

# Answers

```
1. val df_rate = spark.read.format("csv").option("header",
"true").load("C:/Users/ABHINAV/Desktop/Topgear_ass/spark.txt")
```

```
C:\Windows\system32\cmd.exe - spark-shell
at org.apache.spark.deploy.SparkSubmit.doSubmit(SparkSubmit.scala:86)
at org.apache.spark.deploy.SparkSubmit$.anon$2.doSubmit(SparkSubmit.scala:920)
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:929)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-7V7LGEK:4040
Spark context available as 'sc' (master = local[*], app id = local-1597378486070).
Spark session available as 'spark'.
Welcome to

  ____              __
 / ___/  ___/  ___/  /_/_
/ /   / _ \/_ _/___/ __/
/ /___/ ___/____/____/
/____/

version 2.4.6

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 14.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val df_rate = spark.read.format("csv").option("header", "true").load("C:/Users/Abhinav Kumar/Desktop/Topgear_ass/spark.txt")
org.apache.spark.sql.AnalysisException: Path does not exist: file:/C:/Users/Abhinav Kumar/Desktop/Topgear_ass/spark.txt;
at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary$1.apply(DataSource.scala:558)
at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary$1.apply(DataSource.scala:545)
at scala.collection.TraversableLike$.anonfun$flatMap$1.apply(TraversableLike.scala:241)
at scala.collection.TraversableLike$.anonfun$flatMap$1.apply(TraversableLike.scala:241)
at scala.collection.immutable.List.foreach(List.scala:392)
at scala.collection.TraversableLike$class.flatMap(TraversableLike.scala:241)
at scala.collection.immutable.List.flatMap(List.scala:355)
at org.apache.spark.sql.execution.datasources.DataSource.org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary(DataSource.scala:545)
at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:359)
at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:223)
at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:211)
at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:178)
... 49 elided

scala> val df_rate = spark.read.format("csv").option("header", "true").load("C:/Users/ABHINAV/Desktop/Topgear_ass/spark.txt")
df_rate: org.apache.spark.sql.DataFrame = [userId: string, movieId: string ... 2 more fields]

scala>
```

```
2. df_rate.createOrReplaceTempView("rate")
```

```
C:\Windows\system32\cmd.exe - spark-shell
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:929)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-7V7LGEK:4040
Spark context available as 'sc' (master = local[*], app id = local-1597378486070).
Spark session available as 'spark'.
Welcome to

  ____              __
 / ___/  ___/  ___/  /_/_
/ /   / _ \/_ _/___/ __/
/ /___/ ___/____/____/
/____/

version 2.4.6

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 14.0.2)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val df_rate = spark.read.format("csv").option("header", "true").load("C:/Users/Abhinav Kumar/Desktop/Topgear_ass/spark.txt")
org.apache.spark.sql.AnalysisException: Path does not exist: file:/C:/Users/Abhinav Kumar/Desktop/Topgear_ass/spark.txt;
at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary$1.apply(DataSource.scala:558)
at org.apache.spark.sql.execution.datasources.DataSource$.anonfun$org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary$1.apply(DataSource.scala:545)
at scala.collection.TraversableLike$.anonfun$flatMap$1.apply(TraversableLike.scala:241)
at scala.collection.TraversableLike$.anonfun$flatMap$1.apply(TraversableLike.scala:241)
at scala.collection.immutable.List.foreach(List.scala:392)
at scala.collection.TraversableLike$class.flatMap(TraversableLike.scala:241)
at scala.collection.immutable.List.flatMap(List.scala:355)
at org.apache.spark.sql.execution.datasources.DataSource.org$apache$spark$sql$execution$datasources$DataSource$$checkAndGlobPathIfNecessary(DataSource.scala:545)
at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:359)
at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:223)
at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:211)
at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:178)
... 49 elided

scala> val df_rate = spark.read.format("csv").option("header", "true").load("C:/Users/ABHINAV/Desktop/Topgear_ass/spark.txt")
df_rate: org.apache.spark.sql.DataFrame = [userId: string, movieId: string ... 2 more fields]

scala> df_rate.createOrReplaceTempView("rate")

scala>
```

```
3. val df4 = spark.sql("SELECT INT(userId), INT(movieId), INT(rating), int(timestamp) from Rate")
```

```
C:\Windows\system32\cmd.exe - spark-shell

1| 231| 5| null|
1| 235| 4| null|
1| 260| 5| null|
1| 296| 3| null|
1| 316| 3| null|
1| 333| 5| null|
1| 349| 4| null|
-----+-----
only showing top 20 rows

scala> val df4 = spark.sql("SELECT INT(userId), INT(movieId), INT(rating), int(timestamp) from Rate")
df4: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more fields]

scala> df4.show()
-----+-----+-----+-----+
(userId)|(movieId)|(rating)|(timestamp)|
-----+-----+-----+-----+
1| 1| 4| 964982703|
1| 3| 4| 964981247|
1| 6| 4| 964982224|
1| 47| 5| 964983815|
1| 50| 5| 964982931|
1| 70| 3| 964982400|
1| 101| 5| 964808688|
1| 110| 4| 964982176|
1| 151| 5| 964984041|
1| 157| 5| 964984100|
1| 163| 5| 964983650|
1| 216| 5| 964981208|
1| 223| 3| 964980985|
1| 231| 5| 964981179|
1| 235| 4| 964808908|
1| 260| 5| 964981680|
1| 296| 3| 964982967|
1| 316| 3| 964982310|
1| 333| 5| 964981179|
1| 349| 4| 964982563|
-----+-----+-----+-----+
only showing top 20 rows

scala>
```

4. val df5 = spark.sql("SELECT movieId, avg(rating) from Rate group by movieId")

```
C:\Windows\system32\cmd.exe - spark-shell

3.3333333333333335|
3.75|
4.0|
3.625|
3.75|
4.5|
4.5|
-----+-----
only showing top 20 rows

scala> val df5 = spark.sql("SELECT movieId, avg(rating) from Rate group by movieId")
df5: org.apache.spark.sql.DataFrame = [movieId: string, avg(CAST(rating AS DOUBLE)): double]

scala> df5.show()
-----+-----+
(movieId)|(avg(CAST(rating AS DOUBLE)))|
-----+-----+
206| 4.197068403908795|
1090| 3.984126984126984|
115713| 3.9107142857142856|
3210| 3.4761904761904763|
88140| 3.546875|
829| 2.6666666666666665|
2088| 2.5|
2294| 3.2444444444444445|
4821| 3.1|
48738| 3.975|
3959| 3.625|
89864| 3.6315789473684212|
2136| 2.4642857142857144|
691| 3.3333333333333335|
3606| 3.75|
121907| 4.0|
6731| 3.625|
27317| 3.75|
26082| 4.5|
100553| 4.5|
-----+-----+
only showing top 20 rows

scala>
```

5. val df5 = spark.sql("SELECT movieId, count(movieId) as co from Rate group by movieId order by co desc limit 1 ")

```
C:\Windows\system32\cmd.exe - spark-shell
only showing top 20 rows

scala> val df5 = spark.sql("SELECT movieId, count(movieId) as co from Rate group by movieId order by co desc limit by 1 ")
org.apache.spark.sql.catalyst.parser.ParseException:
extraneous input '1' expecting {<EOF>, ';'}(line 1, pos 90)

== SQL ==
SELECT movieId, count(movieId) as co from Rate group by movieId order by co desc limit by 1
-----^^^

at org.apache.spark.sql.catalyst.parser.ParseException.withCommand(ParseDriver.scala:266)
at org.apache.spark.sql.catalyst.parser.AbstractSqlParser.parse(ParseDriver.scala:133)
at org.apache.spark.sql.execution.SparkSqlParser.parse(SparkSqlParser.scala:48)
at org.apache.spark.sql.catalyst.parser.AbstractSqlParser.parsePlan(ParseDriver.scala:81)
at org.apache.spark.sql.SparkSession.$anonfun$sql$2(SparkSession.scala:604)
at org.apache.spark.sql.catalyst.QueryPlanningTracker.measurePhase(QueryPlanningTracker.scala:111)
at org.apache.spark.sql.SparkSession.$anonfun$sql$1(SparkSession.scala:604)
at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:763)
at org.apache.spark.sql.SparkSession.sql(SparkSession.scala:601)
... 47 elided

scala> val df5 = spark.sql("SELECT movieId, count(movieId) as co from Rate group by movieId order by co desc limit 1 ")
df5: org.apache.spark.sql.DataFrame = [movieId: string, co: bigint]

scala> df5.show()
+-----+----+
|movieId| co |
+-----+----+
|      356|329|
+-----+----+

scala>
```

6. val df5 = spark.sql("SELECT userId, avg(rating) as user\_rate from Rate group by userId order by user\_rate ")

```
C:\Windows\system32\cmd.exe - spark-shell

110|3.7254901960784315|
111|3.3397832817337463|
112|3.5923076923076924|
113|3.6466666666666665|
114|3.435483870967742|
115|3.767857142857143|
116|3.4367816091954024|
+-----+-----+
only showing top 20 rows

scala> val df5 = spark.sql("SELECT userId, avg(rating) as user_rate from Rate group by userId order by user_rate ")
df5: org.apache.spark.sql.DataFrame = [userId: string, user_rate: double]

scala> df5.show()
+-----+-----+
|userId| user_rate|
+-----+-----+
|    442|    1.275|
|    139|2.1443298969072164|
|    508|2.1458333333333335|
|    153|2.217877094972067|
|    567|2.2454545454545456|
|    311|2.3392857142857144|
|    298|2.363684771033014|
|    517|    2.38625|
|    308|2.4268869565217393|
|    3|2.4358974358974357|
|    255|2.5681818181818183|
|    22|2.5714285714285716|
|    571|2.5714285714285716|
|    297|2.5972222222222223|
|    19|2.607396870554765|
|    294|2.610983981693364|
|    293|2.619047619047619|
|    287|2.6217105263157894|
|    36|2.6333333333333333|
|    333|    2.64|
+-----+-----+
only showing top 20 rows

scala>
```