Database Schema Creation and Local Usage Guide

Price Collection Project

December 30, 2025

1 Introduction

This document describes the creation and local usage of the MySQL database schema used in the Price Collection project. The schema is shared among all team members to ensure consistency when running ETL pipelines, visualization scripts, and notification services.

The database is designed to store daily snapshots of product prices collected from multiple online platforms.

2 Database Schema Overview

The project uses a single MySQL database named `price_collection`, which contains one table called `PRICE`. Each row in the table represents a price offered by a seller for a specific product on a given date.

The table consists of the following fields:

- `ID`: Auto-incremented primary key
- `Product`: Name of the product
- `Date`: Date on which the price was collected
- `Seller`: Seller or platform offering the product
- `Price`: Price including delivery costs

To prevent duplicate entries, a unique constraint is enforced on the combination of `Product`, `Date`, and `Seller`.

3 Schema Creation

The database schema is shared with the team as a SQL script file named `prices_db.sql`. This file contains all SQL commands required to create the database and the `PRICE` table.

The schema definition is shown below:

```
CREATE DATABASE IF NOT EXISTS price_collection;
USE price_collection;

CREATE TABLE IF NOT EXISTS PRICE (
    ID BIGINT AUTO_INCREMENT PRIMARY KEY,
    Product VARCHAR(255) NOT NULL,
    Date DATE NOT NULL,
    Seller VARCHAR(255) NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    UNIQUE (Product, Date, Seller)
);
```

4 Using the Schema Locally

Each team member should create the database locally before running any ETL or analysis scripts.

4.1 Steps to Initialize the Database

1. Open MySQL Workbench or a MySQL command-line interface.
2. Execute the contents of the file `prices_db.sql` by either:
    - Running `SOURCE prices_db.sql;` in a SQL terminal, or
    - Copying and pasting the file contents into a SQL editor and executing it.

After execution, the following objects should exist locally:

- Database: `price_collection`
- Table: `PRICE`

4.2 Schema Verification

To verify that the schema was created correctly, the following SQL commands can be executed:

```
USE price_collection;
SHOW TABLES;
DESCRIBE PRICE;
```

Successful execution confirms that the database and table are available for use.

5 Sanity Testing

After creating the database, a sanity test was performed to validate correct functionality. This involved inserting a test row into the `PRICE` table, retrieving it using a `SELECT` query, and verifying that the unique constraint prevents duplicate entries for the same product, seller, and date.

Once the test was completed successfully, the test data was removed to keep the database clean for the actual ETL process.

6 Usage in Project Components

All Python components of the project—including Idealo ETL, eBay ETL, Amazon ETL, visualization, and email notification scripts—interact exclusively with the `PRICE` table.

Team members are required to:

- Use the shared schema without modifying table or column names
- Insert prices together with the corresponding collection date
- Rely on the unique constraint to avoid duplicate records

Using a shared schema ensures consistent development and simplifies integration and future deployment to cloud environments such as AWS.