Untitled Session

By wawan — Session (null)

```
>

import pyspark

>

from pyspark.sql import SparkSession

>

from pyspark.sql.types import StructType, StructField, IntegerType, StringType

>

spark = SparkSession.builder.appName("ReadHiveTable").getOrCreate()

Setting spark.hadoop.yarn.resourcemanager.principal to wawan
Setting spark.hadoop.yarn.resourcemanager.principal to wawan

>

spark.sql("SHOW DATABASES")

Hive Session ID = e132468a-30b1-46c1-ba14-ba40a0524902
Hive Session ID = e132468a-30b1-46c1-ba14-ba40a0524902

DataFrame[namespace: string]

>

df = spark.sql("SELECT * FROM mall_customers")
```

AnalysisException: Table or view not found: mall_customers; line 1 pos 14; 'Project [*] +- 'UnresolvedRelation [mall_customers], [], false

AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 df = spark.sql("SELECT * FROM mall_customers") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame` representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'), Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command = proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303 answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value(1305 answer, self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File /opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw) 113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 # Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise converted from None 118 else: 119 raise AnalysisException: Table or view not found: mall_customers; line 1 pos 14; 'Project [*] +- 'UnresolvedRelation [mall_customers], [], false

```
>

df = spark.sql("SELECT * FROM testing_wawan.mall_customers")
```

AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers`. Clients can access this table only if they have the following capabilities: CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not implement

AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 df = spark.sql("SELECT * FROM testing_wawan.mall_customers") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame` representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'), Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command = proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303 answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value(1305 answer, self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File /opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw) 113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 # Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise converted from None 118 else: 119 raise AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers`. Clients can access this table only if they have the following capabilities: CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not implement

```
>

spark.sql("SELECT * FROM testing_wawan.mall_customers")
```

```
AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers`. Clients can access this
table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 spark.sql("SELECT * FROM
testing_wawan.mall_customers") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise AnalysisException: Spark has no access to table
`testing_wawan`.`mall_customers`. Clients can access this table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

>

```
spark.sql("SHOW DATABASES").show()
```

```
+-------------------+
|          namespace|
+-------------------+
|       data_pii_pst|
|       data_pii_stg|
|      data_platform|
|     datalake_ozone|
|       datalake_pst|
|       datalake_stg|
|            default|
|  development_hbase|
|   development_hive|
|development_phoenix|
| information_schema|
|      poc_bsim_test|
|        replication|
|                sys|
|     testing_ruslan|
|      testing_wawan|
|         testranger|
+-------------------+
```

>

```
spark.sql("SELECT * FROM testing_wawan.mall_customers").show()
```

```
AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers`. Clients can access this
table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 spark.sql("SELECT * FROM
testing_wawan.mall_customers").show() File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise AnalysisException: Spark has no access to table
`testing_wawan`.`mall_customers`. Clients can access this table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

>

```
df = spark.sql("SELECT * FROM `testing`_wawan`.`mall_customers`")
```

```
ParseException: mismatched input '`.`' expecting {<EOF>, ';'}(line 1, pos 29) == SQL == SELECT * FROM
`testing`_wawan`.`mall_customers` -----------------------------^^^
```

```
❌ParseException Traceback (most recent call last) Cell In[1], line 1 ----> 1 df = spark.sql("SELECT * FROM
`testing`_wawan`.`mall_customers`") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise ParseException: mismatched input '`.`' expecting {<EOF>, ';'}(line 1,
pos 29) == SQL == SELECT * FROM `testing`_wawan`.`mall_customers` -----------------------------^^^
>
```

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers`")
```

```
❌AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers`. Clients can access this
table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
❌AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 spark.sql("SELECT * FROM
`testing_wawan`.`mall_customers`") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise AnalysisException: Spark has no access to table
`testing_wawan`.`mall_customers`. Clients can access this table only if they have the following capabilities:
CONNECTORREAD,HIVEMANAGEDINSERTREAD,HIVEMANAGEDINSERTWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE.
This table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

```
24/10/14 06:55:43 013 ERROR UserGroupInformation: TGT is expired. Aborting renew thread for wawan@WLEOWLEO.UK.
24/10/14 06:55:43 013 ERROR UserGroupInformation: TGT is expired. Aborting renew thread for wawan@WLEOWLEO.UK.

>
```

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_acid`")
```

```
❌AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers_acid`. Clients can access
this table only if they have the following capabilities:
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
❌AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 spark.sql("SELECT * FROM
`testing_wawan`.`mall_customers_acid`") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise AnalysisException: Spark has no access to table
`testing_wawan`.`mall_customers_acid`. Clients can access this table only if they have the following
capabilities:
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
>
```

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_acid`").show(5)
```

```
❌AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers_acid`. Clients can access
this table only if they have the following capabilities:
```

```
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

```
AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 spark.sql("SELECT * FROM
`testing_wawan`.`mall_customers_acid`").show(5) File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in SparkSession.sql(self, sqlQuery) 707 def
sql(self, sqlQuery): 708 """Returns a :class:`DataFrame` representing the result of the given query. 709 710
.. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'), Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """
--> 723 return DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File
/usr/local/lib/python3.9/site-packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298
command = proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301
proto.END_COMMAND_PART 1303 answer = self.gateway_client.send_command(command) -> 1304 return_value =
get_return_value( 1305 answer, self.gateway_client, self.target_id, self.name) 1307 for temp_arg in
temp_args: 1308 temp_arg._detach() File /opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in
capture_sql_exception.<locals>.deco(*a, **kw) 113 converted = convert_exception(e.java_exception) 114 if not
isinstance(converted, UnknownException): 115 # Hide where the exception came from that shows a non-Pythonic
116 # JVM exception message. --> 117 raise converted from None 118 else: 119 raise AnalysisException: Spark
has no access to table `testing_wawan`.`mall_customers_acid`. Clients can access this table only if they have
the following capabilities:
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

>

```
import pyspark
```

>

```
from pyspark.sql import SparkSession
```

>

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType
```

>

```
spark
```

SparkSession - hive

SparkContext

Spark UI

Version
    v3.2.3.1.20.7172000.0-74
Master
    k8s://https://10.43.0.1:443
AppName
    ReadHiveTable

>

```
df = spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_acid`")
```

```
AnalysisException: Spark has no access to table `testing_wawan`.`mall_customers_acid`. Clients can access
this table only if they have the following capabilities:
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

```
AnalysisException Traceback (most recent call last) Cell In[1], line 1 ----> 1 df = spark.sql("SELECT * FROM
`testing_wawan`.`mall_customers_acid`") File /opt/spark/python/lib/pyspark.zip/pyspark/sql/session.py:723, in
SparkSession.sql(self, sqlQuery) 707 def sql(self, sqlQuery): 708 """Returns a :class:`DataFrame`
representing the result of the given query. 709 710 .. versionadded:: 2.0.0 (...) 721 [Row(f1=1, f2='row1'),
Row(f1=2, f2='row2'), Row(f1=3, f2='row3')] 722 """ --> 723 return
DataFrame(self._jsparkSession.sql(sqlQuery), self._wrapped) File /usr/local/lib/python3.9/site-
packages/py4j/java_gateway.py:1304, in JavaMember.__call__(self, *args) 1298 command =
proto.CALL_COMMAND_NAME +\ 1299 self.command_header +\ 1300 args_command +\ 1301 proto.END_COMMAND_PART 1303
answer = self.gateway_client.send_command(command) -> 1304 return_value = get_return_value( 1305 answer,
self.gateway_client, self.target_id, self.name) 1307 for temp_arg in temp_args: 1308 temp_arg._detach() File
/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py:117, in capture_sql_exception.<locals>.deco(*a, **kw)
113 converted = convert_exception(e.java_exception) 114 if not isinstance(converted, UnknownException): 115 #
Hide where the exception came from that shows a non-Pythonic 116 # JVM exception message. --> 117 raise
converted from None 118 else: 119 raise AnalysisException: Spark has no access to table
`testing_wawan`.`mall_customers_acid`. Clients can access this table only if they have the following
capabilities:
CONNECTORREAD,HIVEFULLACIDREAD,HIVEFULLACIDWRITE,HIVEMANAGESTATS,HIVECACHEINVALIDATE,CONNECTORWRITE. This
table may be a Hive-managed ACID table, or require some other capability that Spark currently does not
implement
```

>

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_external`").show(5)
```

```
[Stage 0:>                                                          (0 + 0) / 1]

[Stage 0:>                                                          (0 + 0) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

[Stage 0:>                                                          (0 + 1) / 1]

+----------+------+----+------------------+----------------------+
|customerid|gender| age|annual income (k$)|spending score (1-100)|
+----------+------+----+------------------+----------------------+
|      null|Gender|null|              null|                  null|
|         1|  Male|  19|                15|                    39|
|         2|  Male|  21|                15|                    81|
|         3|Female|  20|                16|                     6|
|         4|Female|  23|                16|                    77|
+----------+------+----+------------------+----------------------+
only showing top 5 rows
```

>

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_external`")
```

```
DataFrame[customerid: int, gender: string, age: int, annual income (k$): int, spending score (1-100): int]
```

>

```
spark.sql("SELECT * FROM `testing_wawan`.`mall_customers_external`").show(5)
```

```
+----------+------+----+------------------+----------------------+
|customerid|gender| age|annual income (k$)|spending score (1-100)|
+----------+------+----+------------------+----------------------+
|      null|Gender|null|              null|                  null|
|         1|  Male|  19|                15|                    39|
|         2|  Male|  21|                15|                    81|
|         3|Female|  20|                16|                     6|
|         4|Female|  23|                16|                    77|
+----------+------+----+------------------+----------------------+
only showing top 5 rows
```

```
Console will exit automatically if it remains idle for another sixty seconds.
Engine exited with status 129.
```