Департамент образования города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

«Проектный практикум по разработке ETL-решений »

Лабораторная работа №2.1

Тема: «Динамические соединения с базами данных»

Выполнила:

Студентка группы АДЭУ-211

Кравцова Алёна Евгеньевна

Руководитель:

Босенко Т.М

Москва

2024

Цель работы: получить практические навыки создания ETL-процесса для интеграции данных из различных источников с использованием динамических соединений в Pentaho Data Integration, включая обработку повторяющихся данных.

Задачи:

– Создать динамические подключения к различным источникам данных;

– Разработать процесс выявления и обработки дублирующихся записей;

– Реализовать механизм объединения данных в единое хранилище;

– Настроить обработку ошибок при выполнении трансформации.

Программное обеспечение:

– Pentaho Data Integration 9.4;

– MySQL или PostgreSQL;

– CSV или Excel файлы с тестовыми данными.

Вариант 6: Фильтр по прибыли: только Profit > 0; Отчет по категориям; Анализ доставки.

Шаги выполнения:

1) Создание новой базы данных MySQL.

После перехода в php Admin необходимо выполнить скрипт, представленный ниже, для дальнейшей загрузки данных в эти таблицы.

```
-- Таблица заказов (основная информация о продажах)
CREATE TABLE orders (
row_id INT PRIMARY KEY,
order_date DATE,
ship_date DATE,
ship_mode VARCHAR(50),
sales DECIMAL(10,2),
quantity INT,
discount DECIMAL(4,2),
```

```
profit DECIMAL(10,2),
returned TINYINT(1) DEFAULT 0  -- 1 = Yes, 0 = No
);
 -- Таблица клиентов
DROP TABLE IF EXISTS customers;
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id VARCHAR(20) NOT NULL,
    customer_name VARCHAR(100),
    segment VARCHAR(50),
    country VARCHAR(100),
    city VARCHAR(100),
    state VARCHAR(100),
    postal_code VARCHAR(20),
    region VARCHAR(50),
    INDEX idx_customer_id (customer_id),
    INDEX idx_region (region)
);
 -- создаем таблицу products
DROP TABLE IF EXISTS products;
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_id VARCHAR(20) NOT NULL,
    category VARCHAR(50),
    sub_category VARCHAR(50),
    product_name VARCHAR(255),
    person VARCHAR(100),
    INDEX idx_product_id (product_id),
    INDEX idx_category (category),
    INDEX idx_subcategory (sub_category)
);
 -- Создаем индексы для оптимизации запросов
ALTER TABLE orders ADD INDEX idx_order_date (order_date);
ALTER TABLE orders ADD INDEX idx_ship_date (ship_date);
ALTER TABLE customers ADD INDEX idx_region (region);
```

ALTER TABLE products ADD INDEX idx_category (category); -- Установим правильную кодировку

ALTER DATABASE mgpu_ico_etl_prepod CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

ALTER TABLE orders CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

ALTER TABLE customers CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

ALTER TABLE products CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

В результате будут созданы необходимые таблицы (Рис. 1).

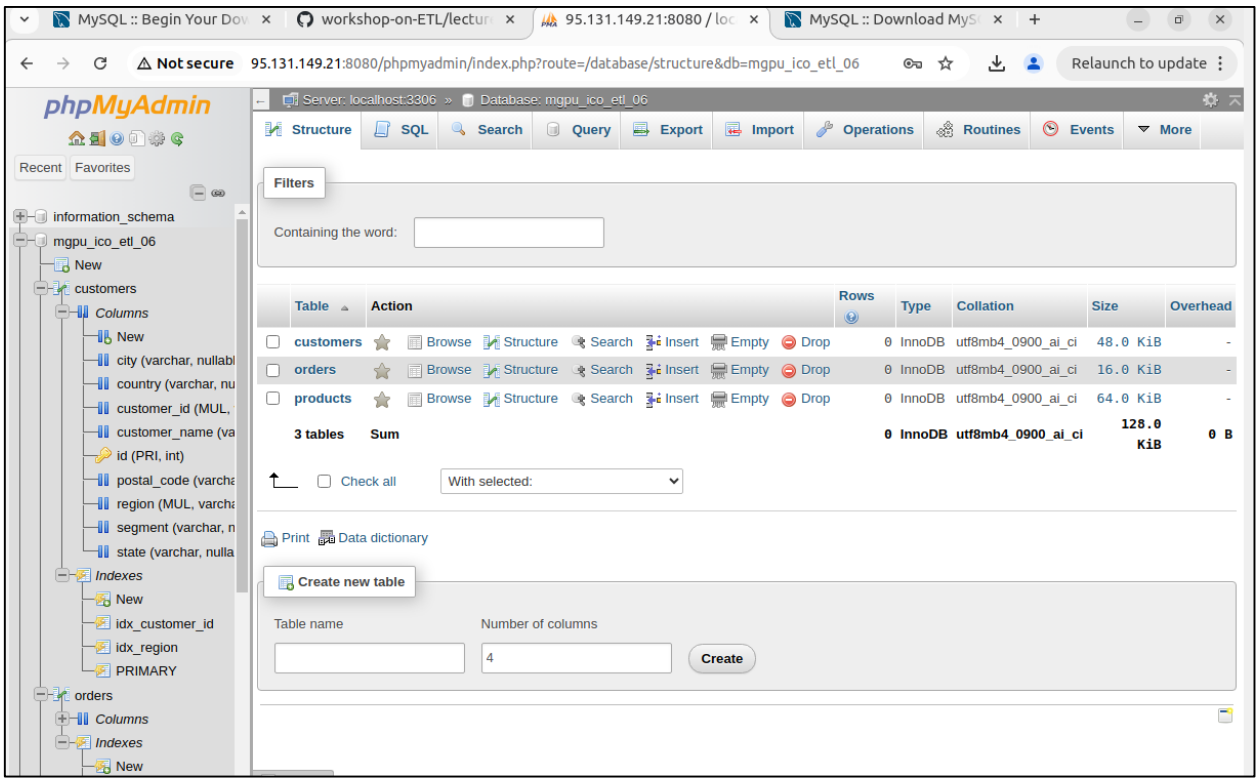

Рис. 1 – Результат созданных таблиц в БД

2) Необходимо загрузить из репозитория нужные трансформации и job, а также csv файл с тестовыми данными. Загрузим эти файлы в созданный каталог datain. Для того, чтобы в pentaho появился доступ к папке необходимо дать разрешение (Рис. 2).

```
dba@dba-vm:~/Downloads/data-integration$ mkdir -p datain
dba@dba-vm:~/Downloads/data-integration$ ls
 Carte.bat                              plugins
 carte.sh                               purge-utility.bat
 classes                                purge-utility.sh
 datain                                 pwd
'Data Integration.app'                  README.txt
'Data Service JDBC Driver'              runSamples.bat
 docs                                   runSamples.sh
 Encr.bat                               samples
 encr.sh                                set-pentaho-env.bat
 Import.bat                             set-pentaho-env.sh
 import-rules.xml                       simple-jndi
 import.sh                              Spoon.bat
 Kitchen.bat                            spoon.command
 kitchen.sh                             SpoonConsole.bat
 launcher                               SpoonDebug.bat
 lib                                    SpoonDebug.sh
 libswt                                 spoon.ico
 LICENSE.txt                            spoon.png
 logs                                   spoon.sh
 Pan.bat                                static
 pan.sh                                 ui
 PentahoDataIntegration_OSS_Licenses.html   yarn.sh
dba@dba-vm:~/Downloads/data-integration$ chmod 755 datain
dba@dba-vm:~/Downloads/data-integration$
```

Рис. 2 – Разрешение на папку datain

3) Выполним трансформацию 1, которая загружает данные о заказах (orders). Также необходимо дополнительно пройтись по узлам, в которых определен путь к загруженному датасету и обновить путь.

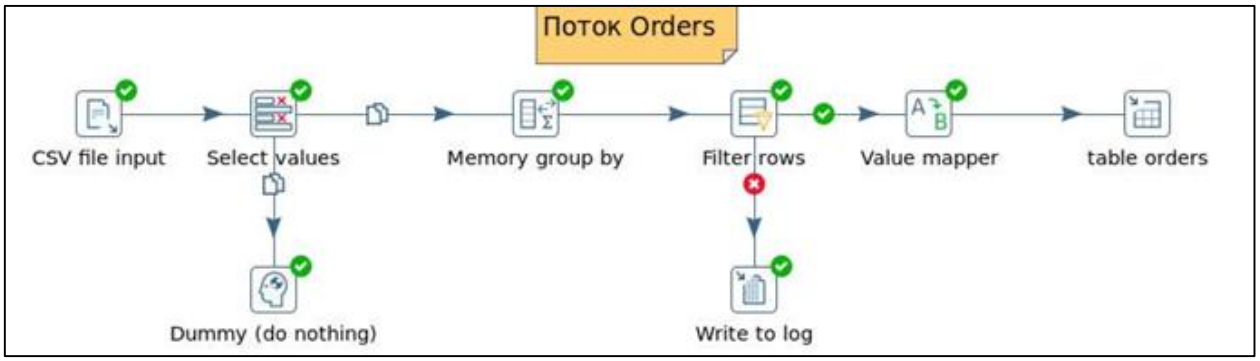

Рис. 3 – Трансформация orders

В узле table orders происходит запись данных в БД, необходимо актуализировать подключение к собственной базе (Рис. 4).

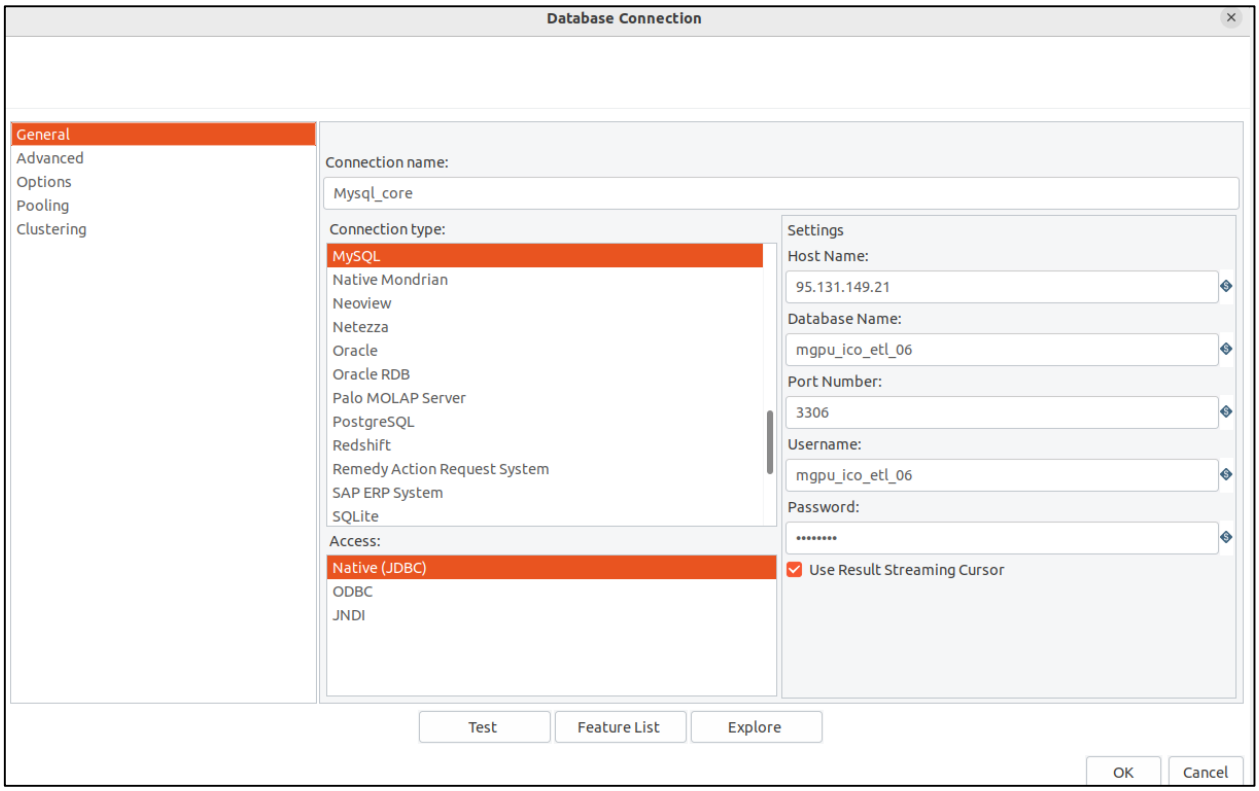

Рис. 4 – Подключение к БД

4) Выполним трансформацию 2, которая загружает данные о заказах (customers). Также выполним шаги, описанные выше. Результат представлен на рисунках 5 и 6.

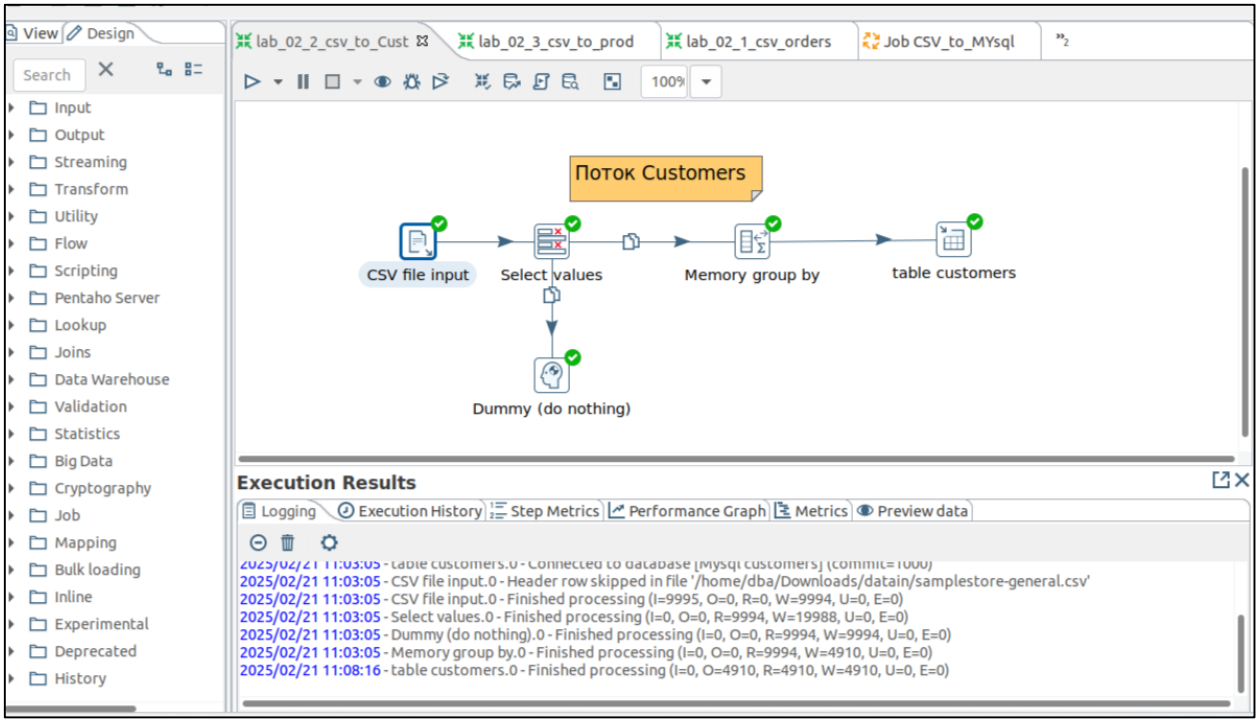

Рис. 5 – Трансформация customers

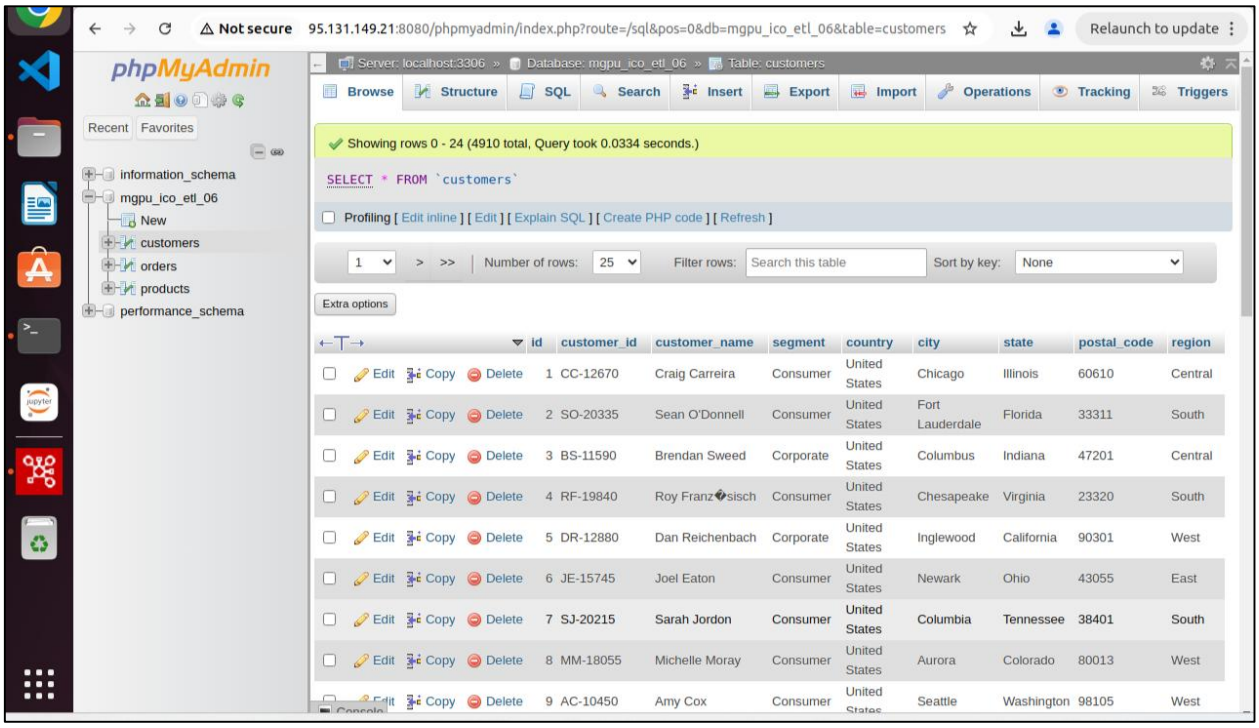

Рис. 6 – Результат загрузки данных в БД

5) Выполним трансформацию 3, которая загружает данные о товарах (products). Также выполним шаги, описанные выше. Результат представлен на рисунках 7 и 8.

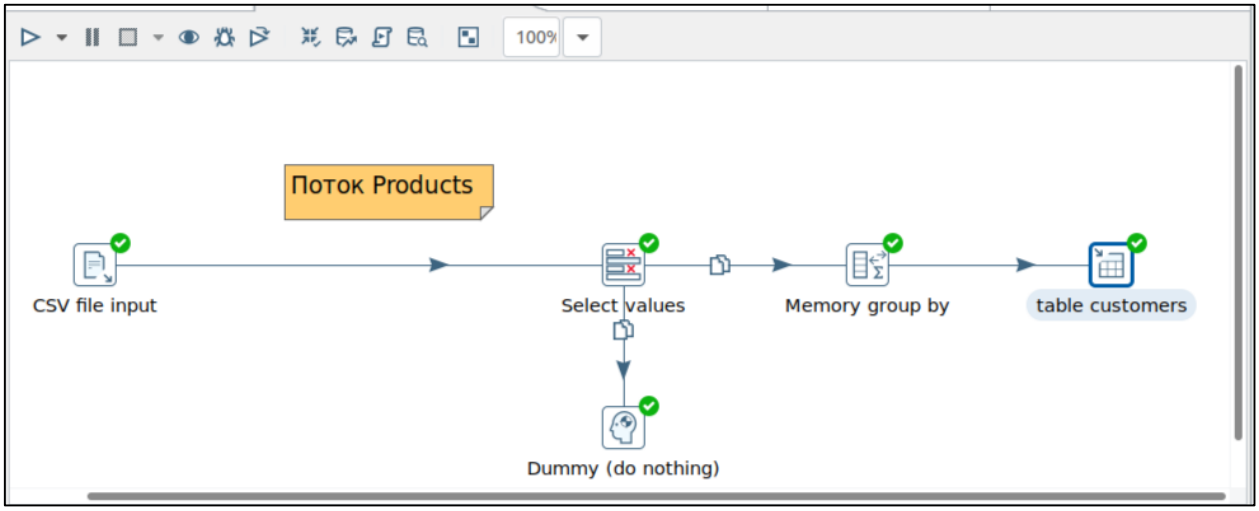

Рис. 7 – Трансформация products

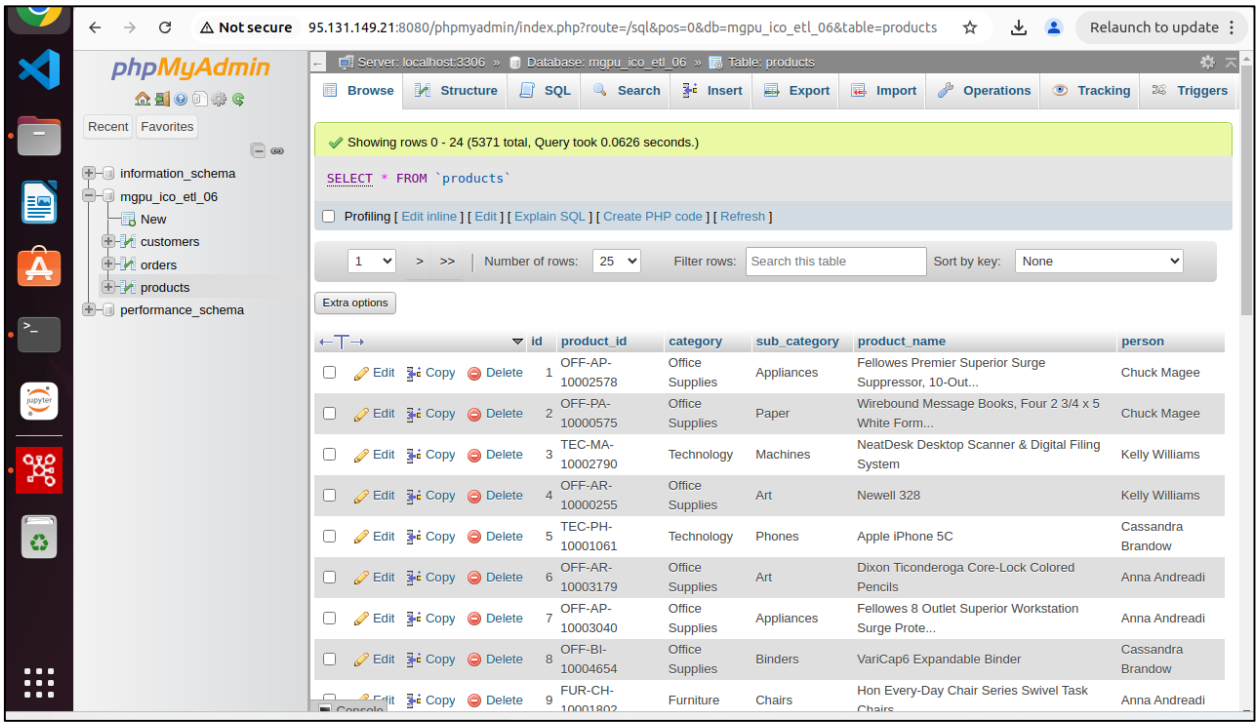

Рис. 8 – Результат загрузки данных в БД

6) Далее выполним job, также предварительно обновив все пути к файлам (Рис. 9).

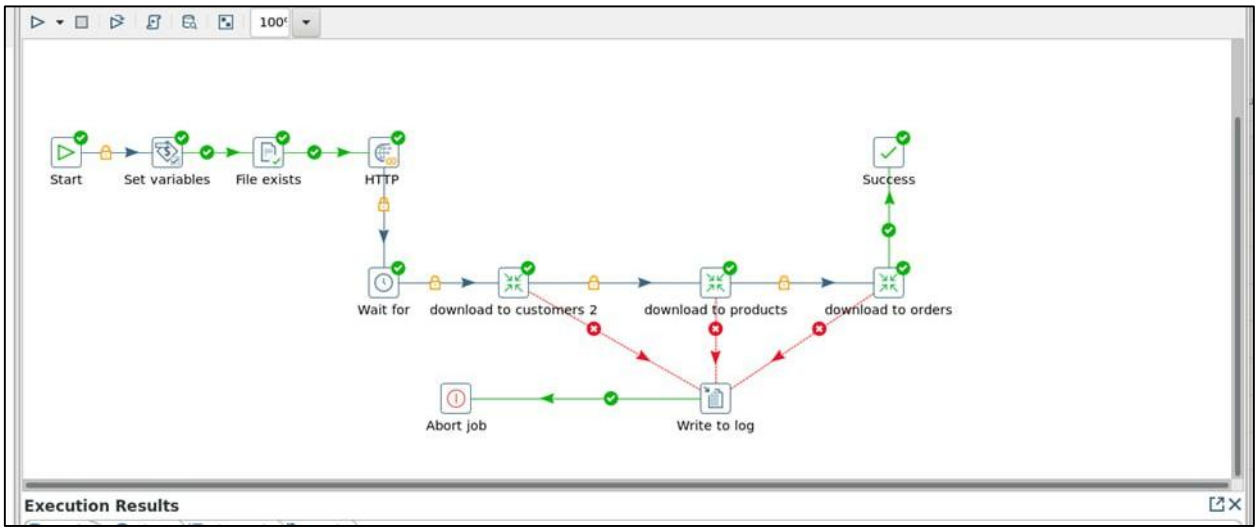

Рис. 9 – Выполнение job

7) Индивидуальное задание.

Сделаем копию трансформации 1, далее необходимо добавить узел Filter. Настроим фильтр по прибыли таким образом, чтобы удовлетворяющие значения (больше 0) записывались в БД, а отрицательные данные фиксировались в логах (Рис. 10).

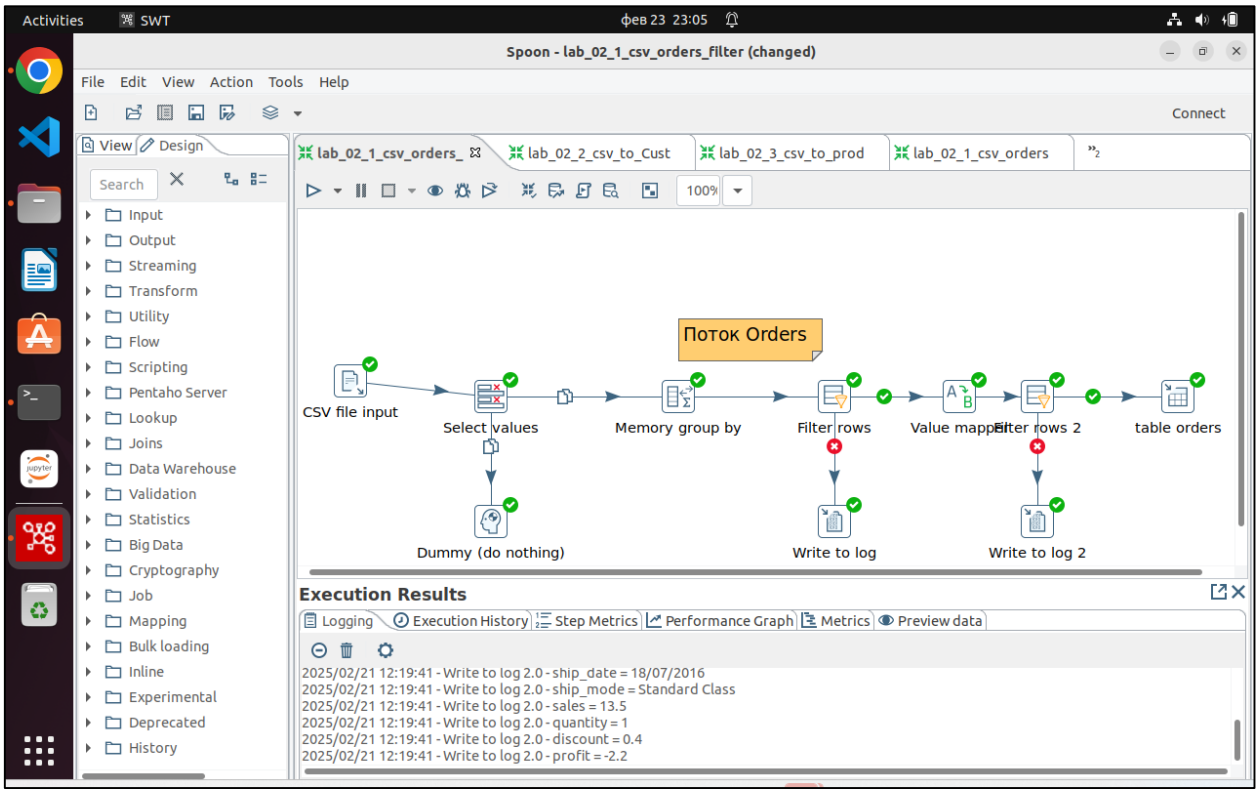

Рис. 10 – Трансформация с фильтром по значению profit

Также установим в фильтре необходимые значения (Рис. 11).

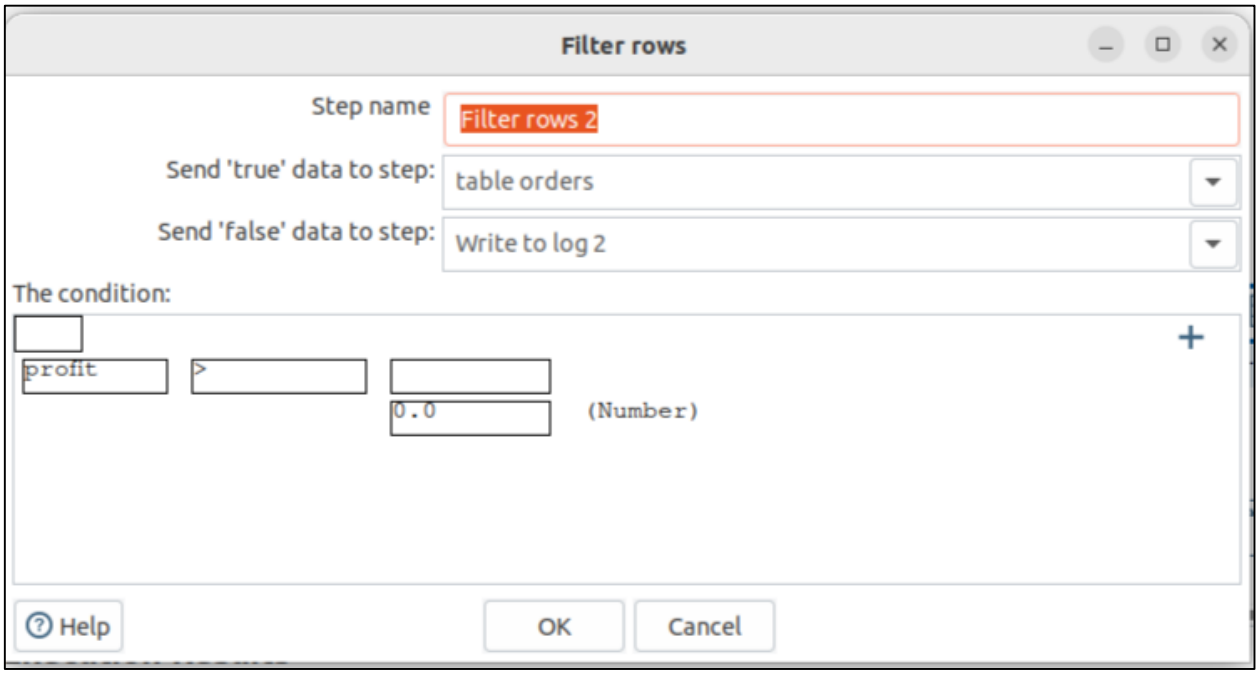

Рис. 11 – Настройка фильтра по столбцу profit

После успешного выполнения в базе появились значения, где profit >0 (Рис. 12).

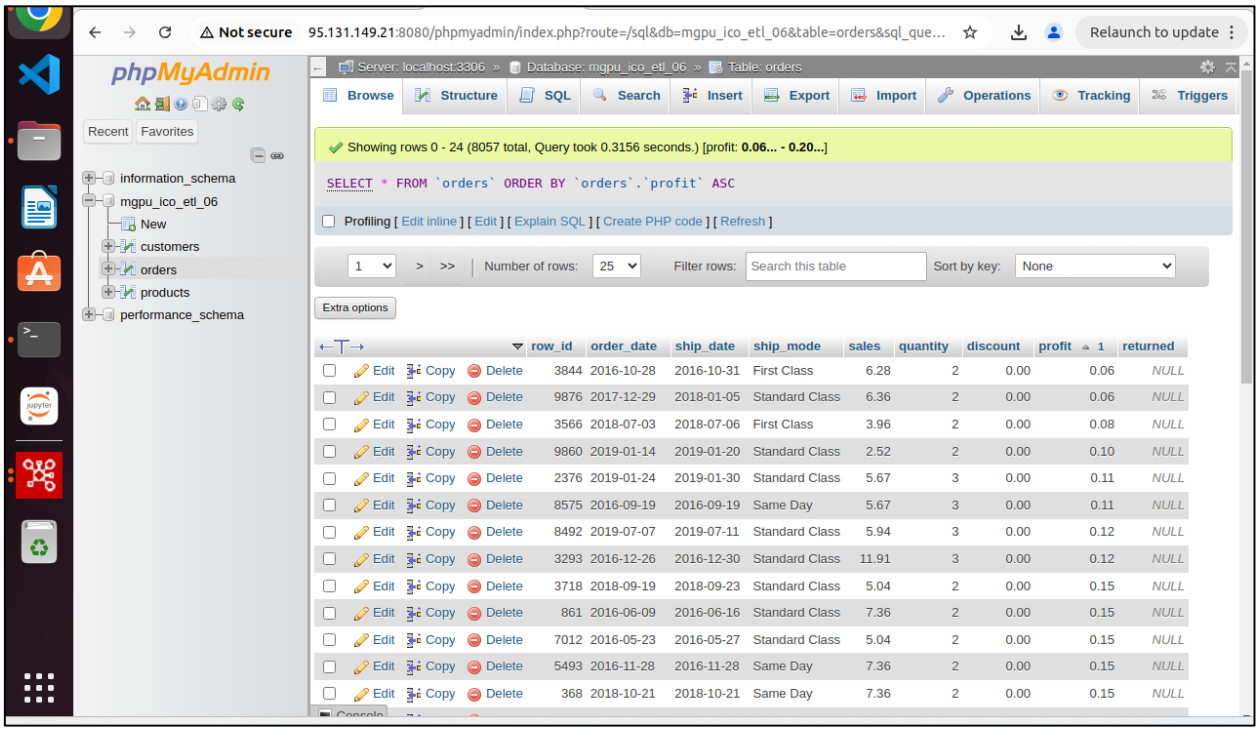

Рис. 12 – Результат загрузки данных в БД