COMP40370 Practical 2

DATA WAREHOUSES AND ASSOCIATION RULES (Part A)

Prof. Tahar KECHADI

Academic year 2023-2024

Assignment Files

`./Practical-02-A.pdf`	Assignment questions (this file).
`./DW_dataset.csv`	Data file for Q1.
`./input_DW_data.csv`	Data file for Q2.

Expected output files

`./Prcatical-02.ipynb`	Python notebook solutions.
`./Prcatical-02.html`	Python notebook in HTML format.

Requirements: Python 3.9+, pandas 1.3+, SQLAlchemy 1.4+, mlxtend 0.20

Part A: Datawarehouse

This practical aims to develop data warehouses for data-driven applications. To do this, you need to use and apply some of the concepts and techniques introduced in the lectures so far. Use PostgreSQL database, Python and its libraries to define and set up a data warehouse for one data-driven application. The assignment should be solved in Python. You can use the following packages for this assignment:

- SQLAlchemy 1.4+ will be used to connect to your database
- You need to install and import all the necessary libraries (e. g. psycopg2 drivers)
- Pandas 1.3+

The documentation of **SQLAlchemy** can be found here: https://docs.sqlalchemy.org/en/14. There are very interesting tutorials you can go through to help you understand how to connect to a DB/DW, how to interact with it, etc. Use those packages as you need them.

The first thing to do is to install PostgreSQL on your computer. Remember the username/password/port that you set during the installation, as you will need them to connect to PostgreSQL using SQLAlchemy

Q1: Data Cube

The given dataset in the DW_dataset.csv file has data about a set of employees in a company. Some pre-processing of data from the original dataset is required to clean them, which is in line with an ETL process in developing a data warehouse.

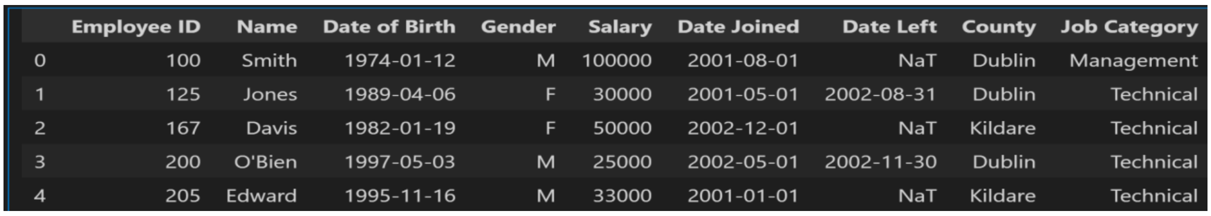
```python
df['Job Title'] = df['Job Title'].str.strip()
df['Gender'] = df['Gender'].str.strip()
df[['Address', 'County']] = df["Address"].str.split(r"\bCo\b", expand=True)
df['County']=df['County'].str.replace(r'.',"", regex=True)
```

```
df['Date of Birth'] =  pd.to_datetime(df['Date of Birth'],
infer_datetime_format=True)
df['Date Joined'] =  pd.to_datetime(df['Date Joined'],
infer_datetime_format=True)
df['Date Left'] =  pd.to_datetime(df['Date Left'], infer_datetime_format=True)

def getJobCategory(x):
    y = x.split(' ');
    if 'Technician' in y:
        return 'Technical'
    elif 'Director' in y:
        return 'Management'
    elif 'Manager' in y:
        return 'Management'

df['Job Category'] = df["Job Title"].apply(getJobCategory)
df.head()
df = df.drop(['Address', 'Job Title'], axis=1)
```

After the pre-processing, the data will look like this:



	Employee ID	Name	Date of Birth	Gender	Salary	Date Joined	Date Left	County	Job Category
0	100	Smith	1974-01-12	M	100000	2001-08-01	NaT	Dublin	Management
1	125	Jones	1989-04-06	F	30000	2001-05-01	2002-08-31	Dublin	Technical
2	167	Davis	1982-01-19	F	50000	2002-12-01	NaT	Kildare	Technical
3	200	O'Bien	1997-05-03	M	25000	2002-05-01	2002-11-30	Dublin	Technical
4	205	Edward	1995-11-16	M	33000	2001-01-01	NaT	Kildare	Technical

After that, create a PostgreSQL database. Then create an Alchemy engine to communicate with it.

```
engine = db.create_engine('postgresql://PGusername:PGpass@localhost:port/DBname')
```

if the connection is successful, use Pandas's to_sql function to store the dataframe into using the engine.

Use sqlalchmy engine to perform the following OLAP queries on PostgreSQL and answers to the following questions.

1) Calculate the average salary of management staff for males and females separately.
2) Calculate the average salaries of employees between the counties of Kildare and Dublin. Then calculate the average salary by gender and by county
3) How many people are employed at the end of 2022 who were born in the 1970s, 1980s and 1990s respectively?
4) If the employee retention rate is the % of staff who stayed during a period (compared to the beginning of that period), what are the employee retention rates in 2001 and 2002?
5) Show the retention rates based on the quarter of the years 2001 and 2002.

Q2: Data Warehouse - Implementation

I-

Suppose that a data warehouse for Big University consists of the four dimensions ***student***, ***course, semester***, and ***instructor***, and two measures ***count*** and ***avg_grade***. At the lowest conceptual level (e.g., for a given student, course, semester, and instructor combination), the

avg_grade measure stores the actual course grade of the student. At higher conceptual levels, ***avg_grade*** stores the average grade for the given combination.

1) Draw a snowflake schema diagram for the data warehouse. You are free to improve the sample `input_DW_data.csv` file with additional rows and columns.
2) Starting with the base cuboid [student, course, semester, instructor], what specific OLAP operations (e.g., roll-up from semester to year) should you perform to list the average grade of CS courses for each Big University student?
3) If each dimension has five levels (including all), such as "*student < major < status < university < all*", how many cuboids will this cube contain (including the base and apex cuboids)?

II-

4) Establish a connection with the database to create tables where you can store and read records and arrays of data. Make sure you follow the PostgreSQL naming convention.
5) Define the following functions to read, write, update and list your data to/from the data warehouse.

```
def read_record (Table, Field, Value, engine):
#Reading a record from a database
     Table:  DB Table name
     Field: the field name to read
     Value: the value to select in WHERE clause
     engine: SQLAlchemy engine

def write_record (Table,[values], engine): …
#Writing a record into a database
     Table:  DB Table name
     Values: values of each columns
     engine: SQLAlchemy engine

def  update_record  (Table,  Updatefield,  value  ,new  value,
SelectField, SelectValue, engine):
#updaing a record in a database
     Table:  DB Table name
     Updatefield: the name of the field to update
     Value: value to select
     New Value: New value to update
     SelectField: the column used to select record to update
     SelectValue: The value to select from updatefirld
     engine: SQLAlchemy engine

def read_dataset (name, engine):
#Read a table from DB and store it in a dataframe
          Table: DB Table name
          engine: SQLAlchemy engine

def write_dataset (name, dataset, engine):
#Writing the dataframe into a database table
          Table: DB Table name
          dataset: name of df
          engine: SQLAlchemy engine
def list_datasets (engine):
        #list all tables in database
```

Please make sure that you have completed this practical. Next week, you will get the second part of the practical.