ETL1 Guide

Table of Contents

[1. Create Postgres Server 3](#_Toc176188318)

[2. Upload Retail Sales Data to Postgres 4](#_Toc176188319)

[3. Build Python Application on Windows VM 4](#_Toc176188320)

[4. Connect Python Application to Postgres 4](#_Toc176188321)

[5. Build a Storage Node w/ NFS 6](#_Toc176188322)

[5. Transform the Data in Python Application 8](#_Toc176188323)

[6. Create Data Warehouse on MySQL 9](#_Toc176188324)

[7. Connect Python to MySQL Database 11](#_Toc176188325)

[8. Connect Grafana to MySQL 12](#_Toc176188326)

# Create Postgres Server

Resource Link: https://www.youtube.com/watch?v=OxIQ\_xJ-yzI

1. Start up Virtual Box and create an Ubuntu VM.

2. Go to Mozilla and download Postgres Ubuntu version from PostgreSQL site.

3. Gain root access by running: su -

4. Run commands from PostgreSQL site.

5. sudo - -i #Sends you into Postgres

6. postgres #Sends you into Postgres DB

7. Exit Postgres and cd into /etc/postgresql/14/main/

8. sudo nano postgresql.conf #This gets you into PG configuration

9. Scroll down to #listen\_addresses line and create a new line with the following: listen\_addresses = '\*'

10. sudo nano pg\_hba.conf

11. Scroll down to # Database administrative login by Unix domain socket and add the follwing line: local all admin md5

12. Scroll to # Allow replication connections... Add the follwoing line underneath it: host all all 0.0.0.0/0 trust

13. sudo service postgresql restart #Restarts PG service

14. sudo -i -u postgres

15. psql

16. create role admin with #Allows to write in admin role

17. Run the following commands to create role configuration:

login

superuser

createdb

createrole

inherit

replication

connection limit -1

password '<password of choice>';

23. Postgres server has been set up and is ready for data!

# 2. Upload Retail Sales Data to Postgres

1. Go to Kaggle and look for Retail Sales Dataset data set: https://www.kaggle.com/datasets/mohammadtalib786/retail-sales-dataset

2. Create a directory called /retail/sales

3. Extract retail\_sales\_dataset.csv in Downloads folder

3. Move the retail\_sales\_dataset.csv.csv file to the /retail/sales file: mv ~/Downloads/\*.csv ~/retail/sales/retail\_sales1.csv

4. Create a databse in Postgres called retail

5. Create a table named sales

6. Import retail\_sales1.csv into sales table: Link: <https://www.warp.dev/terminus/import-csv-to-postgres>

# 3. Build Python Application on Windows VM

1. Download Windows 11 ISO image from this link: https://www.microsoft.com/software-download/windows11

2. Create your Windows 11 VM with the ISO image and wait for the installation to be complete.

3. Install Anaconda Navigator.

# 4. Connect Python Application to Postgres

1. For both VMs, make sure that under Network settings the Bridge Adapter has been selected. This will view them as being on the same network as the host.

2. Get the IPs for both VMs. Windows= ipconfig Ubuntu= hostname -I

3. Make sure firewall connections are open on both VMs:

Windows: Go to Windows Defender Firewall, create New Inbound Rule, specify allowance of port 5432, and name the rule Postgres.

Ubuntu: Run 'sudo ufw status' to check what current FW rules look like. Run ' sudo ufw allow 5432/TCP' and run ' sudo ufw allow <Windows IP> for good measure.

4. Go to your Ubuntu server and nano into your pga\_hba.conf and the following line to # IPv4 local connections and #Replication section: host all all <Windows VM IP>/32 md5

5. Go back to your Windows VM and open up Python and start a script.

6. Run 'pip install psycopg2-binary' to download the psycopg2 packages-pgadmin-org

7. Write the following Python script and run it:

import psycopg2

# Database connection parameters

db\_params = {

'dbname': '<name of db>',

'user': '<user account with access to the db>',

'password': '<password used by account to access database',

'host': '<Ubuntu VM IP>',

'port': '5432'

}

try:

# Connect to the PostgreSQL server

conn = psycopg2.connect(\*\*db\_params)

print("Connection successful")

# Create a cursor object

cursor = conn.cursor()

# Execute a sample query

cursor.execute("SELECT version();")

db\_version = cursor.fetchone()

print(f"PostgreSQL version: {db\_version}")

# Close the cursor and connection

cursor.close()

connection.close()

except Exception as e:

print(f"An error occurred: {e}")

\*Special Note: If you get lost and what the creds are, then swap to your Ubuntu hop into your postgres and run the following command: psql -U <username account> -d <db name>

The creds used here is what you will put in the Python script.

# 5. Build a Storage Node w/ NFS

1. Create an Ubuntu VM. In this example, the storage node is named STR-PROD-100.

2. Switch to Postgres server, PG100, and install NFS package with the following commands:

sudo apt update

sudo apt install nfs-kernel-server

3. Make configurations to the /retail directory with the following commands:

sudo mkdir -p /root/retail

sudo chown nobody:nogroup /root/retail

sudo chmod 755 /root/retail

4. Configure NFS exports by editing the /etc/exports file. Add the following line to /etc/exports:

/root/retail <Ubuntu VM IP>(rw,sync,no\_subtree\_check)

5. Apply the configurations changes by running the following commands:

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

6. Enable the firewall with the following command:

sudo ufw allow from <Ubuntu VM IP> to any port nfs #IP belongs to the storage node

7. Swap to STR-PROD-100

8. Install NFS client package with the following commands:

sudo apt update

sudo apt install nfs-common

9. Create a mount point with this command:

sudo mkdir -p /mnt/retail

10. Mount the /retail directory from PG100 to STR-PROD-100 with this command:

sudo mount <VM IP of Postgres>:/root/retail /mnt/retail

\*Note: This step will have to be manually done whenever the VMs are turned off. To avoid this nano the /etc/fstab and add the following line:

<VM IP of Postgres>:/root/retail/sales /mnt/retail nfs defaults 0 0

11. Verify the mount with this command:

ls /mnt/retail

12. Create a a backup folder called /backups/retail so the data being added to /mnt/retail is immutable.

13. Install inotify-tools by running: sudo apt install inotify-tools

14. Create a folder called /scripts

15. Create a bash script with the following code:

#!/bin/bash

# Define the source directory and backup location

SOURCE\_DIR="/mnt/retail"

BACKUP\_DIR="/path/to/backup/location"

# Run the rsync command

rsync -av --delete "$SOURCE\_DIR/" "$BACKUP\_DIR/"

# Set up inotifywait to monitor the source directory for changes

inotifywait -m -r -e modify,create,delete "$SOURCE\_DIR" | while read -r; do

rsync -av --delete "$SOURCE\_DIR/" "$BACKUP\_DIR/"

done

16. Make the script executable by running: chmod +x auto\_backup.sh

17. Create a cron job to run the script periodically every hour by running the following:

crontab -e

0 \* \* \* \* /scripts/auto\_backup.sh

# 5. Transform the Data in Python Application

1. Log into Python application VM.

2. Import the sales data from Postgres from the Python application using the following code:

import psycopg2

import pandas as pd

# Database connection parameters

db\_params = {

'dbname': 'retail',

'user': '<username of choice',

'password': 'password of choice',

'host': 'VM IP of Postgres', # Replace with your database host

'port': '5432' # Replace with your database port if different

}

# Connect to PostgreSQL

conn = psycopg2.connect(\*\*db\_params)

# Query to fetch data

query = "SELECT \* FROM sales;"

# Use pandas to read the query result into a DataFrame

sales = pd.read\_sql\_query(query, conn)

# Close the connection

conn.close()

# Print the DataFrame

print(sales)

3. Create a dataframe for Clothing products. Use the following code:

clothing = sales[sales['product\_category'] == 'Clothing']

print(clothing)

4. Create a dataframe for Electronics products. Use the following code:

electronics = sales[sales['product\_category'] == 'Electronics']

print(electronics)

# 6. Create Data Warehouse on MySQL

1. Create a Linux VM. \*Note: In this project I built a RHEL9 VM

2. Update system running:

sudo dnf update

3. Create a SQL repo file under /etc/yum.repos.d/mysql.repo with the following:

[mysql]

name=MySQL Community Server

baseurl=https://repo.mysql.com/yum/mysql-8.0-community/el/9/x86\_64/

enabled=1

gpgcheck=1

gpgkey=https://repo.mysql.com/RPM-GPG-KEY-mysql

EOF

4. Install MySQL Server running:

sudo dnf install -y mysql-server

5. Start MySQL by running:

sudo systemctl start mysqld

sudo systemctl enable mysqld

6. Set up secure login on MySQL:

sudo mysql\_secure\_installation

7. Verify service by running:

sudo systemctl status mysqld

8. Go to MySQL Workbench site and download the package

9. Install the MySQL Workbench rpm that was downloaded by running:

sudo dnf localinstall mysql-workbench-community-8.0.38-1.el9.x86\_64.rpm

10. Start Workbench going to the application

11. Create sales database by running:

create database sales;

12. Create clothing & electronics tables by running:

create table sales.clothing(

transaction\_id int auto\_increment primary key,

date timestamp,

customer\_id int,

gender char,

age int,

product\_category char,

quantity numeric,

unit\_price numeric,

total\_amount numeric);

create table sales.electronics(

transaction\_id int auto\_increment primary key,

date timestamp,

customer\_id int,

gender char,

age int,

product\_category char,

quantity numeric,

unit\_price numeric,

total\_amount numeric);

# 7. Connect Python to MySQL Database

1. Ensure connectivity between Python application (Windows VM) and the MySQL server (Linux VM) by performing the following:

On Linux VM:

sudo firewall-cmd --permanent -add-port=3306/tcp

sudo firewall-cmd --reload

sudo firewall-cmd --list-ports

Log into MySQL:

mysql -u root -p

On MySQL:

CREATE USER 'python'@'<Windows VM IP>' IDENTIFIED BY '<password of choice>';

GRANT ALL PRIVILEGES ON sales.\* TO 'python'@'<Windows VM IP>';

FLUSH PRIVILEGES;

2. Assemble Python script to load data into sales database on MySQL:

#Installations

#pip install pandas mysql-connector-python sqlalchemy

import pandas as pd

import mysql.connector

from sqlalchemy import create\_engine

#Connect to MySQL server

connection\_string = 'mysql+mysqlconnector://python:Larry!#@<VM IP of MySQL>:3306/sales'

engine = create\_engine(connection\_string)

#Send dataframe to MySQL database

from sales\_extraction import clothing

table\_name = 'clothing'

clothing.to\_sql(name=table\_name, con=engine, if\_exists='replace', index=False)

from sales\_extraction import electronics

table\_name = 'electronics'

electronics.to\_sql(name=table\_name, con=engine, if\_exists='replace', index=False)

print("Data transferred successfully!")

# 8. Connect Grafana to MySQL

1. Install Grafana Enterprise (On RHEL VM) by running the following:

sudo yum install -y https://dl.grafana.com/enterprise/release/grafana-enterprise-11.2.0-1.x86\_64.rpm

2. Start up Grafana server by running:

sudo systemctl start grafana-server

3. Open firewall connection to allow Grafana to connect to RHEL VM by running:

sudo firewall-cmd -add-port=3000/tcp --permanent

sudo firewall-cmd --reload

4.Create a user in MySQL and assign it privileges to access the sales database by running:

CREATE USER '<username of choice>'@'<VM IP of MySQL>' IDENTIFIED BY '<password of choice>';

GRANT ALL PRIVILEGES ON sales.\* TO '<username of choice>'@'<VM IP of MySQL>';

FLUSH PRIVILEGES;

5. Go to your web browser and enter the following to get to Grafana:

http:<VM IP of MySQL>:3000

6. Once you're on Grafana, click Add your first data source.

7. Fill out the following configuration to connect MySQL to Grafana:

Host URL: <VM IP of MySQL>:3306

Database name: sales

Username: <username created in MySQL>

Password: <password created in MySQL>

8. Click Save & Test

9. The connection should be established and you're now free to run visuals on the data.

Congrats! You completed the ETL!