Hello,
You can find some demos here as well as information for accessing the database and such. Here is our account information.

gmail:
login: votetrackr18@gmail.com
password: thisis220

aws:
login: votetrackr18@gmail.com
password: ThisIs220

If you're planning on doing a lot of queries or modifying the tables a lot, downloading MySQL Workbench can be very helpful. This link: (https://www.linkedin.com/pulse/connecting-aws-rds-mysql-workbench-suresh-kumar/) has a pretty good step by step process for connecting the workbench to the RDS.

I'm figuring out most of this as I go, but feel free to hmu if you have any questions or problems regarding sending queries or accessing the db.
Best,
Hasmik

```
import pandas as pd
import pymysql
```

Connection information extracted from aws console
Most info here can be found under RDS->Instances->votetracker-db

```
host="votetrackr-db.cv1xcgegsskz.us-east-2.rds.amazonaws.com"
port=3306
dbname="votetrackr"
user="VoteTrackrMaster"
password="VotePass"
```

__Establishing a connection__
(Let Hasmik know if this step fails; it usually only fails if there's a problem with the EC2 or the RDS)

```
conn = pymysql.connect(host, user=user,port=port, passwd=password, db=dbname)
```

Let's see what tables there are in the votetrackr schema

```
cursor = conn.cursor()     # get the cursor
```

```
cursor.execute("USE votetrackr") # select the database

print("There are", cursor.execute("SHOW TABLES"), "tables")    # execute 'SHOW
TABLES' (but data is not returned)
print("Their names are", cursor.fetchall())
```

```
There are 4 tables
Their names are (('Bills',), ('Legislators',), ('Users',), ('Votes',))
```

__How to query__

note: as I'm writing the demo the db is empty, so the commands return empty tables

```
pandas_df = pd.read_sql('select * from Bills;', con=conn)
pandas_df
```

.dataframe tbody tr th:only-of-type { vertical-align: middle; } .dataframe tbody tr th { vertical-align: top; } .dataframe thead th { text-align: right; }

BID	Description	Category	DateIntroduced	Status	VotedOn	CongressN	Chamber	Session	DateVoted	RollCallID	URL

```
pandas_df = pd.read_sql('select BID, Description from Bills;', con=conn)
pandas_df
```

.dataframe tbody tr th:only-of-type { vertical-align: middle; } .dataframe tbody tr th { vertical-align: top; } .dataframe thead th { text-align: right; }

BID	Description

```
pandas_df = pd.read_sql('select BID from Bills where VotedOn=1;', con=conn)
pandas_df
```

.dataframe tbody tr th:only-of-type { vertical-align: middle; } .dataframe tbody tr th { vertical-align: top; } .dataframe thead th { text-align: right; }

BID

__To insert a row in the table__

In general, most queries can be executed by cursor.execute(query, args)

query being a string of the query

```
ID = 123
LorU = 1
BID = "testID321"
YayNayIdc = "Idc"
query = "INSERT INTO Votes(ID, LorU, BID, YayNayIdc) " \
            "VALUES(%s,%s,%s,%s)"
args = (ID, LorU, BID, YayNayIdc)
cursor.execute(query, args)
```

```
1
```

1 was returned to indicate success, but let's check if the new row is in the table

```
pd.read_sql('select * from Votes;', con=conn)
```

.dataframe tbody tr th:only-of-type { vertical-align: middle; } .dataframe tbody tr th { vertical-align: top; } .dataframe thead th { text-align: right; }

	ID	LorU	BID	YayNayIdc
0	123	1	testID321	Idc

__To delete the row__
We must identify it with a unique column (e.g. the ID)

```
#Delete record now
sql_Delete_query = "Delete from Votes where ID = 123"
cursor.execute(sql_Delete_query)
```

```
1
```

Check again to make sure the fake entry was removed

```
pd.read_sql('select * from Votes;', con=conn)
```

.dataframe tbody tr th:only-of-type { vertical-align: middle; } .dataframe tbody tr th { vertical-align: top; } .dataframe thead th { text-align: right; }

	ID	LorU	BID	YayNayIdc