

Lab #8 : Using Cloud Relational Database System (RDS)

Exercise 1: Relational Databases on AWS

Lab report screen-shot #1:

The screenshot shows the AWS VPC Management Console. A modal window at the top right indicates that a security group was created successfully. The main page displays the details of the newly created DB Security Group, sg-086d20a894732b490. The details include:

- Security group name:** DB Security Group
- Security group ID:** sg-086d20a894732b490
- Description:** Permit access from Web Security Group
- VPC ID:** vpc-0402bfdd4382f7648
- Owner:** 044674633345
- Inbound rules count:** 1 Permission entry
- Outbound rules count:** 1 Permission entry

Below the details, there are tabs for **Inbound rules**, **Outbound rules**, and **Tags**. A message at the bottom encourages checking network connectivity with the Reachability Analyzer.

Lab report screen-shot #2:

lab-db.c5iulndppurs.us-east-1.rds.amazonaws.com

The screenshot shows the AWS RDS Management Console. The left sidebar is collapsed, and the main area displays the summary for the database lab-db.

Summary:

DB identifier	CPU	Status	Class
lab-db	5.63%	Available	db.t3.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	MySQL Community	us-east-1a

Connectivity & security:

Endpoint & port	Networking	Security
Endpoint lab-db.c5iulndppurs.us-east-1.rds.amazonaws.com	Availability Zone us-east-1a	VPC security groups DB Security Group (sg-086d20a894732b490) Active
Port 3306	VPC Lab VPC (vpc-0402bfdd4382f7648)	Public accessibility No
	Subnet group db-subnet-group	Certificate authority rds-ca-2019
	Subnets	

Lab report screen-shot #3:

A screenshot of a web browser window titled "Workbench" with the URL "Not Secure | 18.206.38.133/rds.php". The page displays an "Address Book" with a table containing two rows of contact information. The table has columns for Last name, First name, Phone, Email, and Admin. An "Add Contact" button is located at the bottom right of the table.

Last name	First name	Phone	Email	Admin
Doe	Jane	010-110-1101	janed@someotheraddress.org	Edit Remove
Johnson	Roberto	123-456-7890	roberto@someaddress.com	Edit Remove

A screenshot of a web browser window titled "Workbench" with the URL "Not Secure | 18.206.38.133/rds.php". The page displays an "Address Book" with a table containing three rows of contact information. The table has columns for Last name, First name, Phone, Email, and Admin. An "Add Contact" button is located at the bottom right of the table.

Last name	First name	Phone	Email	Admin
Danddank	Nalongsone	6129009416	nalongsone.danddank@my.metrost	Edit Remove
Doe	Jane	010-110-1101	janed@someotheraddress.org	Edit Remove
Johnson	Roberto	123-456-7890	roberto@someaddress.com	Edit Remove

Lab report screen-shot #4:

danddank-database.c8wffoskckyb.us-east-1.rds.amazonaws.com

RDS > **Databases** > danddank-database

danddank-database

DB identifier	CPU	Status	Class
danddank-database	0.00%	Available	db.t2.micro

Role	Current activity	Engine	Region & AZ
Instance	0 Connections	MySQL Community	us-east-1f

Connectivity & security

Endpoint & port	Networking	Security
Endpoint danddank-database.c8wffoskcky.us-east-1.rds.amazonaws.com	Availability Zone us-east-1f	VPC security groups default (sg-9a0fe58b) Active
Port 3306	VPC vpc-64fa8419	Public accessibility Yes
	Subnet group default-vpc-64fa8419	Certificate authority rds-ca-2019
	Subnets	

Lab report screen-shot #5:

```

ec2-user@ip-172-31-90-201:~ zsh   ec2-user@ip-172-31-90-201:~ (ssh)   8.8 M
mariadb      x86_64      1:5.5.68-1.amzn2      amzn2-core
=====
Transaction Summary
=====
Install 1 Package

Total download size: 8.8 M
Installed size: 49 M
Is this ok [y/d/N]: y
Downloading packages:
mariadb-5.5.68-1.amzn2.x86_64.rpm | 8.8 MB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : 1:mariadb-5.5.68-1.amzn2.x86_64          1/1
  Verifying   : 1:mariadb-5.5.68-1.amzn2.x86_64          1/1

Installed:
  mariadb.x86_64 1:5.5.68-1.amzn2

Complete!
[ec2-user@ip-172-31-90-201 ~]$ mysql --version
mysql Ver 15.1 Distrib 5.5.68-MariaDB, for Linux (x86_64) using readline 5.1
[ec2-user@ip-172-31-90-201 ~]$ 

```

Lab report screen-shot #6:

```

ec2-user@ip-172-31-90-201:~ -zsh      ec2-user@ip-172-31-90-201:~ (ssh)      361
Database changed
MySQL [danddankdb]> create table danddank_test_table (attr1 int, attr2 int, attr3 varchar(10));
Query OK, 0 rows affected (0.03 sec)

MySQL [danddankdb]> insert into danddank_test_table values (10, 10, "Nalongsone");
Query OK, 1 row affected (0.00 sec)

MySQL [danddankdb]> select * from danddank_test_table;
+-----+-----+-----+
| attr1 | attr2 | attr3   |
+-----+-----+-----+
|    10  |     10 | Nalongsone |
+-----+-----+-----+
1 row in set (0.00 sec)

MySQL [danddankdb]>

```

Lab report screen-shot #7:

The screenshot shows the AWS Cloud9 IDE interface. At the top, there are tabs for 'My Classrooms', 'Workbench' (which is active), and 'Your environments'. Below the tabs is a navigation bar with links for File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run. On the left, there's a file browser showing a directory structure for 'Danddank Lab7' containing files like 'aws', 'create_s3_bucket.py', 'danddank_hello_world.py', 'list_bucket_objects.py', 'my_code_lab7_exerci', 'permissions.py', 'public_policy.json', 'upload_file.py', 'working_with_ec2.py', and 'danddank-db.py'. The main workspace shows a code editor with the following Python script:

```

1 import sys
2
3 # my_name = sys.argv[1]
4 my_name = "Ping"
5
6 print("Hi There "+ my_name)

```

Below the code editor is a terminal window titled 'DanddankLab7/danddank' with the command 'bash -i' running. The terminal output shows the user attempting to install the 'mysql-connector-python' package via pip:

```

vocstartsoft:~/environment (master) $ pip install mysql-connector-python
Defaulting to user installation because normal site-packages is not writeable
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.27-1commercial-cp37-cp37m-manylinux1_x86_64.whl (37.5 MB)
    100% |██████████| 37.5 MB 129 kB/s
Collecting protobuf==3.0.0
  Downloading protobuf-3.19.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
    100% |██████████| 1.1 MB 25.8 MB/s
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.27 protobuf-3.19.0
vocstartsoft:~/environment (master) $

```

Lab report screen-shot #8:

The screenshot shows the AWS Lambda Workbench interface. On the left, a sidebar displays a file tree for a project named "Danddank Lab7". The main area contains a code editor with the following Python script:

```
import mysql.connector
from mysql.connector.constants import ClientFlag

config = {
    'user': 'admin',
    'password': 'admin123',
    'host': 'danddank-database.c8wffoskckyb.us-east-1.rds.amazonaws.com'
}

db_connection = mysql.connector.connect(**config)
cursor = db_connection.cursor()
cursor.execute("show databases")
out = cursor.fetchall()
for row in out:
    print(row)
db_connection.close()
```

Below the code editor is a terminal window titled "DanddankLab7/danddank_..." with the command "danddank-db.py" running. The terminal output shows the following:

```
('danddankdb',)  
('information_schema',)  
('mysql',)  
('performance_schema',)  
('sys',)
```

The terminal concludes with "Process exited with code: 0".

Lab report screen-shot #9:

This screenshot is identical to the one above, showing the AWS Lambda Workbench interface. The file tree on the left is the same, and the code editor contains the same Python script. The terminal window below shows the same successful execution of the script, displaying the list of databases and exiting with code 0.

Lab report screen-shot #10:

The screenshot shows the AWS Lambda Workbench interface. On the left, a sidebar lists files in the 'Danddank Lab7' folder, including 'danddank_hello_world.py', 'danddank-db.py', and 'README.md'. The main workspace displays a Python script named 'danddank-db.py':

```
import mysql.connector
from mysql.connector.constants import ClientFlag

config = {
    'user': 'admin',
    'password': 'admin123',
    'host': 'danddank-database.c8wffoskcky.us-east-1.rds.amazonaws.com'
}

db_connection = mysql.connector.connect(**config)
cursor = db_connection.cursor()
cursor.execute("show databases")
cursor.execute("use danddankdb")
cursor.execute("select * from danddank_test_table")
out = cursor.fetchall()
for row in out:
    print(row)
db_connection.close()
```

The bottom pane shows the terminal output of running the script:

```
(10, 10, 'Nalongsone')

Process exited with code: 0
```

Lab report screen-shot #11:

The screenshot shows the AWS Lambda Workbench interface. On the left, a sidebar lists files in the 'Danddank Lab7' folder, including 'danddank_hello_world.py', 'danddank-db1.py', and 'README.md'. The main workspace displays a Python script named 'danddank-db1.py':

```
config = {
    'user': 'admin',
    'password': 'admin123',
    'host': 'danddank-database.c8wffoskcky.us-east-1.rds.amazonaws.com'
}

print("Enter three values: ")
attr1 = sys.argv[1]
attr2 = sys.argv[2]
attr3 = sys.argv[3]
print("You decide to insert these values to databases: " + attr1 + " " + attr2 + " " + attr3)

db_connection = mysql.connector.connect(**config)
cursor = db_connection.cursor()
cursor.execute("show databases")
cursor.execute("use danddankdb;")
cursor.execute("insert into danddank_test_table values({0}, {1}, {2})".format(attr1, attr2, attr3))
cursor.execute("select * from danddank_test_table;")
out = cursor.fetchall()
for row in out:
    print(row)
db_connection.close()
```

The bottom pane shows the terminal output of running the script:

```
Enter three values:
You decide to insert these values to databases: 12 12 NDanddankTest
(10, 10, 'Nalongsone')
(12, 12, 'NDanddankTest')
Process exited with code: 0
```

Lab report screen-shot #12:

The screenshot shows the AWS Lambda Workbench interface. On the left, a sidebar lists files in the 'Danddank Lab7' folder, including danddank_hello_world.py, danddank-db1.py, and README.md. The main area displays the contents of danddank-db1.py:

```
    'password': 'danddank',
    'host': 'danddank-database.c8wffoskcky.us-east-1.rds.amazonaws.com'
}
11 print("Enter three values: ")
12 attr1 = sys.argv[1]
13 attr2 = sys.argv[2]
14 attr3 = sys.argv[3]
15 print("You decide to insert these values to databases: " + attr1 + " " + attr2 + " "+attr3)
16 db_connection = mysql.connector.connect(**config)
17
18 cursor = db_connection.cursor()
19
20 # cursor.execute("show databases")
21 cursor.execute("use danddankdb;")
22 cursor.execute("insert into danddank_test_table values({0}, {1}, {2})".format(attr1, attr2, attr3))
23 cursor.execute("select * from danddank_test_table;")
24 out = cursor.fetchall()
25 for row in out:
26     print(row)
27
28 db_connection.close()
```

The bottom terminal window shows the execution of the script with three arguments: 12, 12, and NDanddankTest. The output indicates that the values were inserted into the database table.

The screenshot shows the AWS Lambda Workbench interface. On the left, a sidebar lists files in the 'Danddank Lab7' folder, including danddank_hello_world.py, danddank-db1.py, and README.md. The main area displays the contents of danddank-db1.py:

```
    'password': 'danddank',
    'host': 'danddank-database.c8wffoskcky.us-east-1.rds.amazonaws.com'
}
11 # print("Enter three values: ")
12 # attr1 = sys.argv[1]
13 # attr2 = sys.argv[2]
14 # attr3 = sys.argv[3]
15 # print("You decide to insert these values to databases: " + attr1 + " " + attr2 + " "+attr3)
16
17 db_connection = mysql.connector.connect(**config)
18
19 cursor = db_connection.cursor()
20
21 # cursor.execute("show databases")
22 cursor.execute("use danddankdb;")
23 # sql = 'insert into danddank_test_table (attr1, attr2, attr3) values(%s, %s, %s);'
24 # val = (attr1, attr2, attr3)
25 # cursor.execute(sql, val)
26 # db_connection.commit()
27 cursor.execute("select * from danddank_test_table;")
28 out = cursor.fetchall()
29 for row in out:
30     print(row)
31
32 db_connection.close()
```

The bottom terminal window shows the execution of the script with three arguments: 12, 12, and NDanddank. The output indicates that the values were inserted into the database table. A detailed error message is shown at the top of the terminal window.

Lab report screen-shot #13:

```

ec2-user@ip-172-31-90-201:~ -zsh      1
ec2-user@ip-172-31-90-201:~ (ssh)    2
+-----+
| attr1 | attr2 | attr3 |
+-----+
|    10 |     10 | Nalongsone |
+-----+
1 row in set (0.00 sec)

MySQL [danddankdb]> select * from danddank_test_table;
+-----+
| attr1 | attr2 | attr3 |
+-----+
|    10 |     10 | Nalongsone |
|    12 |     12 | NDanddan |
+-----+
2 rows in set (0.00 sec)

MySQL [danddankdb]>

```

Exercise 2: Relational Databases on GCP

Lab report screen-shot #14: 146.148.93.171

Cloud SQL for MySQL: Qwik Start - Overview - SQL - console.cloud.google.com/sql/instances/danddankdb/overview?authuser=4&project=danddank-lap8

Google Cloud Platform - danddank-lap8 - Search products and resources

SQL

PRIMARY INSTANCE

- Overview
- Connections
- Users
- Databases
- Backups
- Replicas
- Operations

All instances > danddankdb

danddankdb

MySQL 5.7

Chart: CPU utilization

No data is available for the selected time frame.

UTC-5 8:50 PM 8:55 PM 9:00 PM 9:05 PM 9:10 PM 9:15 PM 9:20 PM 9:25 PM 9:30 PM 9:35 PM 9:40 PM 9:45 PM 0

Connect to this instance

Public IP address: 146.148.93.171

Connection name: danddank-lap8:us-central1:danddankdb

Configuration

vCPUs: 4	Memory: 26 GB	SSD storage: 100 GB
----------	---------------	---------------------

Database version is MySQL 5.7

Auto start: Enabled

Created danddankdb 9:44:55 PM GMT-5

Uploads and danddank-lap8 operations

Need help connecting?

Review the documentation to learn about the many ways to connect to your instance. [Learn more](#)

Lab report screen-shot #15:

The screenshot shows the Google Cloud Platform Cloud SQL for MySQL Overview page for the instance 'danddankdb'. The left sidebar has 'Overview' selected under 'PRIMARY INSTANCE'. A chart titled 'CPU utilization' shows usage over time, with a significant spike around 9:45 PM. The terminal window below shows the MySQL command `select count(*) from users;` resulting in 943 rows.

```

Query OK, 0 rows affected (0.02 sec)

mysql> select count(*) from users;
+-----+
| count(*) |
+-----+
|      943 |
+-----+
1 row in set (0.01 sec)

mysql>

```

Lab report screen-shot #16:

`mysql -uroot -p -h 146.148.93.171 --ssl-ca=server-ca.pem --ssl-cert=client-cert.pem --ssl-key=client-key.pem`

The screenshot shows the Google Cloud Platform Cloud Shell Editor. The code in the editor is as follows:

```

# databaseconnection.py
# databaseconnection.py > ...
5     'user': 'root',
6     'password': 'password',
7     'host': '146.148.93.171',
8     'client_flags': [ClientFlag.SSL],
9     'ssl_ca': 'ssl/server-ca.pem',
10    'ssl_cert': 'ssl/client-cert.pem',
11    'ssl_key': 'ssl/client-key.pem'
12 ]
13 config['database'] = 'danddank_movie_rating'
14
15 movieratingdb_connection = mysql.connector.connect(**config)
16 cursor = movieratingdb_connection.cursor()
17
18 cursor.execute("SELECT count(*) from users;")
19 out = cursor.fetchall()
20 for row in out:
21     print(row)
22
23 movieratingdb_connection.close()

```

The terminal at the bottom shows the command being run and the output: 943 rows.

```

dalongsonedanddank@cloudshell:~ (danddank-lap8)$ python3 databaseconnection.py
(943,)
dalongsonedanddank@cloudshell:~ (danddank-lap8)$

```

Lab report screen-shot #17:

```

File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
NALONGSONE_DANDDANK
DanddankLab8
DanddankLab7
ssl
client-cert.pem
client-key.pem
server-ca.pem
databaseconnection.py
README-cloudshell.txt
users.csv
PROBLEMS
danddank-lap8
Problems danddank-lap8
nalongsonedanddank@cloudshell:~ (danddank-lap8)$ python3 databaseconnection.py 15
(30, 7, 'M', 'student', '55436\r')
(142, 13, 'M', 'other', '48118\r')
(206, 14, 'F', 'student', '53115\r')
(289, 11, 'M', 'none', '94619\r')
(471, 10, 'M', 'student', '55106\r')
(628, 13, 'F', 'student', '55106\r')
(628, 13, 'M', 'none', '94306\r')
(674, 13, 'F', 'student', '55337\r')
(813, 14, 'F', 'student', '2136\r')
(880, 13, 'M', 'student', '83782\r')
(887, 14, 'F', 'student', '27249\r')
nalongsonedanddank@cloudshell:~ (danddank-lap8)$

```

Lab report screen-shot #18:

```

File Edit Selection View Go Run Terminal Help
EXPLORERS
OPEN EDITORS
NALONGSONE_DANDDANK
DanddankLab8
DanddankLab7
ssl
client-cert.pem
client-key.pem
server-ca.pem
databaseconnection.py
README-cloudshell.txt
users.csv
PROBLEMS
danddank-lap8
Problems danddank-lap8
nalongsonedanddank@cloudshell:~ (danddank-lap8)$ python3 databaseconnection.py 15
(30, 7, 'M', 'student', '55436\r')
(206, 14, 'F', 'student', '53115\r')
(471, 10, 'M', 'student', '55106\r')
(609, 13, 'F', 'student', '55106\r')
(674, 13, 'F', 'student', '55337\r')
(813, 14, 'F', 'student', '2136\r')
(880, 13, 'M', 'student', '83782\r')
(887, 14, 'F', 'student', '27249\r')
There are 8 Total student who younger than 15.
nalongsonedanddank@cloudshell:~ (danddank-lap8)$

```

Lab report screen-shot #19:

Cloud SQL for MySQL | CloudSQLdb - Database | python input integer | Taking input in Python | How to take integer | mysql insert into multi | MySQL Insert Multiple | python insert multiple | Python MySQL Insert | +

console.cloud.google.com/instances/danddankdb/databases?authuser=4&project=danddank-lap8

Google Cloud Platform danddank-lap8

CLOUD SHELL Editor

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS
NALOGSONE_DANDANK
DanddankLab8
DanddankLab7
client-cert.pem
client-key.pem
server-ca.pem
databaseconnection.py
README-cloudshell.txt
users.csv

```

Search products and resources

Open Terminal

```

databaseconnection.py ...
1 import mysql.connector
2 from mysql.connector.constants import ClientFlag
3 import sys
4 config = {
5     'user': 'root',
6     'password': 'password',
7     'host': '146.148.93.171',
8     'client_flags': [ClientFlag.SSL],
9     'ssl_ca': 'ssl/server-ca.pem',
10    'ssl_cert': 'ssl/client-cert.pem',
11    'ssl_key': 'ssl/client-key.pem'
12 }
13 config['database'] = 'danddank_movie_rating'
14 num_user = int(input("How many users to enter: "))
15 users_val = []
16 for i in range(num_user):
17     print("Loop #"+str(i)+" Insert:")
18     userid = input("userid(id>1000): ")
19     age = input("age: ")
20     sex = input("sex: ")
21     occupation = input("occupation: ")
22     zipcode = input("zip code: ")
23     users_val.append(userid, age, sex, occupation, zipcode)
24 movieratingdb_connection = mysql.connector.connect(**config)
25 cursor = movieratingdb_connection.cursor()
26 sql = "insert into users values(%s, %s, %s, %s, %s);"
27 cursor.executemany(sql, users_val)
28 movieratingdb_connection.commit()
29 print("Insert Complete.")
30 print("List user who id > 1000:")
31 cursor.execute("SELECT * from users where userid > 1000;")
32 out = cursor.fetchall()
33 for row in out:
34     print(row)
35
36 movieratingdb_connection.close()

```

Problems

Insert Complete.
list user who id > 1000:
(1001, 35, 'Man', 'Architect', '55106')
(1002, 35, 'Man', 'Programmer', '55123')
(1003, 26, 'Girl', 'Cooker', '55432')

danddank-lap8\$

Cloud SQL for MySQL | CloudSQLdb - Database | python input integer | Taking input in Python | How to take integer | mysql insert into multi | MySQL Insert Multiple | python insert multiple | Python MySQL Insert | +

console.cloud.google.com/instances/danddankdb/databases?authuser=4&project=danddank-lap8

Google Cloud Platform danddank-lap8

CLOUD SHELL Editor

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS
NALOGSONE_DANDANK
DanddankLab8
DanddankLab7
client-cert.pem
client-key.pem
server-ca.pem
databaseconnection.py
README-cloudshell.txt
users.csv

```

Search products and resources

Open Terminal

```

databaseconnection.py ...
1 import mysql.connector
2 from mysql.connector.constants import ClientFlag
3 import sys
4 config = {
5     'user': 'root',
6     'password': 'password',
7     'host': '146.148.93.171',
8     'client_flags': [ClientFlag.SSL],
9     'ssl_ca': 'ssl/server-ca.pem',
10    'ssl_cert': 'ssl/client-cert.pem',
11    'ssl_key': 'ssl/client-key.pem'
12 }
13 config['database'] = 'danddank_movie_rating'
14 num_user = int(input("How many users to enter: "))
15 users_val = []
16 for i in range(num_user):
17     print("Loop #"+str(i)+" Insert:")
18     userid = input("userid(id>1000): ")
19     age = input("age: ")
20     sex = input("sex: ")
21     occupation = input("occupation: ")
22     zipcode = input("zip code: ")
23     users_val.append(userid, age, sex, occupation, zipcode)
24 movieratingdb_connection = mysql.connector.connect(**config)
25 cursor = movieratingdb_connection.cursor()
26 sql = "insert into users values(%s, %s, %s, %s, %s);"
27 cursor.executemany(sql, users_val)
28 movieratingdb_connection.commit()
29 print("Insert Complete.")
30 print("List user who id > 1000:")
31 cursor.execute("SELECT * from users where userid > 1000;")
32 out = cursor.fetchall()
33 for row in out:
34     print(row)
35
36 movieratingdb_connection.close()

```

Problems

How many users to enter: 3
Loop #0 Insert:
userid(id>1000): 1001
age: 35
sex: Man
occupation: Architect
zip code: 55106
Loop #1 Insert:
userid(id>1000): 1002
age: 34
sex: Woman
occupation: Programmer
zip code: 55123
Loop #2 Insert:
userid(id>1000): 1003
age: 26
sex: Girl
occupation: Cooker
zip code: 55432
Insert Complete.
List user who id > 1000:
(1001, 35, 'Man', 'Architect', '55106')
(1002, 35, 'Man', 'Programmer', '55123')
(1003, 26, 'Girl', 'Cooker', '55432')

danddank-lap8\$

Lab report screen-shot #20:

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes tabs for Cloud, dandd, python, Taking, How to, mysql, MySQL, python, Python, My Cls, Workb, Your e, and Dandd. The main workspace displays a file tree for 'Danddank Lab7 Br' containing 'danddank-db.py', 'danddank-hello_world.b', 'databaseconnection.py', and 'README.md'. Below the file tree is a code editor window showing the following Python script:

```
1 import mysql.connector
2 from mysql.connector.constants import ClientFlag
3 import sys
4 config = {
5     'user': 'root',
6     'password': 'password',
7     'host': '146.148.93.171',
8     'client_flags': [ClientFlag.SSL],
9     'ssl_ca': 'ssl/server-ca.pem',
10    'ssl_cert': 'ssl/client-cert.pem',
11    'ssl_key': 'ssl/client-key.pem'
12 }
13 config['database'] = 'danddank_movie_rating'
14
15 movieRatingdb_connection = mysql.connector.connect(**config)
16 cursor = movieRatingdb_connection.cursor()
17
18 print("list user who id > 1000:")
19 cursor.execute("SELECT * from users where userid > 1000;")
20 out = cursor.fetchall()
21 for row in out:
22     print(row)
23
24 print("list user who age == 30:")
25 cursor.execute("SELECT * from users where age = 30;")
26 out = cursor.fetchall()
27 for row in out:
28     print(row)
29
30 print("list user who zip code is 55106")
31 cursor.execute("SELECT * from users where zipcode = 55106 ;")
32 out = cursor.fetchall()
33 for row in out:
34     print(row)
35
36 movieRatingdb_connection.close()
```

The status bar at the bottom indicates '24:34 Python Spaces: 4'. The bottom right corner shows 'AWS: not connected'.

This screenshot shows the AWS Cloud9 IDE after running the 'databaseconnection.py' script. The terminal output pane at the bottom displays the results of the SQL queries executed by the script. The output is as follows:

```
list user who id > 1000:
(1001, 35, 'Man', 'Architect', '55106')
(1002, 34, 'Woman', 'Programmer', '55123')
(1003, 26, 'Girl', 'Cooker', '55432')

list user who age == 30:
(17, 30, 'M', 'programmer', '6355\r')
(23, 30, 'F', 'artist', '48197\r')
(42, 30, 'M', 'administrator', '17870\r')
(77, 30, 'M', 'technician', '19379\r')
(110, 30, 'M', 'scientist', '16609\r')
(125, 30, 'M', 'teacher', '22282\r')
(140, 30, 'F', 'student', '32250\r')
(174, 30, 'F', 'administrator', '52302\r')
(190, 30, 'M', 'administrator', '95938\r')
(199, 30, 'M', 'writer', '17684\r')
(226, 30, 'M', 'librarian', '198285\r')
(260, 30, 'F', 'librarian', '22952\r')
(344, 30, 'F', 'librarian', '94117\r')
(402, 30, 'M', 'engineer', '95129\r')
(410, 30, 'F', 'artist', '94025\r')
(436, 30, 'F', 'administrator', '17345\r')
(440, 30, 'M', 'other', '48076\r')
(447, 30, 'M', 'administrator', '55313\r')
(479, 30, 'M', 'programmer', '75358\r')
(479, 30, 'M', 'operator', '55405\r')
(526, 30, 'M', 'marketing', '97124\r')
(531, 30, 'F', 'salesman', '97408\r')
(557, 30, 'F', 'writer', '11217\r')
(637, 30, 'M', 'other', '74181\r')
(664, 30, 'M', 'engineer', '94122\r')
(676, 30, 'M', 'scientist', '22112\r')
(676, 30, 'M', 'programmer', '32712\r')
(737, 30, 'M', 'programmer', '9872\r')
(756, 30, 'F', 'none', '98247\r')
(774, 30, 'M', 'student', '80827\r')
(776, 30, 'M', 'librarian', '51157\r')
(783, 30, 'M', 'marketing', '77081\r')
(793, 30, 'M', 'operator', '55405\r')
(814, 30, 'M', 'other', '12345\r')
(869, 30, 'M', 'student', '18825\r')
(877, 30, 'M', 'other', '77584\r')
(885, 30, 'F', 'other', '95316\r')
(897, 30, 'M', 'other', '33484\r')
(920, 30, 'F', 'artist', '90008\r')

list user who zip code is 55106
(14, 45, 'M', 'scientist', '55106\r')
(380, 26, 'F', 'programmer', '55106\r')
(609, 13, 'F', 'student', '55106\r')
(1001, 35, 'Man', 'Architect', '55106')
```

The status bar at the bottom indicates 'Process exited with code: 0'. The bottom right corner shows 'AWS: not connected'.