UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Foundations of Databases A.Y. 2021-2022

Homework 3 – Physical Design

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: December 17, 2021

Team acronym	TAGMS	
Last Name	**First Name**	**Student Number**
Arslan	Tunahan	2023640
Basso	Marco	2005796
Cimarosto	Pietro	2027173
Esposito	Vittorio	2005795
Giuliani	Amedeo	2005797
Quiroz	Giannina	2041427
Zanini	Samuele	2019038
Collado	Martin	2039907

Variations to the Relational Schema

We have found some typos within the Data Dictionary section in the previous homework. Below are listed the introduced corrections:

- In Employee the "Registration_date_ID" attribute has been renamed to "Registration_date".
- In Draws_from, the description of the "Lot_ID" attribute is changed from "Foreign Key to Lot, Primary key to Lot" to "Foreign Key to Lot, Primary key"
- In Ship, the "Employee_ID" attribute is not a serial but a text type. In particular, a new domain called "taxnumber" has been defined, to which the Employee entity ID is associated.

Hereafter are reported several clarifications of some choices made during the creation of the DB. The choice to use the data type "numeric" rather than "money" in the price domain is dictated by the company's need to use a more precise decimal notation. In fact, micro price variations may be necessary, which would not be allowed with the "money" type. The company is therefore able to specify a price up to 3 decimal digits. Furthermore, since the company operates only in Italy, the currency used within the system to store the various prices and costs is the euro. Using money, on the other hand, several conversions would have been necessary, as the default currency is in dollars. Furthermore, some constraints (e.g., NOT NULL), have been included within some domain definitions. For example, the "price" domain, which is used in all the fields where an amount must be specified, has been set as "NOT NULL" in the "CREATE DOMAINS" section. It is possible to verify from the "CREATE TABLE" section in the "Physical.tex" file how the choice of unique identifiers has been applied as much as possible (e.g., the VAT number is used for the customer, while the TAX number is used for the employee). In any other case, a self-increasing serial integer attribute has been used.

Physical Schema

```
-- Database Creation

CREATE DATABASE tagmsdb OWNER admin ENCODING = 'UTF8';

-- Create new Schema (you can use the public schema without creating a new one, if you want)

DROP SCHEMA IF EXISTS Tagms CASCADE;
CREATE SCHEMA Tagms;

-- Create new domains

CREATE DOMAIN Tagms.emailaddress AS VARCHAR(255)
CONSTRAINT properemail CHECK (VALUE ~ '^[A-Za-z0-9._%]+@[A-Za-z0-9.]+[.][A-Za-z]+$');

-- Checks for the italian number structure w/ or w/o prefix.
CREATE DOMAIN Tagms.phonenumber AS VARCHAR(17)
CONSTRAINT properphonenumber CHECK (VALUE ~ '^(\((00|\+)39\)|(00|\+)39)?3\d{9}$');

-- Checks for the italian vat number (11 digits).
CREATE DOMAIN Tagms.vatnumber AS CHAR(11)
CONSTRAINT properVAT CHECK (VALUE ~ '^[0-9]{11}$');

-- Checks for the italian tax number.
CREATE DOMAIN Tagms.taxnumber AS CHAR(16)
CONSTRAINT propertaxnumber CHECK (VALUE ~ '^[A-Z]{6}\d{2}[A-Z]\d{2}[A-Z]\d{3}[A-Z]$');

CREATE DOMAIN Tagms.price AS NUMERIC(9, 3)
CONSTRAINT properprice NOT NULL CHECK (VALUE > 0.0);
```

```
CREATE DOMAIN Tagms.quantity AS INTEGER
CONSTRAINT properquantity NOT NULL CHECK (VALUE >= 1);

CREATE DOMAIN Tagms.dimension AS INTEGER
CONSTRAINT properdimension NOT NULL CHECK (VALUE > 0);

CREATE DOMAIN Tagms.percentage AS DECIMAL(4,1)
CONSTRAINT properpercentage CHECK (VALUE >= 0 and VALUE <= 100);

CREATE DOMAIN Tagms.tracknum AS VARCHAR(30)
CONSTRAINT propertracknum CHECK(VALUE ~ '[A-Za-z0-9]');

-- Table Creation (create tables in the correct order, i.e. use FKeys only referring to already created tables

CREATE TABLE Tagms.product_category (
product_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.item_category (
item_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.package_category (
package_category_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.product (
product_id SERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
production_cost Tagms.price,
price_increase Numeric(3, 1) NOT NULL CHECK (price_increase >= 1),
volume Tagms.dimension,
net_weight Tagms.dimension,
package_weight Tagms.dimension,
nutritional_facts VARCHAR(500) NOT NULL,
product_category_id SMALLINT NOT NULL,
FOREIGN KEY (product_category_id) REFERENCES Tagms.product_category(product_category_id)
);

CREATE TABLE Tagms.item (
item_id SERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
quantity Tagms.quantity,
minimum_quantity Tagms.quantity,
item_category_id SMALLINT NOT NULL,
FOREIGN KEY (item_category_id) REFERENCES Tagms.item_category(item_category_id)
);

CREATE TABLE Tagms.package (
package_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500),
weight Tagms.dimension,
```

```
height Tagms.dimension,
width Tagms.dimension,
depth Tagms.dimension,
package_category_id SMALLINT NOT NULL,
FOREIGN KEY (package_category_id) REFERENCES Tagms.package_category(package_category_id)
);

CREATE TABLE Tagms.lot (
lot_id SERIAL PRIMARY KEY,
expiration_date DATE NOT NULL,
product_id INTEGER NOT NULL,
product_quantity Tagms.quantity,
package_id INTEGER NOT NULL,
package_quantity Tagms.quantity,
lot_discount Tagms.percentage  NOT NULL,
vat Tagms.percentage NOT NULL,
lot_price Tagms.price,
FOREIGN KEY (package_id) REFERENCES Tagms.package(package_id),
FOREIGN KEY (product_id) REFERENCES Tagms.product(product_id)
);

CREATE TABLE Tagms.made_up_of_1 (
product_id INTEGER,
item_id INTEGER,
quantity Tagms.quantity,
FOREIGN KEY (product_id) REFERENCES Tagms.product(product_id),
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
PRIMARY KEY (product_id, item_id)
);

CREATE TABLE Tagms.made_up_of_2 (
package_id INTEGER,
item_id INTEGER,
quantity Tagms.quantity,
FOREIGN KEY (package_id) REFERENCES Tagms.package(package_id),
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
PRIMARY KEY (package_id, item_id)
);

CREATE TABLE Tagms.customer (
vat_number Tagms.vatnumber PRIMARY KEY,
customer_name VARCHAR(50) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
address VARCHAR(100) NOT NULL,
email_address Tagms.emailaddress NOT NULL,
registration_date TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Tagms.role (
role_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.employee (
tax_number Tagms.taxnumber PRIMARY KEY,
first_name VARCHAR(30) NOT NULL,
last_name VARCHAR(30) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
email_address Tagms.emailaddress NOT NULL,
birth_date DATE,
```

```
hiring_date DATE NOT NULL,
still_working BOOLEAN NOT NULL,
role_id SMALLINT NOT NULL,
FOREIGN KEY (role_id) REFERENCES Tagms.role(role_id)
);

CREATE TABLE Tagms.department (
department_id SMALLSERIAL PRIMARY KEY,
name VARCHAR(30) NOT NULL,
description VARCHAR(500)
);

CREATE TABLE Tagms.work (
department_id SMALLINT,
employee_id Tagms.taxnumber,
FOREIGN KEY (department_id) REFERENCES Tagms.department(department_id),
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(tax_number),
PRIMARY KEY (department_id, employee_id)
);

CREATE TABLE Tagms.supplier (
vat_number Tagms.vatnumber PRIMARY KEY,
supplier_name VARCHAR(50) NOT NULL,
phone_number Tagms.phonenumber NOT NULL,
email_address Tagms.emailaddress NOT NULL,
address VARCHAR(100) NOT NULL,
registration_date TIMESTAMP WITH TIME ZONE NOT NULL
);

CREATE TABLE Tagms.contract (
contract_id SERIAL PRIMARY KEY,
description VARCHAR(500),
contract_date DATE NOT NULL DEFAULT CURRENT_DATE,
delivery_date DATE NOT NULL,
expiration_date DATE NOT NULL,
supplier_id Tagms.vatnumber NOT NULL,
employee_id Tagms.taxnumber NOT NULL,
FOREIGN KEY (supplier_id) REFERENCES Tagms.supplier(vat_number),
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(tax_number)
);

CREATE TABLE Tagms.specify (
item_id INTEGER,
contract_id INTEGER,
price Tagms.price,
purchased_quantity Tagms.quantity,
FOREIGN KEY (item_id) REFERENCES Tagms.item(item_id),
FOREIGN KEY (contract_id) REFERENCES Tagms.contract(contract_id),
PRIMARY KEY (item_id, contract_id)
);

CREATE TABLE Tagms.order (
order_id SERIAL PRIMARY KEY,
net_price Tagms.price,
taxes Tagms.price,
order_date TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP,
order_paid BOOLEAN NOT NULL,
employee_id Tagms.taxnumber NOT NULL,
customer_id Tagms.vatnumber NOT NULL,
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(tax_number),
FOREIGN KEY (customer_id) REFERENCES Tagms.customer(vat_number)
```

```
);

CREATE TABLE Tagms.draws_from (
lot_id INTEGER PRIMARY KEY,
order_id INTEGER NOT NULL,
FOREIGN KEY (lot_id) REFERENCES Tagms.lot(lot_id),
FOREIGN KEY (order_id) REFERENCES Tagms.order(order_id)
);

CREATE TABLE Tagms.ship (
order_id INTEGER PRIMARY KEY,
employee_id Tagms.taxnumber NOT NULL,
shipping_date TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT CURRENT_TIMESTAMP,
track_num Tagms.tracknum,
FOREIGN KEY (order_id) REFERENCES Tagms.order(order_id),
FOREIGN KEY (employee_id) REFERENCES Tagms.employee(tax_number)
);
```

Populate the Database: Example

Below are reported several examples of the INSERT statement. For the complete list, refer to the "db_population.sql" file.

```
-- Product_Category

INSERT INTO Tagms.product_Category(Name, Description) VALUES
    ('Beer','An alcoholic beverage typically obtained from the fermentation of wort based on barley
        malt'),
    ('Soft Drink','Soft Drink is a sparkling flavored drink'),
    ('Juice Pineapple','Juice Pineapple is a drink made from the extraction or pressing of the
        natural liquid contained in fruit and vegetables'),
    ('Smoothie','Smoothie is a drink made by fruit and vegetables in a blender');



-- Item_Category

INSERT INTO Tagms.item_Category(name, description) VALUES
    ('Soft Sugar','Soft sugar'),
    ('Sugar','Special sugar'),
    ('Coloring','Food coloring'),
    ('Water','Water'),
    ('Tape','Tape employed for packaging'),
    ('Polystyrene','Polystyrene employed for packaging'),
    ('Aluminum','Aluminum employed for packaging'),
    ('Plastic','Plastic employed for packaging'),
    ('Glass','Glass employed for packaging');



-- Package_Category

INSERT INTO Tagms.package_Category(Name, Description) VALUES
    ('Package XS','Extra small package'),
    ('Package S','Small package'),
    ('Package M','Medium package'),
    ('Package L','Large package'),
    ('Package XL','Extra large package');
```

```
-- Product
-- Same products can have different packages (and so different Package_weight) e.g., Beer Atomic in
    aluminum can and Beer Atomic in glass bottle.

INSERT INTO Tagms.product(Name, Description, Production_cost, Price_increase, Volume, Net_weight,
    Package_weight, Nutritional_Facts, Product_Category_ID)  VALUES
    ('Beer Atomic in aluminum can','An alcoholic beverage typically obtained from the fermentation
        of wort based on barley malt',
     '2.00', '2.0', '450', '450', '15', 'Calories: 153, Alcohol: 14 grams, Carbs: 13 grams, Protein:
         2 grams, Fat: 0 grams.', '1'),
    ('Beer Atomic in glass bottle','An alcoholic beverage typically obtained from the fermentation
        of wort based on barley malt',
     '2.00', '2.0', '450', '450', '50', 'Calories: 153, Alcohol: 14 grams, Carbs: 13 grams, Protein:
         2 grams, Fat: 0 grams.', '1'),
    ('Coke Mega in aluminum can','Coca Mega is a carbonated soft drink',
     '1.5', '3', '300', '300', '30', 'Calories: 180, Alcohol: 0 grams, Carbs: 38 grams, Protein: 10
         grams, Fat: 4 grams.', '2'),
    ('Coke Mega in glass bottle','Coca Mega is a carbonated soft drink',
     '1.5', '3', '300', '300', '52', 'Calories: 180, Alcohol: 0 grams, Carbs: 38 grams, Protein: 10
         grams, Fat: 4 grams.', '2'),
    ('Juice Pineapple Turbo','Juice Pineapple Turbo is a drink made from the extraction or pressing
        of the natural liquid contained in fruit and vegetables',
     '1.30', '3.5', '500', '520', '25', 'Calories: 9, Alcohol: 0 grams, Carbs: 2 grams, Protein: 32
         grams, Fat: 8 grams.', '3'),
    ('Fanta explosive','Fanta explosive is a fruit-flavored carbonated soft drinks',
     '1.5', '2.5', '450', '480', '13', 'Calories: 212, Alcohol: 0 grams, Carbs: 12 grams, Protein:
         35 grams, Fat: 120 grams.', '2'),
    ('Strawberry Smoothie','Strawberry Smoothie is a vegetable smoothie rich in vitamins',
     '3.60', '2.8', '300', '340', '28', 'Calories: 2, Alcohol: 0 grams, Carbs: 218 grams, Protein:
         23 grams, Fat: 8 grams.', '3'),
    ('Sprite Refreshing','Sprite Refreshing is a colorless, lemon, lime-flavored and refreshing soft
         drink',
     '2.15', '2.3', '400', '417', '20', 'Calories: 234, Alcohol: 0 grams, Carbs: 80 grams, Protein:
         21 grams, Fat: 48.', '2');



-- Item

INSERT INTO Tagms.item(name, description, quantity, minimum_quantity, item_category_id) VALUES
    ('Sugar C','Raw cane sugar is a less caloric and healthier sugar','3000000','100000','1'),
    ('Sugar W','White sugar for sweeter products','2500000','100000','1'),
    ('Sugar I','Brown sugar is rich in vitamins and minerals such as potassium, calcium and
        magnesium.','500000','200000','2'),
    ('Low mineral water','Low mineral content water employed for producing all products','800000','
        150000','4'),
    ('Coloring E100','Food yellow-orange coloring dye extracted from ground rhizomes of natural
        strains of Curcuma longa','1000000','700000','3'),
    ('Coloring E101','Food yellow natural coloring present in milk, it is also present in many green
         vegetables,','130000','35000','3'),
    ('Coloring E150','Food caramel coloring and can also have the function of amalgamating and
        protecting against oxidation caused by light','2000000','120000','3'),
    ('Tape P','Packing tape to ensure a good seal ','220000','75000','5'),
    ('Tape T','Transparent adhesive tape available in both silent and noisy unwinding','110000','
        50000','5'),
    ('Tape C','Canvas tape equipped with a dense weave that allows the orker to repair, bind and
        bead','300000','80000','5'),
```

```
    ('Polystyrene I','Polystyrene (type I) is an excellent technical insulator','80000','15000','6')
        ,
    ('Pallet 4','Pallet (type 4) is an excellent pellet that ensures great resistance against shocks
        ','80','6','6'),
    ('Pallet 5','Pallet (type 5) is an excellent pellet that ensures great resistance against shocks
        ','90','5','6'),
    ('Aluminum can','Aluminum can with internal coating (coating) which aims to minimize the
        interaction between the beverage and aluminum','5000','130','7'),
    ('Plastic Bottle 1','Plastic bottle (type 1) with heat-insulating','20000','1100','8'),
    ('Glass Bottle 1','Glass bottle (type 1) with heat-insulating','26000','150','9');



-- Package

INSERT INTO Tagms.package(Name, Description, Weight, Height, Width, Depth, Package_Category_ID)
    VALUES
    ('Package XS-1', 'Extra small package, type 1', '10', '3', '5', '2', '1'),
    ('Package XS-2', 'Extra small package, type 2', '13', '3', '6', '3', '1'),
    ('Package S-1', 'Small package, type 1', '13', '6', '6', '5', '2'),
    ('Package M-1', 'Medium package, type 1', '16', '8', '7', '6', '3'),
    ('Package L-1', 'Large package, type 1', '17', '10', '7', '7', '4'),
    ('Package L-2', 'Large package, type 2', '18', '12', '9', '7', '4'),
    ('Package L-3', 'Large package, type 3', '19', '10', '10', '7', '4'),
    ('Package XL-1', 'Extra large package, type 1', '25', '13', '11', '8', '5'),
    ('Package XL-2', 'Extra large package, type 2', '29', '14', '15', '10', '5');



-- Lot

-- 50 beers, 100 beers of 0.33L price 2 without discount
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
    ('2022-03-14','0.0','22.0', '200', '50', '5','1', '1'),
    ('2022-04-14','0.0','22.0', '400', '100', '3','1', '1');

-- 50 cokes, 100 cokes of 0.5L price 1.5 without discount
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
    ('2022-05-14','0.0','22', '225', '50', '2','3', '3'),
    ('2022-05-14','0.0','22', '450', '100', '5','3', '3');

-- 50 juices price 1.3, 100 juices price 3.5 with 10% discount
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
    ('2022-05-14','0.0','22.0', '227.5', '50', '4','5', '5'),
    ('2022-04-14','10.0','22.0', '455', '100', '4','5', '3');

-- 100 fanta explosive price 1.5 with 5% discount, 70 fanta explosive price 1.5 without discount,
    110 fanta explosive price 1.5 with 15% discount
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
    ('2022-11-19','5.0','22.0', '1008.00', '100', '3','5', '6'),
    ('2022-10-14','0.0','22.0', '165.00', '70', '6','3', '6'),
    ('2022-06-19','15.0','22.0', '1008.00', '110', '100','5', '6');

-- 30 smoothie of 0.5L price 3.6 with 10% discount (expired), 40 smoothie of 0.5L price 3.6 with 0%
    discount (expired)
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
```

```
    ('2017-06-19','10.0','22.0', '302.40', '30', '100','5', '7'),
    ('2018-11-24','0.0','22.0', '403.20', '30', '100','5', '7');

-- 30 sprite of 0.5L price 2.15 with 10% discount, 60 sprite of 0.5L price 2.15 with 0% discount (
    expired)
INSERT INTO Tagms.lot(Expiration_date,Lot_discount,VAT, Lot_price, Product_Quantity,
                      Package_Quantity,Package_ID, Product_ID) VALUES
    ('2022-06-10','10.0','22.0', '172.5', '30', '100','5', '8'),
    ('2020-08-17','0.0','22.0', '345', '60', '100','5', '8');



-- Made_Up_Of_1
-- For sake of simplicity, we inserted only few tuples in made_up_of 1 and 2.

-- Beer Atomic: contains low mineral water, sugar w, coloring e100
INSERT INTO Tagms.made_Up_Of_1(Product_ID, Item_ID,Quantity) VALUES
    ('1','4','330'),
    ('1','2','15'),
    ('1','5','3');

-- Coke Mega: contains sugar c, sugar w, low mineral water, coloring e150
INSERT INTO Tagms.made_Up_Of_1(Product_ID, Item_ID,Quantity) VALUES
    ('2','1','3'),
    ('2','2','3'),
    ('2','4','500'),
    ('2','7','4');

-- Juice: contains sugar c, sugar w, coloring e101, low mineral water
INSERT INTO Tagms.made_Up_Of_1(Product_ID, Item_ID,Quantity) VALUES
    ('3','1','2'),
    ('3','2','4'),
    ('3','6','2'),
    ('3','4','600');

-- Fanta: contains sugar c, coloring e100, low mineral water
INSERT INTO Tagms.made_Up_Of_1(Product_ID, Item_ID,Quantity) VALUES
    ('6','1','10'),
    ('6','5','20'),
    ('6','4','550');



-- Made_Up_Of_2
-- For sake of simplicity, we inserted only few tuples in made_up_of 1 and 2.

-- Aluminum Cans 0.33
INSERT INTO tagms.made_up_of_2(Package_ID,Item_ID,Quantity) VALUES
    ('3','14','20'),
    ('3','8','10');

-- Plastic bottle 0.5L
INSERT INTO tagms.made_up_of_2(Package_ID,Item_ID,Quantity) VALUES
    ('5','15','30'),
    ('5','9','15');

-- Glass bottle 1L
INSERT INTO tagms.made_up_of_2(Package_ID,Item_ID,Quantity) VALUES
    ('1','16','10'),
    ('1','10','10');
```

```
-- Customers

INSERT INTO Tagms.customer VALUES
    ('74853719034','Conad','+393895562233','Via Prato 25, Pordenone','conad_pordenone@mail.com',
        current_date),
    ('56983717634','Coop','+393827762376','Via America 30, Belluno','coop_belluno@mail.com',
        current_date),
    ('62347815297','Famila','+393336941253','Viale dei Mille 100, Parma','famila_parma@mail.com',
        current_date),
    ('12564378963','Eurospin','+393346901723','Via 20 Settembre 21, Lecce','eurospin_lecce@mail.com'
        ,current_date),
    ('12564378964','PAM','+393315589636','Via Giappone 45, Taranto','pam_taranto@mail.com',
        current_date);



-- Roles

INSERT INTO Tagms.role(Name,Description) VALUES
    ('Manager','The managers are in charge of interacting with suppliers and concluding contracts
        with them'),
    ('Worker','The workers take care of production and shipments'),
    ('Salesman','The sellers have to deal with customers and create orders'),
    ('Data Analyst','The data analyst has access to the acquired data for inventory, cost and profit
         analysis');



-- Employees

INSERT INTO Tagms.employee VALUES
    ('FGDVSF30C62D012T','John','Smith','3516235214','johnsmith@gmail.com','1995-12-30','2018-12-12',
        'true',1),
    ('BDBBHH72S22C841N','Adam','Willis','3516931849','adamwillis@gmail.com','1993-11-20','2019-11-11
        ','true',2),
    ('HGDVSF40C62D012T','Bruce','De Forest','3319856644','deforest@gmail.com','1992-11-20','
        2012-11-11','true',3),
    ('MGHRLQ80B49H779S','Lena','Choi','3358853696','deforest@gmail.com','1992-11-20','2014-05-23','
        true',3),
    ('QMUHFE38E25G815G','Ronald','Vickers','3352317485','rvickers@gmail.com','1998-07-11','
        2010-04-23','true',4),
    ('DKLERT23D96A316T','Gurver','Maata','00393403631287','gmaata@gmail.com','1953-06-24','
        2017-09-14','true',1),
    ('RNNZLB88S21B046D','Solomon','Ayala','00393666549821','sayala@gmail.com','1952-09-10','
        2016-10-22','true',1);



-- Departments

INSERT INTO Tagms.department(Name, Description) VALUES
    ('Operations','Responsible for converting raw materials and packaging materials into finished
        goods and work to improve the efficiency of the production.'),
    ('Sales','Responsible for the sale of lots.'),
    ('Administration','Responsible for the administration of the accounting of employment contracts
        and management of personnel, customers and suppliers.'),
    ('Maintenance','Department responsible for the management and maintenance of the different
        machinery used for production.'),
    ('Shipping','Responsible for marking and shipping lots.');
```

```
-- Work
-- Operations: 2 employees Sales: 2 employees Administration: 1 employee Maintenance: 1 employee
    Shipping: 2 employees.

INSERT INTO Tagms.work(Department_ID, Employee_ID) VALUES
    ('1','FGDVSF30C62D012T'),
    ('1','BDBBHH72S22C841N'),
    ('2','HGDVSF40C62D012T'),
    ('2','RNNZLB88S21B046D'),
    ('3','DKLERT23D96A316T'),
    ('2','MGHRLQ80B49H779S'),
    ('4','QMUHFE38E25G815G'),
    ('5','QMUHFE38E25G815G'),
    ('5','RNNZLB88S21B046D');



-- Suppliers

INSERT INTO Tagms.supplier(vat_number, supplier_name, phone_number, email_address, address,
    registration_date) VALUES
    ('86334519757','Reg s.r.l.','3794567845','regsrl@gmail.com','Via Roma 4 35100 Padova','
        2021-12-04'),
    ('23932902833','VisumH2O','3396547835','vish2o@gmail.com','Via Trieste 4 31100 Treviso','
        2021-10-07'),
    ('23232503889','EnergyFeed s.p.a.','3365447309','feed4life@gmail.com','Via Venezia 7 19100
        Spezia','2021-04-08'),
    ('22134054695','ReSugar','3366244789','sugar@gmail.com','Via Marsalis 1 81290 Firenze','
        2021-01-02'),
    ('45288712635','MaterialPCK','3366244789','matforpack@gmail.com','Via Resto 1 23202 Rovigo','
        2021-02-011');



-- Contracts

INSERT INTO tagms.contract (description,contract_date,delivery_date,expiration_date,supplier_id,
    employee_id) VALUES
    ('contract with Reg s.r.l.','2020-01-18','2021-12-06','2023-12-31','86334519757','
        FGDVSF30C62D012T'),
    ('contract with Reg s.r.l.','2021-02-01','2021-12-08','2024-12-31','86334519757','
        DKLERT23D96A316T'),
    ('contract with VisumH2O','2019-01-25','2021-12-07','2022-12-31','23932902833','FGDVSF30C62D012T
        '),
    ('contract with ReSugar','2017-06-11','2021-12-21','2025-04-01','22134054695','RNNZLB88S21B046D'
        ),
    ('contract with MaterialPCK','2021-01-17','2021-12-01','2023-05-22','45288712635','
        RNNZLB88S21B046D'),
    ('contract with EnergyFeed s.p.a.','2018-02-07','2021-12-05','2022-03-14','45288712635','
        DKLERT23D96A316T'),
    ('contract with Reg s.r.l.','2020-03-12','2021-11-06','2023-10-31','86334519757','
        FGDVSF30C62D012T'),
    ('contract with Reg s.r.l.','2020-04-17','2021-10-06','2023-12-30','86334519757','
        FGDVSF30C62D012T');



-- Specify
```

```
-- Attributes order: item_id, contract_id, price, purchased_quantity.

INSERT INTO tagms.specify VALUES
    (1,4,10200,7500),
    (2,4,12200,1000),
    (3,4,10231,5000),
    (8,5,331,400),
    (9,5,671,222),
    (10,5,349,380),
    (11,5,293,200);



-- Orders

INSERT INTO Tagms.order(net_price, taxes, order_date, order_paid, employee_id, customer_id) VALUES
    (877.5,50,'2020-04-25','true','HGDVSF40C62D012T','56983717634'),
    (225,75,'2021-12-01','true','HGDVSF40C62D012T','12564378963'),
    (400,20,'2021-04-25','true','MGHRLQ80B49H779S','56983717634'),
    (455,40,current_date,'true','MGHRLQ80B49H779S','74853719034'),
    (1008,200,current_date,'false','MGHRLQ80B49H779S','62347815297');



-- Draws_from

INSERT INTO Tagms.draws_from(Lot_ID, Order_ID) VALUES
    ('1','1'),
    ('3','2'),
    ('2','3'),
    ('4','1'),
    ('5','1'),
    ('6','4'),
    ('7','5');



-- Ships

INSERT INTO Tagms.ship(Order_ID, Employee_ID, Shipping_date,Track_num) VALUES
    (1,'BDBBHH72S22C841N','2021-04-27','LM00392619SIT'),
    (2,'BDBBHH72S22C841N','2021-12-03','SL69065769SIT'),
    (3,'BDBBHH72S22C841N','2021-04-27','RT463451577IT');
```

Principal Queries

Hereafter are listed the principal queries:

1. List all contracts between a specific manager and a specific supplier.
2. Show all the details of the employees in the organization
3. Decrease the quantity of the items involved in the production of a lot (After inserting a new lot with identifier Lot_id)
4. Increment the quantities of items in stock (after inserting a new contract with identifier Contract_id, and in the delivery date)

5. Lists all available lots (unsold and that won't expire in 6 months) containing a particular product having a given Product_id as identifier, sorted by expiration date (oldest lots must be sold first).
6. Get the net sales and paid taxes in a given time interval; Get the cost of materials in a given time interval; Get the production cost in a given time interval
7. Given an order with Order_id, compute the order's net_price and total taxes, and update the order.
8. Given a lot with Lot_id, compute the lot price, and update the lot.
9. Given an item (which is used to create a product) having Item_id as identifier and a time interval (actually, two dates), find the total quantity of that item that has been used for production or packaging during that time.
10. Return the list of unsellable lots which are in stock (that will expire in less than 6 months)
11. For each year, return the number of lots sold, the net sales and the added taxes paid by the customers

```
-- List all contracts between a specific manager and a specific supplier.

SELECT c.contract_id,
       c.description,
       c.contract_date AS contract_date,
       c.expiration_date AS expiration_date,
       e.first_name AS manager_name,
       e.last_name AS manager_surname
                FROM tagms.contract AS c
                    INNER JOIN tagms.employee AS e ON c.employee_id = e.tax_number
                    INNER JOIN tagms.supplier AS s ON c.supplier_id = s.vat_number
WHERE e.tax_number='FGDVSF30C62D012T'
  AND s.supplier_name='Reg s.r.l.';
```

	contract_id integer	description character varying (500)	contract_date date	expiration_date date	manager_name character varying (30)	manager_surname character varying (30)
1	1	contract with Reg s.r.l.	2020-01-18	2023-12-31	John	Smith
2	7	contract with Reg s.r.l.	2020-03-12	2023-10-31	John	Smith
3	8	contract with Reg s.r.l.	2020-04-17	2023-12-30	John	Smith

Figure 1: Output of query #1.

```
-- Show all the details of the employees in the organization

SELECT e.tax_number,
       e.first_name,
       e.last_name,
       e.phone_number,
       e.email_address,
       e.birth_date,
       e.hiring_date,
       r.name,
       d.name
FROM tagms.employee AS e
    INNER JOIN tagms.work AS w ON e.tax_number = w.employee_id
```

```
    INNER JOIN tagms.department AS d ON w.department_id = d.department_id
    INNER JOIN tagms.role AS r ON r.role_id = e.role_id
WHERE e.still_working = TRUE;
```

	tax_number character (16)	first_name character varying (30)	last_name character varying (30)	phone_number character varying (17)	email_address character varying (255)	birth_date date	hiring_date date	name character varying (30)	name character varying (30)
1	FGDVSF30C62D012T	John	Smith	3516235214	johnsmith@gmail.com	1995-12-30	2018-12-12	Manager	Operations
2	BDBBHH72S22C841N	Adam	Willis	3516931849	adamwillis@gmail.com	1993-11-20	2019-11-11	Worker	Operations
3	HGDVSF40C62D012T	Bruce	De Forest	3319856644	deforest@gmail.com	1992-11-20	2012-11-11	Salesman	Sales
4	RNNZLB88S21B046D	Solomon	Ayala	00393666549821	sayala@gmail.com	1952-09-10	2016-10-22	Manager	Sales
5	DKLERT23D96A316T	Gurver	Maata	00393403631287	gmaata@gmail.com	1953-06-24	2017-09-14	Manager	Administration
6	MGHRLQ80B49H779S	Lena	Choi	3358853696	deforest@gmail.com	1992-11-20	2014-05-23	Salesman	Sales
7	QMUHFE38E25G815G	Ronald	Vickers	3352317485	rvickers@gmail.com	1998-07-11	2010-04-23	Data Analyst	Maintenance
8	QMUHFE38E25G815G	Ronald	Vickers	3352317485	rvickers@gmail.com	1998-07-11	2010-04-23	Data Analyst	Shipping
9	RNNZLB88S21B046D	Solomon	Ayala	00393666549821	sayala@gmail.com	1952-09-10	2016-10-22	Manager	Shipping

Figure 2: Output of query #2.

```
-- After inserting a new lot with identifier Lot_id (see the "populate" section)
-- decrease the quantity of the items involved in the production of a lot
-- Note: it is necessary to insert the Lot_id twice in the query

UPDATE tagms.item AS i SET
    quantity = c.quantity
FROM (
        SELECT i.item_id, i.quantity - l.product_quantity * m1.quantity AS quantity FROM tagms.lot
            AS l
            INNER JOIN tagms.made_up_of_1 AS m1 ON l.product_id = m1.product_id
            INNER JOIN tagms.item AS i ON m1.item_id = i.item_id
        WHERE l.lot_id = '3'
    UNION
        SELECT i.item_id, i.quantity - l.package_quantity * m2.quantity AS quantity FROM tagms.lot
            AS l
            INNER JOIN tagms.made_up_of_2 AS m2 ON l.package_id = m2.package_id
            INNER JOIN tagms.item AS i ON m2.item_id = i.item_id
        WHERE l.lot_id = '3'
     )
    AS c(item_id, quantity)
WHERE c.item_id = i.item_id
RETURNING i.item_id, name, i.quantity, minimum_quantity;
```

	item_id [PK] integer	name character varying (30)	quantity integer	minimum_quantity integer
1	1	Sugar C	2999900	100000
2	2	Sugar W	2499800	100000
3	4	Low mineral water	770000	150000
4	6	Coloring E101	129900	35000
5	8	Tape P	219980	75000
6	14	Aluminum can	4960	130

Figure 3: Output of query #3.

```
-- After inserting a new contract with identifier Contract_id,
-- in the delivery date the quantities of items in stock must be incremented

UPDATE tagms.item AS i SET
    quantity = c.quantity
FROM (
    SELECT i.item_id, i.quantity + s.purchased_quantity AS quantity FROM tagms.contract AS c
        INNER JOIN tagms.specify AS s ON c.contract_id = s.contract_id
        INNER JOIN tagms.item AS i ON s.item_id = i.item_id
    WHERE c.contract_id = '4'
     )
     AS c(item_id, quantity)
WHERE c.item_id = i.item_id
RETURNING i.item_id, name, description, i.quantity, minimum_quantity, item_category_id;
```

	item_id [PK] integer	name character varying (30)	description character varying (500)	quantity integer	minimum_quantity integer	item_category_id smallint
1	3	Sugar I	Brown sugar is rich in vitamins and minerals such as potassium, calcium and magnesium.	505000	200000	2
2	1	Sugar C	Raw cane sugar is a less caloric and healthier sugar	3007400	100000	1
3	2	Sugar W	White sugar for sweeter products	2500800	100000	1

Figure 4: Output of query #4.

```
-- Lists all available lots (unsold and that won't expire in 6 months) containing
-- a particular product having a given Product_id as identifier,
-- sorted by expiration date (oldest lots must be sold first).

SELECT l.lot_id,
       l.expiration_date AS expiration_date,
       l.product_id,
       l.product_quantity,
       ROUND(l.lot_price * (1 + l.vat/100) * (1 - l.lot_discount/100), 2) AS gross_price
FROM tagms.lot AS l
    LEFT OUTER JOIN tagms.draws_from AS df ON l.lot_id = df.lot_id
WHERE l.product_id = '6'
  AND l.expiration_date > (current_date + interval '6 months')
  AND df.order_id IS NULL
ORDER BY l.expiration_date ASC;
```

	lot_id [PK] integer	expiration_date date	product_id integer	product_quantity integer	gross_price numeric
1	9	2022-06-19	6	110	1045.30
2	8	2022-10-14	6	70	201.30

Figure 5: Output of query #5.

```
-- Get the net sales and paid taxes in a given time interval
SELECT SUM(o.net_price) AS net_sales, SUM(o.taxes) AS taxes FROM tagms.order AS o
    WHERE DATE(o.order_date) >= '2021-01-01' AND
          DATE(o.order_date) <= '2021-12-31' AND
          o.order_paid = TRUE;

-- Get the cost of materials in a given time interval
SELECT SUM(sp.price) AS cost_of_material FROM tagms.specify as sp
    INNER JOIN tagms.contract AS c ON c.contract_id = sp.contract_id
WHERE c.contract_date >= '2021-01-01' AND c.contract_date <= '2021-12-31';

-- Get the production cost in a given time interval
SELECT SUM(p.production_cost * l.product_quantity) AS production_cost FROM tagms.lot AS l
    JOIN tagms.product AS p ON l.product_id = p.product_id
    JOIN tagms.draws_from AS df ON l.lot_id = df.lot_id
    JOIN tagms.order AS o ON df.order_id = o.order_id
WHERE DATE(o.order_date) >= '2021-01-01' AND DATE(o.order_date) <= '2021-12-31';
```

	net_sales numeric	taxes numeric
1	1080.000	135.000

(a) Output of query #6a.

	cost_of_material numeric
1	1644.000

(b) Output of query #6b.

	production_cost numeric
1	575.000

(c) Output of query #6c.

Figure 6

```
-- When inserting an order, the operator can choose a custom price (e.g., decided with the customer)
-- or use the following query.
-- Given an order with Order_id, compute the order's net_price and total taxes.
-- Note: the given Order_id must be written twice in the query

UPDATE tagms.order AS o
SET net_price = tmp.net_price,
    taxes = tmp.taxes
FROM (
         SELECT SUM(l.lot_price * (1 - l.lot_discount / 100)) AS net_price,
                SUM(l.lot_price * l.VAT/100) as taxes
         FROM tagms.draws_from AS df
                  INNER JOIN tagms.lot AS l ON df.lot_id = l.lot_id
        WHERE df.order_id = '1'
    ) AS tmp(net_price, taxes)
WHERE o.order_id = '1'
RETURNING o.order_id, o.net_price, o.taxes;
```

	order_id [PK] integer	net_price numeric (9,3)	taxes numeric (9,3)
1	1	877.500	193.050

Figure 7: Output of query #7.

```
-- When inserting a lot, the operator can choose a custom price
-- or use the following query.
-- Given a lot with Lot_id, compute the lot price
-- Note: the given Lot_id must be written twice in the query

UPDATE tagms.lot AS l
SET lot_price = tmp.lot_price
FROM (
         SELECT p.production_cost*p.price_increase*l.product_quantity
             AS lot_price FROM tagms.lot AS l
            INNER JOIN tagms.product AS p ON l.product_id = p.product_id
         WHERE l.lot_id = '3'
     ) AS tmp(lot_price)
WHERE l.lot_id = '3'
RETURNING *;
```

	lot_id [PK] integer	expiration_date date	product_id integer	product_quantity integer	package_id integer	package_quantity integer	lot_discount numeric (4,1)	vat numeric (4,1)	lot_price numeric (9,3)	lot_price numeric
1	3	2022-05-14	3	50	3	2	0.0	22.0	225.000	225.0000

Figure 8: Output of query #8.

```
-- Given an item (which is used to create a product) having Item_id as identifier
-- and a time interval (actually, two dates),
-- find the total quantity of that item that has been used for production or packaging during that
    time.

SELECT SUM(l.product_quantity * m1.quantity) AS quantity FROM tagms.made_up_of_1 AS m1
    INNER JOIN tagms.lot AS l ON m1.product_id = l.product_id
    INNER JOIN tagms.draws_from AS df ON l.lot_id = df.lot_id
    INNER JOIN tagms.order AS o ON df.order_id = o.order_id
WHERE m1.item_id = '1'
  AND DATE(o.order_date) >= '2021-01-01'
  AND DATE(o.order_date) <= '2021-12-31';
```

	quantity bigint
1	1300

Figure 9: Output of query #9.

```
-- Return the list of unsellable lots which are in stock (that will expire in less than 6 months)

SELECT
       l.lot_id,
       l.expiration_date AS expiration_date,
       l.product_id,
       l.product_quantity,
       l.package_id,
       l.package_quantity,
       l.lot_discount,
       l.vat,
       l.lot_price
       FROM tagms.lot AS l
    LEFT OUTER JOIN tagms.draws_from AS df ON l.lot_id = df.lot_id
WHERE l.expiration_date <= (current_date + interval '6 months')
  AND df.order_id IS NULL;


-- For each year, return the number of lots sold, the net sales and the added taxes paid by the
    customers

SELECT
    EXTRACT(year FROM o.order_date) AS year,
    COUNT(*) AS orders,
    SUM(o.net_price) AS net_sales,
    SUM(o.taxes) AS taxes
FROM tagms.order AS o
WHERE o.order_paid = TRUE
GROUP BY year
```

	lot_id [PK] integer	expiration_date date	product_id integer	product_quantity integer	package_id integer	package_quantity integer	lot_discount numeric (4,1)	vat numeric (4,1)	lot_price numeric (9,3)
1	11	2018-11-24	7	30	5	100	0.0	22.0	403.200
2	12	2022-06-10	8	30	5	100	10.0	22.0	172.500
3	10	2017-06-19	7	30	5	100	10.0	22.0	302.400
4	13	2020-08-17	8	60	5	100	0.0	22.0	345.000

Figure 10: Output of query #10.

```
    HAVING SUM(o.net_price) > '100';
```

	year double precision	orders bigint	net_sales numeric	taxes numeric
1	2020	1	877.500	193.050
2	2021	3	1080.000	135.000

Figure 11: Output of query #11.

JDBC Implementations of the Principal Queries and Visualization

In the following are listed two examples of java classes that execute two different INSERT operations.

```
//In the first example, the java class lists all available lots (unsold and that won't expire in 6
    months) containing a particular product having a given Product_id as identifier, sorted by
    expiration date (oldest lots must be sold first).
package com.example.tagmswebapp;

import java.sql.*;

public class ListAvailableLots {

    /* *
     * The JDBC driver to be used
     */
    private static final String DRIVER = "org.postgresql.Driver";
    /* *
     * The URL of the database to be accessed
     */
    private static final String DATABASE ="jdbc:postgresql://localhost:5432/tagmsdb";
    /* *
     * The username for accessing the database
     */
    private static final String USER = "nopass";
    /* *
     * The password for accessing the database
     */
    private static final String PASSWORD = "";

    /* *
```

```
 * Lists all available lots (unsold and that won't expire in 6 months) containing
 * a particular product having a given Product_id as identifier,
 * sorted by expiration date (oldest lots must be sold first).
 *
 * @param args command - line arguments ( not used ).
 */
public static void main(String[] args) {

    // the connection to the DBMS
    Connection con = null;
    // the statement to be executed
    Statement stmt = null;
    // the results of the statements execution
    ResultSet rs = null;
    // start time of a statement
    long start;
    // end time of a statement
    long end;

    // "data structures" for the data to be read from the database
    try {
        // register the JDBC driver
        Class.forName ( DRIVER );
        System.out.printf ("Driver %s successfully registered .%n", DRIVER );
    }  catch ( ClassNotFoundException e ) {
        System.out.printf(
                "Driver %s not found : %s. %n",DRIVER, e.getMessage());
        // terminate with a generic error code
        System.exit(-1);
    }

    try {
        // connect to the database
        start = System.currentTimeMillis ();
        con = DriverManager.getConnection (DATABASE , USER , PASSWORD);
        end = System.currentTimeMillis ();
        System.out.printf (
                "Connection to database %s successfully established in %d milliseconds .%n",
                DATABASE , end - start );
        // create the statement to execute the query
        start = System.currentTimeMillis();
        stmt = con.createStatement ();
        end = System.currentTimeMillis ();
        System.out.printf (
                "Statement successfully created in %d milliseconds .%n",
                end - start);

        start = System.currentTimeMillis();



        // Note: the following variable must be properly initialized

        int product_id = 6;

        String sql ="SELECT l.lot_id,\n" +
                "       l.expiration_date AS expiration_date,\n" +
                "       l.product_id,\n" +
                "       l.product_quantity,\n" +
                "       ROUND(l.lot_price * (1 + l.vat/100) * (1 - l.lot_discount/100), 2) AS
                    gross_price\n" +
```

```
            "FROM tagms.lot AS l\n" +
            "    LEFT OUTER JOIN tagms.draws_from AS df ON l.lot_id = df.lot_id\n" +
            "WHERE l.product_id = '" + product_id + "'\n" +
            "  AND l.expiration_date > (current_date + interval '6 months')\n" +
            "  AND df.order_id IS NULL\n" +
            "ORDER BY l.expiration_date ASC;";

    rs = stmt.executeQuery(sql);

    int lot_id;
    Date expiration_date;
    int product_quantity;
    float gross_price;

    System.out.println(
            pad("LOT_ID", 15) +
                    pad("PRODUCT_ID", 15) +
                    pad("EXPIRATION_DATE", 15) +
                    pad("PRODUCT_QUANTITY", 15) +
                    pad("GROSS_PRICE", 15)
    );

    while ( rs.next()) {

        lot_id = rs.getInt("lot_id");  // read item id
        product_id = rs.getInt("product_id"); // read product id
        product_quantity = rs.getInt("product_quantity"); // read product quantity
        gross_price=rs.getFloat("gross_price"); // read gross price
        expiration_date=rs.getDate("expiration_date"); // read expiration date

        System.out.println(
                pad(String.valueOf(lot_id), 15) +
                        pad(String.valueOf(lot_id), 15)  +
                        pad(String.valueOf(expiration_date), 15) +
                        pad(String.valueOf(product_quantity), 15) +
                        pad(String.valueOf(gross_price), 15)
        );
    }

    rs.close ();
    stmt.close ();
    con.close ();
    end = System.currentTimeMillis ();
    System.out.printf ("%n Data correctly extracted and visualized in %d milliseconds .%n",
            end - start );
} catch (SQLException e ) {
    System.out.printf ("Database access error :%n");
    // cycle in the exception chain
    while ( e != null ) {
        System.out.printf ("- Message : %s%n", e.getMessage ());
        System.out.printf ("- SQL status code : %s%n", e . getSQLState ());
        System.out.printf ("- SQL error code : %s%n", e . getErrorCode ());
        System.out.printf ("%n");
        e = e.getNextException ();
    }

} finally {
    try {
        // close the used resources
        if (!rs.isClosed()){
            start = System.currentTimeMillis();
```

```
                    rs.close();
                    end = System.currentTimeMillis();
                    System.out.printf(" Result set successfully closed in finally block in %d
                        milliseconds .%n",
                            end - start );
                }
                if (!stmt.isClosed ()) {
                    start = System.currentTimeMillis();
                    stmt.close();
                    end = System.currentTimeMillis();
                    System.out.printf(" Statement successfully closed " +
                                    "in finally block in %d milliseconds .%n",
                            end - start );
                }
                if (! con.isClosed ()) {
                    start = System.currentTimeMillis();
                    con.close();
                    end = System.currentTimeMillis();
                    System.out.printf(" Connection successfully closed in finally block in %d
                        milliseconds .%n",
                            end - start );
                }
            } catch (SQLException e) {
                System.out.printf (" Error while releasing resources in finally block :%n");
                // cycle in the exception chain
                while ( e != null ) {
                    System.out.printf(" - Message : %s %n",e.getMessage());
                    System.out.printf(" - SQL status code : %s %n",e.getSQLState());
                    System.out.printf(" - SQL error code : %s %n",e.getErrorCode());
                    System.out.printf(" %n");
                    e = e.getNextException();
                }
            } finally {
                // release resources to the garbage collector
                rs = null ;
                stmt = null ;
                con = null ;
                System.out.printf (" Resources released to the garbage collector .%n");
            }
        }
        System.out.printf (" Program end .%n");
    }

    private static String pad(String text, int length) {
        return String.format("%-" + length + "." + length + "s", text);
    }

}
```

```
//In the second example the java class, after inserting a new lot with identifier Lot_id (see the "
    populate" section), decrease the quantity of the items involved in the production of a lot.
package com.example.tagmswebapp;

import java.sql.*;

public class DecrementQuantityItem {

    /* *
```

```
 * The JDBC driver to be used
 */
private static final String DRIVER = "org.postgresql.Driver";
/* *
 * The URL of the database to be accessed
 */
private static final String DATABASE ="jdbc:postgresql://localhost:5432/tagmsdb";
/* *
 * The username for accessing the database
 */
private static final String USER = "nopass";
/* *
 * The password for accessing the database
 */
private static final String PASSWORD = "";
/* *
 * After inserting a new lot with identifier Lot_id (see the "populate" section)
 * decrease the quantity of the items involved in the production of a lot
 *
 * @param args command - line arguments ( not used ).
 */
public static void main(String[] args) {

    // the connection to the DBMS
    Connection con = null;
    // the statement to be executed
    Statement stmt = null;
    // the results of the statements execution
    ResultSet rs = null;
    // start time of a statement
    long start;
    // end time of a statement
    long end;

    // "data structures" for the data to be read from the database
    try {
        // register the JDBC driver
        Class.forName ( DRIVER );
        System.out.printf ("Driver %s successfully registered .%n", DRIVER );
    }  catch ( ClassNotFoundException e ) {
        System.out.printf(
                "Driver %s not found : %s. %n",DRIVER, e.getMessage());
        // terminate with a generic error code
        System.exit(-1);
    }

    try {
        // connect to the database
        start = System.currentTimeMillis ();
        con = DriverManager.getConnection (DATABASE , USER , PASSWORD);
        end = System.currentTimeMillis ();
        System.out.printf (
                "Connection to database %s successfully established in %d milliseconds .%n",
                DATABASE , end - start );
        // create the statement to execute the query
        start = System.currentTimeMillis();
        stmt = con.createStatement ();
        end = System.currentTimeMillis ();
        System.out.printf (
                "Statement successfully created in %d milliseconds .%n",
                end - start);
```

```
start = System.currentTimeMillis();


// NOTE: it is necessary to properly initialize the following variable

int lot_id = 3;

final String STATEMENT = "UPDATE tagms.item AS i SET\n" +
        "    quantity = c.quantity\n" +
        "FROM (\n" +
        "        SELECT i.item_id, i.quantity - l.product_quantity * m1.quantity AS
            quantity FROM tagms.lot AS l\n" +
        "            INNER JOIN tagms.made_up_of_1 AS m1 ON l.product_id = m1.product_id
            \n" +
        "            INNER JOIN tagms.item AS i ON m1.item_id = i.item_id\n" +
        "        WHERE l.lot_id = '" + lot_id + "'\n" +
        "    UNION\n" +
        "        SELECT i.item_id, i.quantity - l.package_quantity * m2.quantity AS
            quantity FROM tagms.lot AS l\n" +
        "            INNER JOIN tagms.made_up_of_2 AS m2 ON l.package_id = m2.package_id
            \n" +
        "            INNER JOIN tagms.item AS i ON m2.item_id = i.item_id\n" +
        "        WHERE l.lot_id = '" + lot_id + "'\n" +
        "    )\n" +
        "    AS c(item_id, quantity)\n" +
        "WHERE c.item_id = i.item_id\n" +
        "RETURNING i.item_id, name, i.quantity, minimum_quantity;";

rs = stmt.executeQuery(STATEMENT);

int item_id;
String name;
int quantity;
int minimum_quantity;


System.out.println(
        pad("ITEM_ID", 15) +
                pad("NAME", 35) +
                pad("QUANTITY", 15) +
                pad("MINIMUM_QUANTITY", 20)
);


// cycle on the query results ( i . e . for each employee we will select the events to
    be printed )
while ( rs.next()) {

    // read item id
    item_id = rs.getInt("item_id");

    // read item name
    name = rs.getString("name");

    // read item quantities
    quantity = rs.getInt("quantity");
    minimum_quantity = rs.getInt("minimum_quantity");


    System.out.println(
```

```
                pad(String.valueOf(item_id), 15) +
                        pad(name, 35) +
                        pad(String.valueOf(quantity), 15) +
                        pad(String.valueOf(minimum_quantity), 20)
            );
        }


        rs.close();
        stmt.close();
        con.close();
        end = System.currentTimeMillis();
        System.out.printf ("%n Data correctly extracted and visualized in %d milliseconds .%n",
                end - start );
    } catch (SQLException e ) {
        System.out.printf ("Database access error :%n");
        // cycle in the exception chain
        while ( e != null ) {
            System.out.printf ("- Message : %s%n", e.getMessage ());
            System.out.printf ("- SQL status code : %s%n", e . getSQLState ());
            System.out.printf ("- SQL error code : %s%n", e . getErrorCode ());
            System.out.printf ("%n");
            e = e.getNextException ();
        }

    } finally {
        try {
            // close the used resources
            if (!rs.isClosed()){
                start = System.currentTimeMillis();
                rs.close();
                end = System.currentTimeMillis();
                System.out.printf(" Result set successfully closed in finally block in %d
                    milliseconds .%n",
                        end - start );
            }
            if (!stmt.isClosed()) {
                start = System.currentTimeMillis();
                stmt.close();
                end = System.currentTimeMillis();
                System.out.printf(" Statement successfully closed " +
                                "in finally block in %d milliseconds .%n",
                        end - start );
            }
            if (! con.isClosed()) {
                start = System.currentTimeMillis();
                con.close();
                end = System.currentTimeMillis();
                System.out.printf(" Connection successfully closed in finally block in %d
                    milliseconds .%n",
                        end - start );
            }
        } catch (SQLException e) {
            System.out.printf (" Error while releasing resources in finally block :%n");
            // cycle in the exception chain
            while ( e != null ) {
                System.out.printf(" - Message : %s %n",e.getMessage());
                System.out.printf(" - SQL status code : %s %n",e.getSQLState());
                System.out.printf(" - SQL error code : %s %n",e.getErrorCode());
                System.out.printf(" %n");
                e = e.getNextException();
```

```
                }
            } finally {
                // release resources to the garbage collector
                rs = null ;
                stmt = null ;
                con = null ;
                System.out.printf (" Resources released to the garbage collector .%n");
            }
        }
        System.out.printf (" Program end .%n");
    }

    private static String pad(String text, int length) {
        return String.format("%-" + length + "." + length + "s", text);
    }

}
```

Group Members Contributions

- **Variations to the Relational Schema:** Esposito, Basso
- **Physical Schema:** Giuliani, Collado, Esposito, Basso
- **Populate the Database: Example:** Zanini, Collado, Esposito, Basso
- **Principal Queries:** Giuliani, Collado, Esposito, Basso
- **JDBC Implementations of the Principal Queries and Visualization:** Zanini, Collado, Esposito, Basso