Hands-on Lab: ETL using shell scripts

Estimated time needed: **30** minutes

Objectives

After completing this lab you will be able to:

- Extract data from a delimited file.
- Transform text data.
- Load data into a database using shell commands.

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open-source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. to complete this lab, we will be using the Cloud IDE based on Theia and Postgres running in a Docker container.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persisted. Every time you connect to this lab, a new environment is created for you. Any data you may have saved in the earlier session would get lost. Plan to complete these labs in a single session, to avoid losing your data.

Getting the environment ready

Open a new terminal, by clicking on the menu bar and selecting **Terminal**->**New Terminal**, as in the image below. This will open a new terminal at the bottom of the screen.

| File | Edit | Selection | View | Go | Run | Terminal | Help |

SKILLS NETWO...

∨ **DATABASES**

MySQL **INACTIVE**

PostgreSQL **IDLE**

Cassandra **INACTIVE**

MongoDB **INACTIVE**

> **BIG DATA**

> **CLOUD**

> **EMBEDDABLE AI**

> **OTHER**

🚀 Launch Application

| New Terminal | ` |
| Split Terminal | |

Run Task...
Run Build Task
Run Test Task
Rerun Last Task   K

Show Running Tasks...
Restart Running Task...
Terminate Task...
Attach Task...

Configure Tasks...

theia@theiadocker-lavanyas: /home/project ✕

theia@theiadocker-lavanyas:/home/project$ ☐
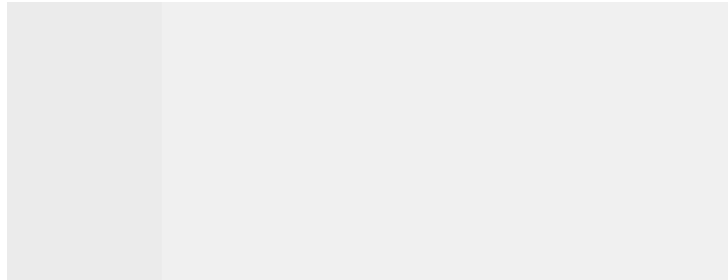
Run all the commands on the newly opened terminal. (You can copy the code by clicking on the little copy button on the bottom right of the codeblock below and then paste it, wherever you wish.)

Exercise 1 - Extracting data using 'cut' command

The filter command `cut` helps us extract selected characters or fields from a line of text.

  1. Extracting characters.

The command below shows how to extract the first four characters.

1. 1

1. `echo "database" | cut -c1-4`

Copied!   Executed!

You should get the string 'data' as output.

The command below shows how to extract 5th to 8th characters.

1. 1

1. `echo "database" | cut -c5-8`

Copied!   Executed!

You should get the string 'base' as output.

Non-contiguous characters can be extracted using the comma.

The command below shows how to extract the 1st and 5th characters.

1. 1

1. `echo "database" | cut -c1,5`

Copied!   Executed!

You get the output : 'db'

  2. Extracting fields/columns

We can extract a specific column/field from a delimited text file, by mentioning

- the delimiter using the `-d` option, or
- the field number using the `-f` option.

The /etc/passwd is a ":" delimited file.

The command below extracts usernames (the first field) from /etc/passwd.

1. 1

    1. `cut -d":" -f1 /etc/passwd`

Copied! | Executed!

The command below extracts multiple fields 1st, 3rd, and 6th (username, userid, and home directory) from /etc/passwd.

    1. 1

    1. `cut -d":" -f1,3,6 /etc/passwd`

Copied! | Executed!

The command below extracts a range of fields 3rd to 6th (userid, groupid, user description and home directory) from /etc/passwd.

    1. 1

    1. `cut -d":" -f3-6 /etc/passwd`

Copied! | Executed!

Exercise 2 - Transforming data using 'tr'

`tr` is a filter command used to translate, squeeze, and/or delete characters.

  1. Translate from one character set to another

The command below translates all lower case alphabets to upper case.

    1. 1

    1. `echo "Shell Scripting" | tr "[a-z]" "[A-Z]"`

Copied! | Executed!

You could also use the pre-defined character sets also for this purpose:

    1. 1

    1. `echo "Shell Scripting" | tr "[:lower:]" "[:upper:]"`

Copied! | Executed!

The command below translates all upper case alphabets to lower case.

    1. 1

    1. `echo "Shell Scripting" | tr  "[A-Z]" "[a-z]"`

Copied! | Executed!

  2. Squeeze repeating occurrences of characters

The `-s` option replaces a sequence of a repeated characters with a single occurrence of that character.

The command below replaces repeat occurrences of 'space' in the output of `ps` command with one 'space'.

    1. 1

    1. `ps | tr -s " "`

Copied! | Executed!

In the above example, the space character within quotes can be replaced with the following : `"[\:space\:]"`.

  3. Delete characters

We can delete specified characters using the `-d` option.

The command below deletes all digits.

1. 1

1. `echo "My login pin is 5634" | tr -d "[:digit:]"`

Copied! Executed!

The output will be : 'My login pin is'

Exercise 3 - Start the PostgreSQL database.

1. From the SkillsNetwork tools, under Databases choose `PostgreSQL` Database server and click `Start` to start the server. This will take a few mins.

2. Click `PostgresSQL CLI` on the screen to start interacting wit the PostgresSQL server.

SKILLS NETWORK TOOL...   ⊡ ⑦ | 🗃 MongoDB | Welcome | 🗃 PostgreSQL ✕

∨ DATABASES

MySQL INACTIVE

PostgreSQL ACTIVE

Cassandra INACTIVE

MongoDB INACTIVE

∨ BIG DATA

Apache Airflow INACTIVE

Apache Hive COMING SOON

Apache Spark COMING SOON

> CLOUD

> EMBEDDABLE AI

> OTHER

🚀 Launch Application

PostgreSQL   [ACTIVE]

🗃 v13.2 | 👥 v5.0 | ⌦ v14.5

Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.

Stop

Summary   **Connection Information**   **Details**

Your database and pgAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate PostgreSQL, please check out the Details section.

Username: captainfedo1 ⧉

Password: Mjk2MDktY2FwdGFp ⧉

You can manage PostgreSQL via:

pgAdmin ⧉

Or to interact with the database in the terminal, select one of

these options:

PostgreSQL CLI   New Terminal

This will start the interactive `psql` client which connects to the PostgreSQL server with `postgres=#` prompt as shown below.

```
theia@theiadocker-lavanyas:/home/project$ psql --username=postgres --host=localhost
psql (15.2 (Ubuntu 15.2-1.pgdg18.04+1), server 13.2)
Type "help" for help.

postgres=# ▮
```

Exercise 4 - Create a table

In this exercise we will create a table called `users` in the PostgreSQL database using PostgresSQL CLI. This table will hold the user account information.

The table `users` will have the following columns:

1. uname

2. uid

3. home

4. You will connect to `template1` database which is already available by default. To connect to this database, run the following command at the 'postgres=#' prompt.

   1. 1

   1. `\c template1`

   Copied!

You will get the following message.

`You are now connected to database "template1" as user "postgres".`

Also, your prompt will change to 'template1=#'.

2. Run the following statement at the 'template1=#' prompt to create the table.

   1. 1

   1. `create table users(username varchar(50),userid int,homedirectory varchar(100));`

   Copied!

If the table is created successfully, you will get the message below.

CREATE TABLE

Exercise 5 - Loading data into a PostgreSQL table.

In this exercise, you will create a shell script which does the following.

- Extract the user name, user id, and home directory path of each user account defined in the /etc/passwd file.
- Save the data into a comma separated (CSV) format.
- Load the data in the csv file into a table in PostgreSQL database.

1. Open a new Terminal.
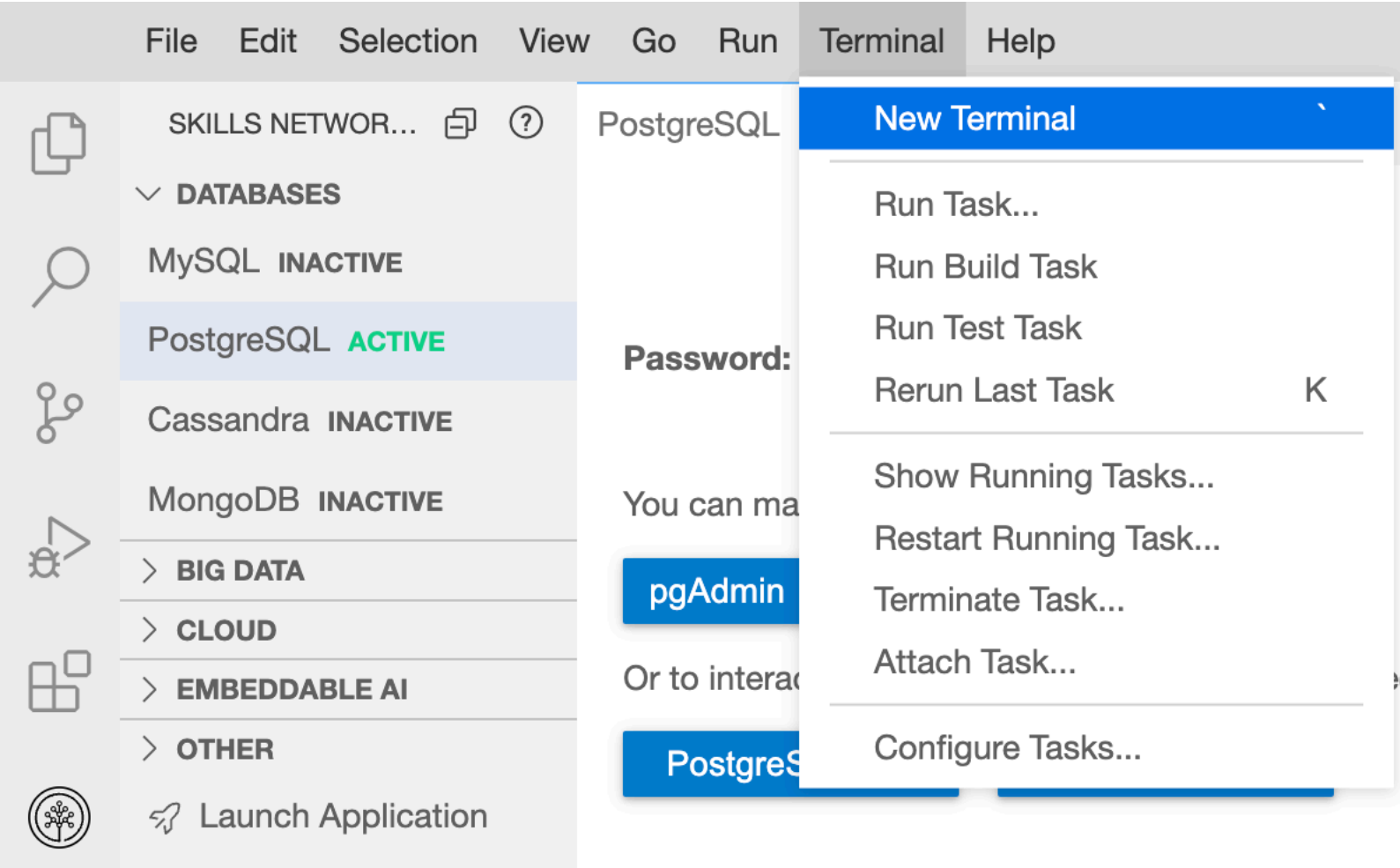
2. In the terminal, run the following command to create a new shell script named `csv2db.sh`.

1. 1

1. `touch csv2db.sh`

Copied! | Executed!

3. Open the file in the editor. Copy and paste the following lines into the newly created file.

Open **csv2db.sh** in IDE

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```
1. # This script
2. # Extracts data from /etc/passwd file into a CSV file.
3.
4. # The csv data file contains the user name, user id and
5. # home directory of each user account defined in /etc/passwd
6.
7. # Transforms the text delimiter from ":" to ",".
8. # Loads the data from the CSV file into a table in PostgreSQL database.
```

Copied!

4. Save the file by presseing `Ctrl+s` or by using the **File->Save** menu option.

5. You need to add lines of code to the script that will xtract user name (field 1), user id (field 3), and home directory path (field 6) from /etc/passwd file using the `cut` command.

Copy the following lines and paste them to the end of the script and save the file.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```
1. # Extract phase
2.
3. echo "Extracting data"
4.
5. # Extract the columns 1 (user name), 2 (user id) and
6. # 6 (home directory path) from /etc/passwd
7.
8. cut -d":" -f1,3,6 /etc/passwd
```

Copied!

6. Run the script.

```
1. 1
```

```
1. bash csv2db.sh
```

Copied!   Executed!

7. Verify that the output contains the three fields, that you extracted.

8. Change the script to redirect the extracted data into a file named `extracted-data.txt`

Replace the cut command at end of the script with the following command.

```
1. 1
```

```
1. cut -d":" -f1,3,6 /etc/passwd > extracted-data.txt
```

Copied!

9. Run the script.

```
1. 1
```

```
1. bash csv2db.sh
```

Copied! | Executed!

10. Run the command below to verify that the file `extracted-data.txt` is created, and has the content.

```
1. 1
```

```
1. cat extracted-data.txt
```

Copied! | Executed!

11. The extracted columns are separated by the original ":" delimiter. You need to convert this into a "," delimited file. Add the below lines at the end of the script and save the file.

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. # Transform phase
2. echo "Transforming data"
3. # read the extracted data and replace the colons with commas.
4.
5. tr ":" "," < extracted-data.txt  > transformed-data.csv
```

Copied!

12. Run the script.

```
1. 1
```

```
1. bash csv2db.sh
```

Copied! | Executed!

13. Run the command below to verify that the file `transformed-data.csv` is created, and has the content.

```
1. 1
```

```
1. cat transformed-data.csv
```

Copied!

14. To load data from a shell script, you will use the `psql` client utility in a non-interactive manner. This is done by sending the database commands through a command pipeline to `psql` with the help of `echo` command.

PostgreSQL command to copy data from a CSV file to a table is `COPY`.

The basic structure of the command which we will use in our script is,

```
COPY table_name FROM 'filename' DELIMITERS 'delimiter_character' FORMAT;
```

Now, add the lines below to the end of the script 'csv2db.sh' and save the file.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
```

```
1. # Load phase
2. echo "Loading data"
3. # Set the PostgreSQL password environment variable.
```

```
4. # Replace <yourpassword> with your actual PostgreSQL password.
5. export PGPASSWORD=<yourpassword>;
6. # Send the instructions to connect to 'template1' and
7. # copy the file to the table 'users' through command pipeline.
8. echo "\c template1;\COPY users  FROM '/home/project/transformed-data.csv' DELIMITERS ',' CSV;" | psql --username=postgres --host=postgres
```

Copied!

Exercise 6 - Execute the final script

1. Run the script.

1. `1`

1. `bash csv2db.sh`

Copied!

2. Now, add the line below to the end of the script 'csv2db.sh' and save the file.

1. `1`

1. `echo "SELECT * FROM users;" | psql --username=postgres --host=postgres template1`

Copied!

3. Run the script to verify that the table users is populated with the data.

1. `1`

1. `bash csv2db.sh`

Copied!

Congratulations! You have created an ETL script using shell scripting.

Practice exercises

1. Copy the data in the file 'web-server-access-log.txt.gz' to the table 'access_log' in the PostgreSQL database 'template1'.

   The file is available at the location : https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Bash%20Scripting/ETL%20using%20shell%20scripting/web-server-access-log.txt.gz

   The following are the columns and their data types in the file:

   - a. timestamp - TIMESTAMP
   - b. latitude - float
   - c. longitude - float
   - d. visitorid - char(37)
   - e. accessed_from_mobile - boolean
   - f. browser_code - int

   The columns which we need to copy to the table are the first four coumns : timestamp, latitude, longitude and visitorid.

   NOTE: The file comes with a header. So use the 'HEADER' option in the 'COPY' command.

The problem may be solved by completing the following tasks:

1. Go to the SkillsNetwork Tools menu and start the Postgres SQL server if it is not already running.

2. Create a table named `access_log` to store the timestamp, latitude, longitude and visitorid.

▶ Click here for Hint

▼ **Click here for Solution**

Step 1: Open the Postgres SQL CLI, if it is not already open.

Step 2: At the `postgres=#` prompt, run the following command to connect to the database 'template1'.

    1. 1

    1. \c template1;

    Copied!

Step 3: Once you connect to the database, run the command to create the table called 'access_log':

    1. 1

    1. CREATE TABLE access_log(timestamp TIMESTAMP, latitude float, longitude float, visitor_id char(37));

    Copied!

Task 3. Create a shell script named `cp-access-log.sh` and add commands to complete the remaining tasks to extract and copy the data to the database.

Create a shell script to add commands to complete the rest of the tasks.

▶ Click here for Hint

Task 4. Download the access log file.

Add the `wget` command to the script to download the file.

    1. 1

    1. wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Bash%20Scripting/ETL%20using%20shell%20scripting/web-server-access-log.txt.gz"

    Copied!

Task 5. Unzip the gzip file.

Add the code, to run the gunzip command to unzip the `.gz` file and extract the `.txt` file, to the script.

    1. 1
    2. 2

    1. # Unzip the file to extract the .txt file.
    2. gunzip -f web-server-access-log.txt.gz

    Copied!

      The **-f** option of gunzip is to overwrite the file if it already exists.

Task 6. Extract required fields from the file.

Extract timestamp, latitude, longitude and visitorid which are the first four fields from the file using the `cut` command.

The columns in the web-server-access-log.txt file is delimited by '#'.

▶ Click here for Hint
▼ Click here for Solution

Step 1: Copy the following lines and add them to the end of the script.

    1. 1
    2. 2
    3. 3
    4. 4
    5. 5
    6. 6
    7. 7

8. 8

1. # Extract phase
2.
3. echo "Extracting data"
4.
5. # Extract the columns 1 (timestamp), 2 (latitude), 3 (longitude) and
6. # 4 (visitorid)
7.
8. cut -d"#" -f1-4 web-server-access-log.txt

Copied!

Step 2: Save the file.

Step 3: Run the script.

1. 1

1. bash cp-access-log.sh

Copied! | Executed!

Verify that the output contains all the four fields that we extracted.

Task 7. Redirect the extracted output into a file.

Redirect the extracted data into a file named `extracted-data.txt`

▶ Click here for Hint
▼ Click here for Solution

Step 1: *Replace* the cut command at end of the script with the following command and save the file.

1. 1

1. cut -d"#" -f1-4 web-server-access-log.txt > extracted-data.txt

Copied!

Step 2: Run the script.

1. 1

1. bash cp-access-log.sh

Copied!

Step 3: Run the command below to verify that the file `extracted-data.txt` is created, and has the content.

1. 1

1. cat extracted-data.txt

Copied!

Task 8. Transform the data into CSV format.

The extracted columns are separated by the original "#" delimiter.

We need to convert this into a "," delimited file.

▶ Click here for Hint
▼ Click here for Solution

Step 1: Add the lines below at the end of the script.

1. 1

```
2. 2
3. 3
4. 4
5. 5
6. 6
```

```
1. # Transform phase
2. echo "Transforming data"
3.
4. # read the extracted data and replace the colons with commas and
5. # write it to a csv file
6. tr "#" "," < extracted-data.txt > transformed-data.csv
```

Copied!

Step 2: Save the file.

Step 3: Run the script.

```
1. 1
```

```
1. bash cp-access-log.sh
```

Copied!

Step 4: Run the command below to verify that the file 'transformed-data.csv' is created, and has the content.

```
1. 1
```

```
1. cat transformed-data.csv
```

Copied!

Task 9. Load the data into the table `access_log` in PostgreSQL

PostgreSQL command to copy data from a CSV file to a table is `COPY`.

The basic structure of the command is,

```
1. 1
```

```
1. COPY table_name FROM 'filename' DELIMITERS 'delimiter_character' FORMAT;
```

Copied!

The file comes with a header. So use the 'HEADER' option in the 'COPY' command.

Invoke this command from the shellscript, by sending it as input to 'psql' filter command.

▶ Click here for Hint
▼ Click here for Solution

Step 1: Add the lines below to the end of the script 'cp-access-log.sh' and save the file.

```
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
```

```
1. # Load phase
2. echo "Loading data"
3.
4. # Send the instructions to connect to 'template1' and
5. # copy the file to the table 'access_log' through command pipeline.
6.
7. echo "\c template1;\COPY access_log  FROM '/home/project/transformed-data.csv' DELIMITERS ',' CSV HEADER;" | psql --username=postgres --host=localhost
```

Copied!

Task 10. Execute the final script.

Run the final script.

▼ Click here for Solution

Run the following command at the terminal:

    1. 1

    1. bash cp-access-log.sh

Copied!

The completed bash script would be as below.

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
```

```
 1. # cp-access-log.sh
 2. # This script downloads the file 'web-server-access-log.txt.gz'
 3. # from "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Bash%20Scripting/ETL%20using%20shell%20scripting/".
 4.
 5. # The script then extracts the .txt file using gunzip.
 6.
 7. # The .txt file contains the timestamp, latitude, longitude
 8. # and visitor id apart from other data.
 9.
10. # Transforms the text delimeter from "#" to "," and saves to a csv file.
11. # Loads the data from the CSV file into the table 'access_log' in PostgreSQL database.
12. # Download the access log file
```

```
13.
14. wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Bash%20Scripting/ETL%20using%20shell%20scripting/web-server-access-log.txt.gz"
15. # Unzip the file to extract the .txt file.
16. gunzip -f web-server-access-log.txt.gz
17.
18. # Extract phase
19.
20. echo "Extracting data"
21.
22. # Extract the columns 1 (timestamp), 2 (latitude), 3 (longitude) and
23. # 4 (visitorid)
24.
25.
26.
27. cut -d"#" -f1-4 web-server-access-log.txt > extracted-data.txt
28.
29. # Transform phase
30. echo "Transforming data"
31.
32. # read the extracted data and replace the colons with commas.
33. tr "#" "," < extracted-data.txt > transformed-data.csv
34.
35. # Load phase
36. echo "Loading data"
37.
38. # Send the instructions to connect to 'template1' and
39. # copy the file to the table 'access_log' through command pipeline.
40.
41. echo "\c template1;\COPY access_log  FROM '/home/project/transformed-data.csv' DELIMITERS ',' CSV HEADER;" | psql --username=postgres --host=localhost
```

Copied!

Task 11. Verify by querying the database.

▶ Click here for Hint
▼ Click here for Solution

Run the command below at Postgres SQL CLI prompt.

```
1. 1
```

```
1. SELECT * from access_log;
```

Copied!

You should see the records displayed on screen.

Authors

Ramesh Sannareddy

Other Contributors

Rav Ahuja