**Task 1: Creating database, tables (**Tuomas Pasanen**)**

Note: Text marked in *italics* means a variable name, replace it with the desired name of your own.

Note: psql supports multiline commands, so remember to use a semicolon to end them.

Note: psql is not case sensitive, but it is commonplace to type commands in uppercase.

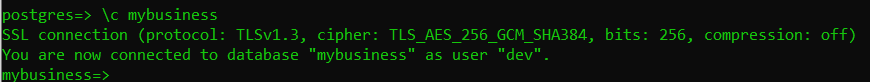
Note: In strings (text), single quotes -> ‘ ‘ are used instead of double quotes -> “ “.

A database can be created with the command “CREATE DATABASE *dbname*[*options*]”:

A picture containing graphical user interface

Description automatically generated

“\l” lists the databases, where we can see that the “mybusiness” database has been created.



With “\c *dbname*”, we can connect to the database. Notice the reticle changing in the terminal, indicating the database we are currently in.

Doing “\dt” (short for “\dtables”) lists the tables in the database, which we have none of currently.



Let’s create the first table in the task, the command to do it is “CREATE TABLE *tablename(data1 datatype, data2 datatype)*”.

Inside the parentheses we have our columns

Text

Description automatically generated

Notice the multiline command to make the command easier to type and read.

Now doing “\dt” shows our table: Text

Description automatically generated

Furthermore, doing “SELECT \* FROM *salesman*” shows the data inside our table, and how it is structured:

A screenshot of a computer

Description automatically generated with medium confidence

Now we want to alter the columns a bit, to make them fit the requirements of the task.

First is constricting the commission -column to values between 0 and 1. We can alter the column and do a check everytime something is inserted, giving an error if our check returns false:



Second is having the serial salesman\_id start from 5000 instead of 0, this can be done with “ALTER SEQUENCE *table*\_*column­\_seq* RESTART WITH *integer*“:



Finally, we can start adding rows to our table with the syntax: “INSERT INTO *tablename*(*data1 datatype, data2 datatype…*) VALUES (*value1, value2…*)”. The data and values in this command have to be in the same order, but the data does not have to be in the same order as the columns in the table.

Since the salesman\_id is a serial type, it is automatically incremented and we don’t (shouldn’t) give it a value when inserting, as it affects the incrementing and can create unwanted results. Serial can also be given the value of “default”, this may be advisable.

Text

Description automatically generated

I will next insert the rest of the rows using a single multiline command by just giving multiple values () fields.

Text

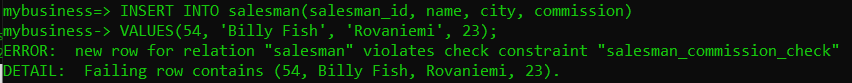
Description automatically generated

A picture containing text

Description automatically generated

Despite the salesman\_id field being serial, in the interest of the task I gave the id’s manually. It might be better practice to just not specify a salesman\_id field, or give it the value of “default”.

We can try out our checks we implemented by trying to insert a row which breaks the check:



An error is thrown and the row is not added (Also an id value was specified, so it does not follow the increment). However if we change commission to follow our constraint:

A picture containing text

Description automatically generated

It gets added. I will remove this row in the interest of the task, with “DELETE FROM *tablename* WHERE *variable = somethingelse*”:

A screenshot of a computer

Description automatically generated with low confidence

Now to create the second table, which introduces foreign keys.

The command is basically the same: create a new table, specify the columns, add the rows.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Notice that the salesman\_id references the salesman\_id -column of the salesman table. This means that it references the salesman table, if for example we wanted to find out more about the salesmen who’ve done the most sales or information about an individual salesman, we can easily reference the id’s.

We can again start the incrementation from a different number:



Now to just insert the rows inside the table:

Text

Description automatically generated

Finally, we can use the affromentioned commands to create the last table, orders:

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Again, customer\_id and salesman\_id are foreign keys, representing the id fields of their respective tables. DATE is a new datatype, but it just represents a date (duh), where it is recommended to use the ISO 8601 format (YYYY-MM-DD), wrapped in single quotes (‘ ‘).  
Text

Description automatically generated

Chart

Description automatically generated with medium confidence