Assignment 2: ETL with Bash

Course: Data Acquisition and Management
Instructor: Muhammad Shahin

PGDM PREDICTIVE ANALYTICS

GURDARSHAN SINGH
Due Date: February 10, 2025

Executive Summary:

The goal of this assignment is to design an ETL pipeline that downloads COVID-19 data, processes it, and loads it into a PostgreSQL database for further analysis. The key steps of this project include:

1. **Extracting** raw COVID-19 data from an external source.
2. **Transforming** the data by selecting relevant columns and cleaning invalid or missing values.
3. **Loading** the transformed data into a PostgreSQL database using Python.
4. **Scheduling** the ETL job to run at specified intervals using cron jobs.
5. **Logging** the process and checking the logs for successful execution.

This process is vital for automating data management tasks, especially when handling large datasets, ensuring that the data is stored efficiently and available for analysis.

1. Creating the **etl.sh** file in the bash then changing its permission and verifying it existence and permissions in the system.

2. **Extract - Downloading the Data:**
   Theory: The first step in any ETL process is **extraction**. In this case, we are downloading the COVID-19 dataset from a publicly available source. The dataset provides important information like the number of COVID-19 cases for various countries and regions. This raw data is typically available in formats like CSV or JSON, and it needs to be fetched from a remote location using a method like HTTP requests or APIs.
   Implementation:
   The data is downloaded using **curl** in the shell script. The **curl** command fetches the data from a specified URL and saves it to the local machine.

Purpose: *This is the foundational step of the ETL pipeline, providing raw data that will be processed and transformed.*

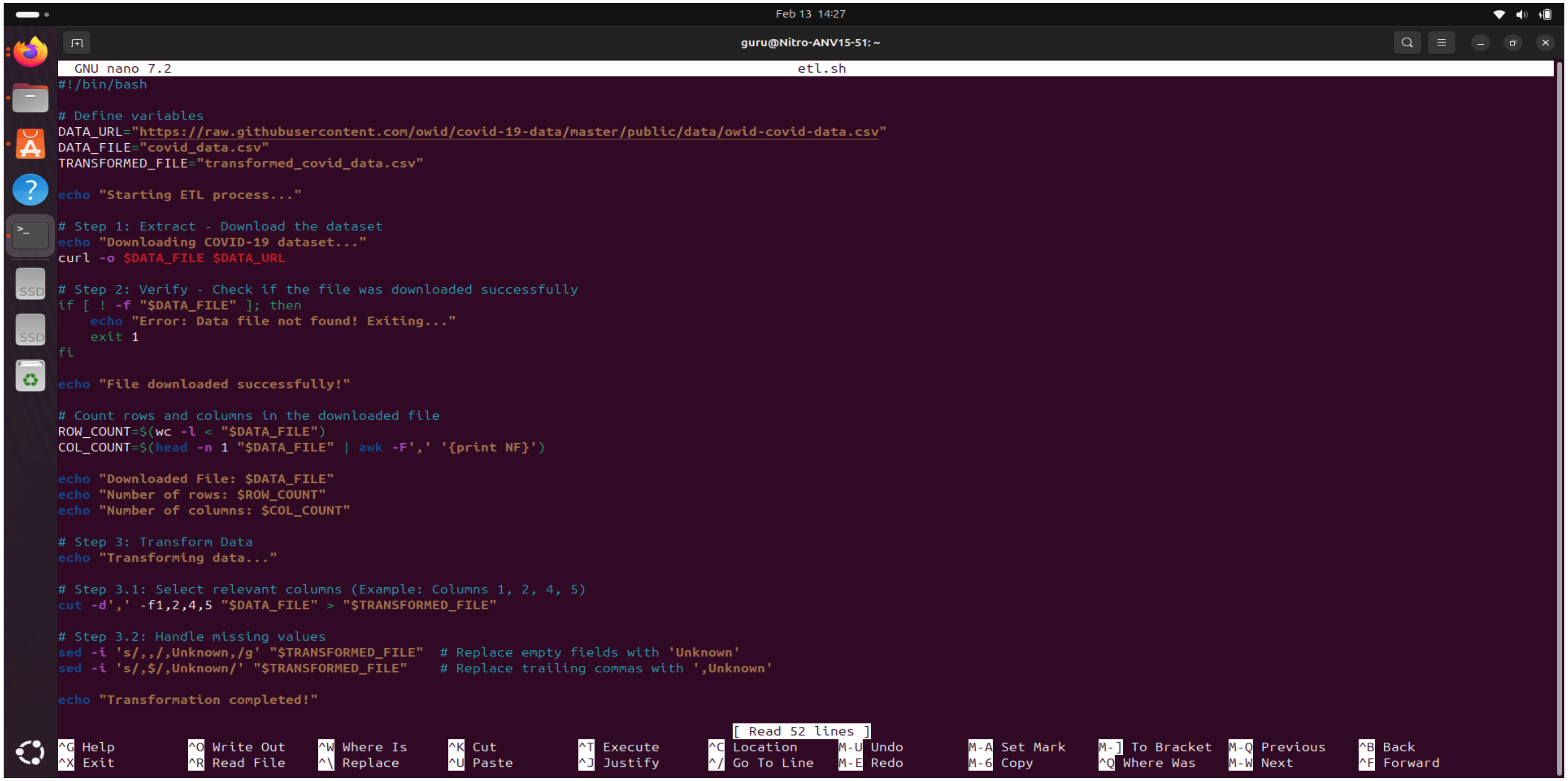
3. **Transform - Cleaning and Formatting the Data:**
   Theory: The **transformation** step involves modifying the extracted data into a suitable format for storage. This might involve several operations like:

- **Selecting columns**: Only the necessary columns should be kept, such as iso_code, location, date, and total_cases.
- **Cleaning missing or invalid values**: Missing or invalid data (e.g., "Unknown" or empty strings) should be replaced with a consistent placeholder like "Unknown" or NULL.
- **Formatting the data**: Ensure the data is in the correct format for the database, e.g., dates should be in a valid YYYY-MM-DD format, and numerical values should be consistent (integers or floats).

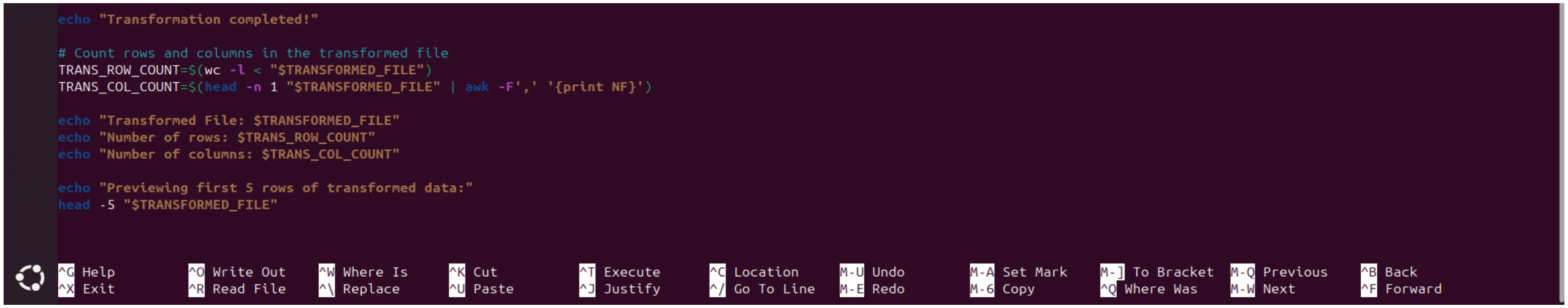
Implementation:

- Using Bash commands like **cut**, **sed**, and **awk**, the CSV file is processed to select relevant columns and clean any invalid or missing data.

Purpose: *The transformation step ensures that the data is cleaned, properly formatted, and ready for efficient storage in the PostgreSQL database.*

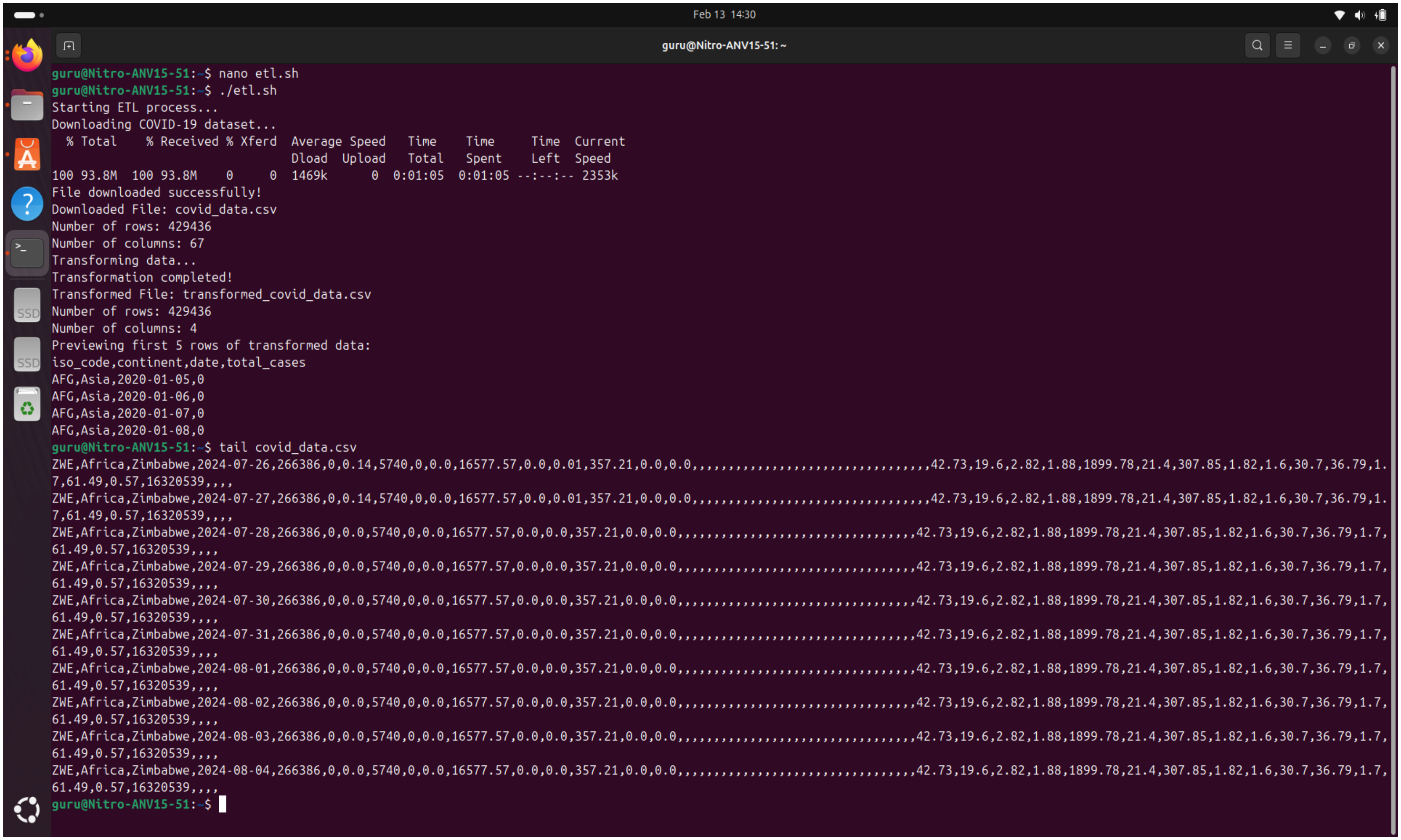


```
#!/bin/bash

# Define variables
DATA_URL="https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv"
DATA_FILE="covid_data.csv"
TRANSFORMED_FILE="transformed_covid_data.csv"

echo "Starting ETL process..."

# Step 1: Extract - Download the dataset
echo "Downloading COVID-19 dataset..."
curl -o $DATA_FILE $DATA_URL

# Step 2: Verify - Check if the file was downloaded successfully
if [ ! -f "$DATA_FILE" ]; then
    echo "Error: Data file not found! Exiting..."
    exit 1
fi

echo "File downloaded successfully!"

# Count rows and columns in the downloaded file
ROW_COUNT=$(wc -l < "$DATA_FILE")
COL_COUNT=$(head -n 1 "$DATA_FILE" | awk -F',' '{print NF}')

echo "Downloaded File: $DATA_FILE"
echo "Number of rows: $ROW_COUNT"
echo "Number of columns: $COL_COUNT"

# Step 3: Transform Data
echo "Transforming data..."

# Step 3.1: Select relevant columns (Example: Columns 1, 2, 4, 5)
cut -d',' -f1,2,4,5 "$DATA_FILE" > "$TRANSFORMED_FILE"

# Step 3.2: Handle missing values
sed -i 's/,,/,Unknown,/g' "$TRANSFORMED_FILE"  # Replace empty fields with 'Unknown'
sed -i 's/,$/,Unknown/' "$TRANSFORMED_FILE"    # Replace trailing commas with ',Unknown'

echo "Transformation completed!"

                                       [ Read 52 lines ]
```



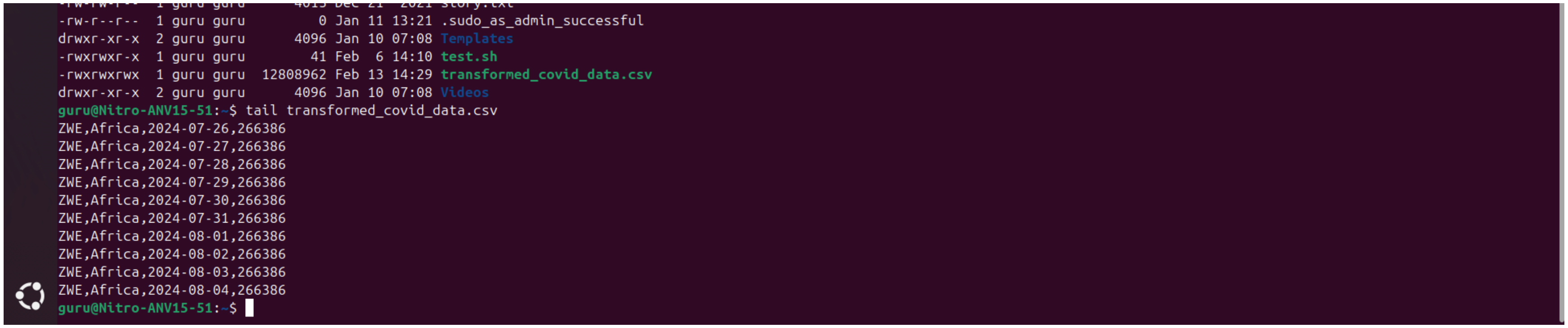
```
echo "Transformation completed!"

# Count rows and columns in the transformed file
TRANS_ROW_COUNT=$(wc -l < "$TRANSFORMED_FILE")
TRANS_COL_COUNT=$(head -n 1 "$TRANSFORMED_FILE" | awk -F',' '{print NF}')

echo "Transformed File: $TRANSFORMED_FILE"
echo "Number of rows: $TRANS_ROW_COUNT"
echo "Number of columns: $TRANS_COL_COUNT"

echo "Previewing first 5 rows of transformed data:"
head -5 "$TRANSFORMED_FILE"
```

Extraction and transformation process after running etl.sh file.



```
guru@Nitro-ANV15-51:~$ nano etl.sh
guru@Nitro-ANV15-51:~$ ./etl.sh
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  1469k      0  0:01:05  0:01:05 --:--:-- 2353k
File downloaded successfully!
Downloaded File: covid_data.csv
Number of rows: 429436
Number of columns: 67
Transforming data...
Transformation completed!
Transformed File: transformed_covid_data.csv
Number of rows: 429436
Number of columns: 4
Previewing first 5 rows of transformed data:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
guru@Nitro-ANV15-51:~$ tail covid_data.csv
ZWE,Africa,Zimbabwe,2024-07-26,266386,0,0.14,5740,0,0.0,16577.57,0.0,0.01,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.
7,61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-07-27,266386,0,0.14,5740,0,0.0,16577.57,0.0,0.01,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.
7,61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-07-28,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-07-29,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-07-30,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-07-31,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-08-01,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-08-02,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-08-03,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
ZWE,Africa,Zimbabwe,2024-08-04,266386,0,0.0,5740,0,0.0,16577.57,0.0,0.0,357.21,0.0,0.0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,42.73,19.6,2.82,1.88,1899.78,21.4,307.85,1.82,1.6,30.7,36.79,1.7,
61.49,0.57,16320539,,,,
guru@Nitro-ANV15-51:~$
```

Above screenshot show the details about rows and columns of extracted file (covid_data) and as well as for the transformed file(transformed_covid_data).



```
-rw-rw-r--  1 guru guru      4015 Dec 21  2021 story.txt
-rw-r--r--  1 guru guru         0 Jan 11 13:21 .sudo_as_admin_successful
drwxr-xr-x  2 guru guru      4096 Jan 10 07:08 Templates
-rwxrwxr-x  1 guru guru        41 Feb  6 14:10 test.sh
-rwxrwxrwx  1 guru guru  12808962 Feb 13 14:29 transformed_covid_data.csv
drwxr-xr-x  2 guru guru      4096 Jan 10 07:08 Videos
guru@Nitro-ANV15-51:~$ tail transformed_covid_data.csv
ZWE,Africa,2024-07-26,266386
ZWE,Africa,2024-07-27,266386
ZWE,Africa,2024-07-28,266386
ZWE,Africa,2024-07-29,266386
ZWE,Africa,2024-07-30,266386
ZWE,Africa,2024-07-31,266386
ZWE,Africa,2024-08-01,266386
ZWE,Africa,2024-08-02,266386
ZWE,Africa,2024-08-03,266386
ZWE,Africa,2024-08-04,266386
guru@Nitro-ANV15-51:~$ 
```

3. Load - Inserting Transformed Data into PostgreSQL:

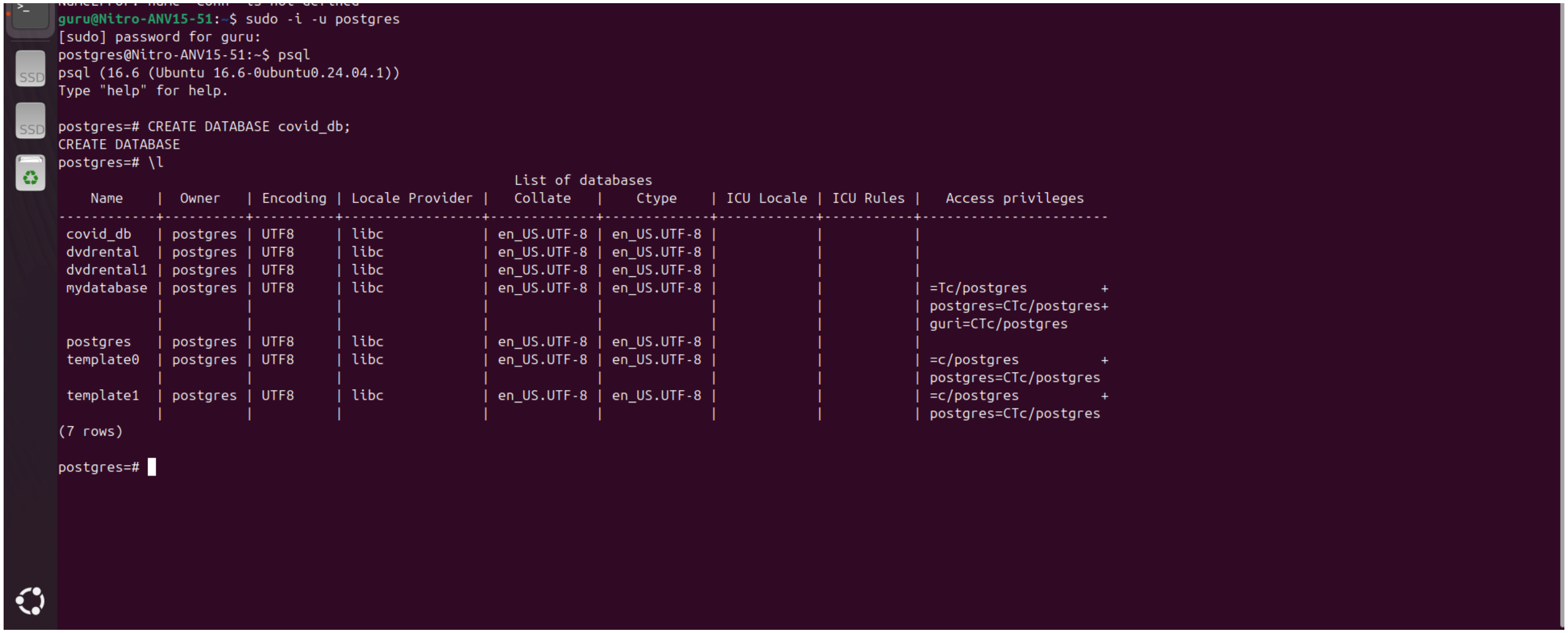
Theory: In the **load** step, the transformed data is stored in a PostgreSQL database. PostgreSQL is a powerful relational database management system (RDBMS) that can efficiently handle large datasets. The load process involves creating a database and a table with the appropriate structure (columns and data types) and inserting the cleaned data into the table.

Implementation:

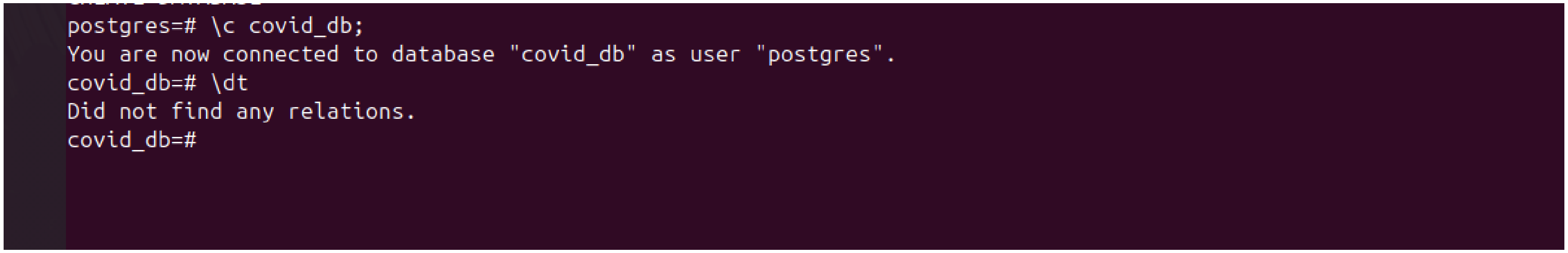
- A Python script (load_data.py) is used to connect to the PostgreSQL database and insert the transformed data into the table.
- The COPY command in PostgreSQL is used to efficiently load the data from the CSV file into the covid table. The COPY command is optimized for bulk loading and is much faster than inserting rows one by one.

Purpose: This step allows us to store the data in a structured manner, making it available for querying and analysis in PostgreSQL.

First, I have created a database named covid_db using postgres.

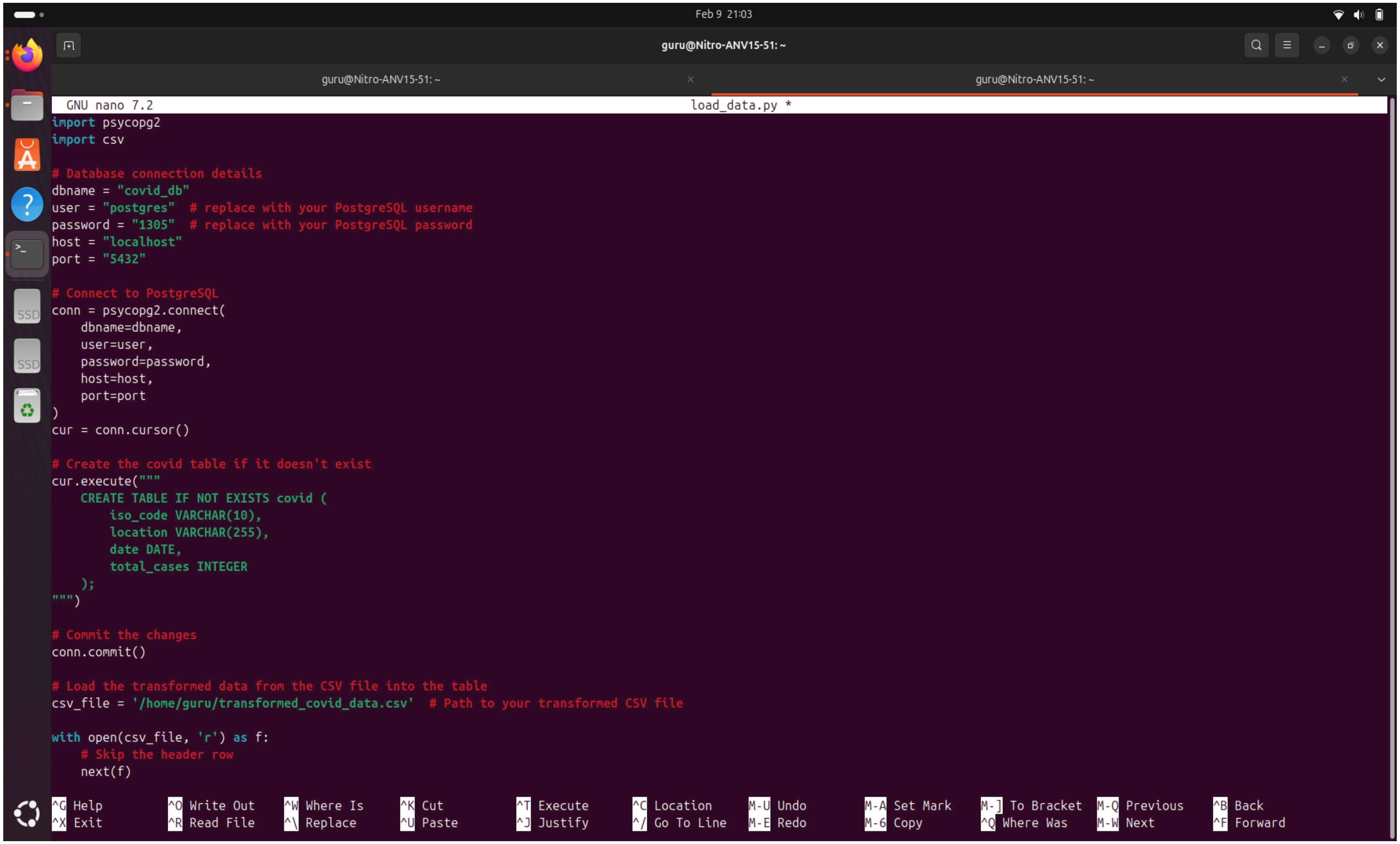


```
NameError: name 'conn' is not defined
guru@Nitro-ANV15-51:~$ sudo -i -u postgres
[sudo] password for guru:
postgres@Nitro-ANV15-51:~$ psql
psql (16.6 (Ubuntu 16.6-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# CREATE DATABASE covid_db;
CREATE DATABASE
postgres=# \l
                                                        List of databases
    Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
------------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 covid_db   | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 dvdrental  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 dvdrental1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 mydatabase | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =Tc/postgres         +
            |          |          |                 |             |             |            |           | postgres=CTc/postgres+
            |          |          |                 |             |             |            |           | guri=CTc/postgres
 postgres   | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
            |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
            |          |          |                 |             |             |            |           | postgres=CTc/postgres
(7 rows)

postgres=#
```

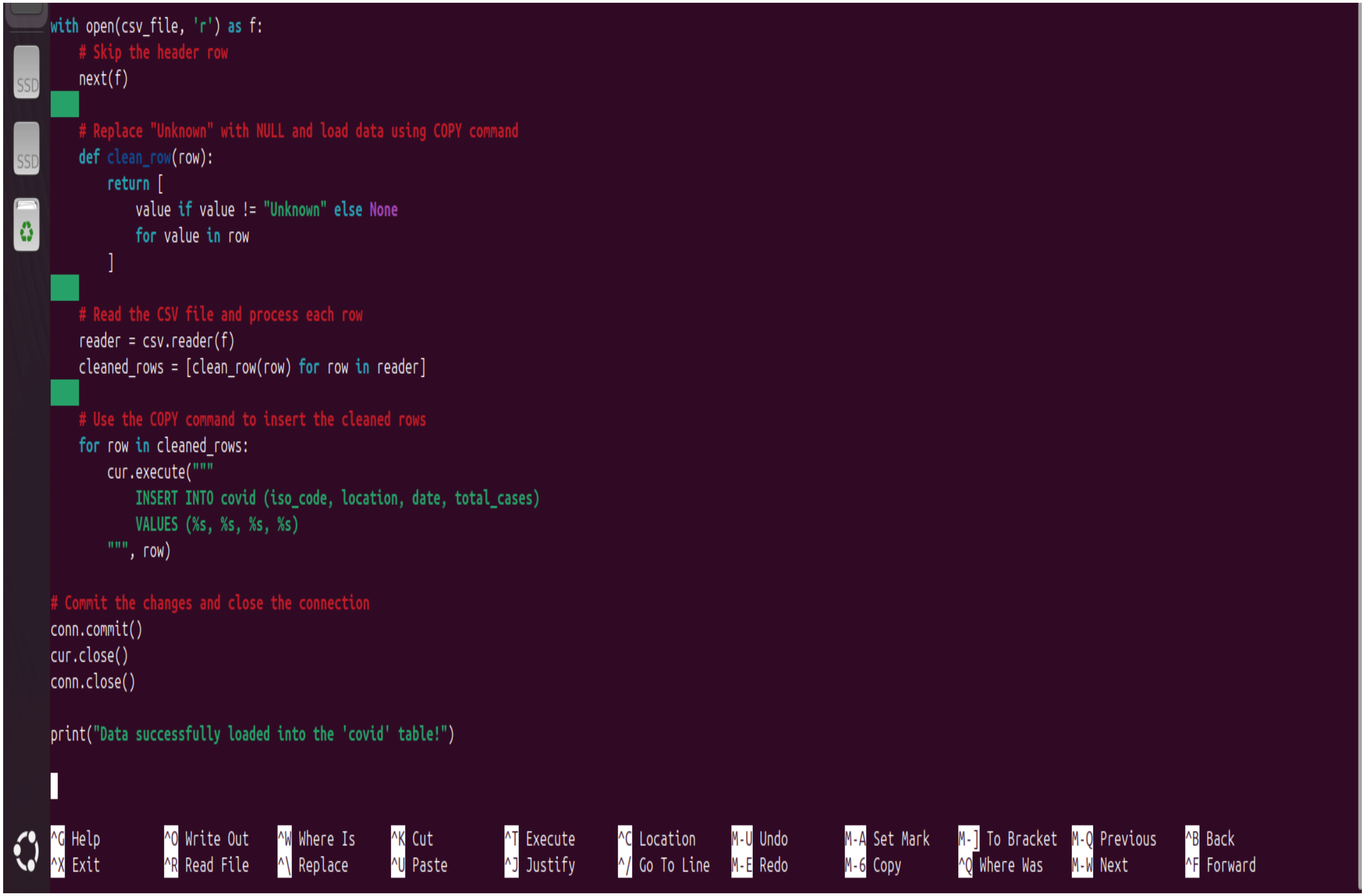


```
CREATE DATABASE
postgres=# \c covid_db;
You are now connected to database "covid_db" as user "postgres".
covid_db=# \dt
Did not find any relations.
covid_db=#
```

Then created python script of name load_data.py and write the code in that to load the transformed data in the covid_db.

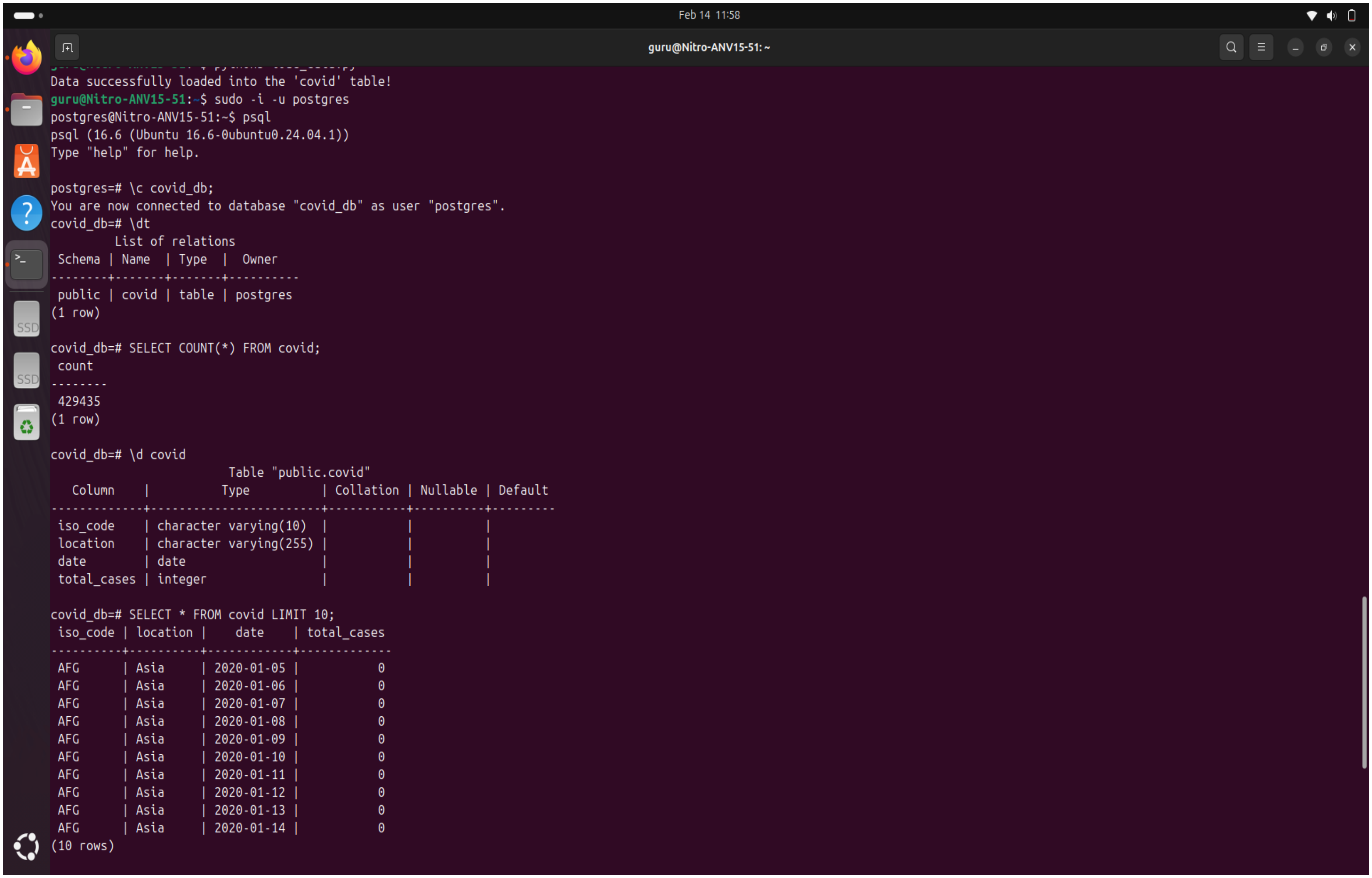


```
import psycopg2
import csv

# Database connection details
dbname = "covid_db"
user = "postgres"  # replace with your PostgreSQL username
password = "1305"  # replace with your PostgreSQL password
host = "localhost"
port = "5432"

# Connect to PostgreSQL
conn = psycopg2.connect(
    dbname=dbname,
    user=user,
    password=password,
    host=host,
    port=port
)
cur = conn.cursor()

# Create the covid table if it doesn't exist
cur.execute("""
    CREATE TABLE IF NOT EXISTS covid (
        iso_code VARCHAR(10),
        location VARCHAR(255),
        date DATE,
        total_cases INTEGER
    );
""")

# Commit the changes
conn.commit()

# Load the transformed data from the CSV file into the table
csv_file = '/home/guru/transformed_covid_data.csv'  # Path to your transformed CSV file

with open(csv_file, 'r') as f:
    # Skip the header row
    next(f)
```

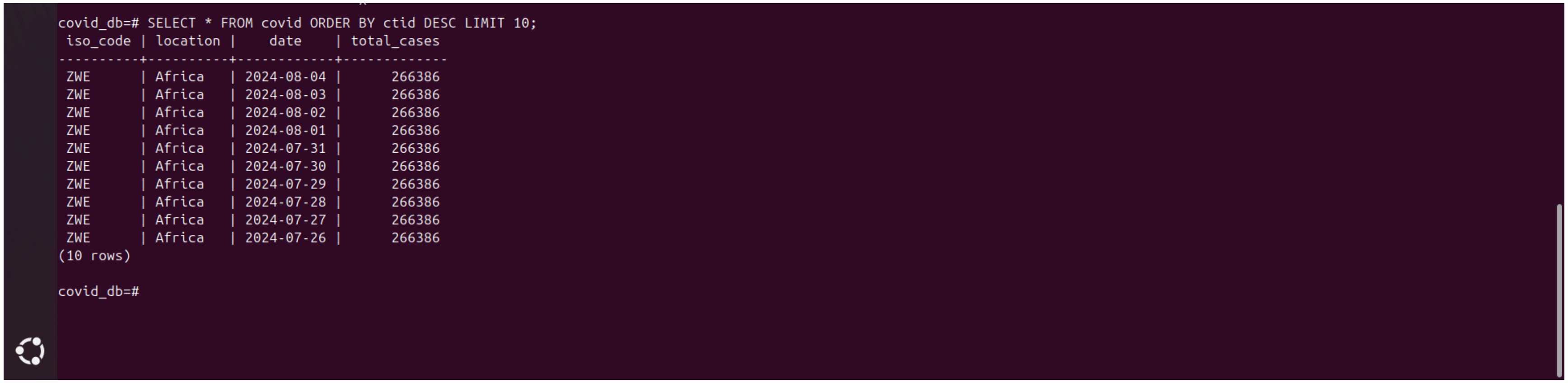


```
with open(csv_file, 'r') as f:
    # Skip the header row
    next(f)

    # Replace "Unknown" with NULL and load data using COPY command
    def clean_row(row):
        return [
            value if value != "Unknown" else None
            for value in row
        ]

    # Read the CSV file and process each row
    reader = csv.reader(f)
    cleaned_rows = [clean_row(row) for row in reader]

    # Use the COPY command to insert the cleaned rows
    for row in cleaned_rows:
        cur.execute("""
            INSERT INTO covid (iso_code, location, date, total_cases)
            VALUES (%s, %s, %s, %s)
        """, row)

# Commit the changes and close the connection
conn.commit()
cur.close()
conn.close()

print("Data successfully loaded into the 'covid' table!")
```

After running the load_data.py file



```
Data successfully loaded into the 'covid' table!
guru@Nitro-ANV15-51:~$ sudo -i -u postgres
postgres@Nitro-ANV15-51:~$ psql
psql (16.6 (Ubuntu 16.6-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# \c covid_db;
You are now connected to database "covid_db" as user "postgres".
covid_db=# \dt
         List of relations
 Schema | Name  | Type  |  Owner
--------+-------+-------+----------
 public | covid | table | postgres
(1 row)

covid_db=# SELECT COUNT(*) FROM covid;
 count
--------
 429435
(1 row)

covid_db=# \d covid
                         Table "public.covid"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 iso_code    | character varying(10)  |           |          |
 location    | character varying(255) |           |          |
 date        | date                   |           |          |
 total_cases | integer                |           |          |

covid_db=# SELECT * FROM covid LIMIT 10;
 iso_code | location |    date    | total_cases
----------+----------+------------+-------------
 AFG      | Asia     | 2020-01-05 |           0
 AFG      | Asia     | 2020-01-06 |           0
 AFG      | Asia     | 2020-01-07 |           0
 AFG      | Asia     | 2020-01-08 |           0
 AFG      | Asia     | 2020-01-09 |           0
 AFG      | Asia     | 2020-01-10 |           0
 AFG      | Asia     | 2020-01-11 |           0
 AFG      | Asia     | 2020-01-12 |           0
 AFG      | Asia     | 2020-01-13 |           0
 AFG      | Asia     | 2020-01-14 |           0
(10 rows)
```



```
covid_db=# SELECT * FROM covid ORDER BY ctid DESC LIMIT 10;
 iso_code | location |    date    | total_cases
----------+----------+------------+------------
 ZWE      | Africa   | 2024-08-04 |      266386
 ZWE      | Africa   | 2024-08-03 |      266386
 ZWE      | Africa   | 2024-08-02 |      266386
 ZWE      | Africa   | 2024-08-01 |      266386
 ZWE      | Africa   | 2024-07-31 |      266386
 ZWE      | Africa   | 2024-07-30 |      266386
 ZWE      | Africa   | 2024-07-29 |      266386
 ZWE      | Africa   | 2024-07-28 |      266386
 ZWE      | Africa   | 2024-07-27 |      266386
 ZWE      | Africa   | 2024-07-26 |      266386
(10 rows)

covid_db=#
```

4. Automate the ETL Process with Cron:

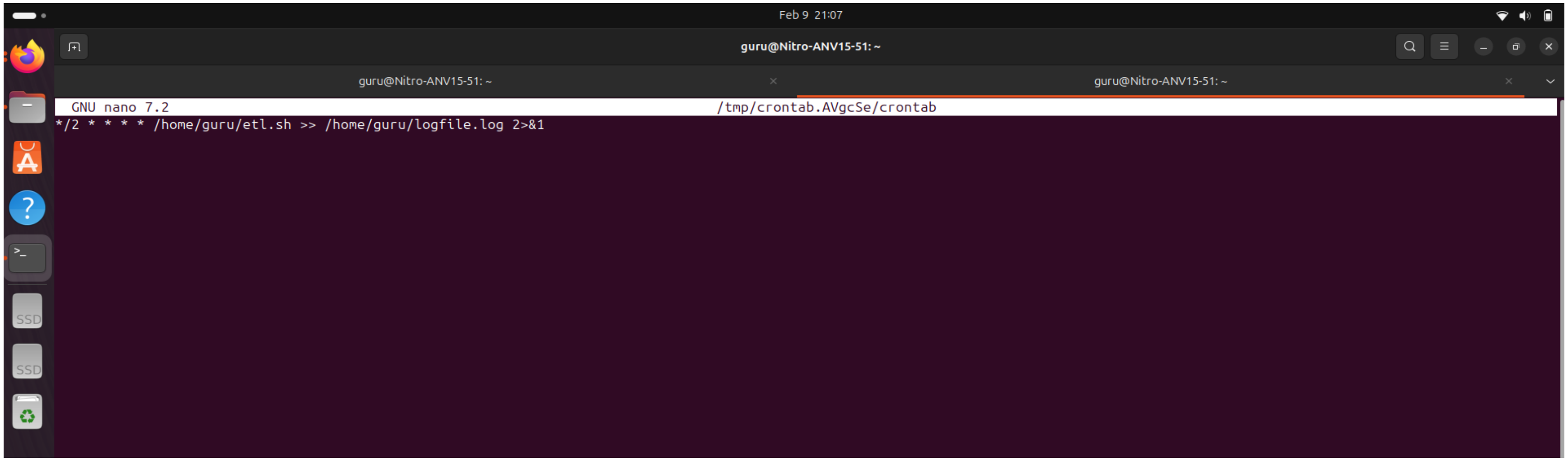
Theory: The **automation** of the ETL process is essential for keeping the database up to date. A cron job is a time-based job scheduler in Unix-like operating systems. By creating a cron job, the ETL process can be scheduled to run automatically at specified intervals, ensuring that the data is refreshed regularly (e.g., every day at midnight).

Implementation:

- For sampling, A cron job is configured to run the etl.sh script every 2 min.
- The goal is to create a cron job for running the process at 12:10 everyday for which I will configure the code later after running the cron job every 2 min first.
- The cron job will run the script at the specified time, automatically performing the download, transformation, and loading steps.

Purpose: This automation ensures that the ETL process runs without manual intervention, making it efficient and reliable.

Sample cron job for running process atonmatically every 2 min.



```
  GNU nano 7.2                                              /tmp/crontab.AVgcSe/crontab
*/2 * * * * /home/guru/etl.sh >> /home/guru/logfile.log 2>&1
```

5. Logging and Monitoring the Process:

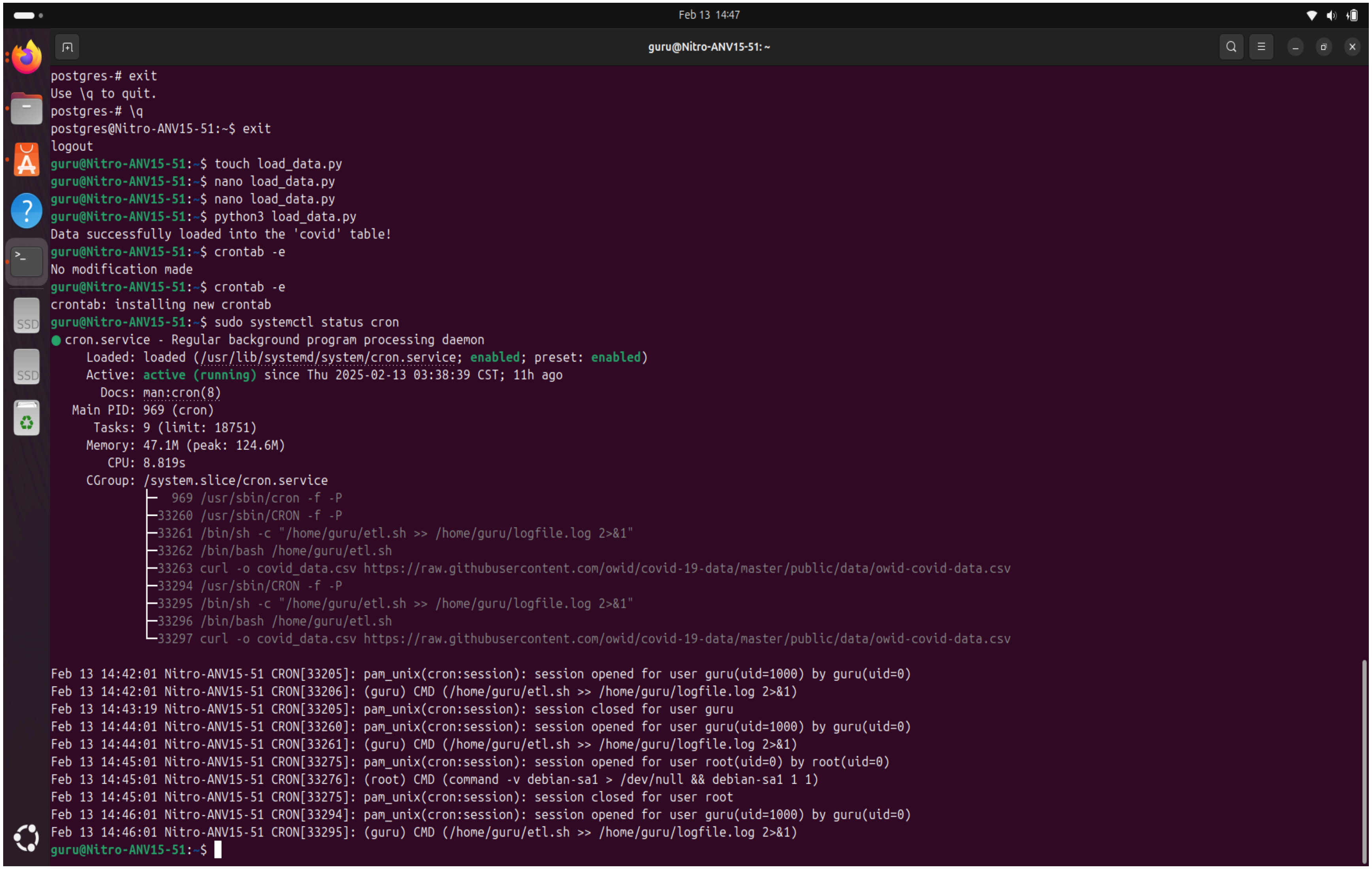
Theory: Logging is crucial for debugging, monitoring, and ensuring that the ETL pipeline is functioning as expected. By logging the process, any issues that arise during execution can be traced and addressed quickly. Logs also help in keeping track of the execution status and errors.

Implementation:

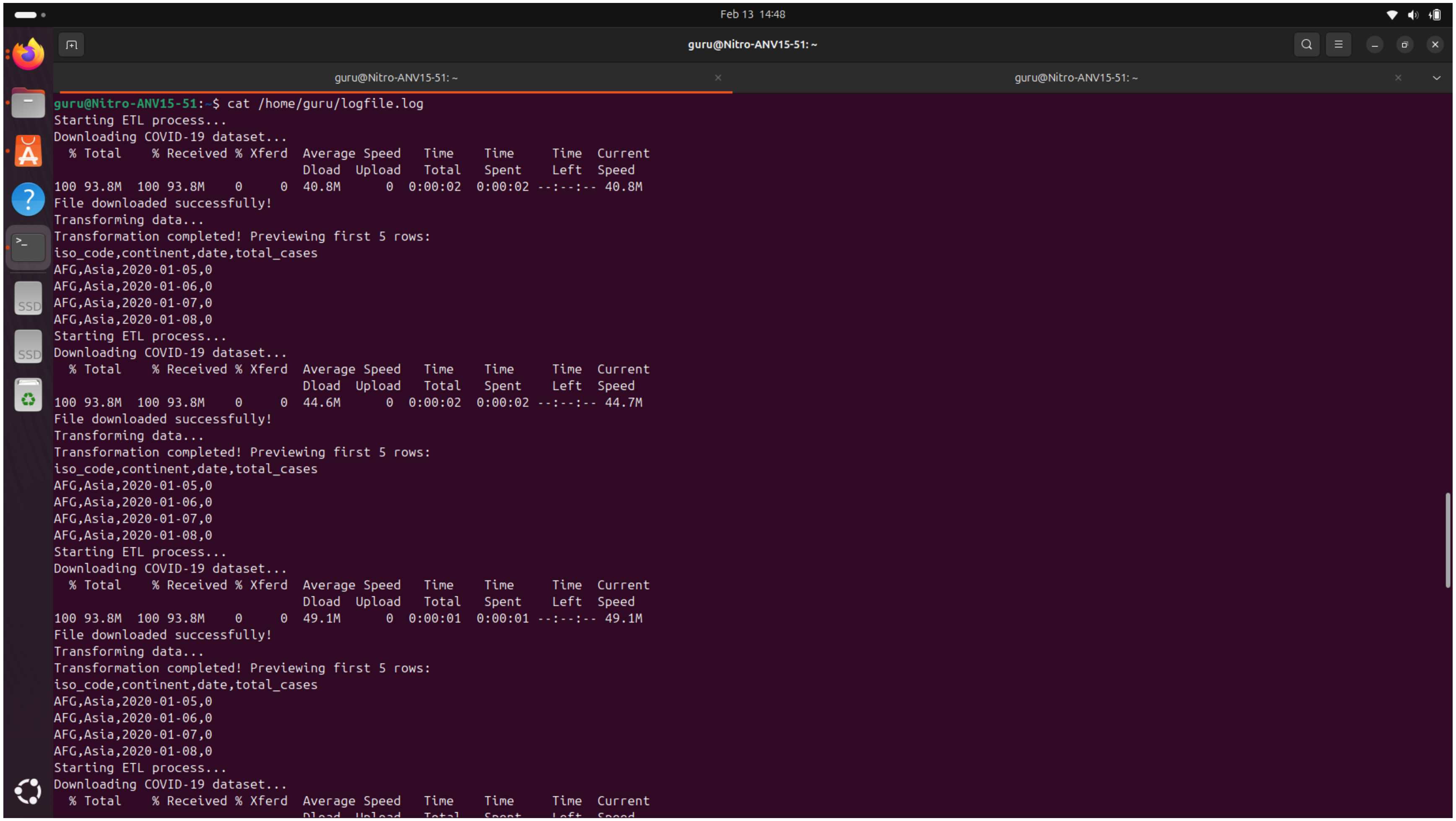
- The cron job is configured to log the output of the etl.sh script to a log file.
- The log file can be reviewed using tools like tail to ensure the script runs successfully and to troubleshoot errors if any.

Purpose: Logging ensures that the ETL process is traceable, errors are captured, and the system is monitored effectively.

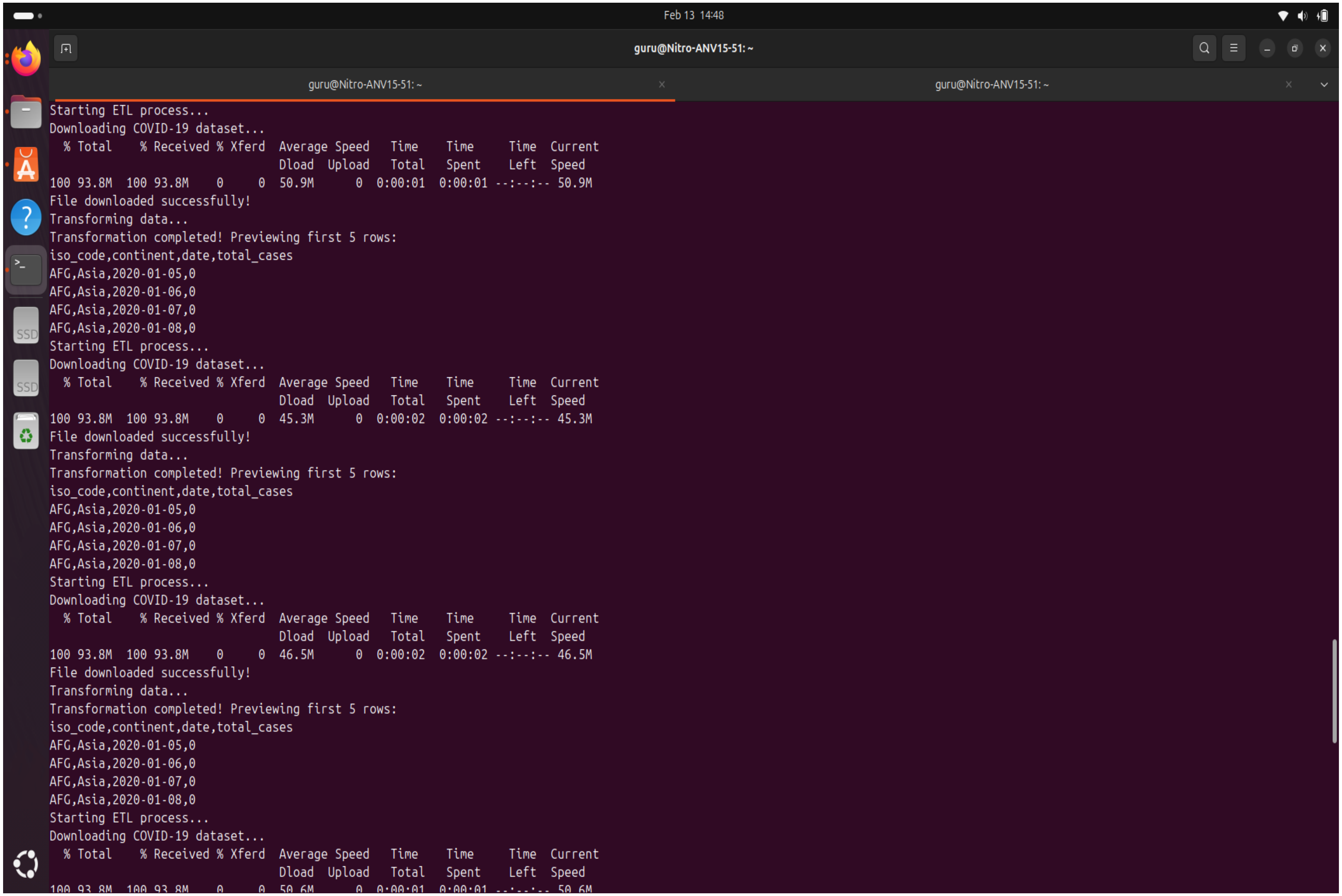
After our sample running of cron job every 2 min here are results:



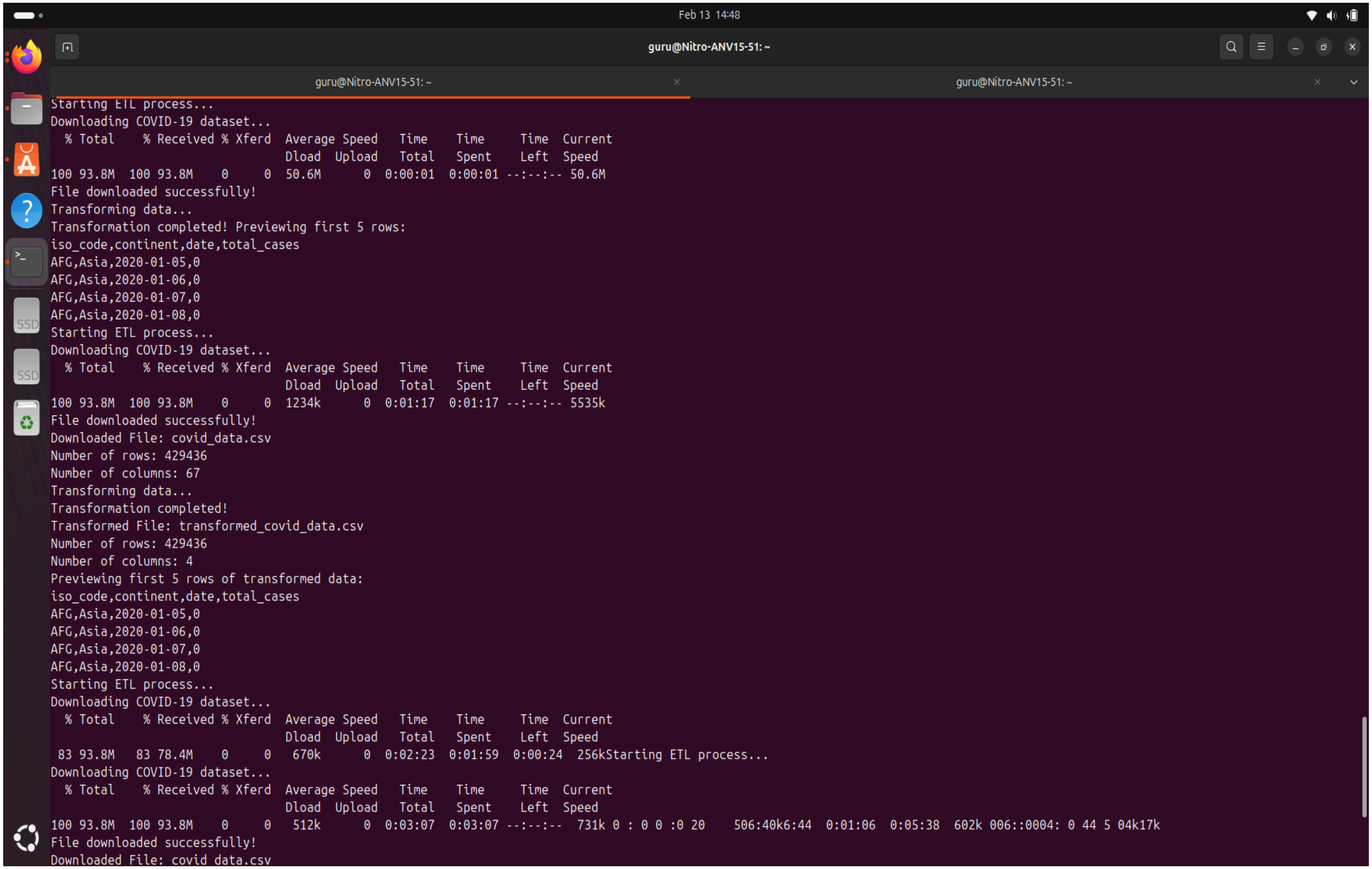
```
postgres-# exit
Use \q to quit.
postgres-# \q
postgres@Nitro-ANV15-51:~$ exit
logout
guru@Nitro-ANV15-51:~$ touch load_data.py
guru@Nitro-ANV15-51:~$ nano load_data.py
guru@Nitro-ANV15-51:~$ nano load_data.py
guru@Nitro-ANV15-51:~$ python3 load_data.py
Data successfully loaded into the 'covid' table!
guru@Nitro-ANV15-51:~$ crontab -e
No modification made
guru@Nitro-ANV15-51:~$ crontab -e
crontab: installing new crontab
guru@Nitro-ANV15-51:~$ sudo systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/usr/lib/systemd/system/cron.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-02-13 03:38:39 CST; 11h ago
       Docs: man:cron(8)
   Main PID: 969 (cron)
      Tasks: 9 (limit: 18751)
     Memory: 47.1M (peak: 124.6M)
        CPU: 8.819s
     CGroup: /system.slice/cron.service
             ├─  969 /usr/sbin/cron -f -P
             ├─33260 /usr/sbin/CRON -f -P
             ├─33261 /bin/sh -c "/home/guru/etl.sh >> /home/guru/logfile.log 2>&1"
             ├─33262 /bin/bash /home/guru/etl.sh
             ├─33263 curl -o covid_data.csv https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv
             ├─33294 /usr/sbin/CRON -f -P
             ├─33295 /bin/sh -c "/home/guru/etl.sh >> /home/guru/logfile.log 2>&1"
             ├─33296 /bin/bash /home/guru/etl.sh
             └─33297 curl -o covid_data.csv https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/owid-covid-data.csv

Feb 13 14:42:01 Nitro-ANV15-51 CRON[33205]: pam_unix(cron:session): session opened for user guru(uid=1000) by guru(uid=0)
Feb 13 14:42:01 Nitro-ANV15-51 CRON[33206]: (guru) CMD (/home/guru/etl.sh >> /home/guru/logfile.log 2>&1)
Feb 13 14:43:19 Nitro-ANV15-51 CRON[33205]: pam_unix(cron:session): session closed for user guru
Feb 13 14:44:01 Nitro-ANV15-51 CRON[33260]: pam_unix(cron:session): session opened for user guru(uid=1000) by guru(uid=0)
Feb 13 14:44:01 Nitro-ANV15-51 CRON[33261]: (guru) CMD (/home/guru/etl.sh >> /home/guru/logfile.log 2>&1)
Feb 13 14:45:01 Nitro-ANV15-51 CRON[33275]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0)
Feb 13 14:45:01 Nitro-ANV15-51 CRON[33276]: (root) CMD (command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Feb 13 14:45:01 Nitro-ANV15-51 CRON[33275]: pam_unix(cron:session): session closed for user root
Feb 13 14:46:01 Nitro-ANV15-51 CRON[33294]: pam_unix(cron:session): session opened for user guru(uid=1000) by guru(uid=0)
Feb 13 14:46:01 Nitro-ANV15-51 CRON[33295]: (guru) CMD (/home/guru/etl.sh >> /home/guru/logfile.log 2>&1)
guru@Nitro-ANV15-51:~$
```



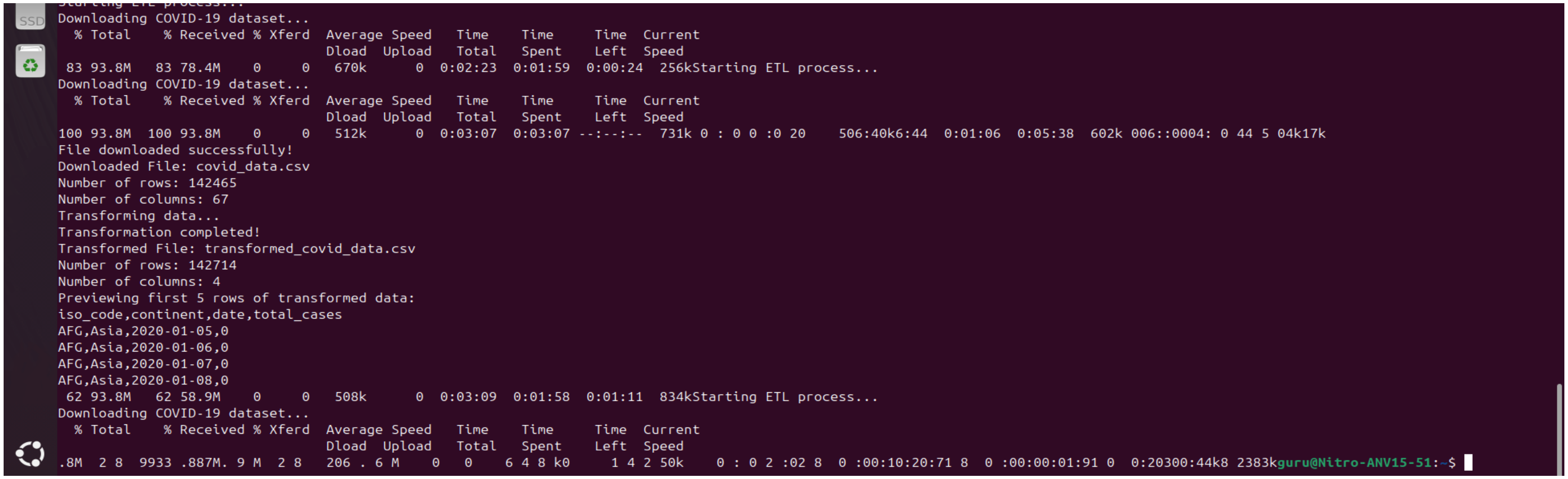
```
guru@Nitro-ANV15-51:~$ cat /home/guru/logfile.log
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  40.8M      0  0:00:02  0:00:02 --:--:-- 40.8M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  44.6M      0  0:00:02  0:00:02 --:--:-- 44.7M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  49.1M      0  0:00:01  0:00:01 --:--:-- 49.1M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
```



```
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  50.9M      0  0:00:01  0:00:01 --:--:-- 50.9M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  45.3M      0  0:00:02  0:00:02 --:--:-- 45.3M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  46.5M      0  0:00:02  0:00:02 --:--:-- 46.5M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  50.6M      0  0:00:01  0:00:01 --:--:-- 50.6M
```

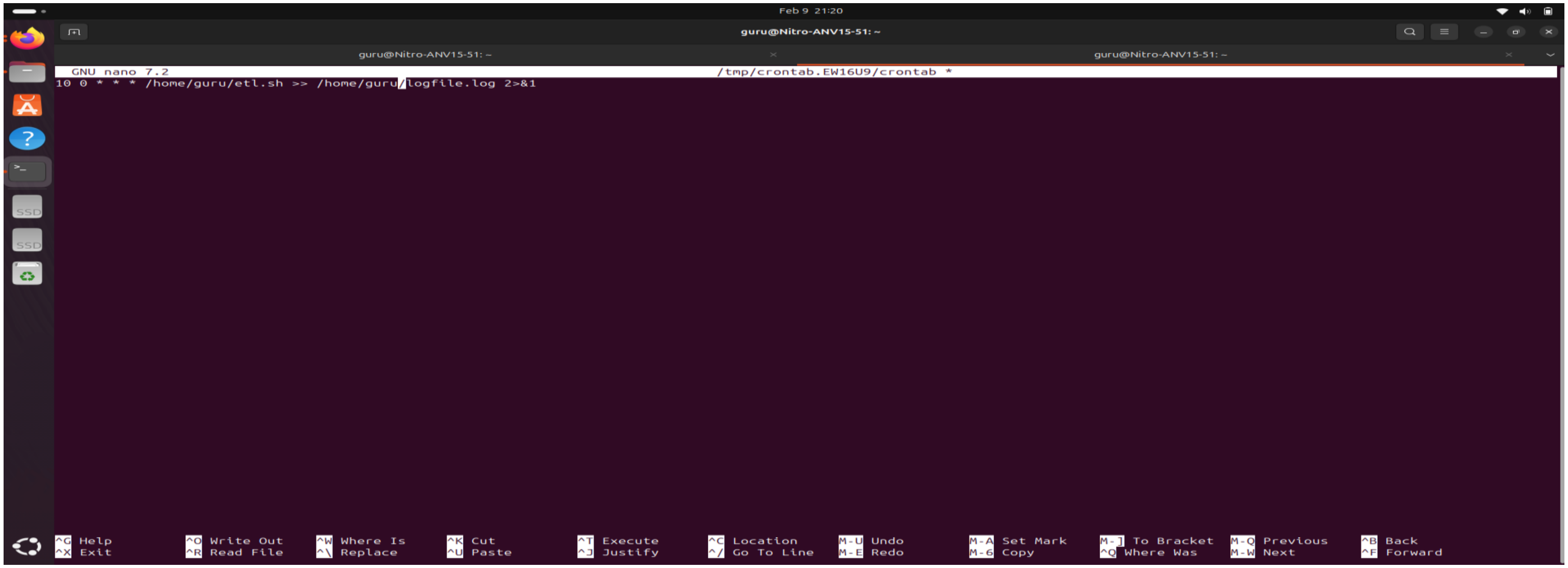


```
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  50.6M      0  0:00:01  0:00:01 --:--:-- 50.6M
File downloaded successfully!
Transforming data...
Transformation completed! Previewing first 5 rows:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0  1234k      0  0:01:17  0:01:17 --:--:-- 5535k
File downloaded successfully!
Downloaded File: covid_data.csv
Number of rows: 429436
Number of columns: 67
Transforming data...
Transformation completed!
Transformed File: transformed_covid_data.csv
Number of rows: 429436
Number of columns: 4
Previewing first 5 rows of transformed data:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 83 93.8M   83 78.4M    0     0   670k      0  0:02:23  0:01:59  0:00:24  256kStarting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0   512k      0  0:03:07  0:03:07 --:--:--  731k 0 : 0 0 :0 20    506:40k6:44  0:01:06  0:05:38  602k 006::0004: 0 44 5 04k17k
File downloaded successfully!
Downloaded File: covid data.csv
```



```
Starting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
 83 93.8M   83 78.4M    0     0   670k      0  0:02:23  0:01:59  0:00:24  256kStarting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 93.8M  100 93.8M    0     0   512k      0  0:03:07  0:03:07 --:--:--  731k 0 : 0 0 :0 20    506:40k6:44  0:01:06  0:05:38  602k 006::0004: 0 44 5 04k17k
File downloaded successfully!
Downloaded File: covid_data.csv
Number of rows: 142465
Number of columns: 67
Transforming data...
Transformation completed!
Transformed File: transformed_covid_data.csv
Number of rows: 142714
Number of columns: 4
Previewing first 5 rows of transformed data:
iso_code,continent,date,total_cases
AFG,Asia,2020-01-05,0
AFG,Asia,2020-01-06,0
AFG,Asia,2020-01-07,0
AFG,Asia,2020-01-08,0
 62 93.8M   62 58.9M    0     0   508k      0  0:03:09  0:01:58  0:01:11  834kStarting ETL process...
Downloading COVID-19 dataset...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
.8M  2 8  9933 .887M. 9 M  2 8   206 . 6 M    0   0    6 4 8 k0     1 4 2 50k    0 : 0 2 :02 8  0 :00:10:20:71 8  0 :00:00:01:91 0  0:20300:44k8 2383kguru@Nitro-ANV15-51:~$ 
```

Then I configured the cron job to run at 12:10 everyday.



```
  GNU nano 7.2                          /tmp/crontab.EW16U9/crontab *
10 0 * * * /home/guru/etl.sh >> /home/guru/logfile.log 2>&1
```

Conclusion:

This ETL pipeline serves as a robust solution for automating the process of downloading, transforming, and loading COVID-19 data into a PostgreSQL database. By following a well-structured approach, the pipeline ensures that the data is cleaned, correctly formatted, and updated at regular intervals. The use of cron jobs and logs makes the system efficient and easily manageable. This process not only helps in data automation but also in ensuring that the database is always up to date for analysis and reporting.

This pipeline can be further extended or modified for other datasets, improving scalability and making the solution adaptable for different use cases.