P12: Acquiring Postgre Procedural Language superpowers

INTRODUCTION

##### PL/pgSQL (Procedural Language/PostgreSQL) is a procedural programming language supported by the PostgreSQL ORDBMS. It closely resembles Oracle's PL/SQL language.

##### PL/pgSQL, as a fully featured programming language, allows much more procedural control than [SQL](https://en.wikipedia.org/wiki/SQL), including the ability to use loops and other control structures. SQL statements and [triggers](https://en.wikipedia.org/wiki/Database_trigger) can call functions created in the PL/pgSQL language.

##### **You have to solve all the proposed problems and document it in a professional way.**

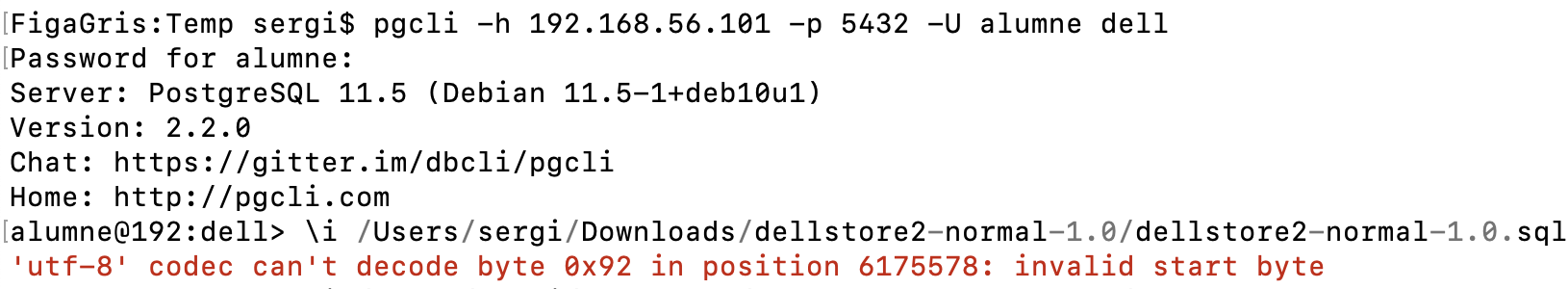
## PART A. Preparing the development environment

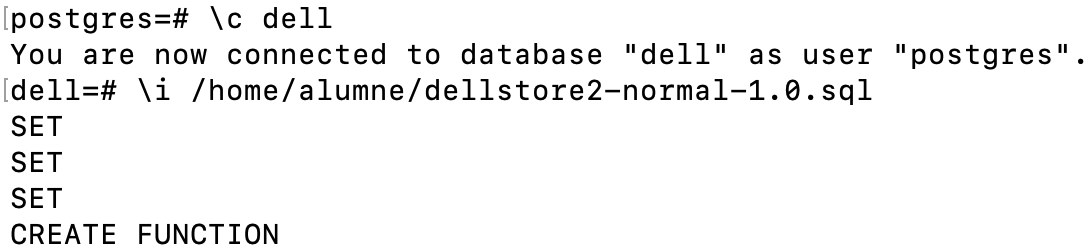
In order to solve this practise you can choose any client-server architecture and choose any of the database clients seen in class.

Download the following file ([dellstore2-normal-1.0.tar.gz](https://ftp.postgresql.org/pub/projects/pgFoundry/dbsamples/dellstore2/dellstore2-normal-1.0/dellstore2-normal-1.0.tar.gz)) and import it into a new database ‘dell’ and into that database into the schema ‘data’ (owner of everything ‘alumne’). Revoke all permission from public to that database and to that schema.

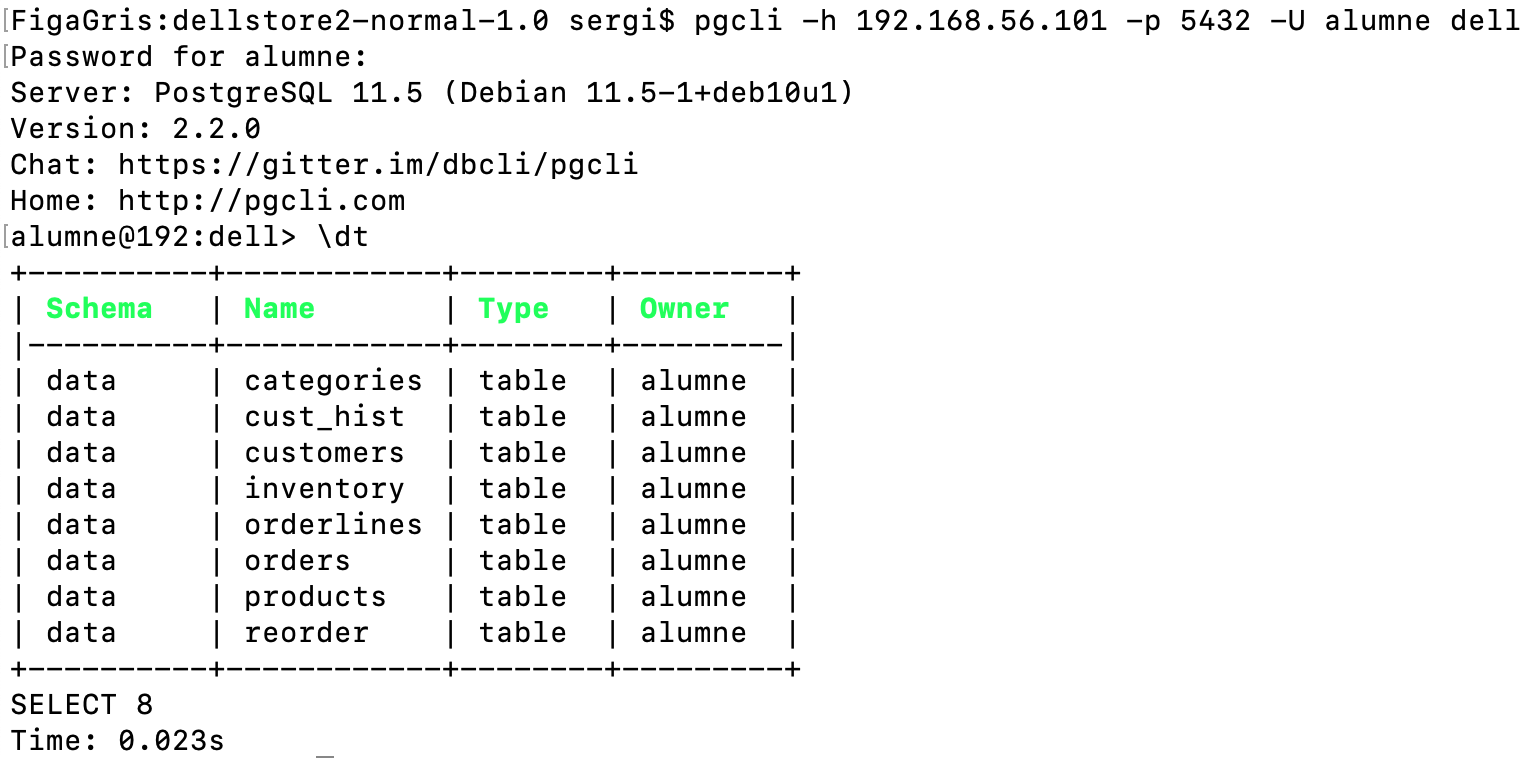
**Clue:**

* **Using pgcli you can get errors importing the database, use psql...**

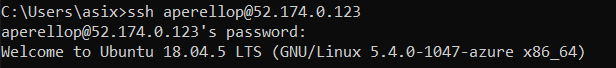




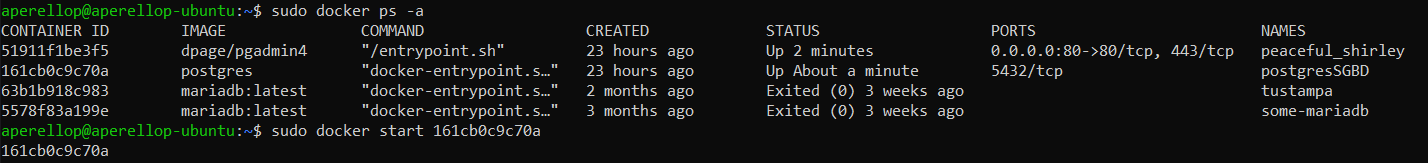
**The result database should be something like this:**



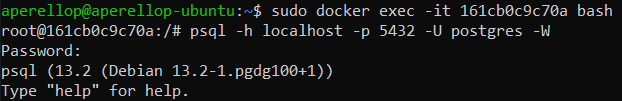
We acces our virtual machine



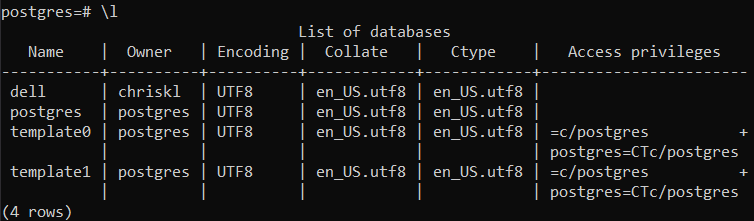
We see our dockers and start postgres



We enter tot the bash of our virtual machine and from it we access postgres



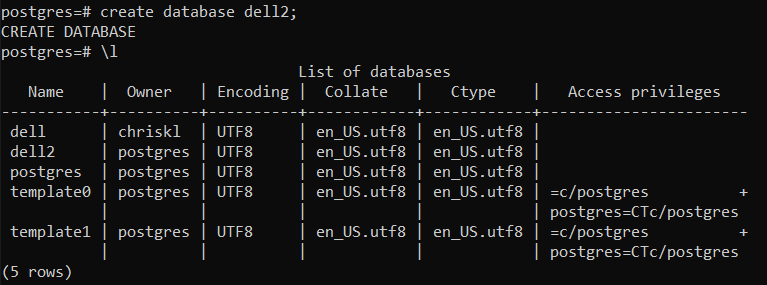
We list all databases



We check wich database we are conected to

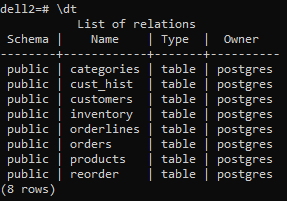


We create the dell database, connect to it and enter all the content

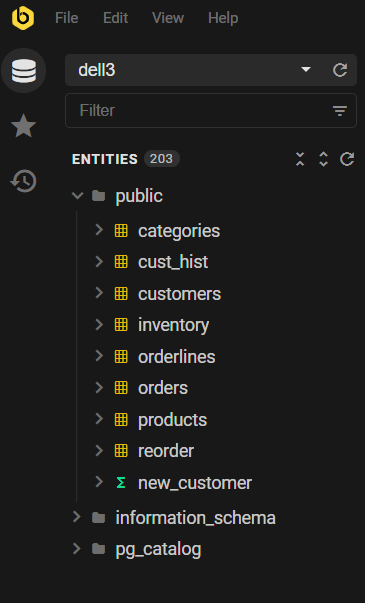




We list the tables of database to check that the import has gone well



And we link it to our graphics tool

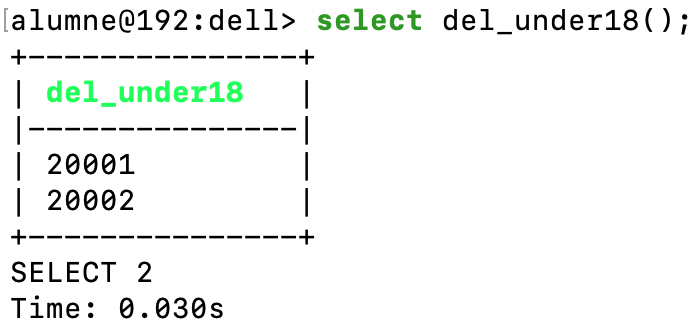


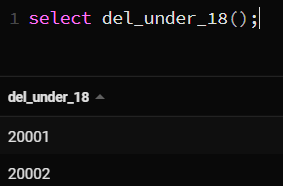
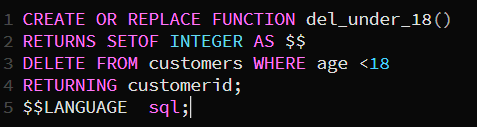
## PART B. QUESTIONS

1.- Create a function ‘del\_under18’ to delete the customers under 18 years old. The function must return the deleted customer’s ids. To test if the function works insert:

INSERT INTO data.customers VALUES (20001, 'Sergi', 'González', '6224597470 Dell Way', NULL, 'DVCINXG', NULL, 0, 'Australia', 2, 'sg@dell.com', '6224597470', 3, '1869697669055999', '2010/07', 'user20001', 'password', 17, 40000, 'F');

INSERT INTO data.customers VALUES (20002, 'Pep', 'López', '6224597470 Dell Way', NULL, 'DVCINXG', NULL, 0, 'Australia', 2, 'sg@dell.com', '6224597470', 3, '1869697669055999', '2010/07', 'user20002', 'password', 17, 40000, 'F');

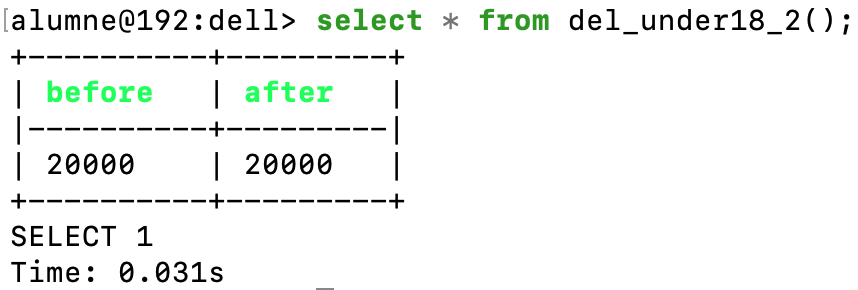
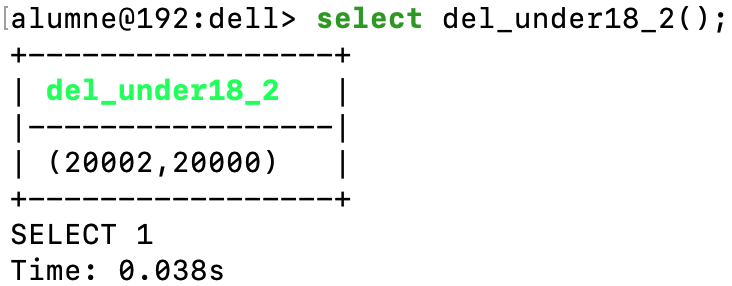


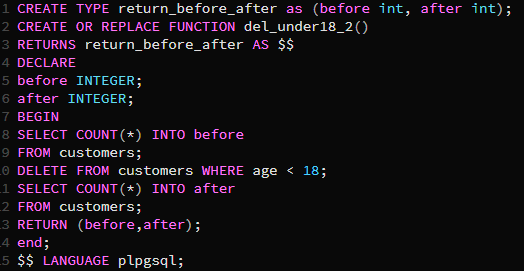
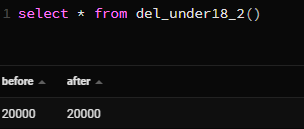


2.- Create a new function ‘del\_under18\_2’ returning the number of customers before and after deleting the ones under 18 y.o. Create a type to return both values in a row. To test if the function works insert:

INSERT INTO data.customers VALUES (20001, 'Sergi', 'González', '6224597470 Dell Way', NULL, 'DVCINXG', NULL, 0, 'Australia', 2, 'sg@dell.com', '6224597470', 3, '1869697669055999', '2010/07', 'user20001', 'password', 17, 40000, 'F');

INSERT INTO data.customers VALUES (20002, 'Pep', 'López', '6224597470 Dell Way', NULL, 'DVCINXG', NULL, 0, 'Australia', 2, 'sg@dell.com', '6224597470', 3, '1869697669055999', '2010/07', 'user20002', 'password', 17, 40000, 'F');



3.- Create a function to insert new categories. Do four versions: the first one with named parameters, the second one with numbered parameters, the third one with parameters with the same name that the table fields (category, categoryname), and the last one with only a single parameter (data type ‘categories’). Use language sql. Headers:

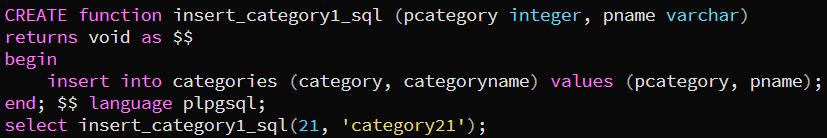
CREATE FUNCTION insert\_category1\_sql(pcategory integer, pname varchar) RETURNS void AS

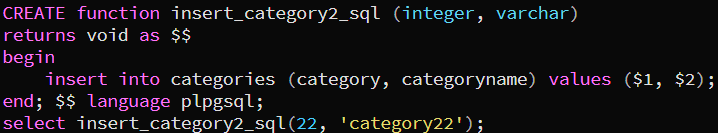
CREATE FUNCTION insert\_category2\_sql(integer, varchar) RETURNS void AS

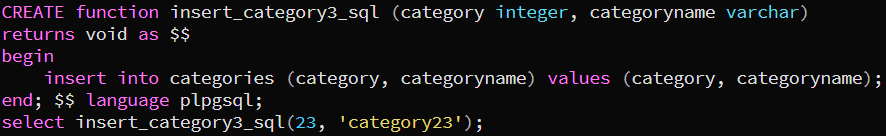
CREATE FUNCTION insert\_category3\_sql(category integer, categoryname varchar) RETURNS void AS

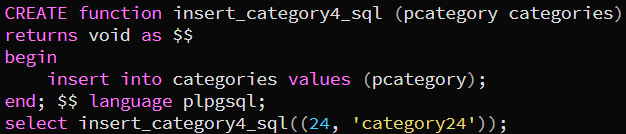
CREATE FUNCTION insert\_category4\_sql(pcategory categories) RETURNS void AS

|  |  |
| --- | --- |
|  |  |
|  |  |









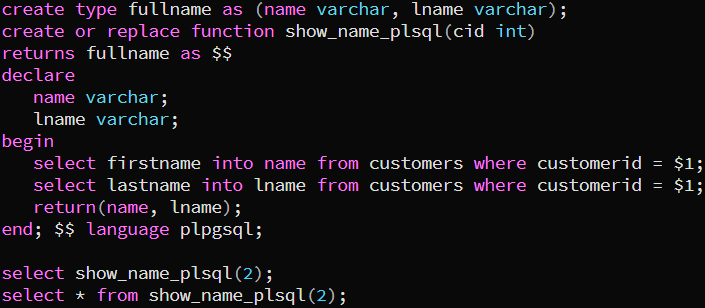
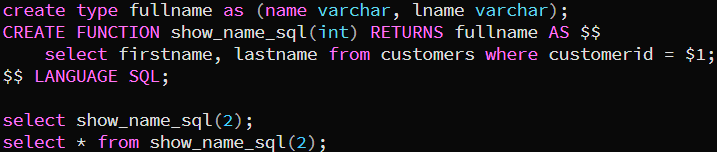
4.- Create the following functions to return the first name and the last name of a customer. Write two versions: one with language sql and another one with plpgsql. It’s mandatory to use the following headers:

CREATE FUNCTION show\_name\_sql(id integer, OUT first varchar, OUT last varchar) AS

CREATE FUNCTION show\_name\_plpgsql(id integer, OUT first varchar, OUT last varchar) AS

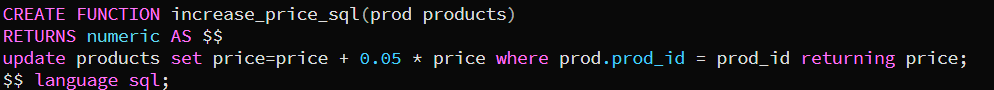
|  |  |
| --- | --- |
|  |  |



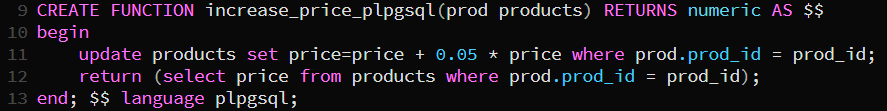


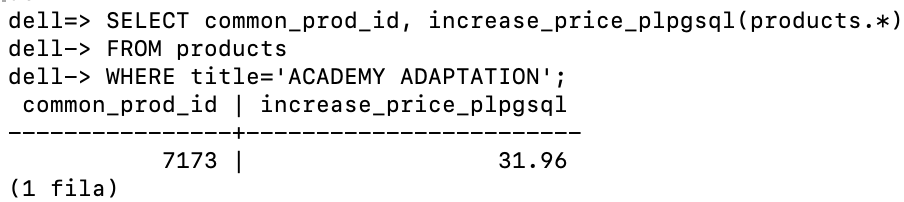
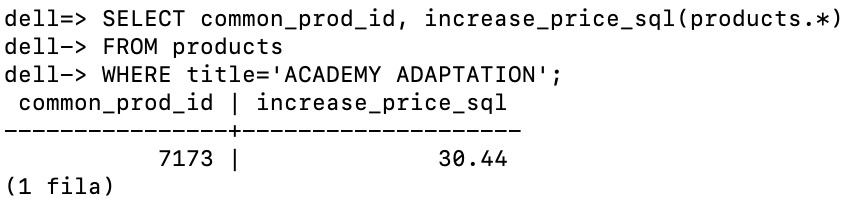
5.- Create a function to increase the price of the products by 5%. You return the new price (use RETURNING CLAUSE). Write two versions: one with language sql and another one with plpgsql. Headers:

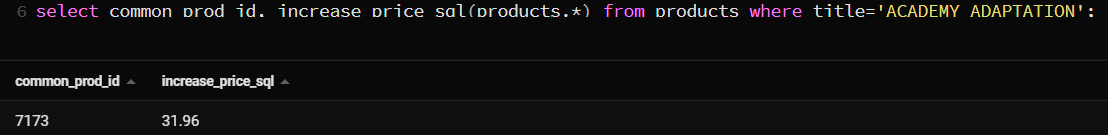
CREATE FUNCTION increase\_price\_sql(prod products) RETURNS numeric AS

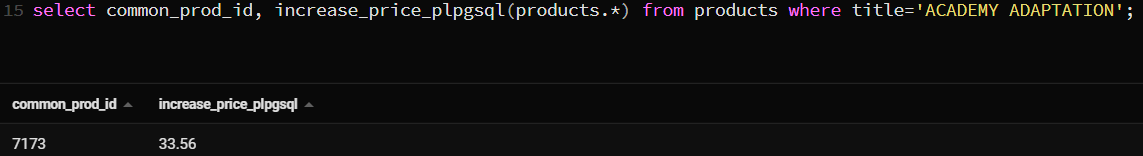


CREATE FUNCTION increase\_price\_plpgsql(prod products) RETURNS numeric AS







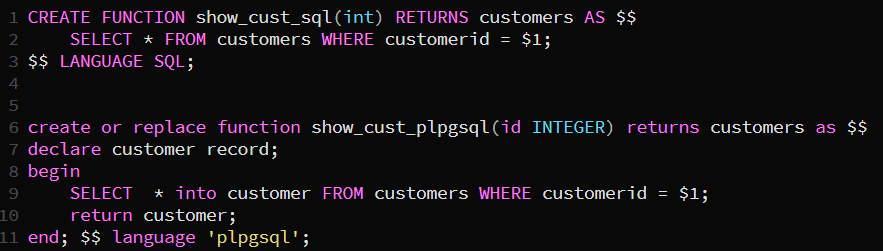


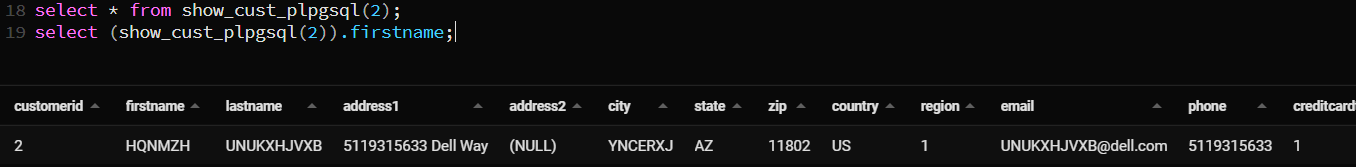
6.- Create a function to return all data of a customer. Write two versions: one with language sql and another one with plpgsql. Headers:

CREATE FUNCTION show\_cust\_sql(id integer) RETURNS customers AS

CREATE or replace FUNCTION show\_cust\_plpgsql(id integer) RETURNS customers AS

|  |  |
| --- | --- |
|  |  |



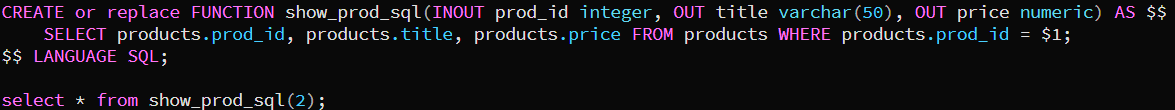


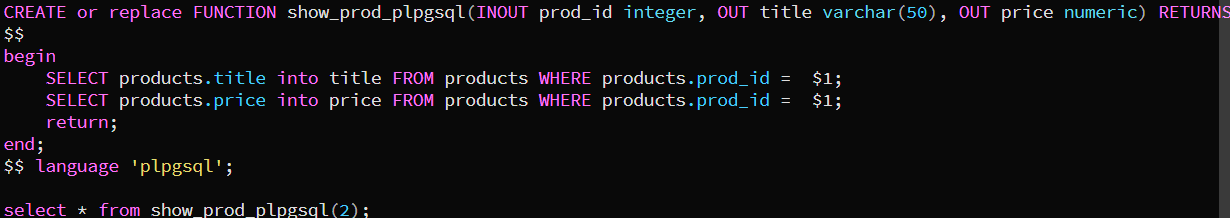
7.- Create a function to return the id, name and price of a product passing its identifier. Write two versions: one with language sql and another one with plpgsql. Headers:

CREATE or replace FUNCTION show\_prod\_sql(INOUT prod\_id integer, OUT title varchar(50), OUT price numeric) AS

CREATE or replace FUNCTION show\_prod\_plpgsql(INOUT prod\_id integer, OUT title varchar(50), OUT price numeric) RETURNS RECORD AS

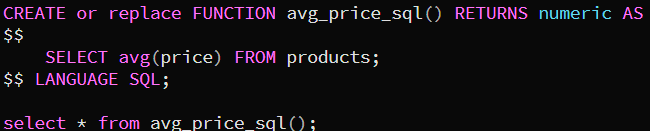
|  |  |
| --- | --- |
|  |  |



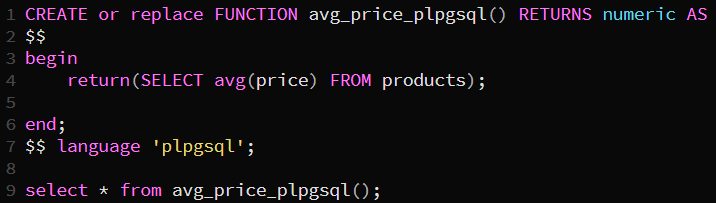


8.- Write an average function (without parameters) that returns the average price of the products. Write two versions: one with language sql and another one with plpgsql. Make a third version not using avg. Headers:

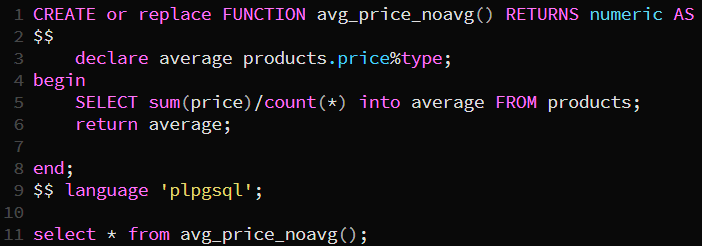
CREATE or replace FUNCTION avg\_price\_sql() RETURNS numeric AS



CREATE or replace FUNCTION avg\_price\_plpgsql() RETURNS numeric AS

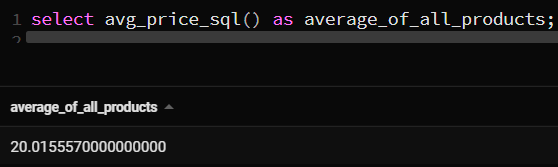


CREATE or replace FUNCTION avg\_price\_noavg() RETURNS numeric AS

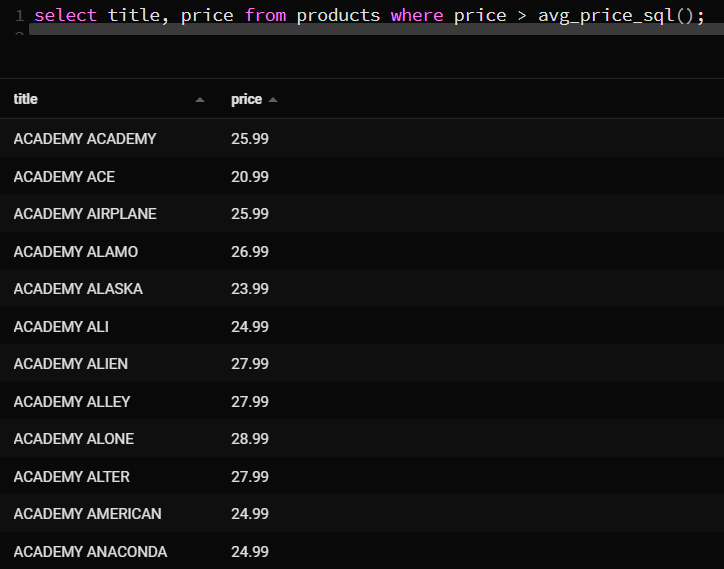


9.- Using the function ‘avg\_price\_sql’, do the following queries:

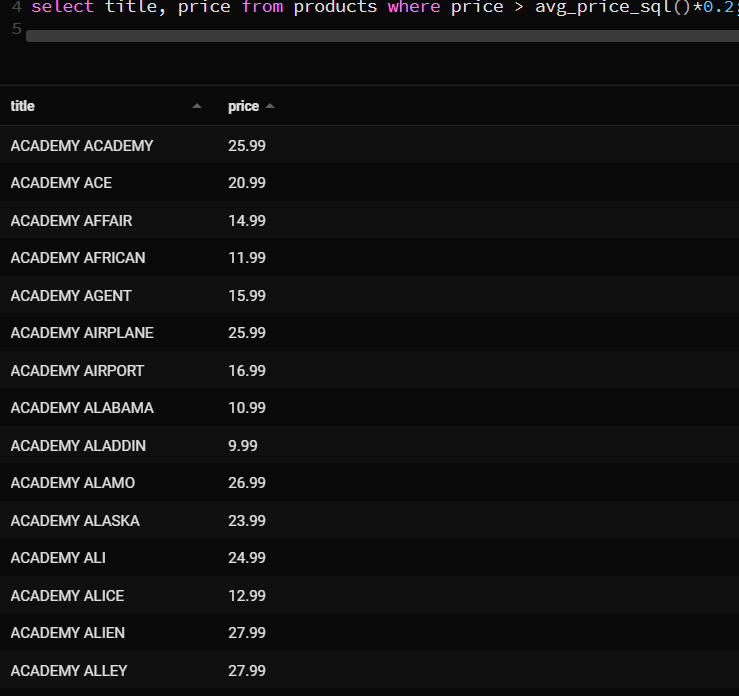
1. Average price of all the products.



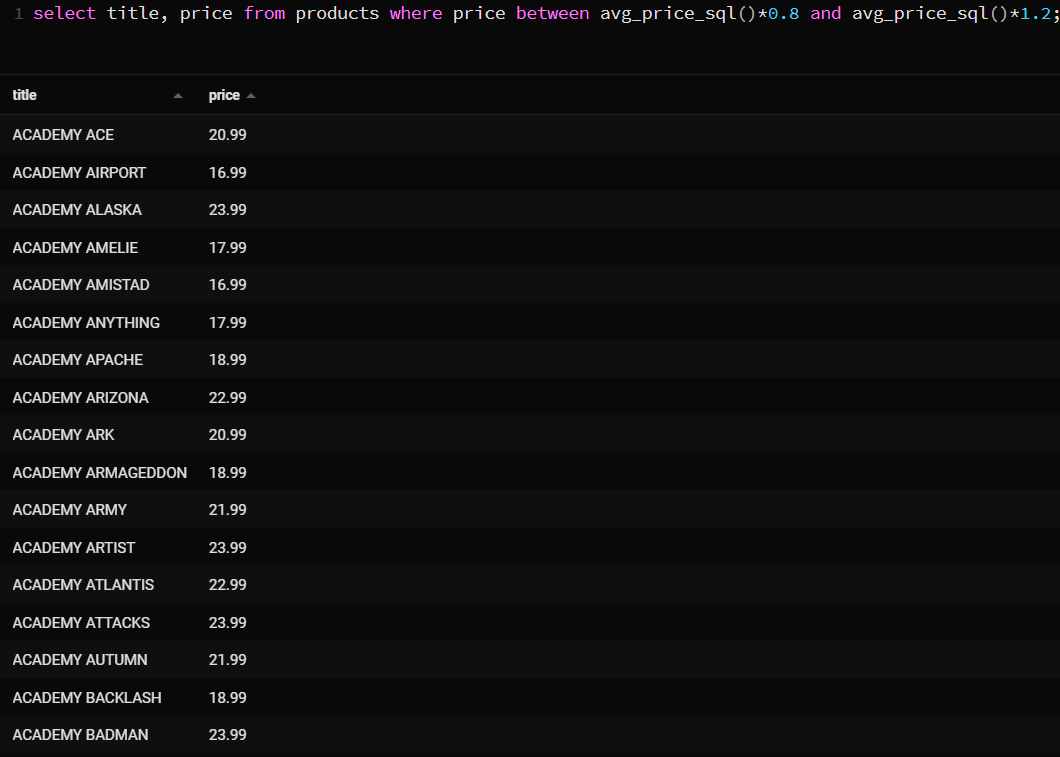
1. Title and price of products that cost more than the average product price.



1. Title and price of products that cost more than a 20% to the average price.



1. Title and price of products that theirs price is equal to the average price to 20% (ie those that theirs price is between 80% and 120% of the average price).

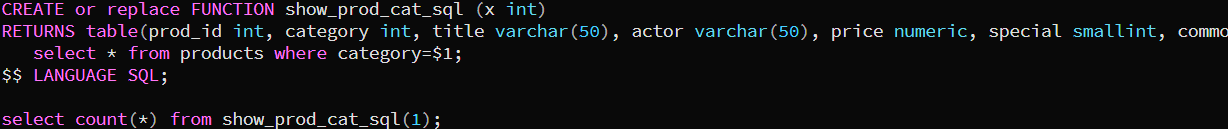


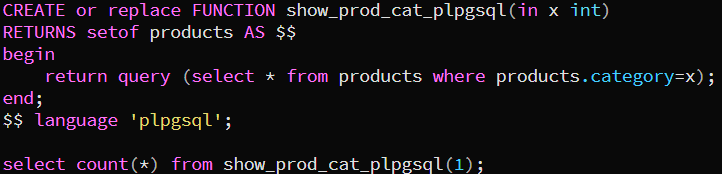
10.- Create a function to return all products of a category. Write two versions: one with language sql and another one with plpgsql.

Clue 1: SETOF.

Clue 2: For the plpgsql version check the slide 28.

|  |  |
| --- | --- |
|  |  |





11.- Repeat exercise 10 (only the plpgsql version) using TABLE to return the values. Clue: Slide 25. Header of the function:

CREATE or replace FUNCTION show\_prod\_cat2\_plpgsql(catid integer) RETURNS TABLE(

prod\_id integer,

category integer,

title character varying(50),

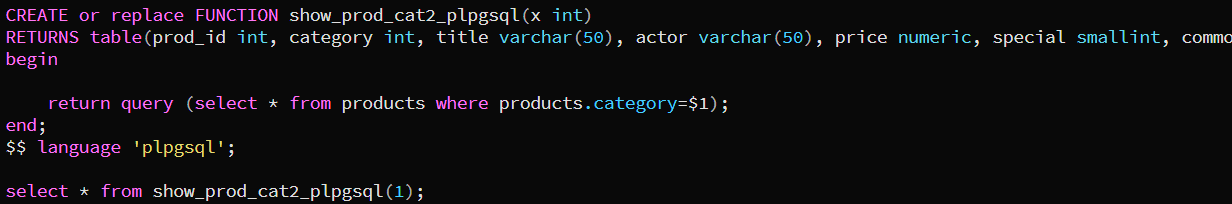
actor character varying(50),

price numeric(12,2),

special smallint,

common\_prod\_id integer

) AS

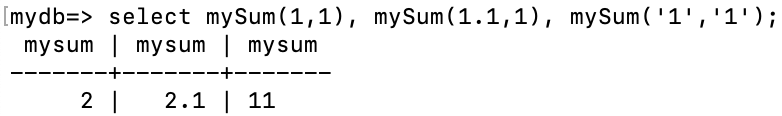


DANGER: 12.- Repeat the last query using a CURSOR (we’ll see this topic later…). Use this header:

CREATE or replace FUNCTION show\_prod\_cat\_plpgsql(catid integer) RETURNS SETOF products AS

Clue: slide 108 (or searching on the Internet).

DANGER: 13.- Create a function “mySum” to add integers, decimals and strings. Note that if you define a function only for integers an error will appear… CLUE: slide 53.



# 

# PART C. CONTROL VERSION ON THE GENERATED CODE

Upload all the final code in sql format into your github repository in a correct order and structure and provide its URL.