                                password="Welcome@123").mode('append').save()
```



```
-------------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------
EMP_NO,EMPLOYEE_NAME,DEPT,SALARY
123,Raju,10,10000
234,Ravi,20,10000
345,pawan,30,20000
456,kalyan,20,20000-




>>> df = spark.read.format("csv").option("header", "true").load("C://Users//kurkumar//Desktop//EMP_BONUS.csv
)
```

```
>>> df.collect()
[Row(EMP_NO='123', EMPLOYEE_NAME='Raju', DEPT='10', SALARY='10000'), Row(EMP_NO='234', EMPL
YEE_NAME='Ravi', DEPT='20', SALARY='10000'), Row(EMP_NO='345', EMPLOYEE_NAME='pawan', DEPT
'30', SALARY='20000'), Row(EMP_NO='456', EMPLOYEE_NAME='kalyan', DEPT='20', SALARY='20000')]

>>> df.show()
+------+-------------+----+------+
|EMP_NO|EMPLOYEE_NAME|DEPT|SALARY|
+------+-------------+----+------+
|   123|         Raju|  10| 10000|
|   234|         Ravi|  20| 10000|
|   345|        pawan|  30| 20000|
|   456|       kalyan|  20| 20000|
+------+-------------+----+------+



>>> df.printSchema()
root
 |-- EMP_NO: string (nullable = true)
 |-- EMPLOYEE_NAME: string (nullable = true)
 |-- DEPT: string (nullable = true)
 |-- SALARY: string (nullable = true)


>>> df.count()
4
>>> df.columns
['EMP_NO', 'EMPLOYEE_NAME', 'DEPT', 'SALARY']
>>> len(df.columns)
4
>>> df.describe('EMP_NO').show()
+-------+------------------+
|summary|            EMP_NO|
+-------+------------------+
|  count|                 4|
|   mean|             289.5|
| stddev|143.30038380967443|
|    min|               123|
|    max|               456|
+-------+------------------+

>>> df.describe('SALARY').show()
+-------+-----------------+
|summary|           SALARY|
+-------+-----------------+
|  count|                4|
|   mean|          15000.0|
| stddev|5773.502691896258|
|    min|            10000|
```

```
|   max|           20000|
+-------+-----------------+
```

```
>>> df.select('EMP_NO','EMPLOYEE_NAME').show(2)
+------+-------------+
|EMP_NO|EMPLOYEE_NAME|
+------+-------------+
|  123|         Raju|
|  234|         Ravi|
+------+-------------+
only showing top 2 rows
```

```
>>> df.select('EMP_NO','EMPLOYEE_NAME','SALARY').orderBy(df.SALARY).show(2)
+------+-------------+------+
|EMP_NO|EMPLOYEE_NAME|SALARY|
+------+-------------+------+
|  234|         Ravi| 10000|
|  123|         Raju| 10000|
+------+-------------+------+
only showing top 2 rows
```

```
>>> df.select('EMP_NO','EMPLOYEE_NAME','SALARY').orderBy(df.SALARY).show()
+------+-------------+------+
|EMP_NO|EMPLOYEE_NAME|SALARY|
+------+-------------+------+
|  123|         Raju| 10000|
|  234|         Ravi| 10000|
|  345|        pawan| 20000|
|  456|       kalyan| 20000|
+------+-------------+------+
```



```
>>> df.select('EMP_NO','EMPLOYEE_NAME','SALARY').orderBy(df.SALARY,ascending=False).show()
+------+-------------+------+
|EMP_NO|EMPLOYEE_NAME|SALARY|
+------+-------------+------+
|  345|        pawan| 20000|
|  456|       kalyan| 20000|
|  123|         Raju| 10000|
|  234|         Ravi| 10000|
+------+-------------+------+
```

```
>>> df.select('EMP_NO','EMPLOYEE_NAME','SALARY').orderBy(df.SALARY,ascending=False).show(2)
+------+-------------+------+
|EMP_NO|EMPLOYEE_NAME|SALARY|
+------+-------------+------+
|  456|       kalyan| 20000|
```

```
|   345|       pawan| 20000|
+------+------------+------+
only showing top 2 rows

>>> df.groupby('DeptNo').show()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'GroupedData' object has no attribute 'show'
>>> df.show()
+------+-------------+----+------+
|EMP_NO|EMPLOYEE_NAME|DEPT|SALARY|
+------+-------------+----+------+
|   123|         Raju|  10| 10000|
|   234|         Ravi|  20| 10000|
|   345|        pawan|  30| 20000|
|   456|       kalyan|  20| 20000|
+------+-------------+----+------+


>>> df.groupby('Dept').count().show()
+----+-----+
|Dept|count|
+----+-----+
|  10|    1|
|  20|    2|
|  30|    1|
+----+-----+

>>> df.groupby('Dept').count().show()
+----+-----+
|Dept|count|
+----+-----+
|  10|    1|
|  20|    2|
|  30|    1|
+----+-----+

>>>https://dzone.com/articles/pyspark-dataframe-tutorial-introduction-to-datafra

AssertionError: col should be Column



from pyspark.sql.functions import lit
>>> df.withColumn("Bonus",lit('1000'))

df.drop("Bonus").collect()

df2=df_new.withColumn("Bonus","sal"/10)

df_new = df.withColumn("sal", df["salary"].cast(IntegerType()))
```

```
from pyspark.sql.types import IntegerType
>>> df_new = df.withColumn("sal", df["salary"].cast(IntegerType()))


df_new.withColumn("Bonus",lit(1000))


df5 =df_new.withColumn("bonus",df_new.sal/10)
```