Mockup Project

Part 1 Database and Queries

Supervisor
Panagiotis Tampakis

Made by
Denis og Søren

Abstract

In this report we'll see how you can implement a database in PostgreSQL and construct simple SQL queries to get the desired relation data.

Preface

This is a voluntarily project on the second semester of computer science in the course DM576 Database design. The report is short and concise only documenting key decisions since we dedicated only a few hours to the project as a whole.

Indhold

1 Introduction

The goal of this project is get hands-on experience setting up a database in PostgreSQL and looking up data in it. Furthermore the purpose is about us getting feedback as early in the course as possible. The report reflects and documents the work carried out by us *(Denis and Søren)* in the project.

1.1 Formulation of the problem

Create a database and insert data, then query data.

1.2 Requirements

The database should be deployed in PostgreSQL with appropriate commands to create the tables, by taking into account and enforcing all the integrity constants. About 5-10 records in each table.

The schema to be implemented is as follows:

- `PRODUCT(ProductID, CategoryID, ProductName, Description)`
  Information about a product. A product is assoiciated only with a single category to fullfill the assignment decription.
- `PRODUCT_CATEGORY(CategoryID, Name, Description)`
  The category is the type of the product.
- `SUPPLIER(SupplierVAT, SupplierName, Address, Phone, Email)`
  A supplier supplies the products to a retail store.
- `SUPPLY(InvoiceID, SupplierVAT, Date)`
  Relationship between the supplier, the supplied products and the date og supplying.
- `PRODUCT_SUPPLY(InvoiceID, ProductID, Quantity, Value)`
  Product that has been supplied by the supplier. The price is what the store pay the supplier pr. unit.
- `SALE(SaleID, Date)`
  A sale is between retail and their customers. We interpret this as a sale that could be any combination of the products in stock.
- `SALE_OF_PRODUCT(SaleID, ProductID, Quantity, Value)`
  The price is what customer pay pr. unit of a particular product.
- `PRODUCT_RETURN(SaleID, ProductID, Date, Quantity)`
  A costumer can return the products to the store that they've bought if they're unhappy.
- `STOCK(ProductID, Quantity)`
  The stock is the total

1.3 Scope

There is not much to this project. Just create the database, and discuss the methods of retrieving the data the assignment requires.

2 Database implementation

In this section we'll go through the reasoning behind our implementation of the database. Integrity constraints are discussed but what data have been chosen for the database entries.

2.1 Record data

All record data are shown in Appendix 4.1.

2.2 Integrity constraints

All integrity constraints are shown in Appendix 4.2.

3 Queries

3.1 1: The total value of sale each month of 2022

We chose to combine each `date` of sale with its `value, quantity` attributes a using a `sale NATURAL JOIN sale_of_product` for all sales of 2022 in which returns a relation with all we need to group by `month` and summed the value sold in each month of 2022. To be able to group by month we had to use the `date` type for our date. Then we could `EXTRACT` the month to its own column.

```sql
SELECT SUM(value*quantity) AS totalsalevalue, EXTRACT(MONTH FROM date) AS month
FROM sale NATURAL JOIN sale\_of\_product
WHERE EXTRACT(YEAR FROM date) = 2022
GROUP BY month
ORDER BY month ASC;
```

Listing 1: Total value of sales by month in 2022

3.2 2: Sale with the highest value

Grouping `sale_of_product` by `saleid` in case a sale is of several products. By ordering in a descending manner according to their sum of value and limiting the result to 1 relation instance we are sure that the result is the sale that has the highest value of all the sales.

```sql
SELECT saleid, SUM(value*quantity) AS salevalue
FROM sale_of_product
GROUP BY saleid
ORDER BY salevalue DESC
LIMIT 1
```

Listing 2: Sale with highest value

3.3 3: Categories with maximum and minumum sold item in 2022

Using the same method as with the highest value sale. Here we just group by the `categoryid` from the `product` relation. This query is done two times. One in ascending order and the other in descending. By limiting both results to return 1 relation instance accomplishes that of finding the categories of maximum and minimun sold items. The final result is a union of these queries. Both of these queries consists of `sale_of_prodct NATURAL JOIN product_category`

```sql
 (SELECT categoryid, sum(quantity) AS numberofsolditems
FROM sale_of_product NATURAL JOIN product
GROUP BY categoryid
ORDER BY numberofsolditems ASC
LIMIT 1)
UNION
 (SELECT categoryid, sum(quantity) AS numberofsolditems
FROM sale_of_product NATURAL JOIN product
GROUP BY categoryid
ORDER BY numberofsolditems DESC
LIMIT 1)
```

Listing 3: Categories with minimum and maximum sold items in 2022

3.4 4: Product with the greatest profit

This query was difficult for us which it probably shouldn't have been.

```sql
SELECT productid, SUM(TotalSaleValue - TotalReturnValue -TotalCostValue) AS profit
    FROM
    (SELECT sale_of_product.productID,
    SUM((sale_of_product.quantity-product_return.quantity)*(sale_of_product.value-product_supply.value)) AS revenue
    FROM product_return
    LEFT OUTER JOIN sale_of_product ON product_return.saleid = sale_of_product.saleid
    LEFT OUTER JOIN product_supply ON sale_of_product.productid = product_supply.productid
    GROUP BY sale_of_product.productID)
GROUP BY productid;
```

Listing 4: Product with most profit

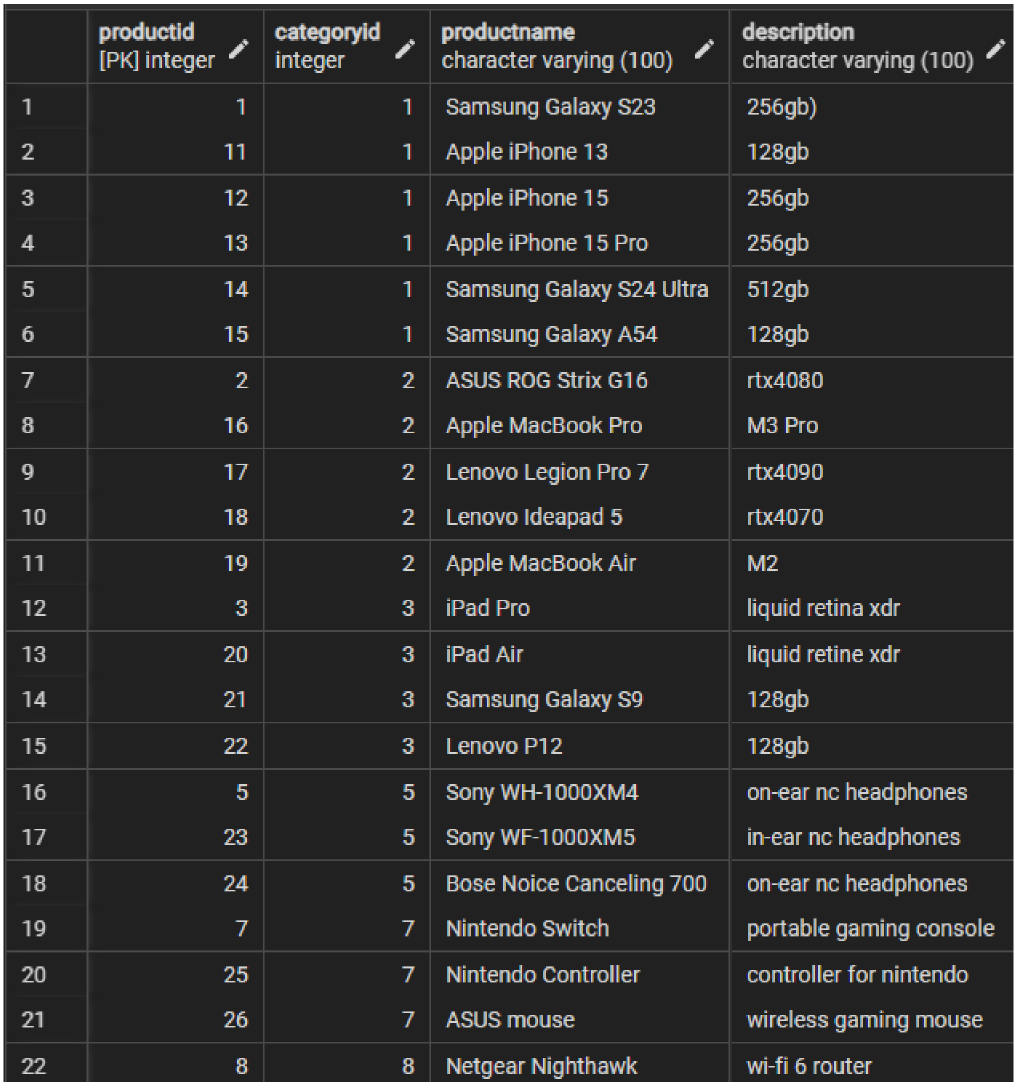
3.5 5: Category profit each month of 2022

TODO

4 Appendix

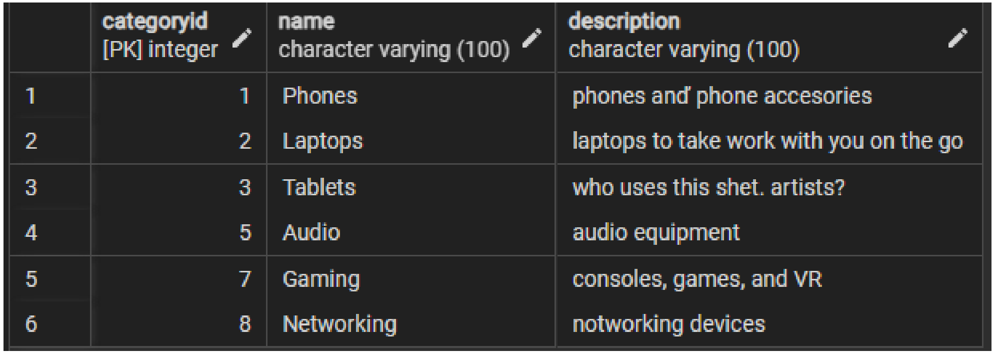
4.1 Tables

product:



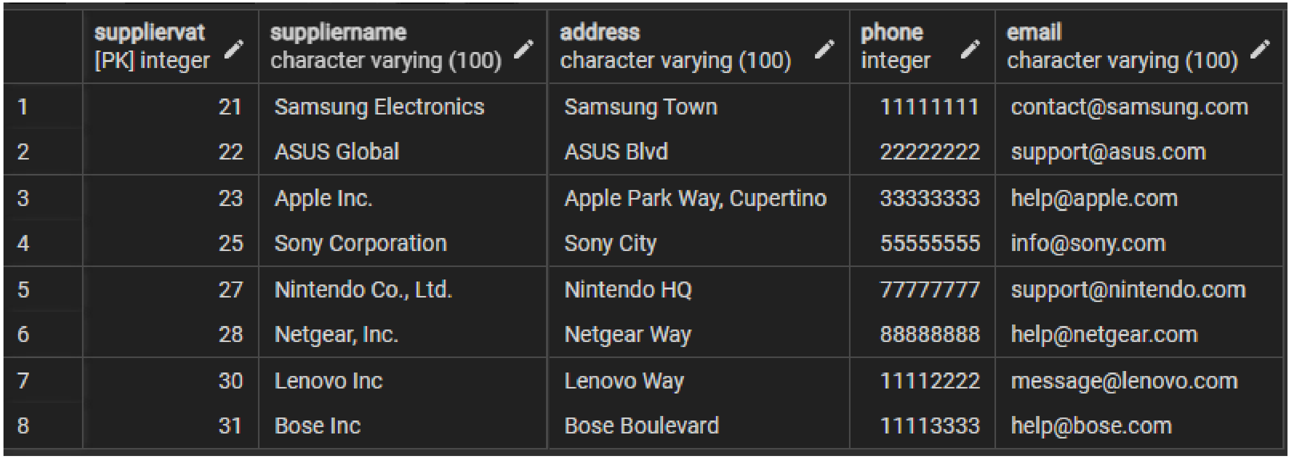
	productid [PK] integer	categoryid integer	productname character varying (100)	description character varying (100)
1	1	1	Samsung Galaxy S23	256gb)
2	11	1	Apple iPhone 13	128gb
3	12	1	Apple iPhone 15	256gb
4	13	1	Apple iPhone 15 Pro	256gb
5	14	1	Samsung Galaxy S24 Ultra	512gb
6	15	1	Samsung Galaxy A54	128gb
7	2	2	ASUS ROG Strix G16	rtx4080
8	16	2	Apple MacBook Pro	M3 Pro
9	17	2	Lenovo Legion Pro 7	rtx4090
10	18	2	Lenovo Ideapad 5	rtx4070
11	19	2	Apple MacBook Air	M2
12	3	3	iPad Pro	liquid retina xdr
13	20	3	iPad Air	liquid retine xdr
14	21	3	Samsung Galaxy S9	128gb
15	22	3	Lenovo P12	128gb
16	5	5	Sony WH-1000XM4	on-ear nc headphones
17	23	5	Sony WF-1000XM5	in-ear nc headphones
18	24	5	Bose Noice Canceling 700	on-ear nc headphones
19	7	7	Nintendo Switch	portable gaming console
20	25	7	Nintendo Controller	controller for nintendo
21	26	7	ASUS mouse	wireless gaming mouse
22	8	8	Netgear Nighthawk	wi-fi 6 router

product_category:



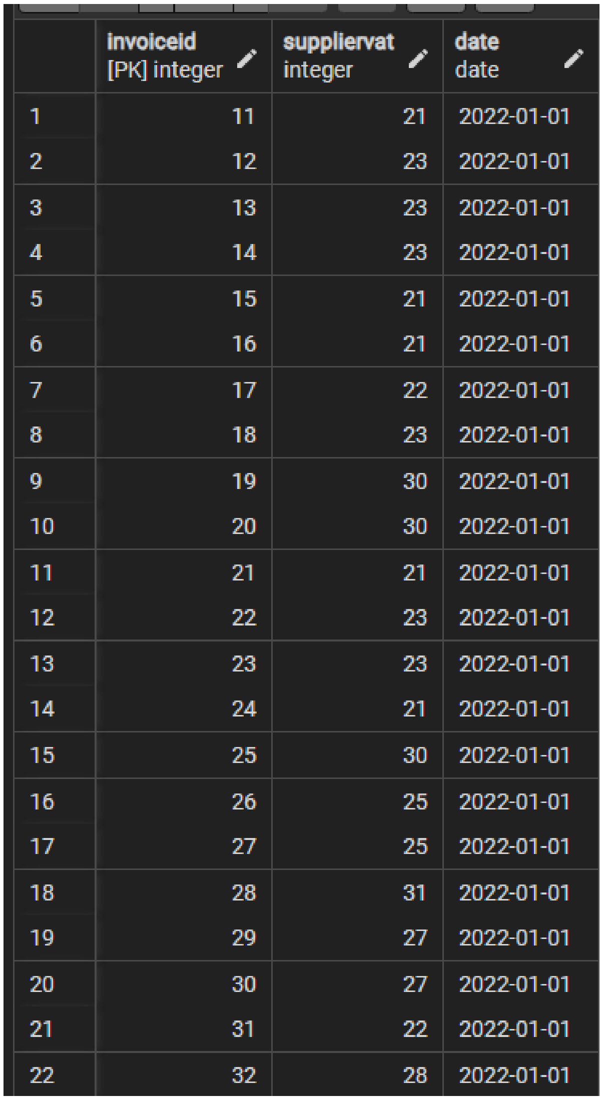
	categoryid [PK] integer	name character varying (100)	description character varying (100)
1	1	Phones	phones and' phone accesories
2	2	Laptops	laptops to take work with you on the go
3	3	Tablets	who uses this shet. artists?
4	5	Audio	audio equipment
5	7	Gaming	consoles, games, and VR
6	8	Networking	notworking devices

supplier:



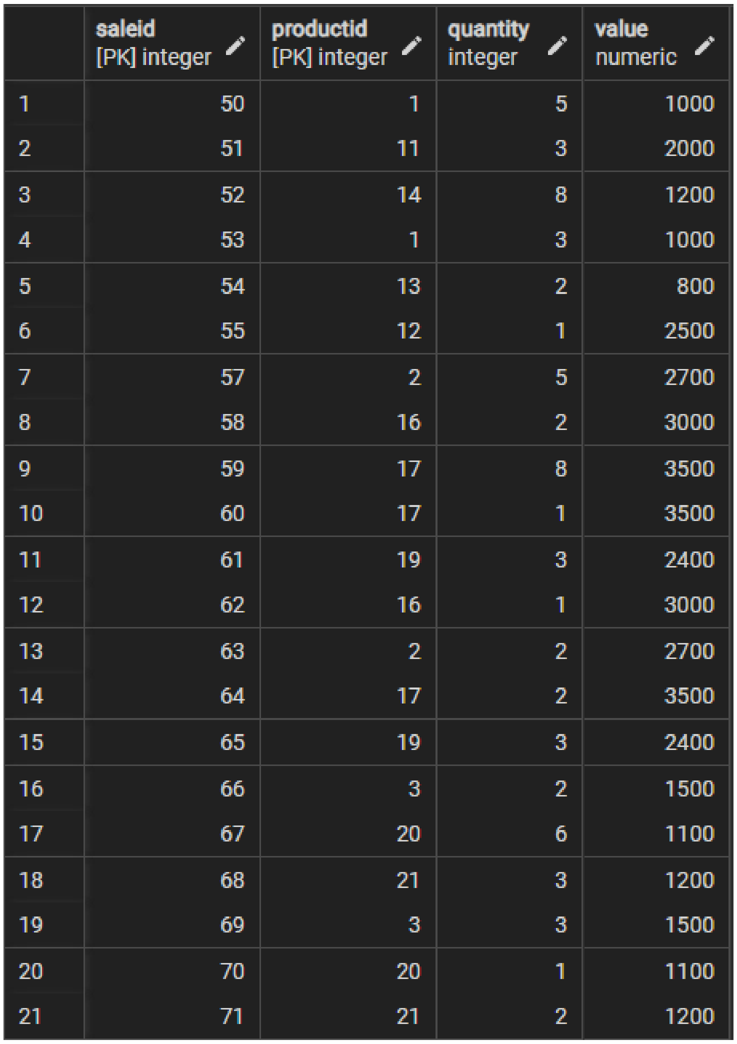
	suppliervat [PK] integer	suppliername character varying (100)	address character varying (100)	phone integer	email character varying (100)
1	21	Samsung Electronics	Samsung Town	11111111	contact@samsung.com
2	22	ASUS Global	ASUS Blvd	22222222	support@asus.com
3	23	Apple Inc.	Apple Park Way, Cupertino	33333333	help@apple.com
4	25	Sony Corporation	Sony City	55555555	info@sony.com
5	27	Nintendo Co., Ltd.	Nintendo HQ	77777777	support@nintendo.com
6	28	Netgear, Inc.	Netgear Way	88888888	help@netgear.com
7	30	Lenovo Inc	Lenovo Way	11112222	message@lenovo.com
8	31	Bose Inc	Bose Boulevard	11113333	help@bose.com

supply:



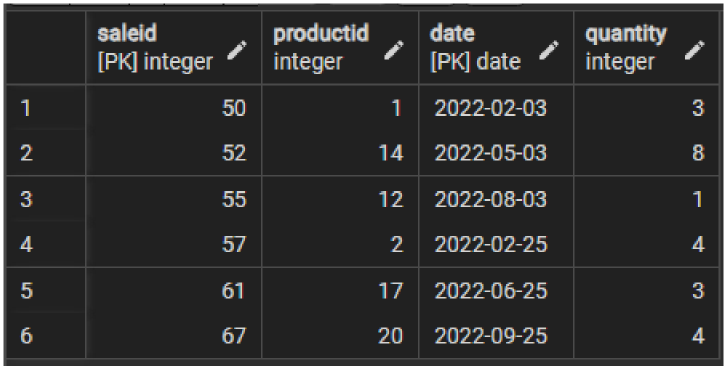
	invoiceid [PK] integer	suppliervat integer	date date
1	11	21	2022-01-01
2	12	23	2022-01-01
3	13	23	2022-01-01
4	14	23	2022-01-01
5	15	21	2022-01-01
6	16	21	2022-01-01
7	17	22	2022-01-01
8	18	23	2022-01-01
9	19	30	2022-01-01
10	20	30	2022-01-01
11	21	21	2022-01-01
12	22	23	2022-01-01
13	23	23	2022-01-01
14	24	21	2022-01-01
15	25	30	2022-01-01
16	26	25	2022-01-01
17	27	25	2022-01-01
18	28	31	2022-01-01
19	29	27	2022-01-01
20	30	27	2022-01-01
21	31	22	2022-01-01
22	32	28	2022-01-01

sale_of_product:



	saleid [PK] integer	productid [PK] integer	quantity integer	value numeric
1	50	1	5	1000
2	51	11	3	2000
3	52	14	8	1200
4	53	1	3	1000
5	54	13	2	800
6	55	12	1	2500
7	57	2	5	2700
8	58	16	2	3000
9	59	17	8	3500
10	60	17	1	3500
11	61	19	3	2400
12	62	16	1	3000
13	63	2	2	2700
14	64	17	2	3500
15	65	19	3	2400
16	66	3	2	1500
17	67	20	6	1100
18	68	21	3	1200
19	69	3	3	1500
20	70	20	1	1100
21	71	21	2	1200

product_return:



	saleid [PK] integer	productid integer	date [PK] date	quantity integer
1	50	1	2022-02-03	3
2	52	14	2022-05-03	8
3	55	12	2022-08-03	1
4	57	2	2022-02-25	4
5	61	17	2022-06-25	3
6	67	20	2022-09-25	4

4.2 Constraints

```sql
--
CREATE TABLE PRODUCT_CATEGORY(
        CategoryID INT PRIMARY KEY,
        Name VARCHAR(100),
        Description VARCHAR(100));
--
CREATE TABLE PRODUCT(
        ProductID INT PRIMARY KEY,
        CategoryID INT,
        Productname VARCHAR(100),
        Description VARCHAR(100),
        FOREIGN KEY (CategoryID));
--
CREATE TABLE SUPPLIER(
        SupplierVAT INT PRIMARY KEY,
        SupplierName VARCHAR(100),
        Address VARCHAR(100),
        Phone INT,
        Email VARCHAR(100));
--
CREATE TABLE SUPPLY(
        InvoiceID INT PRIMARY KEY,
        SupplierVAT INT,
        Date DATE,
        (FOREIGN KEY (SupplierVAT)));
--
CREATE TABLE PRODUCT_SUPPLY(
        InvoiceID INT,
        ProductID INT,
        Quantity INT,
        Value DECIMAL,
        PRIMARY KEY (InvoiceID ,ProductID),
        FOREIGN KEY (ProductID),
        FOREIGN KEY (InvoiceID));
--
CREATE TABLE SALE(
        SaleID INT PRIMARY KEY,
        Date DATE);
--
CREATE TABLE SALE_OF_PRODUCT(
        SaleID INT ,
        ProductID INT,
        Quantity INT,
        Value DECIMAL,
        PRIMARY KEY (SaleID, ProductID),
        FOREIGN KEY (SaleID),
        FOREIGN KEY (ProductID));
--
CREATE TABLE PRODUCT_RETURN(
        SaleID INT,
        ProductID INT,
        Date DATE,
        Quantity INT,
        PRIMARY KEY (SaleID, ProductID, Quantity),
        FOREIGN KEY (SaleID),
        FOREIGN KEY (ProductID));
```

```
--
CREATE TABLE STOCK(
        ProductID INT PRIMARY KEY,
        Quantity INT);
```

Listing 5: constraints