UNIVERSITEIT TWENTE.

Data Science [201400174]

Course year 2022/2023, Quarter 1B

DATE
November 11, 2022

EXCERPT

Data Preparation and Visualization [DPV]

TEACHERS
Faizan Ahmed
Ellen-Wien Augustijn
Nacir Bouali
Faiza Bukhsh
Rolf de By
Karin Groothuis-Oudshoorn
Maurice van Keulen
Mahdi Khodadadzadeh
Elena Mocanu
Estefania Talavera
Brenda Voorthuis
Shenghui Wang

PROJECT OWNERS
Faiza Bukhsh
Karin Groothuis-Oudshoorn
Maurice van Keulen
Elena Mocanu
Mannes Poel
Michel van Putten
Mohsen Jafari Songhori
Luc Wismans

COURSE COORDINATOR
Karin Groothuis-Oudshoorn (quartile 1A)
Maurice van Keulen (quartile 1B)
Faizan Ahmed (quartile 2A)

Topic 1

Data Preparation and Visualization [DPV]

1.1 Introduction

Topic teachers: Maurice van Keulen (general) and Faizan Ahmed (R-based part).

Data Warehousing, OLAP and Data Visualization are in essence technologies developed for Business Intelligence. They are, however, also effective for data science. The topic will teach (a) data warehousing techniques for extracting and transforming data (ETL), (b) modeling data for analytic purposes using the multidimensional modeling approach of OLAP, and (c) data visualisation techniques.

The topic uses open source or otherwise (temporarily) free to use tools. The method and working of these tools are representative for what is used in practice, both in the fields of Business Intelligence as well as Data Science.

1.1.1 Global description of the practicum and project

The process of obtaining visualisations from raw data follows four steps:

1. *Determine the business questions.*
2. *Design a data warehouse,*
   i.e., design a database schema that can answer the business questions based on the multidimensional modelling method. You can in principle use any conceptual data modelling tool for this, but we do this by hand (i.e., with pen and paper). By setting up a database (we use PostgreSQL Server as DBMS) with this schema, you have obtained the *structure* of the data warehouse suitable for analysis ... but no data yet.
3. *Prepare data and fill the data warehouse,*
   i.e., extract relevant data from the raw sources, transform it, clean it, and store it in the database based on the principles of ETL. You can do this with a self-written program possibly accompanied by SQL statements, but many industrial tools (also) that support a kind of visual programming for ETL. In this topic, you accomplish this task in the programming language R. After this step, one has obtained a *filled* data warehouse.
4. *Visualize the data,*
   i.e., use a visualisation tool (we use R or Tableau) that connects to the data warehouse and presents the data in graphs that effectively answer the business questions.

The practicum assignments follow a different order: the first exercise is about understanding the concepts of the multidimentional model, one then practices with steps 3 and 4, and then with steps 1 and 2. Information on the data sets and database can be found in Section 1.2.

If you decide to apply this technology in a project, you are strongly advised to use the multidimensional approach and follow the four steps for the given data set. You can distinguish yourself by formulating original business questions, by paying extra attention to visualisation or cleaning, and/or by constructing complex transformations that dig deeper in the data.

1.1.2 Study material and tools

- Data Warehousing Book - "Multidimensional Databases and Data Warehousing", Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen
- Data Warehouse: PostgreSQL Server (pre-installed on a remote server)
- Database administration: PhpPgAdmin (pre-installed on the same remote server; web-based user-interface)
- R and R-studio or Python with an IDE (such as Jupyter Notebook or Spyder)
- Data visualization: programming language or visualization tool (see Section 1.3.1 for details)

1.1.3 Deliverables and obligatory items

The practicum assignments specify what should be delivered. You are asked to include all those deliverables of all assignments of DPV into *one PDF file*. Submit the PDF-file to Canvas after you have completed all DPV assignments.

Tips:

- Whatever you have written with pen on paper, simply take a picture of it and include that.
- For R or Python, include the program code in the PDF
- Include also pictures of your visualisations (PDF, JPG or PNG). Either save/export the picture using your programming language or tool, or make a screenshot.

1.2 Data set and database

For the practicum assignments, we use CSV-files (BI_Raw_Data.csv and SuperSales.zip) containing orders to a warehouse in the US. The files can be found in the Canvas site. The attributes in BI_Raw_Data.csv have the following meaning (the ones in SuperSales.zip are a straightforward extension):

attribute	description
Order_ID	Unique identifier for the order. Since one order can have more than one order line, i.e., one order can contain purchases for more than one product, there are multiple rows with the same Order_ID.
Order_Date_Year	Year of the order
Order_Date_Month	Month of the order
Order_Date_Day	Day of the order
Customer_Name	Name of the customer who placed the order
Customer_Country	The customer's country
Product_Name	The name of the product being ordered
Product_Category	The category of the product
Order_Price_Total	The total amount of the order
Product_Order_Unit_Price	The price of one unit of the product
Product_Order_Quantity	The number of product units ordered
Product_Order_Price_Total	The total price of the purchase of all units of that product, i.e., the multiplication of the previous two values

You will use a DBMS (database management system) to store and share your cubes. A DBMS is software that runs in the background and manages tables with data (the cubes you create are essentially also stored

as tables with data). All other tools (including R and Tableau) *connect* to this software to get access to and manipulate the data. A DBMS has already been pre-installed for you: PostgreSQL 10.16 running on server `bronto.ewi.utwente.nl`.

Each group has their own database on this server. You can obtain a database for your group as follows (only one person of the group needs to do this to obtain a database for the whole group).

- Go to DAB: `https://bronto.ewi.utwente.nl/dab/`
- If you do not have an account in DAB, create one with "Register here". You need to use your student email address.
  NB1: there is a known bug with the system that it sometimes produces an error if you click on "Register here". The problem is that there is a "dab" missing in the URL: please change it from `https://bronto.ewi.utwente.nl/register` to `https://bronto.ewi.utwente.nl/dab/register`
  NB2: DAB doesn't seem to work properly in all browsers. For example, Internet explorer, Chrome and Safari do not seem to always work properly, but Edge and FireFox seem to work fine.
  NB3: there have been cases of people using a supported browser but still have problems. Sometimes it was just a matter of clearing the cache and cookies of the browser, so this may also be something to try when you experience problems.
- Sign in and choose the course "ds22231a" which stands for "Data Science 2022/2023 quartile 1A".
- Fill in your Canvas group number and click on "Get credentials".
- A database will be created for you and the system provides you with the credentials: a username (which is the same as the database name) and a password.
- For your convenience, your database already has three *schemas*: `ass2` and `ass3` for assignments 2 and 3, and `project` for use in the project. In this way, you can keep the tables for these apart.
- You can always return to DAB to look at your credentials again and to reset your database.
  Warning: you loose everything in the database when you reset it, i.e., for all schemas in the database.

The same server also runs a web-based database administration tool, called PhpPgAdmin. You can access it with this link: `http://bronto.ewi.utwente.nl/phppgadmin`. If you login with the credentials you obtained from DAB, you will see your database and the three schemas (there is also a fourth; just ignore it). You can easily create, inspect and drop the tables in your database with this tool[1]

1.3 Description of the practical assignments

1.3.1 Tools

The tools that you would need to install for the topic assignments are:

- R and Rstudio **OR** Python with an IDE (such as Jupyter Notebook or Spyder)
- Some means to create visualisations. You can choose between making visualisations in a programming language such as R or Python or using a tool like Tableau, Microsoft PowerBI, or Qlikview. Our advice in decreasing order of preference: R, Tableau, PowerBI, Python, Qlikview (the order of preference is that for the preferred ones, we have the most expertise to help you solving problems).
  NB: Additional requirement! You can do the data preparation and the visualisation both with R or Python, but if you do that, you are required to do so in ***two separate programs*** (e.g., `dataprep.py` and `dataviz.py`). We require this to show how different tools work together by both connecting to the same database for data exchange. With only one program, you could simply visualise what you have in your internal variables without the exchange via the databases hence skipping this step.

NB: choose either R or Python: Student can choose between R and Python

NB: No spreadsheets! The idea of this topic is to go beyond what spreadsheet software such as Excel and Numbers (Mac) can do. Furthermore, data manipulation you do with a spreadsheet is manual and one-off. The idea of using a programming language is that you truly automate the process: it can be repeated or adapted for changing circumstances. Therefore, no use of a spreadsheet other than a quick exploration of the data or solving a particular problem.

[1] You can also use any other *PostgreSQL client* such as PgAdmin (host = bronto.ewi.utwente.nl; port = 5432).

Please find below installation instructions for the needed software
NB: You need not install all of them! Depending on your choices above for programming language and/or tools, you need to only install the associated items. For R you need both R and Rstudio, for Python you need both Python and an IDE, and depending on whether you chose to do the visualisations with a tool, then you may need to additionally install that one.

Installation R and RStudio

- Install the latest version of R from `https://cran.r-project.org`.
- Install the latest version of RStudio (desktop version, free license) from `http://www.rstudio.com/`.
- Open RStudio
- Go to File > new file > R script: you will see the file in the upper left part of Rstudio. In this file you can type syntax and run in later. Do not forget to save it once and a while.
- Go to the right lower pane and click on 'Install'. Now you can install the libraries `DBI`, `RPostgreSQL`. `readr`, `dplyr` and `lubridate`.
- After installing these packages you need to activate them by e.g. type the following piece of R script, select it all and click on 'Run':
```
library(DBI)
library(RPostgreSQL)
library(readr)
library(dplyr)
library(lubridate)
```
- Instead of installing packages via the menu as described above you could have also used (e.g.): `install.packages("dplyr")`.
- By typing `help(<name>)` with the name of the function you want to get the helpfile from you will see the documentation of that function in the lower right corner of R-studio for the tab Help.
- It is advised to work in RStudio with projects. So make first a new directory on your laptop with the name 'datascience' (off course you can choose another name) and a subdirectory with the name 'data'. Then in RStudio goto File > New Project and select `Existing directory` and navigate to the just made directory 'datascience'. If you now open a new script file it will be automaticaly saved in that directory. Moreover, if you put the datafiles (e.g. BI_Raw_Data.csv) in the data subdirectory you need only to specify 'data/filename' when you need to import some data.
- For more background on R we refer to the book "R for Datascience", H. Wickham (see `http://r4ds.had.co.nz/`).

Installation Python We strongly recommend installing the Anaconda Distribution, which includes Python, Jupyter Notebook. It also includes all major libraries.

- Install the latest version of anaconda from `https://www.anaconda.com/products/individual`
  NB: google Colaboratory: You can also use the browser-based google Colaboratory (`https://colab.research.google.com/`). If you use google Colaboratory you do not need to install any software on your computer. However, to use google Colaboratory you need a google account. Most instructions given below also work for google Colaboratory.
- Open Anaconda
- Click Launch below the Jupyter Notebook 

  You can consult the official website to learn more about how to start working with Jupyter Notebook `https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html`
- For this assignment you need three packages: `Pandas`, `Numpy` and `sqlalchemy`. These packages are part of basic Anaconda system, so you do not need to install them.
  - For a quick introduction to Pandas package see (although most commands needed for this assignment are give below): `https://pandas.pydata.org/pandas-docs/stable/user_`

guide/10min.html

NB. Installing packages: You can follow the instructions given at https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/ to install a package if needed.

- To be able to use these packages one need to import them. You can do that as follows:

```
import pandas as pd # needed for most operation
import numpy as np # needed for some array operations
from sqlalchemy import create_engine # needed for DB connection.
```

Installation Tableau

- Get an academic license for a year at http://www.tableau.com/academic/students. While waiting for the license code, you may want to already install a trial version from http://www.tableau.com/products/desktop (version 9.2 or higher) and install it as instructed.
  They will need a proof of your registration. In the past, some students had some trouble proving this. What seems to work best is the following: When you login to Osiris under personal details, you can click on status registration for the degree programme. It will send you to a new page and there you can click on declaration of enrolment, which gives a signed pdf of the UT that you are registered. Make a screenshot of that and provide it to Tableau as proof of bein a student of UT.
- Connect ... to a server ... PostgreSQL. You are likely to get an error message complaining that a driver is missing. Simply download the driver(s) from the speficied place and install it. Then try again to connect.
  Some students mentioned that in their version, there is no button to download the drivers from. If that happens to you, just go to https://www.tableau.com/support/drivers and download the drivers for your operating system. Put the downloaded files in Tableau's driver folder (for windows, this is C:\Program Files\Tableau\Drivers\) and execute the installer.

Installation Microsoft PowerBI The student license you obtain from the UT for Microsoft Office also includes "PowerBI". This is also a very powerful visualisation environment often used in practice just like Tableau.
NB: The assignments of this topic require you to make a connection to a database server. This is only supported in the PowerBI Desktop tool which is only available for Windows. Therefore, unfortunately Mac users cannot use PowerBI for the DPV topic assignments.

- Go to http://portal.office.com and log in with your UT credentials.
- Click on the "app launcher": this is the button with the 9 dots in the upper left corner. Select "All apps". Then click on "PowerBI".
- You can use the tool on-line inside your browser, but for being able to make a connection to a database (which is needed for the assigments), you need to use the PowerBI Desktop client. To download and install it, click on the '...' button in the top right corner, select "Download" and then "PowerBI". Follow the instructions.
- To connect to the database, first click on "Get data" and a pop up will appear. Look for the option "PostgreSQL database", then fill in for "Server" bronto.ewi.utwente.nl and the database name. Click "OK". A window will appear where you fill in your username and password. After completing these steps, an error will pop up saying "Unable to connect". Go to File → Options and Settings → Data Source Settings, and edit the permissions under Encryption: untick the setting for encrypt connections. Then redo the steps from "Get data". You should now get a pop up window with all the available tables from the various schemas: select the ones you need and press "OK".

1.3.2 Assignment 1: Facts and dimensions

This is an on-paper assignment. You don't need any tools for it. [2]

☞ **1.1** See Figure 1.1 which contains data on events happening in the city of Utrecht. We illustrate the

[2] Both examples are simplified versions of actual data from http://data.overheid.nl.

Events

dayofweek	year	nrofevents	totalparticipants	location
Friday	2016	2	15000	Euclideslaan
Friday	2016	1	5000	Domplein
Tuesday	2016	5	25000	Domplein
Wednesday	2016	5	30000	Domplein
Tuesday	2016	6	25000	Muntstraat
⋮	⋮	⋮	⋮	⋮

Figure 1.1: Example of a one-table cube containing data on events happening in the city of Utrecht.

Institutes

instID	institute	cityID
1	University of Twente	153
2	Technical University Eindhoven	772
3	Leiden University	546
4	Maastricht University	935
5	Transnational University Limburg	935
⋮	⋮	⋮

Cities

cityID	city	province
153	Enschede	Overijssel
546	Leiden	Zuid-Holland
772	Eindhoven	Noord-Brabant
935	Maastricht	Limburg
⋮	⋮	⋮

Registrations

instID	study	Phase	year	sex	nrofregistrations
1	Technical Computer Science	Bachelor	2015	Female	13
1	Technical Computer Science	Bachelor	2015	Male	214
1	Technical Computer Science	Bachelor	2016	Female	22
1	Technical Computer Science	Bachelor	2016	Male	270
1	Computer Science	Master	2016	Female	23
1	Computer Science	Master	2016	Male	126
2	Technical Computer Science	Bachelor	2015	Female	19
2	Technical Computer Science	Bachelor	2015	Male	216
2	Technical Computer Science	Bachelor	2016	Female	30
2	Technical Computer Science	Bachelor	2016	Male	280
⋮	⋮	⋮	⋮	⋮	⋮

Figure 1.2: Example of a multi-table cube containing data on the numbers of registered students in The Netherlands

meaning of the data by explaining what the first row means: there were in total 2 events organised in Euclideslaan on a Friday in 2016 which both together drew 15000 participants.

(a) Which attributes are the facts; which attributes are the dimensions?
(b) Draw a conceptual star schema for this cube.

□

☞ **1.2** See Figure 1.2 which contains data on the numbers of registered students in The Netherlands. We illustrate the meaning of the data by explaining what the first row of table "Registrations" means: In 2015, there were 13 female students registered for the bachelor study "Technical Computer Science" at the University of Twente which is located in the city of Enschede in the province of Overijssel.

(a) Which attributes are the facts; which attributes are the dimensions?
(b) Draw a conceptual star schema for this cube.

□

Deliverable: Submit the answers to all DPV assignments as *one PDF file*. For this assignment, include your answers of course, which may also be a scan of handwritten answers.

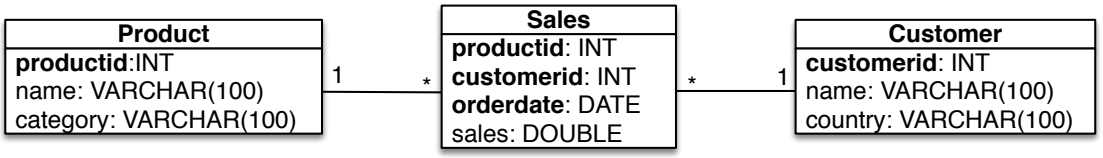

Figure 1.3: Starschema for Mr Bezos' warehouse. Primary key in **bold** (if more attributes are in bold, they are *together* a primary key).

1.3.3 Assignment 2: Re-create the demo from the lecture

Our data file for this assignment, is a spreadsheet file **BI_Raw_Data.csv**. The *encoding* of this file is "ISO-8859-1".[3]

A large warehouse located in the US has many customers for its different product from around the world. The warehouse manager Mr Jack Bezos has many large customers from over the world ordering different products from his warehouse. We follow the method of Section 1.1.1 where all steps are more or less already given (and demoed at the lecture).

Step 1: Business questions

Mr. Bezos wants to know answers for the following questions

- Who are his top-5 most valued customers?
- What are his top-5 most important products?

Step 2a: Multidimensional model

Notice that Mr. Bezos wants to know something about customers and products. These are *dimensions*. Time usually also is a dimension if you are interested in trends, so we also include the dimension 'orderdate'.

What is it he wants to know? It is 'most valued' and 'most important', but what does this mean concretely? How can we *measure* value and importance? Obviously in terms of money: the most valued customer is the one who bought for the most money, and the most important product is the one which was sold for the most money. Therefore, our *fact* is 'amount' (of money), i.e., 'sales'.

Step 2b: Create tables in your database

We create a database schema for this multidimensional model given in Figure 1.3. There is already an empty *schema* in your PostgreSQL database with the name "ass2" intended to hold the tables for this assignment. We use the web-based database management tool PhpPgAdmin to create the tables and attributes in that schema. Open your browser at `http://bronto.ewi.utwente.nl/phppgadmin` to access the tool. Click on "PostgreSQL" under "Servers" to login and expand the panel on the left. You can create tables by navigating to "Schemas" → "ass2" → "Tables".

The database consists of three tables:[4]

- 'customer' with attributes customerid:integer (primary key, NOT NULL), name:character varying(100), country:character varying(100)
- 'product' with attributes productid: integer (primary key, NOT NULL), name:character varying(100), category:character varying(100)
- 'sales' with attributes orderdate:date (primary key, NOT NULL), customerid:integer (primary key, NOT NULL), productid: integer (primary key, NOT NULL), sales:double precision

[3] 'Encoding' is the standard which is used to represent accented and other strange characters. The file has encoded characters such as 'ö' encoded using encoding standard "ISO-8859-1".

[4] Although usually a good thing, we do not create any foreign key declarations, because that will hinder you in re-running your data transformations.

Note: The orderid is *not a key*! A customer can order more than one product in one order which results in several rows for one order. Furthermore, in its purest form, a cube does not store information on individual transactions or cases, but only aggregated data for each combination of dimension values. In the case of Mr Bezos, the cube does not store individual orders, but the sales (fact) for all combinations of the dimensions orderdate, customer, and product. In other words, the orderdate, customerid, and productid together are the primary key!

Step 3: ETL — Prepare data and fill the database

General advice on programming Some advice on programming in general:

- *Small do-test steps*
  Do: Add only one or two small bits, then execute and verify the result, before continuing.
  Do not: Add many steps and then don't know where the mistake is when you receive an error.
- *Read the error message carefully*
  It may contain a lot of gibberish you don't understand, but part of it may provide clues to what is wrong.
- *GIYF: Google Is Your Friend*
  You may think Googling is not academic, but the internet is full of information on what may have caused certain errors and what you can do to fix them.
- *Verify*
  Check your table contents with PhpPgAdmin (Browse) to verify that the data stored by your program really arrived properly in the tables of your database.

NB: We have a description of this step for both the programming language R as well as Python. Students who chose the latter, please look further down for the part "Using Python".

Using R We use R/RStudio to extract the data from the .csv file, to transform it into three tables: one facts table 'sales' and two dimension tables 'product' and 'customer'), and load it into the database. Open first a new syntax file to put your R code in. A usefull package to work with tables is `dplyr`. With the package `readr` you can import data into R, e.g., comma separated values files (CSV). So start with installing and loading both packages in R/RStudio.

```
library(readr)
library(dplyr)
```

Then, we load the data by using the command "read_delim" and put it into the object "data0":

```
data0 <- read_delim(file = "data/BI_Raw_data.csv",
         delim = ";", col_names = TRUE, col_types = NULL,
         locale = locale(encoding="ISO-8859-1"))

head(data0)
```

Use the function 'head' to inspect the first five rows of the imported data.

NB: Under certain circumstances, you may witness that a column like "profit" is not recognised as a column containing numbers. You can see this in the type of the column being "chr". This may, for example, happen if you have set your laptop to Dutch: numbers are represented with a ',' (comma) as a decimal separator instead of the internationally used '.' (period). You can solve this problem by setting the `locale` option to `locale = locale(decimal_mark = ",", encoding = "ISO-8859-1")`.

We first concentrate on producing the tables for the dimensions. A dimension table can be produced in several steps (we use "product" as an example here)

- **Select relevant columns**
  You first need to select the columns from the data0 object that are attributes of products (with function select()), i.e. the name (`Product_Name`) and the category (`Product_Category`).

- **Rename columns**
  Rename the column `Product_Name` to `name` and the column `Product_Category` to `category` with function `rename()` .
- **Remove duplicates**
  In the original .csv file each row is a sales transaction so the products and category columns contain a lot of duplicates. In the products table you want to have only one row for each product/category combination and a productid, the primary key. So for each name and category combination you need to delete the duplicate rows, this can be done by first grouping with function `group_by()` the rows based on `name` and `category` and then use the function `distinct()`. Finaly, you have to ungroup the data again with function `ungroup()`.
- **Generate an identifier for the dimension**
  We now attach a new column with variable name `productid` to the table with the function `mutate()` with `row_number()`. This 'productid' attribute can be used as the primary key of the dimension table.

An example of all steps to make the product table is as follows (using the pipe operator):

```
# Step 1: make Product table 'product':
product <- data0 %>%
  select(Product_Name, Product_Category) %>%
  rename(name = Product_Name, category = Product_Category) %>%
  arrange(name, category) %>%
  group_by(name, category) %>%
  distinct() %>%
  ungroup() %>%
  mutate(productid = row_number())
```

Check that you have now created a table `product` that contains the required columns and 77 rows. You can make the customer table in an analogous manner.

For the sales table you first need to select the columns from the original data that you need and then join the product and customer table to add the productid and customerid to the sales table and finally you need to drop columns that are redundant in the sales table.

Joining two tables can be done as follows (a join between sales and product in the example below)

```
sales <- sales %>%
  full_join(product, by = c("Product_Name" = "name",
        "Product_Category" = "category")) %>%
  select( -Product_Name, -Product_Category)
```

Having now constructed all data for the dimension and fact tables, you can now store them in the database. From R you can connect to the PostgreSQL database server with the functions `dbDriver`, `dbConnect` in packages `DBI` and `RPostgreSQL`. So, install these packages and load them first. The code to store the constructed R tables in the database is as follows (for schema 'ass2').

```
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, port = 5432, host = "bronto.ewi.utwente.nl",
     dbname = "<name db>", user = "<username>", password = "<passwd>",
     options="-c search_path=ass2")
dbWriteTable(con, "product", value = product, overwrite = T, row.names = F)
dbWriteTable(con, "customer", value = customer, overwrite = T, row.names = F)
dbWriteTable(con, "sales", value = sales, overwrite = T, row.names = F)
```

Check now with PhpPgAdmin (`http://bronto.ewi.utwente.nl/phppgadmin/`) that indeed the three tables are there filled with the expected data.

Using Python We use Python/Jupyter Notebook to extract the data from the .csv file, to transform it into three tables: one facts table 'sales' and two dimension tables 'product' and 'customer'), and load it into the database. Open first a new Python file in your Jupyter Notebook to put your Python code in. A useful package to manipulate tabular data is `pandas`. It can also be used to read *.csv files. For some array

operations we need `numpy` package. These packages come pre-installed with Anaconda, so you do not need to install them. However, you need to import them.

Caution: Do not copy paste code in the Notebook, since it might introduce special characters. And it is difficult to spot such errors.

```
import pandas as pd # needed for most operation
import numpy as np # needed for some array operations
from sqlalchemy import create_engine # needed for DB connection.
```

Then, we load the data by using the command `read_csv` and put it into the dataframe "data0":

```
data0=pd.read_csv('data/BI_Raw_Data.csv',sep=';',encoding="ISO-8859-1")
# read first five rows of data
data0.head()
```

NB Note that the csv file must be located in the folder named 'data'. The folder must be in the current python working directory (use the function `pwd()` to know the current Python working directory). Alternatively, you can also provide complete path to the file.

Use the function 'head()' to inspect the first five rows of the imported data.

We first concentrate on producing the tables for the dimensions. A dimension table can be produced in several steps (we use "product" as an example here)

- **Select relevant columns**
  You first need to select the columns from the data0 object that are attributes of products (you can do it by column name e.g. `data0[['Product_Name', 'Product_Category']]` will return a dataframe with only two columns Product_Name and 'Product_Category'), i.e. the name (`Product_Name`) and the category (`Product_Category`).
- **Rename columns**
  Rename the column `Product_Name` to `name` and the column `Product_Category` to `category` with function `rename()` .
- **Remove duplicates**
  In the original .csv file each row is a sales transaction so the products and category columns contain a lot of duplicates. In the products table you want to have only one row for each product/category combination and a productid, the primary key. So for each name and category combination you need to delete the duplicate rows, this can be done by function `drop_duplicates`.
- **Generate an identifier for the dimension**
  We now attach a new column with variable name `productid` to the table with the function `reset_index().index`. This 'productid' attribute can be used as the primary key of the dimension table.

An example of all steps to make the product table is as follows:

```
# Create Product table.
# Step-1: Select the columns related to Product such Product_Name and Product_Category
product=data0[['Product_Name', 'Product_Category']]
print(product.shape)
# Step 2: Rename the columns according to your star schema
product=product.rename(columns={'Product_Name':'name','Product_Category':'category'})
# Step 3: Remove duplicatess.
# Note ignore_index will work only for Pandas version 1.0.0 or higher.
#If it gives error do not use it or update pandas version.
product=product.drop_duplicates(ignore_index=True,keep='last')
# Step 4: Generate an identifier for the dimension
# Resetting index to be sure it starts from zero.
product['productid'] = product.reset_index().index
# Step 5: re order the columns
product=Product[['productid','name','category']]
# Print the shape of data((3,77) should be printed)
```

```
print(product.shape)
#Inspect first five rows( just a sanity check)
product.head()
```

Check that you have now created a table `product` that contains the required columns and 77 rows. You can make the customer table in an analogous manner.

For the sales table you first need to select the columns from the original data that you need and then join the product and customer table to add the productid and customerid to the sales table and finally you need to drop columns that are redundant in the sales table.

You can use the function `merge` to join data from two tables. Joining two tables can be done as follows (a join between sales and product in the example below)

```
sales= pd.merge(sales,product,how='outer',
    left_on = ['Product_Name','Product_Category'],
                right_on = ['name','category'])
```

NB: The following is a description of some of the optional arguments used above. the description is taken from the official documentation.

- **how='outer'** this refers to SQL's full outer join. This means that if for a row on the left or right there is no match on the other side, the row is still added to the result, but with 'None' in every attribute of the other side. Keys are sorted lexicographically.
- **left_on** Column or index level names to join on in the left DataFrame. Can also be an array or list of arrays of the length of the left DataFrame. These arrays are treated as if they are columns.
- **right_on** Column or index level names to join on in the right DataFrame. Can also be an array or list of arrays of the length of the right DataFrame. These arrays are treated as if they are columns.

Having now constructed all data for the dimension and fact tables, you can now store them in the database. From Python you can connect to the PostgreSQL database server with the functions `create_engine`, `to_sql` in packages `sqlalchemy`. The code to store the constructed Python tables in the database is as follows (for schema 'ass2').

```
# first create link to database
# Replace username with the user name password with the password
driver='postgresql'
username='use_your_own_username'
dbname=username # it is the same as the username
password='use_your_own_password'
server='bronto.ewi.utwente.nl'
port='5432'
# Creating the connetcion pool for SQL
engine = create_engine(f'{driver}://{username}:{password}@{server}:{port}/{dbname}')

product.to_sql('product', engine,schema='ass2', index=False)
customer.to_sql('customer', engine,schema='ass2',index=False)
sales.to_sql('sales', engine,schema='ass2',index=False)
```

Note that 'db_name' is the same as your username. You must use your username and password.

NB: Above we have used string interpolation to form the address to be passed to engine. The code works only for Python 3.6 or higher.

Check now with PhpPgAdmin (`http://bronto.ewi.utwente.nl/phppgadmin/`) that indeed the three tables are there filled with the expected data.

Step 4: Visualize

Now that we have our data ready in a form suitable for analysis, it is time to address the two business problems raised in the beginning. We create a dashboard with metrics to measure the required KPIs. We describe below how to do this for R, Python, and Tableau. Other programming languages and tools have similar steps. Don't use spreadsheet software for this.

NB: You can do the data preparation and the visualisation both with R or Python, but if you do that, you are required to do so in ***two separate programs*** (e.g., `dataprep.py` and `dataviz.py`). We require this to show how different tools work together by both connecting to the same database for data exchange. With only one program, you could simply visualise what you have in your internal variables without the exchange via the databases hence skipping this step.

R We advise to use the package "ggplot2" for this. See `https://www.r-graph-gallery.com/ggplot2-package.html` for an overview, detailed documentation, and example programs for each type of visualisation.

For the data preparation, you already used packages *DBI* and *RPostgreSQL* with functions "dbDriver", "dbConnect", and "dbWriteTable". It obviously also has a "dbReadTable" function.

Python We advise to use the package "matplotlib" for this. See `https://matplotlib.org/stable/tutorials/index` for an overview, detailed documentation, and example programs for each type of visualisation. You can also use `df.plot(kind='bar')` on the created dataframe (uses matplotlib) e.g.

```
sales['Customer_Name'].value_counts().plot(kind='bar', figsize=(20,5))
```

Tableau Connect it to the PostgreSQL database. For this you have to choose PostgreSQL and then you get a pop-up menu where you can select the server ('`bronto.ewi.utwente.nl`'), the database ('name of your database'), username and password. If all is well, you should see the three tables you created. Drag them to the top area starting with the fact table "Sales", then the other two. Notice that Tableau automatically recognizes how the relationships between the tables are: a benefit from a database schema conforming to a multidimensional model. Click "Update now" to read the data into Tableau (or use "Update automatically").

To start our dashboard, click on "Sheet 1", and drag Customer.Name in Y-axes and Sales.Sales in X-axes. Then click "Sort name ascending by sales". You now have a visualization that answers the first business question about the most valued customer. Make another one for product answering the question about most important product.

Deliverables: Include in your PDF file the following

- Include the R source code **OR** Include Python source code
- Export the dashboard visualizations as JPG and also include those as pictures in the PDF.
- Either include the output of the following R code (do not copy-paste):

```
dbGetQuery(con,
           "SELECT table_name FROM information_schema.tables
           WHERE table_schema='ass2'")  ## to get the tables from schema ass2
str(dbReadTable(con, c("ass2", "customer")))
str(dbReadTable(con, c("ass2", "product")))
str(dbReadTable(con, c("ass2", "sales")))
```

  OR include the output of the following Python code:

```
# Notice the use of three quotes. DO NOT COPY PASTE
engine.execute("""SELECT table_name FROM information_schema.tables
               where table_schema='ass2'""").fetchall() ## to get the tables from schema ass2
pd.read_sql_table('product', engine,schema='ass2').info()
pd.read_sql_table('customer', engine,schema='ass2').info()
pd.read_sql_table('sales', engine,schema='ass2').info()
```

  Notice the use of the function `info()`.

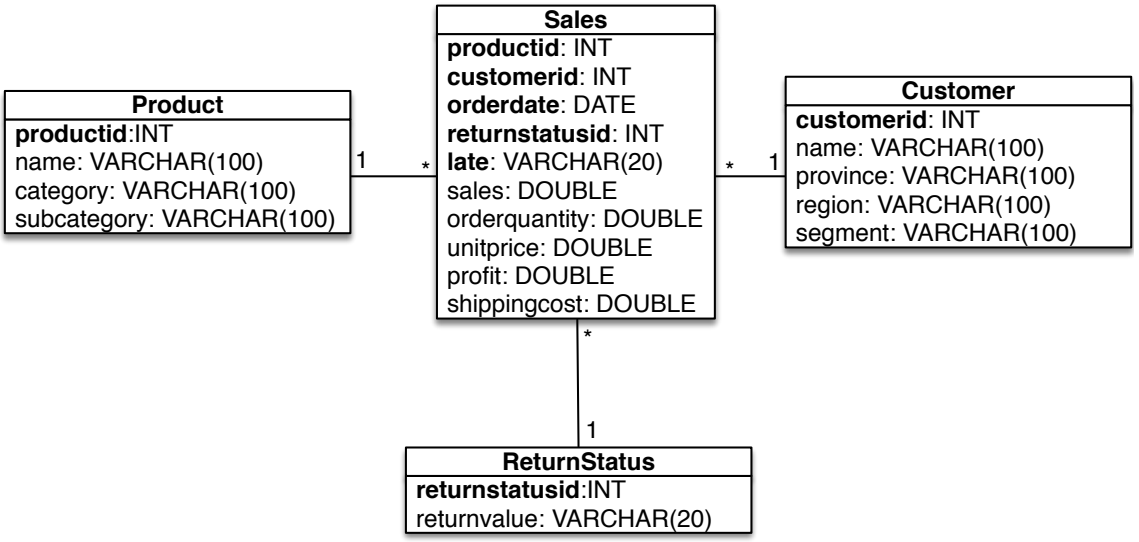

Figure 1.4: Starschema for SuperSales

1.3.4 Assignment 3: Do it yourself

Use the spreadsheets in **SuperSales.zip**.

We follow the method of Section 1.1.1 again, but now only steps 1 and 2 are given; the assignment is to do steps 3 and 4 yourself.

Step 1: Business questions

For this new warehouse in Canada, Mr. Bezos wants to know answers for the following questions

- Which products/product categories made the most loss?
- Which products/product categories were shipped really late (more than 2 days)?
- Which products/product categories were returned the most?

Step 2: Multidimensional model and database structure

Create a table structure in the database according to the star schema of Figure 1.4 for Mr. Bezos to answer his business questions. We require that

- The dimension "ReturnStatus" has a *separate table* containing two rows representing the only two values "Returned" and "NotReturned".
- The dimension "Late" is an *inlined dimension* representing the only two values "Late" and "NotLate".

Step 3: ETL — Prepare data and fill the database

The main goal of this assignment is that you do this step yourself.

Note: There is an important complication in this case. Notice that the star schema models a proper cube in the pure sense of multidimensional modeling. For each combination of dimension values, there should be exactly one row in the fact table. Suppose that the same customer orders the same product twice on the same day and both products are not late and not returned. In other words, all dimensions are the same for these two orders. Therefore, we need to combine the facts of these two orders, so that we end up with exactly one row for this particular combination of dimension values.

As you know from assignment 2, the suggested way of accomplishing this is by using the function `group_by()` and then `summarise()`. Note that combining facts may mean different things for different facts: sales, profit, orderquatity and shippingcost should be summed, but unitprice should not.

Tips:

- The dates in the CSV file are just strings, but to be able to calculate with dates, e.g., to determine the number of days a product is too late you have to transform them into real dates (i.e., values of type 'date'). The function `dmy()` takes a string and transforms it into the right class. When you have obtained the orderdate and shipdate in the appropriate type, you can calculate the number of days with: `interval(orderdate, shipdate)/ddays()`. Next you can use the function `if_else` to make a column with two values "Late" and "NotLate".
- There is more than one CSV-file for which the information needs to be combined. A handy function to accomplish such a combination is `full_join()`. The rows of two tables are joined if they have the same value(s) for one or more key attributes (e.g., "orderid").

Step 4: Visualization

Realize a dashboard that answers these business questions.

Deliverable: Include in your PDF file the following

- Include a picture of your multi-dimensional model / database schema
- Include the R source code Or the Python source code
- Export the dashboard visualizations as JPG and also include those as pictures in the PDF.
- Include the output of the following R code:

```
dbGetQuery(con,
           "SELECT table_name FROM information_schema.tables
           WHERE table_schema='ass2'")  ## to get the tables from schema ass2
str(dbReadTable(con, c("ass2", "customer")))
str(dbReadTable(con, c("ass2", "product")))
str(dbReadTable(con, c("ass2", "returnstatus")))
str(dbReadTable(con, c("ass2", "sales")))
```

or include the output of the following Python code:

```
# Notice the use of three quotes. DO NOT COPY PASTE
engine.execute("""SELECT table_name FROM information_schema.tables
               where table_schema='ass3'""").fetchall() ## to get the tables from schema ass3
pd.read_sql_table('product', engine,schema='ass3').info()
pd.read_sql_table('customer', engine,schema='ass3').info()
pd.read_sql_table('sales', engine,schema='ass3').info()
pd.read_sql_table('returnstatus', engine,schema='ass3').info()
```

1.3.5 Assignment 4: multidimensional modeling. Case "Mobile app beta tester service"

This is an on-paper assignment. You don't need any tools for it.

The company *Pear* sells smartphones. For downloading and updating mobile applications they have the "Pear app store". Pear management has a new business idea: a beta tester service for game developers. The purpose of beta testing is mainly for getting feedback on game play: do you become bored too quickly, is it challenging enough, etc. The purpose is not so much finding bugs. Good gamers can offer themselves as beta tester and developers can hire them. Developers pay Pear for a subscription plus a fee for hiring a beta tester of which a small percentage goes to Pear.

A subscription includes support for developers in finding the "best" beta testers for their particular application. Pear needs to realize a data warehouse for this purpose with all gaming data of the beta testers. They have in their databases timestamps of start of game app, switch to other app, and relevant events (turn, next level, completion of game, etc.) as well as scores of completed games, and data on the apps: name, solitary/multiplayer, name of developer, a fine-grained category, etc.
(you may assume more data to be available if it is reasonable a company as Pear would have it; make these assumptions explicit)

A beta tester receives a small amount of money for joining in exchange for allowing Pear to disclosing their

data to developers. A second purpose of the data warehouse is that during a beta test, Pear provides the developer with information on how much time the beta tester has devoted to beta testing the game.

(a) For being able to determine the right fact(s) for the star schema an important question is "What is *good*", i.e., "What makes a particular gamer a good beta tester for a particular kind of game?" More concretely, how can you determine a score that quantifies this 'goodness' or 'suitability', which obviously needs to be calculated from the available data.
Propose a formula for "goodness score" and explain why a high value is an indicator for a good beta tester.

(b) What is the business question, or what are the business questions in this case? Formulate them as accurately as possible.

(c) Give a star schema. Explain your design by describing the most important design choices and considerations.

Deliverable: Include in your PDF file the answers to the questions. Also include a picture of the star schema. It may be a scan or handwritten answers.