In [1]:
```
# Import necessary module
import pandas as pd
import numpy as np
import mysql.connector
from mysql.connector import errorcode
from sqlalchemy import create_engine
```

In [3]:
```
DATABASE_URI='mysql+mysqlconnector://{user}:{password}@{server}/{database}'.fo
rmat(user='root', password='', server='localhost', database='netflixstudy')

# engine = create_engine("mysql://root:@localhost/netflixstudy",echo = True)
engine = create_engine(DATABASE_URI,echo = True)
```

In [4]:
```
df = pd.read_csv('../data/processed/df.csv')
```

In [ ]:
```
tbl = 'ratings'
# load data local INFILE 'C:\Users\sb\OneDrive\Documents\BowlerConsulting\Soft
wareDevelopment\UTRGVmsit\CSCI6370\netflixstudy\data\processed\df.csv'
#       INTO TABLE `ratings` FIELDS TERMINATED BY ',' (,Cust_Id,Rating,Movie_I
d);
```

In [6]:
```
df.to_sql(tbl, engine, index=False, if_exists="replace")
```

```
2020-10-07 07:45:13,435 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIK
E 'sql_mode'
2020-10-07 07:45:13,446 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,453 INFO sqlalchemy.engine.base.Engine SHOW VARIABLES LIK
E 'lower_case_table_names'
2020-10-07 07:45:13,454 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,460 INFO sqlalchemy.engine.base.Engine SELECT DATABASE()
2020-10-07 07:45:13,461 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,468 INFO sqlalchemy.engine.base.Engine SELECT CAST('test
plain returns' AS CHAR(60)) AS anon_1
2020-10-07 07:45:13,470 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,480 INFO sqlalchemy.engine.base.Engine SELECT CAST('test
unicode returns' AS CHAR(60)) AS anon_1
2020-10-07 07:45:13,481 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,494 INFO sqlalchemy.engine.base.Engine DESCRIBE `ratings`
2020-10-07 07:45:13,497 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,511 INFO sqlalchemy.engine.base.Engine DESCRIBE `ratings`
2020-10-07 07:45:13,513 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,528 INFO sqlalchemy.engine.base.Engine SHOW FULL TABLES F
ROM `netflixstudy`
2020-10-07 07:45:13,529 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,564 INFO sqlalchemy.engine.base.Engine SHOW CREATE TABLE
`ratings`
2020-10-07 07:45:13,564 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,570 INFO sqlalchemy.engine.base.Engine
DROP TABLE ratings
2020-10-07 07:45:13,572 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,610 INFO sqlalchemy.engine.base.Engine COMMIT
2020-10-07 07:45:13,616 INFO sqlalchemy.engine.base.Engine
CREATE TABLE ratings (
        `Unnamed: 0` BIGINT,
        `Cust_Id` BIGINT,
        `Rating` FLOAT(53),
        `Movie_Id` BIGINT
)


2020-10-07 07:45:13,617 INFO sqlalchemy.engine.base.Engine {}
2020-10-07 07:45:13,643 INFO sqlalchemy.engine.base.Engine COMMIT
2020-10-07 07:46:12,004 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2020-10-07 18:14:02,590 INFO sqlalchemy.engine.base.Engine INSERT INTO rating
s (`Unnamed: 0`, `Cust_Id`, `Rating`, `Movie_Id`) VALUES (%(Unnamed: 0)s, %(C
ust_Id)s, %(Rating)s, %(Movie_Id)s)
2020-10-07 18:14:02,742 INFO sqlalchemy.engine.base.Engine ({'Unnamed: 0': 69
6, 'Cust_Id': 712664, 'Rating': 5.0, 'Movie_Id': 3}, {'Unnamed: 0': 697, 'Cus
t_Id': 1331154, 'Rating': 4.0, 'Movie_Id': 3}, {'Unnamed: 0': 698, 'Cust_Id':
2632461, 'Rating': 3.0, 'Movie_Id': 3}, {'Unnamed: 0': 699, 'Cust_Id': 44937,
'Rating': 5.0, 'Movie_Id': 3}, {'Unnamed: 0': 700, 'Cust_Id': 656399, 'Ratin
g': 4.0, 'Movie_Id': 3}, {'Unnamed: 0': 701, 'Cust_Id': 439011, 'Rating': 1.
0, 'Movie_Id': 3}, {'Unnamed: 0': 703, 'Cust_Id': 1644750, 'Rating': 3.0, 'Mo
vie_Id': 3}, {'Unnamed: 0': 704, 'Cust_Id': 2031561, 'Rating': 4.0, 'Movie_I
d': 3}  ... displaying 10 of 71833509 total bound parameter sets ...  {'Unnam
ed: 0': 100497352, 'Cust_Id': 1922916, 'Rating': 3.0, 'Movie_Id': 17769}, {'U
nnamed: 0': 100497354, 'Cust_Id': 1704416, 'Rating': 2.0, 'Movie_Id': 17769})
2020-10-07 18:41:45,498 INFO sqlalchemy.engine.base.Engine ROLLBACK
```

```
---------------------------------------------------------------------------
OverflowError                             Traceback (most recent call last)
<ipython-input-6-be99aa867443> in <module>
----> 1 df.to_sql(tbl, engine, index=False, if_exists="replace")

~\anaconda3\lib\site-packages\pandas\core\generic.py in to_sql(self, name, co
n, schema, if_exists, index, index_label, chunksize, dtype, method)
   2651         from pandas.io import sql
   2652 
-> 2653         sql.to_sql(
   2654             self,
   2655             name,

~\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql(frame, name, con, sc
hema, if_exists, index, index_label, chunksize, dtype, method)
    510         )
    511 
--> 512     pandas_sql.to_sql(
    513         frame,
    514         name,

~\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql(self, frame, name, i
f_exists, index, index_label, schema, chunksize, dtype, method)
   1315         )
   1316         table.create()
-> 1317         table.insert(chunksize, method=method)
   1318         if not name.isdigit() and not name.islower():
   1319             # check for potentially case sensitivity issues (GH7815)

~\anaconda3\lib\site-packages\pandas\io\sql.py in insert(self, chunksize, met
hod)
    753 
    754                 chunk_iter = zip(*[arr[start_i:end_i] for arr in data
_list])
--> 755                 exec_insert(conn, keys, chunk_iter)
    756 
    757     def _query_iterator(

~\anaconda3\lib\site-packages\pandas\io\sql.py in _execute_insert(self, conn,
keys, data_iter)
    667         """
    668         data = [dict(zip(keys, row)) for row in data_iter]
--> 669         conn.execute(self.table.insert(), data)
    670 
    671     def _execute_insert_multi(self, conn, keys, data_iter):

~\anaconda3\lib\site-packages\sqlalchemy\engine\base.py in execute(self, obje
ct_, *multiparams, **params)
   1012             )
   1013         else:
-> 1014             return meth(self, multiparams, params)
   1015 
   1016     def _execute_function(self, func, multiparams, params):

~\anaconda3\lib\site-packages\sqlalchemy\sql\elements.py in _execute_on_conne
ction(self, connection, multiparams, params)
    296     def _execute_on_connection(self, connection, multiparams, params)
```

```
:
    297         if self.supports_execution:
--> 298             return connection._execute_clauseelement(self, multiparam
s, params)
    299         else:
    300             raise exc.ObjectNotExecutableError(self)

~\anaconda3\lib\site-packages\sqlalchemy\engine\base.py in _execute_clauseele
ment(self, elem, multiparams, params)
   1125             )
   1126 
-> 1127         ret = self._execute_context(
   1128             dialect,
   1129             dialect.execution_ctx_cls._init_compiled,

~\anaconda3\lib\site-packages\sqlalchemy\engine\base.py in _execute_context(s
elf, dialect, constructor, statement, parameters, *args)
   1315 
   1316         except BaseException as e:
-> 1317             self._handle_dbapi_exception(
   1318                 e, statement, parameters, cursor, context
   1319             )

~\anaconda3\lib\site-packages\sqlalchemy\engine\base.py in _handle_dbapi_exce
ption(self, e, statement, parameters, cursor, context)
   1513                 )
   1514             else:
-> 1515                 util.raise_(exc_info[1], with_traceback=exc_info[2])
   1516 
   1517         finally:

~\anaconda3\lib\site-packages\sqlalchemy\util\compat.py in raise_(***failed r
esolving arguments***)
    176 
    177         try:
--> 178             raise exception
    179         finally:
    180             # credit to

~\anaconda3\lib\site-packages\sqlalchemy\engine\base.py in _execute_context(s
elf, dialect, constructor, statement, parameters, *args)
   1255                             break
   1256                 if not evt_handled:
-> 1257                     self.dialect.do_executemany(
   1258                         cursor, statement, parameters, context
   1259                     )

~\anaconda3\lib\site-packages\sqlalchemy\engine\default.py in do_executemany
(self, cursor, statement, parameters, context)
    588 
    589     def do_executemany(self, cursor, statement, parameters, context=N
one):
--> 590         cursor.executemany(statement, parameters)
    591 
    592     def do_execute(self, cursor, statement, parameters, context=None)
:
```

```
~\anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in executemany(s
elf, operation, seq_params)
    350             stmt = self._batch_insert(operation, seq_params)
    351             if stmt is not None:
--> 352                 return self.execute(stmt)
    353 
    354         rowcnt = 0

~\anaconda3\lib\site-packages\mysql\connector\cursor_cext.py in execute(self,
operation, params, multi)
    262 
    263         try:
--> 264             result = self._cnx.cmd_query(stmt, raw=self._raw,
    265                                          buffered=self._buffered,
    266                                          raw_as_string=self._raw_as_s
tring)

~\anaconda3\lib\site-packages\mysql\connector\connection_cext.py in cmd_query
(self, query, raw, buffered, raw_as_string)
    485             if not isinstance(query, bytes):
    486                 query = query.encode('utf-8')
--> 487             self._cmysql.query(query,
    488                                raw=raw, buffered=buffered,
    489                                raw_as_string=raw_as_string)

OverflowError: size does not fit in an int
```