Laboratório Prático: ETL usando scripts de shell

Tempo estimado: **30** minutos

Objetivos

Após concluir este laboratório, você será capaz de:

- Extrair dados de um arquivo delimitado.
- Transformar dados de texto.
- Carregar dados em um banco de dados usando comandos de shell.

Sobre o Skills Network Cloud IDE

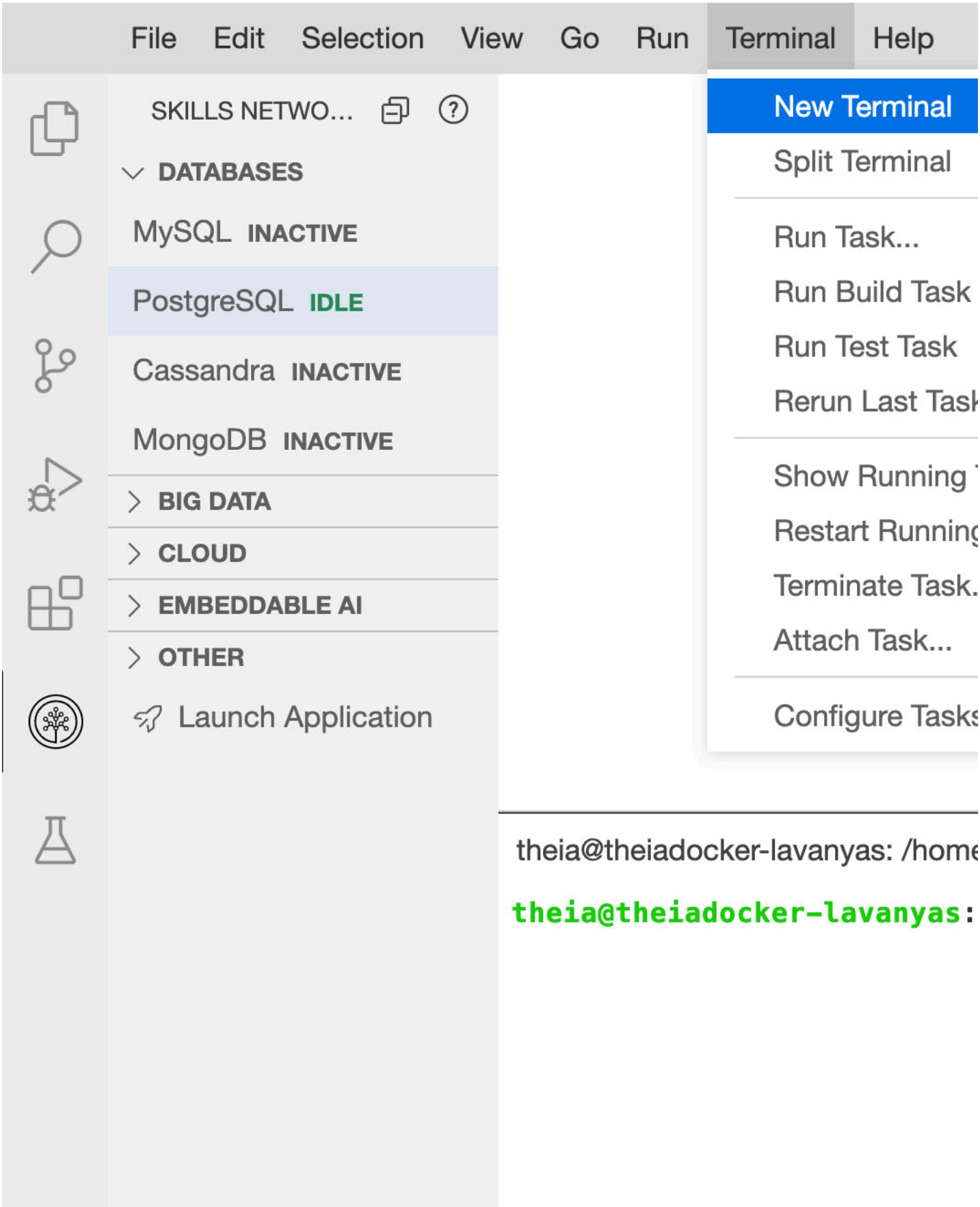
O Skills Network Cloud IDE (baseado em Theia e Docker) fornece um ambiente para laboratórios práticos relacionados a cursos e projetos. Theia é um IDE de código aberto (Ambiente de Desenvolvimento Integrado), que pode ser executado no desktop ou na nuvem. Para completar este laboratório, usaremos o Cloud IDE baseado em Theia e Postgres rodando em um contêiner Docker.

Aviso Importante sobre este ambiente de laboratório

Por favor, esteja ciente de que as sessões deste ambiente de laboratório não são persistidas. Toda vez que você se conectar a este laboratório, um novo ambiente será criado para você. Quaisquer dados que você possa ter salvo na sessão anterior serão perdidos. Planeje completar esses laboratórios em uma única sessão, para evitar a perda de seus dados.

Preparando o ambiente

Abra um novo terminal, clicando na barra de menu e selecionando **Terminal**->**Novo Terminal**, como na imagem abaixo. Isso abrirá um novo terminal na parte inferior da tela.



Execute todos os comandos no terminal recém-aberto. (Você pode copiar o código clicando no pequeno botão de copiar no canto inferior direito do bloco de código abaixo e, em seguida, colá-lo onde desejar.)

Exercício 1 - Extraindo dados usando o comando 'cut'

O comando de filtro `cut` nos ajuda a extrair caracteres ou campos selecionados de uma linha de texto.

1. Extraindo caracteres.

O comando abaixo mostra como extrair os primeiros quatro caracteres.

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

Você deve obter a string ‘data’ como saída.

O comando abaixo mostra como extrair os 5º a 8º caracteres.

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

Você deve obter a string ‘base’ como saída.

Caracteres não contíguos podem ser extraídos usando a vírgula.

O comando abaixo mostra como extrair o 1º e o 5º caracteres.

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

Você obtém a saída: ‘db’

2. Extraindo campos/colunas

Podemos extrair uma coluna/campo específico de um arquivo de texto delimitado, mencionando

- o delimitador usando a opção `-d`, ou
- o número do campo usando a opção `-f`.

O /etc/passwd é um arquivo delimitado por “:”.

O comando abaixo extrai nomes de usuário (o primeiro campo) do /etc/passwd.

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

O comando abaixo extrai múltiplos campos 1º, 3º e 6º (nome de usuário, id do usuário e diretório home) do /etc/passwd.

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

O comando abaixo extrai um intervalo de campos do 3º ao 6º (userid, groupid, descrição do usuário e diretório home) do /etc/passwd.

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

Exercício 2 - Transformando dados usando 'tr'

`tr` é um comando de filtro usado para traduzir, comprimir e/ou deletar caracteres.

1. Traduzir de um conjunto de caracteres para outro

O comando abaixo traduz todas as letras minúsculas para maiúsculas.

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

Você também pode usar os conjuntos de caracteres pré-definidos para este propósito:

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

O comando abaixo traduz todas as letras maiúsculas para minúsculas.

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

2. Compactar ocorrências repetidas de caracteres

A opção `-s` substitui uma sequência de caracteres repetidos por uma única ocorrência desse caractere.

O comando abaixo substitui ocorrências repetidas de ‘espaço’ na saída do comando `ps` por um único ‘espaço’.

```
ps | tr -s " "
```

No exemplo acima, o caractere de espaço entre aspas pode ser substituído por: `"[\:space\:]"`.

3. Excluir caracteres

Podemos excluir caracteres especificados usando a opção `-d`.

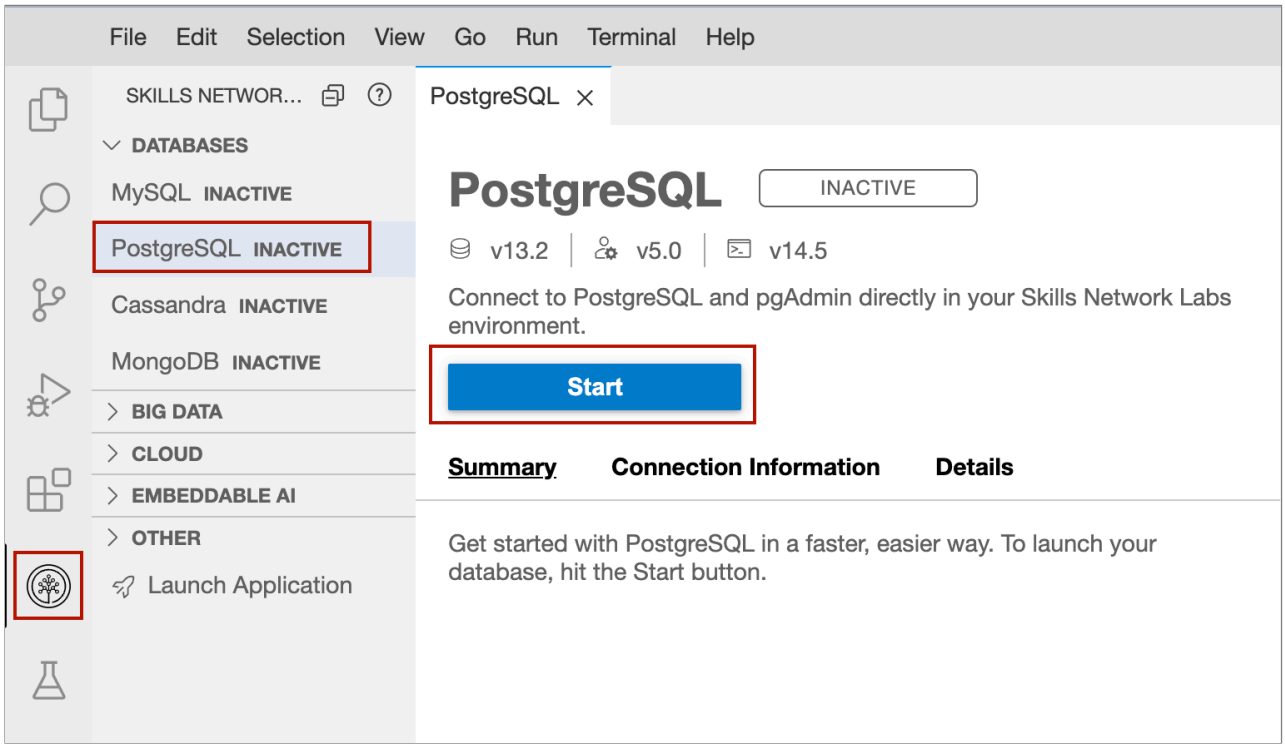
O comando abaixo exclui todos os dígitos.

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

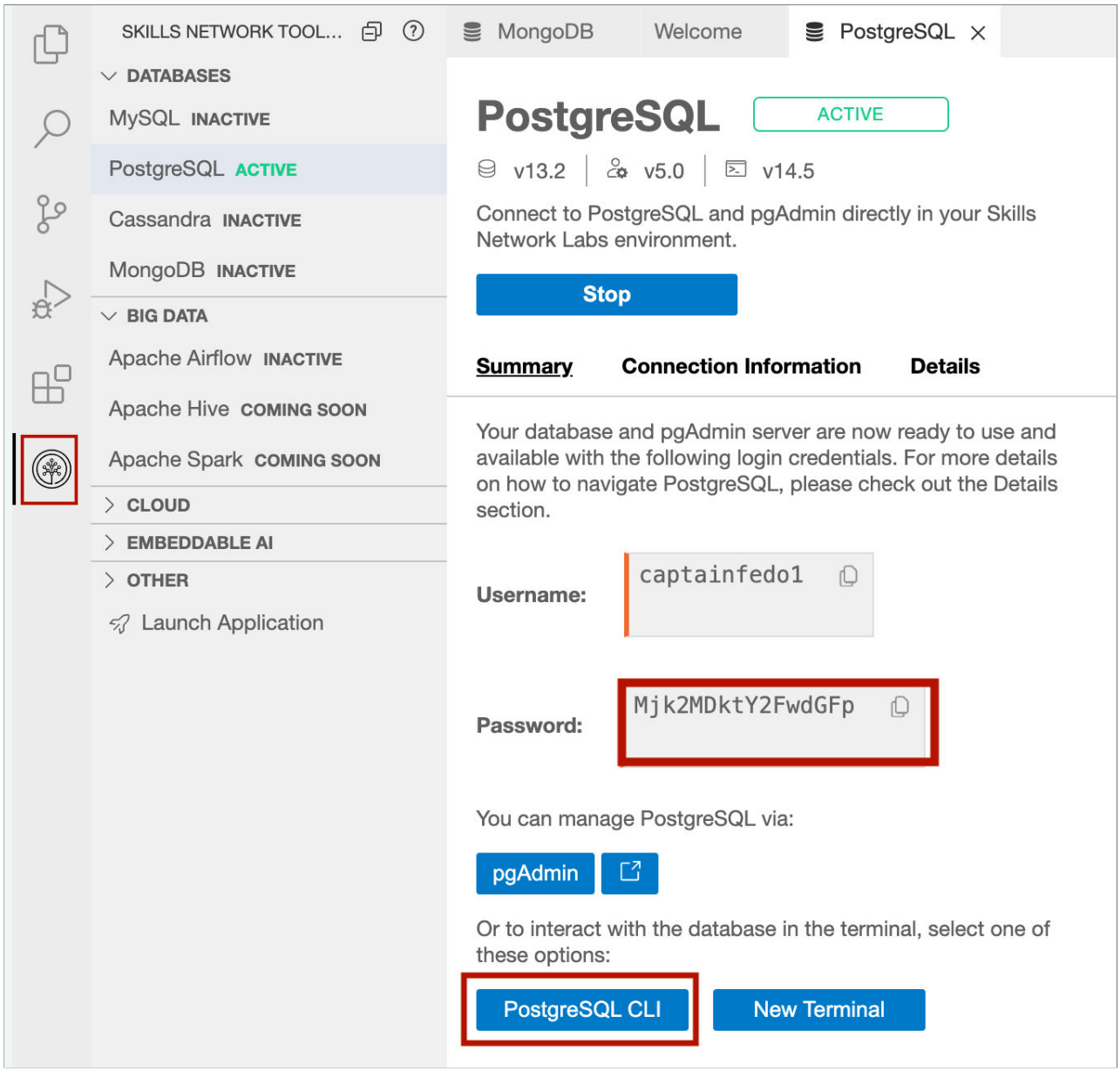
A saída será: ‘Meu PIN de login é’

Exercício 3 - Iniciar o banco de dados PostgreSQL.

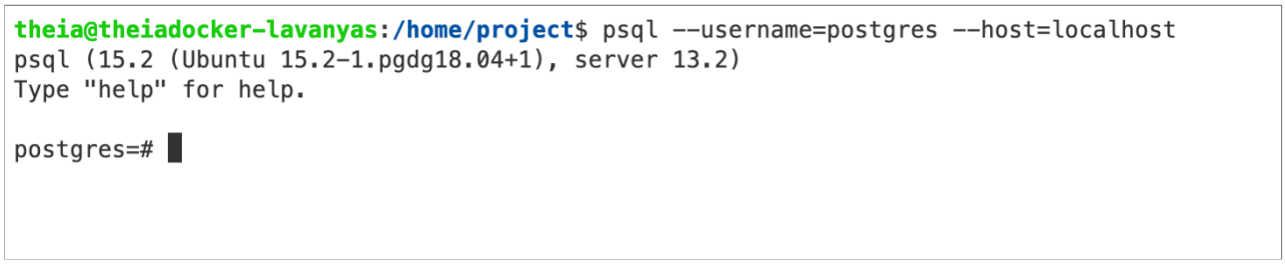
1. Nos ferramentas do SkillsNetwork, em Bancos de Dados, escolha o servidor de banco de dados `PostgresSQL` e clique em `Iniciar` para iniciar o servidor. Isso levará alguns minutos.



2. Clique em `PostgresSQL CLI` na tela para começar a interagir com o servidor PostgresSQL.



Isso iniciará o cliente interativo `psql`, que se conecta ao servidor PostgreSQL com o prompt `postgres=#`, conforme mostrado abaixo.



Exercício 4 - Criar uma tabela

Neste exercício, criaremos uma tabela chamada `users` no banco de dados PostgreSQL usando a CLI do PostgreSQL. Esta tabela armazenará as informações da conta do usuário.

A tabela `users` terá as seguintes colunas:

1. uname
2. uid
3. home
4. Você se conectará ao banco de dados `template1`, que já está disponível por padrão. Para se conectar a este banco de dados, execute o seguinte comando no prompt 'postgres=#'.

```
\c template1
```

Você receberá a seguinte mensagem.

```
Você está agora conectado ao banco de dados "template1" como usuário "postgres".
```

Além disso, seu prompt mudará para 'template1=#'.

2. Execute a seguinte instrução no prompt 'template1=#' para criar a tabela.

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

Se a tabela for criada com sucesso, você receberá a mensagem abaixo.

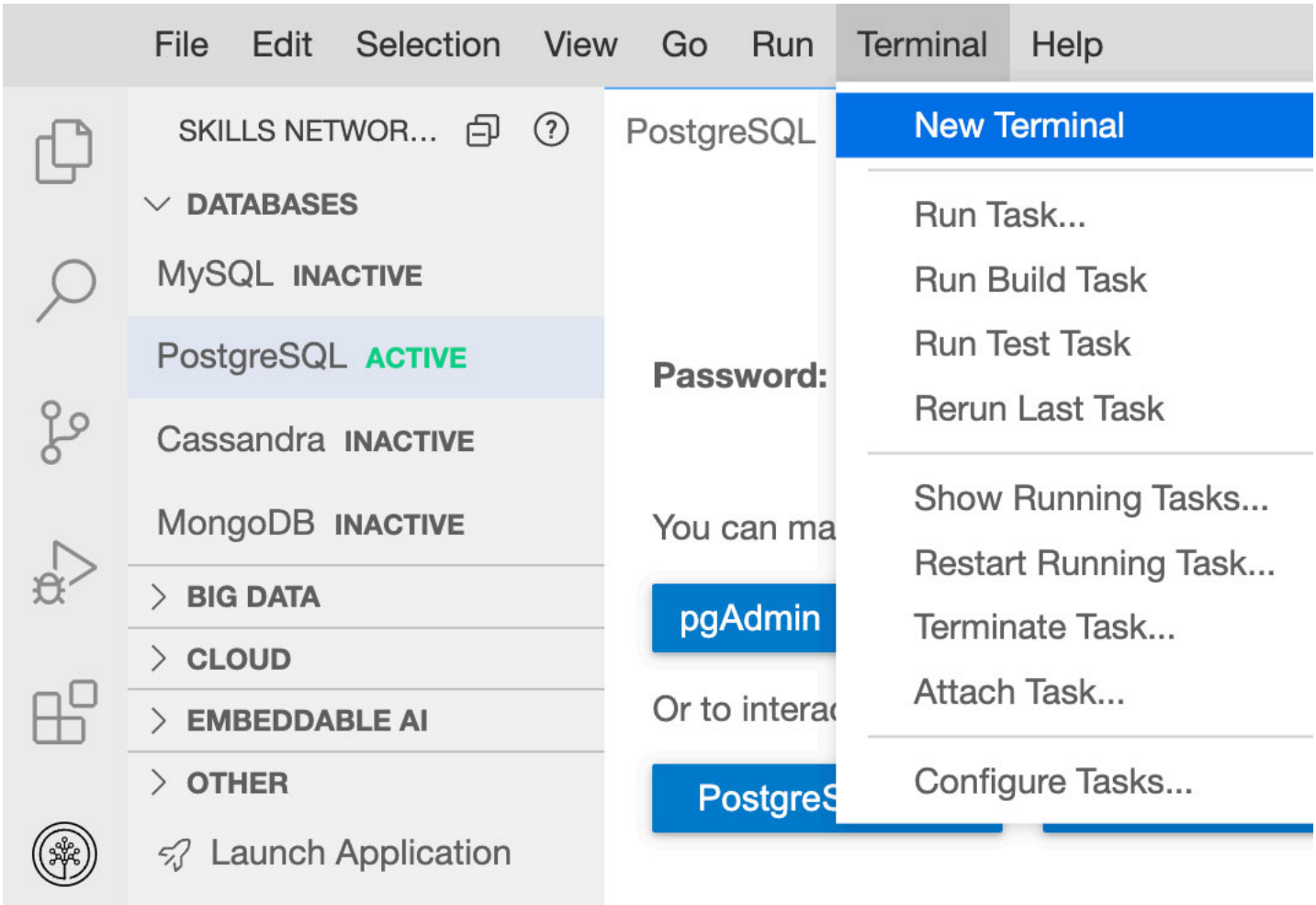
```
CREATE TABLE
```

Exercício 5 - Carregando dados em uma tabela PostgreSQL.

Neste exercício, você criará um script shell que faz o seguinte.

- Extrai o nome do usuário, o ID do usuário e o caminho do diretório home de cada conta de usuário definida no arquivo /etc/passwd.
- Salva os dados em um formato separado por vírgulas (CSV).
- Carrega os dados do arquivo CSV em uma tabela no banco de dados PostgreSQL.

1. Abra um novo Terminal.



2. No terminal, execute o seguinte comando para criar um novo script shell chamado `csv2db.sh`.

```
touch csv2db.sh
```

3. Abra o arquivo no editor. Copie e cole as seguintes linhas no arquivo recém-criado.

Open **csv2db.sh** in IDE

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

4. Salve o arquivo pressionando `Ctrl+s` ou usando a opção de menu **Arquivo->Salvar**.

5. Você precisa adicionar linhas de código ao script que extrairão o nome do usuário (campo 1), o ID do usuário (campo 3) e o caminho do diretório inicial (campo 6) do arquivo /etc/passwd usando o comando `cut`.

Copie as seguintes linhas e cole-as no final do script e salve o arquivo.

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

6. Execute o script.

```
bash csv2db.sh
```

7. Verifique se a saída contém os três campos que você extraiu.

8. Altere o script para redirecionar os dados extraídos para um arquivo chamado `extracted-data.txt`

Substitua o comando cut no final do script pelo seguinte comando.

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

9. Execute o script.

```
bash csv2db.sh
```

10. Execute o comando abaixo para verificar se o arquivo `extracted-data.txt` foi criado e contém o conteúdo.

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

11. As colunas extraídas são separadas pelo delimitador original “:”. Você precisa converter isso em um arquivo delimitado por “,”. Adicione as linhas abaixo ao final do script e salve o arquivo.

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

12. Execute o script.

    ```
    bash csv2db.sh
    ```

13. Execute o comando abaixo para verificar se o arquivo `transformed-data.csv` foi criado e contém o conteúdo.

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

14. Para carregar dados de um script shell, você usará a utilidade cliente `psql` de forma não interativa. Isso é feito enviando os comandos do banco de dados através de um pipeline de comandos para o `psql` com a ajuda do comando `echo`.

O comando do PostgreSQL para copiar dados de um arquivo CSV para uma tabela é `COPY`.

A estrutura básica do comando que usaremos em nosso script é,

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

Agora, adicione as linhas abaixo ao final do script ‘csv2db.sh’ e salve o arquivo.

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

Exercício 6 - Execute o script final

1. Execute o script.

```
bash csv2db.sh
```

2. Agora, adicione a linha abaixo ao final do script 'csv2db.sh' e salve o arquivo.

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

3. Execute o script para verificar se a tabela users está populada com os dados.

```
bash csv2db.sh
```

Parabéns! Você criou um script ETL usando shell scripting.

Exercícios de prática

1. Copie os dados do arquivo 'web-server-access-log.txt.gz' para a tabela 'access_log' no banco de dados PostgreSQL 'template1'.

   O arquivo está disponível no seguinte local: 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

   As seguintes são as colunas e seus tipos de dados no arquivo:

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

   As colunas que precisamos copiar para a tabela são as quatro primeiras colunas: timestamp, latitude, longitude e visitorid.

   NOTA: O arquivo vem com um cabeçalho. Portanto, use a opção 'HEADER' no comando 'COPY'.

O problema pode ser resolvido completando as seguintes tarefas:

1. Vá ao menu Ferramentas do SkillsNetwork e inicie o servidor SQL Postgres, se ainda não estiver em execução.
2. Crie uma tabela chamada `access_log` para armazenar o timestamp, latitude, longitude e visitorid.

▶ Clique aqui para Dica
▶ Clique aqui para Solução

```
\c template1;
```

Passo 3: Depois de se conectar ao banco de dados, execute o comando para criar a tabela chamada 'access_log':

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

Tarefa 3. Crie um script shell chamado `cp-access-log.sh` e adicione comandos para completar as tarefas restantes para extrair e copiar os dados para o banco de dados.

Crie um script shell para adicionar comandos que completem o restante das tarefas.

▶ Clique aqui para Dica

Tarefa 4. Baixar o arquivo de log de acesso.

Adicione o comando `wget` ao script para baixar o arquivo.

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Bash%20Scripting/ETL%20using%20shell%20
```

Tarefa 5. Descompactar o arquivo gzip.

Adicione o código para executar o comando gunzip para descompactar o arquivo `.gz` e extrair o arquivo `.txt` ao script.

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

A opção **-f** do gunzip é para sobrescrever o arquivo se ele já existir.

Tarefa 6. Extraia os campos necessários do arquivo.

Extraia timestamp, latitude, longitude e visitorid, que são os quatro primeiros campos do arquivo usando o comando `cut`.

As colunas no arquivo web-server-access-log.txt são delimitadas por '#'.

▶ Clique aqui para Dica
▶ Clique aqui para Solução

Tarefa 7. Redirecionar a saída extraída para um arquivo.

Redirecione os dados extraídos para um arquivo chamado `extracted-data.txt`

▶ Clique aqui para Dica
▶ Clique aqui para Solução

Tarefa 8. Transforme os dados em formato CSV.

As colunas extraídas são separadas pelo delimitador original "#".

Precisamos converter isso em um arquivo delimitado por ",".

▶ Clique aqui para Dica
▶ Clique aqui para Solução

Tarefa 9. Carregar os dados na tabela `access_log` no PostgreSQL

O comando PostgreSQL para copiar dados de um arquivo CSV para uma tabela é `COPY`.

A estrutura básica do comando é,

```
COPY nome_tabela FROM 'nome_arquivo' DELIMITERS 'caractere_delimitador' FORMAT;
```

O arquivo vem com um cabeçalho. Portanto, use a opção 'HEADER' no comando 'COPY'.

Invocar este comando a partir do shellscript, enviando-o como entrada para o comando de filtro 'psql'.

▶ Clique aqui para Dica
▶ Clique aqui para Solução

Tarefa 10. Execute o script final.

Execute o script final.

▶ Clique aqui para a Solução

Tarefa 11. Verifique consultando o banco de dados.

▶ Clique aqui para Dica
▶ Clique aqui para Solução