SCD2 Project

a project based on slow changing data table with Python & SQL.

By: Anudyoti Deb

email: anudyotideb@gmail.com

## Overview

Database has a SCD table which contains information about mobile brands with their current price. Every time a new brand comes in it gets added to the SCD table. Also, if there is a change in price on the existing models, those too gets added to the table without replacing the older price. The table keeps the record of the older data with date range.

## Tools

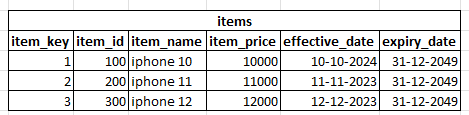
Oracle Database SQL

Python with Pandas

## Problem Statement

Assume the database has a table name “items’ with the mobile model information along with price and dates as shown below.

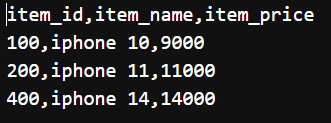
**Example Input**



Our goal is to create a ETL system which will keep appending new data into this table keeping the older record in place.

**Input Format**

Input can be in any file format or any kind of source. However, for this project we are using a simple text file in CSV format. The input data comes in format shown below.



There are three columns: 1st will be the “item\_id”,2nd will be the “item\_name” & 3rd will be the “item\_price” value.

**The input data may contain any of these:**

1. A new mobile model that doesn’t exist in the database table with price.
2. Existing mobile model with new price.
3. Existing mobile model with the same price. (can be possible)

for scenario 1, append the data into the database table as a new row with today’s date and a default expiry date.

for scenario 2, append the data into the database table as a new row with today’s date and a default expiry date. Do not overwrite the older model. However, once this is added change the expiry date of the older model to previous day.

for scenario 3, do not add any row. This should be ignored.

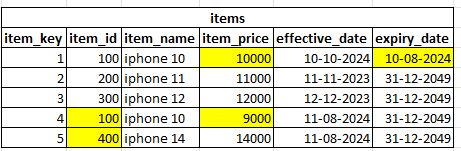
In our case, we will include all three scenarios as we can see from the images.

item\_id 100 comes with a new price. (scenario 2)

item\_200 already exist with the same price. (scenario 3)

item\_id 400 is a completely new model. (scenario 1)

**Expected Output**



We can see two items has been added to the database table with item\_key 4 & 5.

Item\_key 1 expiry date has been changed.

Every time a new set of record comes in; it should follow the rules and update the database table correctly.

## Initial Setup

Let’s create a database table with all the dummy info. Below is the source code link for all the SQL queries needed.

<https://github.com/anudyoti-deb/DE-Projects>

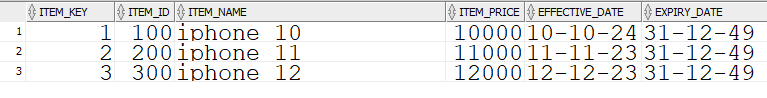
**SQL query explanation**

A sequence & trigger object has been created so that every time a row is inserted , it will add a primary key in the item\_key column.

Seps for query execution sequence for table creation with values provided below or run the script.

1. CREATE TABLE
2. CREATE SEQUENCE
3. CREATE TRIGGER
4. INSERT VALUES
5. COMMIT

Once the table is created, we can see the table in our database.



## Solution

Please find the python source code link below:

<https://github.com/anudyoti-deb/DE-Projects>

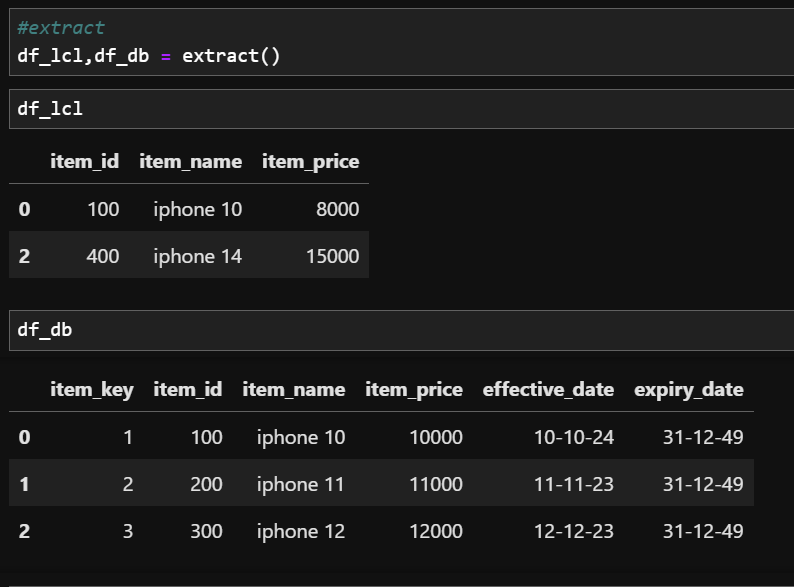
**Code Sequence**

1. Imported all the required libraries & modules.
2. Connected to the database engine (Oracle)
3. Created four functions for ETL.
4. Call the functions.

**Explanation**

extract () function: it pulls the data from both the tables (external source [text file in this case] & database table) into the pandas ecosystem and keeps it as a dataframe in variables df\_lcl & df\_db.

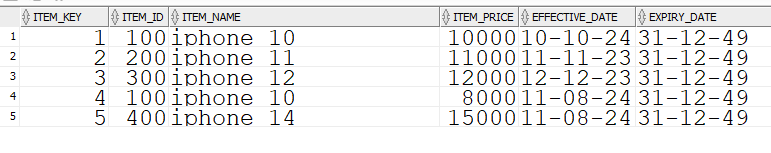
Please note that we are extracting the data from database based on certain condition on expiry date. This filter will prevent any changes in the models with the older prices prior to the current one. This will only give the rows for which the expiry date needs to be updated. That is model price that are currently active. Also, to solve the scenario 3 problem, we are removing the duplicates (both values) from the external file.



*Image showing both the dataframes after extraction. Note that item\_id 200 has been removed from external source as the item was duplicate.*

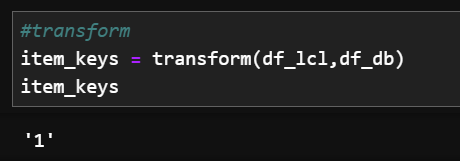
insert() function : it takes the dataframe df\_lcl (external source data) returned by the extract() function and adds two new columns with name effective\_date & expiry\_date. It adds values into those columns for each row. Adds today’s date in the effective\_date column & a fixed end data '31-12-49'. Once the data are added, it will append the entire dataframe into the database table “items”.

Please note that we are not adding any value for item\_key column as the data will auto generate based on sequence every time a row is added.



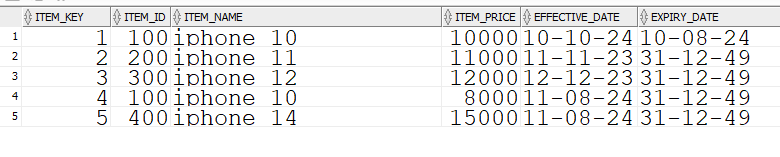
*Final output of insert function. We can see index 4 i.e. item\_id 100 has been added with the updated price of 8000 without removing the older model at index 1 with price 10000. Also, at index 5 , the new model has been added. Both the inserted rows has today’s effective date & the default expiry date. Also, item\_id 200 from the external source has not been added in this final table as the price was unchanged.*

transform () function: it will merge both the dataframes df\_lcl & df\_db coming from the previous function. We will do an inner join so that we get only the rows that are common in both the tables. This will help in getting the item\_keys for which we need to update the expiry dates. This will return a list of item\_keys in string format.



*this will give the item\_key 1 where the date need to be changed.*

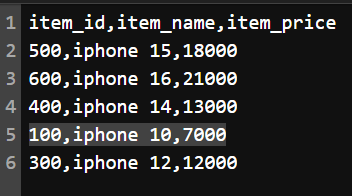
Updates() function : it will update the expiry date to yesterday’s date for the previous active items based on the item\_keys.



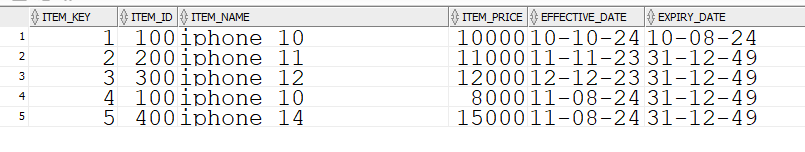
*note for item\_key 1, expiry date is updated.*

We will call the functions in sequence and our database table will get updated with the latest price keeping the older data intact.

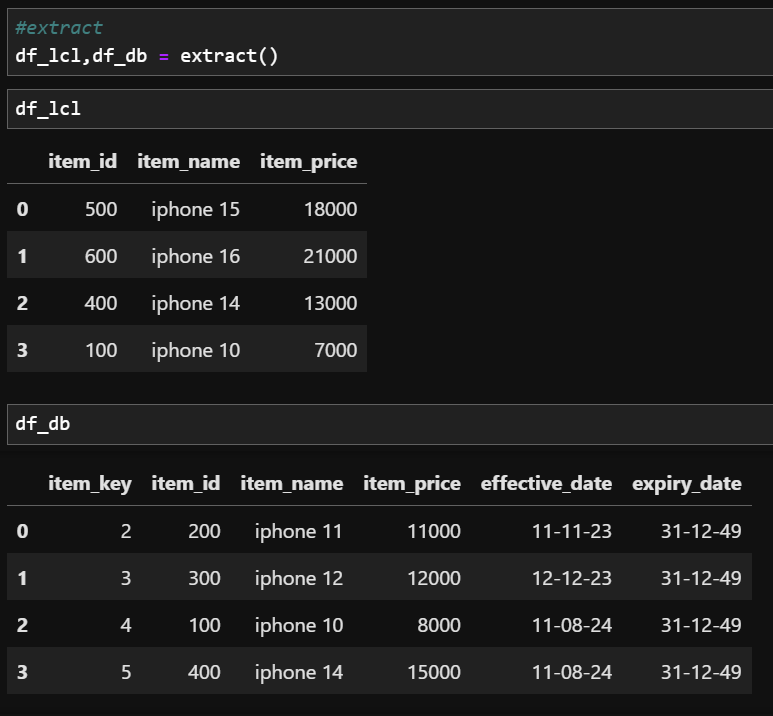
## Test Case: New dataset test



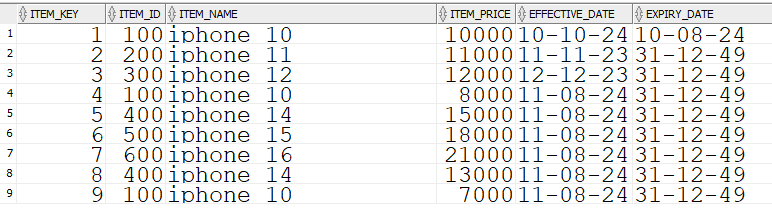
*New incoming data.*



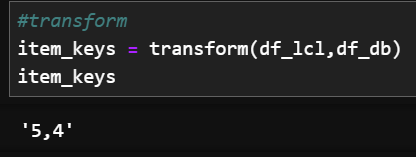
*Existing data from database table before execution.*



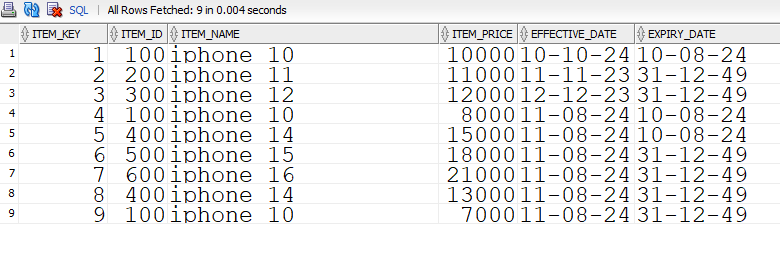
*both the datasets after extraction.*



*database table after insert*



*transform function giving item\_keys*



*final update on the database table*

--- End of Document ---