Lab - ETL

Estimated time needed: **30** minutes.

Scenario

You are a data engineer at an e-commerce company. You need to keep data synchronized between different databases/data warehouses as a part of your daily routine. One task that is routinely performed is the sync up of staging data warehouse and production data warehouse. Automating this sync up will save you a lot of time and standardize your process. You will be given a set of python scripts to start with. You will use/modify them to perform the incremental data load from MySQL server which acts as a staging warehouse to the IBM DB2 or PostgreSQL which is a production data warehouse. This script will be scheduled by the data engineers to sync up the data between the staging and production data warehouse.

Objectives

In this assignment you will write a python program that will:

- Connect to IBM DB2 or PostgreSQL data warehouse and identify the last row on it.
- Connect to MySQL staging data warehouse and find all rows later than the last row on the datawarehouse.
- Insert the new data in the MySQL staging data warehouse into the IBM DB2 or PostgreSQL production data warehouse.

About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, 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 MySQL database running in a Docker container. You will also need an instance of DB2 running in IBM Cloud or PostgreSQL database running in a Docker container.

Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session 1will get lost. To avoid losing your data, please plan to complete these labs in a single session.

Software Required

- MySQL Server
- IBM DB2 or PostgreSQL

Note - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. You will need these screenshots to either answer graded quiz questions or to upload as your submission for peer review at the end of this course. You can use various free screengrabbing tools to do this or use your operating system's shortcut keys to do this (for example Alt+PrintScreen in Windows).

Prepare the lab environment

Before you start the assignment:

Step 1: Start MySQL server

Step 2: Create a database named `sales`

Step 3: Download the file below

sales.sql

Step 4: Import the data in the file `sales.sql` into the `sales` database.

Step 5: Download the mysqlconnect.py python programs from link below.

mysqlconnect.py

Step 6: `mysqlconnect.py` has the sample code to help you understand how to connect to MySQL using Python.

Step 7: Modify `mysqlconnect.py` suitably and make sure you are able to connect to the MySQL server instance on the Theia environment.

> Note: Before executing `mysqlconnect.py` note that you install the connector using the command
> `python3.11 -m pip install mysql-connector-python;`

In order to complete the tasks below, you have the option to complete them on either a DB2 database (Option A) or on PostgreSQL (Option B).

If you choose **Option A**, please follow tasks 8-11. If you choose **Option B**, please follow tasks 12-16.

Option A: If you choose DB2 as the data warehouse:

> Note: Verify that you can access your cloud instance of IBM DB2 server.

Step 8: Download the db2connect.py python program from the link below.

db2connect.py

db2connect.py has the sample code to help you understand how to connect to the cloud instance of IBM DB2 using Python.

> Note: Before executing `db2connect.py` note that you install the connector using the command
>
> ```
> pip install --force-reinstall ibm_db==3.1.0 ibm_db_sa==0.3.3
> ```

Step 9: Modify `db2connect.py` suitably and make sure you are able to connect to your cloud instance of IBM DB2 from the Theia environment.

Step 10: Download the file below

sales.csv

Step 11: Load sales.csv into a table named `sales_data` on your cloud instance of IBM DB2 database.

> Note: By default, the sales.csv file contains price and timestamp columns, which are not present in sales.sql. Therefore, after loading the CSV into the sales_data table, you should run the script below. This script ensures that for any new values, the price column will be updated to 0 by default, and the timestamp will be updated with the current timestamp.

```
ALTER TABLE sales_data ALTER COLUMN timestamp SET data type timestamp;
ALTER TABLE sales_data ALTER COLUMN timestamp SET NOT NULL;
ALTER TABLE sales_data ALTER COLUMN timestamp SET DEFAULT CURRENT_TIMESTAMP;
ALTER TABLE sales_data ALTER COLUMN price SET data type decimal;
ALTER TABLE sales_data ALTER COLUMN price SET DEFAULT 0;
```

At any stage, if you encounter an error like -

SQL0668N>SQL0668N Operation not allowed for reason code <reason-code> on table <table-name>

```
call sysproc.admin_cmd('reorg table sales_data');
```

OR

Option B: If you choose PostgreSQL as the data warehouse:

Step 12: Download the postgresqlconnect.py python program from the link below.

postgresqlconnect.py

`postgresqlconnect.py` has the sample code to help you understand how to connect to the PostgreSql data warehouse using Python.

> Note: Before executing `postgresqlconnect.py` note that you install the connector using the command

```
python3 -m pip install psycopg2
```

Step 13: Modify postgresqlconnect.py suitably and make sure you are able to connect to PostgreSql from the Theia environment.

Step 14: Download the file below

sales.csv

> Note: By default, the sales.csv file contains price and timestamp columns, which are not present in sales.sql. Therefore, you can use the below lines of code in your script to include price and timestamp columns when creating the table in Postgres.
>
> ```
> price decimal DEFAULT 0.0 NOT NULL,
> timeestamp timestamp without time zone DEFAULT CURRENT_TIMESTAMP NOT NULL
> ```

Step 15: Create a table called `sales_data` using the columns `rowid`, `product_id`, `customer_id`, `price`, `quantity`
`timeestamp`. Load sales.csv into the table `sales_data` on your PostgreSql database.

> Note: Ensure that you upload the file to this path: /var/lib/pgadmin/

Step 16: Download the `automation.py` from the following URL : automation.py

You will be using `automation.py` as a scafolding program to execute the tasks in this assignment

Exercise 1 - Automate loading of incremental data into the data warehouse

One of the routine tasks that is carried out around a data warehouse is the extraction of daily new data from the operational database and loading it into the data warehouse. In this exercise you will automate the extraction of incremental data, and loading it into the data warehouse.

In order to complete Tasks 1 and 3 below, you have an option to complete the tasks on a DB2 database (Option A), or on PostgreSQL (Option B).

Task 1 - Implement the function get_last_rowid()

In the program automation.py implement the function get_last_rowid()

Option A: If you choose DB2 as the data warehouse:

This function must connect to the DB2 data warehouse and return the last rowid.

Option B: If you choose PostgreSQL as the data warehouse:

This function must connect to the PostgreSql as the data warehouse and return the last rowid.

Take a screenshot of the python code clearly showing the implementation of the function `get_last_rowid()`. Also save the code block separately as a text for later use.

Name the screenshot `get_last_rowid.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 2 - Implement the function `get_latest_records()`

In the program `automation.py` implement the function `get_latest_records()`.

This function must connect to the MySQL database and return all records later than the given `last_rowid`.

Take a screenshot of the python code clearly showing the implementation of the function `get_latest_records()`. Also save the code block separately as a text for later use.

Name the screenshot `get_latest_records.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 3 - Implement the function insert_records()

In the program automation.py implement the function insert_records()

Option A: If you choose DB2 as the data warehouse:

This function must connect to the DB2 data warehouse and insert all the given records.

Option B: If you choose PostgreSQL as the data warehouse:

This function must connect to the PostgreSQL data warehouse and insert all the given records.

Take a screenshot of the python code clearly showing the implementation of the function `insert_records()`. Also save the code block separately as a text for later use.

Name the screenshot `insert_records.jpg`. (Images can be saved with either the .jpg or .png extension.)

Task 4 - Test the data synchronization

Run the program `automation.py` and test if the synchronization is happening as expected.

Take a screenshot of the program output .

Name the screenshot `synchronization.jpg`. (Images can be saved with either the .jpg or .png extension.)

Conclusion

You have succesfully completed all the relevant tasks of ETL and pipelining set up as required.

Author

Ramesh Sannareddy

Other Contributors

Rav Ahuja
Abhishek Gagneja