**Semester 3: QAP 2 – Databases**

1. How many hours did it take you to complete this assessment? (Please keep try to keep track of how many hours you have spent working on each individual part of this assessment as best you can - an estimation is fine; we just want a rough idea.)

Problem 1 (5 Exercises) – 5 hours

Problem 2 – 1 hour

2. What online resources you have used? (My lectures, YouTube, Stack overflow etc.)

* <https://www.datensen.com/blog/er-diagram/many-to-many-relationships/>
* https://blog.devart.com/how-to-create-many-to-many-relationships-between-tables.html
* W3 Schools
* Lectures/Powerpoints
* Oracle
* <https://vertabelo.com/blog/many-to-many-relationship/>

3. Did you need to ask any of your friends in solving the problems. (If yes, please mention name of the friend. They must be amongst your class fellows.)

- No

4. Did you need to ask questions to any of your instructors? If so, how many questions did you ask (or how many help sessions did you require)?

- No

5. Rate (subjectively) the difficulty of each question from your own perspective, and whether you feel confident that you can solve a similar but different problem requiring some of the same techniques in the future now that you’ve completed this one.

- Problem 1: Medium – Hard, easier now.

- Problem 2: Easy

**Exercise 1:**

**SQL for Exercise 1:**

CREATE DATABASE exercise1

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

BEGIN;

CREATE TABLE IF NOT EXISTS public.city (

city\_id serial NOT NULL,

city\_name varchar NOT NULL,

prov\_id integer NOT NULL,

CONSTRAINT city\_pkey PRIMARY KEY (city\_id),

CONSTRAINT prov\_id FOREIGN KEY (prov\_id) REFERENCES public.province (prov\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public.province (

prov\_id serial NOT NULL,

prov\_name varchar NOT NULL,

CONSTRAINT province\_pkey PRIMARY KEY (prov\_id));

END;

First table is the province table, that has the primary key serial integer prov\_id, the char attribute prov\_name, but all attributes cannot have null values. The purpose of the table is to identify the province region of each city in address/location queries.

Second table is the city table, that has the primary key serial integer city\_id, the char attribute city\_name, but all attributes cannot have null values. This table also includes the foreign key prov\_id, from the province table. The purpose of the table is to identify the city location in each province in address/location queries.

The relationship between the parent table province, and the child table city is 1 to Many. In a one-to-many relationship, one record in a table can be associated with one or more records in another table. There may be many cities in each province, however, there is only one province per city.

A screenshot of a computer

Description automatically generated

**Exercise 2:**

**SQL for Exercise 2:**

CREATE DATABASE "Exercise2"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

BEGIN;

CREATE TABLE IF NOT EXISTS public.artwork (

art\_id SERIAL NOT NULL,

art\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

gallery\_id INTEGER NOT NULL,

CONSTRAINT art\_id PRIMARY KEY (art\_id),

CONSTRAINT gallery\_id FOREIGN KEY (gallery\_id) REFERENCES public.gallery (gallery\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public.automobile (

auto\_id SERIAL NOT NULL,

auto\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

dmv\_id INTEGER NOT NULL,

CONSTRAINT auto\_id PRIMARY KEY (auto\_id),

CONSTRAINT dmv\_id FOREIGN KEY (dmv\_id) REFERENCES public."motor vehicle department" (dmv\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public.faculty (

faculty\_id SERIAL NOT NULL,

faculty\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

uni\_id INTEGER NOT NULL,

CONSTRAINT faculty\_id PRIMARY KEY (faculty\_id),

CONSTRAINT uni\_id FOREIGN KEY (uni\_id) REFERENCES public.university (uni\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public.gallery (

gallery\_id SERIAL NOT NULL,

gallery\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT gallery\_pkey PRIMARY KEY (gallery\_id)

);

CREATE TABLE IF NOT EXISTS public."household furniture" (

furniture\_id SERIAL NOT NULL,

furniture\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "household furniture\_pkey" PRIMARY KEY (furniture\_id)

);

CREATE TABLE IF NOT EXISTS public.menu (

food\_id SERIAL NOT NULL,

food\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

res\_id INTEGER NOT NULL,

CONSTRAINT food\_id PRIMARY KEY (food\_id),

CONSTRAINT res\_id FOREIGN KEY (res\_id) REFERENCES public.restaurant (res\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public."motor vehicle department" (

dmv\_id SERIAL NOT NULL,

dmv\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "motor vehicle department\_pkey" PRIMARY KEY (dmv\_id)

);

CREATE TABLE IF NOT EXISTS public.restaurant (

res\_id SERIAL NOT NULL,

res\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT restaurant\_pkey PRIMARY KEY (res\_id)

);

CREATE TABLE IF NOT EXISTS public.students

(

stu\_id VARCHAR(255) COLLATE pg\_catalog."default" NOT NULL,

stu\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

uni\_id INTEGER NOT NULL,

CONSTRAINT students\_pkey PRIMARY KEY (stu\_id),

CONSTRAINT uni\_id FOREIGN KEY (uni\_id) REFERENCES public.university (uni\_id) MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public.university

(

uni\_id INTEGER NOT NULL DEFAULT nextval('"University\_uni\_id\_seq"'::regclass),

uni\_name VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "University\_pkey" PRIMARY KEY (uni\_id)

);

CREATE INDEX IF NOT EXISTS fki\_uni\_id ON public.students(uni\_id);

END;

**Table – Column & Attribute Descriptions:**

**“University”** table is the table that has the primary key serial integer uni\_id, the varchar attribute uni\_name, but all attributes cannot have null values. The purpose of the table is to identify the university that both faculty and students can belong to.

**“Student”** table is the table that has the primary key serial integer *stu\_id*, the varchar attribute *stu\_name,* but all attributes cannot have null values. This table also includes the foreign key *uni\_id,* from the **“University”** table. The purpose of the table is to identify the university that each student belongs to. Students can only belong to one university, however, a university has many students.

**“Faculty”** table is the table that has the primary key serial integer *faculty\_id*, the varchar attribute *faculty\_name,* but all attributes cannot have null values. This table also includes the foreign key *uni\_id,* from the **“University”** table. The purpose of the table is to identify the university that each faculty member belongs to. Faculty can only belong to one university, however, a university has many faculty members.

**“Gallery”** table is the table that has the primary key serial integer *gallery\_id*, the varchar attribute *gallery\_name,* but all attributes cannot have null values. The purpose of the table is to identify the gallery that each piece of artwork belongs to. Artwork can only belong to one gallery at a time, however, a gallery (usually) has many pieces of art.

**“Artwork”** table is the table that has the primary key serial integer *art\_id*, the varchar attribute *art\_name,* but all attributes cannot have null values. This table also includes the foreign key *gallery\_id,* from the **“Gallery”** table. The purpose of the table is to identify the gallery that the artwork belongs to. Artwork can only belong to one gallery, however, a gallery has many pieces of art.

**“Motor Vehicle Department”** table is the table that has the primary key serial integer *dmv\_id*, the varchar attribute *dmv\_name,* but all attributes cannot have null values. The purpose of the table is to identify the department of motor vehicle that each automobile belongs to. Automobiles can only belong to one department, however, a motor vehicle department has many automobiles registered.

**“Automobile”** table is the table that has the primary key serial integer *auto\_id*, the varchar attribute *auto\_name,* but all attributes cannot have null values. This table also includes the foreign key *dmv\_id,* from the **“Motor Vehicle Department”** table. The purpose of the table is to identify the DMV that each vehicle belongs to. Automobiles can only belong to one Motor Vehicle Department, however, a department has many automobiles.

**“Restaurant”** table is the table that has the primary key serial integer *res\_id*, the varchar attribute res*\_name,* but all attributes cannot have null values. The purpose of the table is to identify the restaurant that each menu item/food belongs to. Menu items can only belong to one restaurant however, a restaurant has many food items on its menu.

**“Menu”** table is the table that has the primary key serial integer *food\_id*, the varchar attribute *food\_name,* but all attributes cannot have null values. This table also includes the foreign key *res\_id,* from the **“Restaurant”** table. The purpose of the table is to identify the food item that is on each restaurants menu. Food items can only be on one restaurants menu (unique recipes), however, a restaurant has many food items.

**“Household Furniture”** table is the table that has the primary key serial integer *furniture\_id*, the varchar attribute *furniture\_name,* but all attributes cannot have null values. The purpose of the table is to identify household furniture. This table has no relationships.

A screenshot of a computer screen

Description automatically generated

**Exercise 3:**

**SQL Statement for Exercise 3:**

CREATE DATABASE exercise4

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

BEGIN;

CREATE TABLE IF NOT EXISTS public.publication (

pub\_id SERIAL NOT NULL,

pub\_title VARCHAR COLLATE pg\_catalog."default" NOT NULL,

last\_rev DATE NOT NULL,

CONSTRAINT publication\_pkey PRIMARY KEY (pub\_id)

);

CREATE TABLE IF NOT EXISTS public.usage\_downloads (

inv\_id SERIAL NOT NULL,

month\_use\_date DATE NOT NULL,

use\_count INTEGER NOT NULL,

pub\_id INTEGER NOT NULL,

CONSTRAINT inv\_id PRIMARY KEY (inv\_id),

CONSTRAINT pub\_id FOREIGN KEY (pub\_id) REFERENCES public.publication (pub\_id) MATCH SIMPLE

);

END;

**Exercise 3:**

**“Publication”** table is the table that has the primary key serial integer *pub\_id*, the varchar attribute *pub\_name,* and the *lastRev* date attribute, but all attributes cannot have null values. The purpose of the table is to identify the publication that can be downloaded. Publications can be downloaded multiple times, but changes over time.

**“Usage/Downloads”** table is the table that has the primary key serial integer *inv\_id* to identify the specific data entry, the varchar attribute *jan24use* to show the usage during that month for each publication*,* but all attributes cannot have null values. This table also includes the foreign key *pub\_id,* from the **“Publication”** table. The purpose of the table is to identify the publication whose usage is being recorded. Publications can only be invoiced once a month, but many publications can be assessed during these months.

**A screenshot of a computer

Description automatically generated**

**Exercise 4:**

**Create Database:**

CREATE DATABASE exercise4

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

BEGIN;

CREATE TABLE IF NOT EXISTS public."Author" (

"author\_ID" SERIAL NOT NULL,

"authorFirstName" VARCHAR COLLATE pg\_catalog."default" NOT NULL,

"authorLastName" VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Author\_pkey" PRIMARY KEY ("author\_ID")

);

CREATE TABLE IF NOT EXISTS public."Ebook-Author" (

"ebook\_ID" INTEGER NOT NULL,

"author\_ID" INTEGER NOT NULL,

CONSTRAINT "Ebook\_Author\_pkey" PRIMARY KEY ("ebook\_ID", "author\_ID"),

CONSTRAINT "Ebook\_Author\_author\_ID\_fkey" FOREIGN KEY ("author\_ID") REFERENCES public."Author" ("author\_ID") MATCH SIMPLE,

CONSTRAINT "Ebook\_Author\_ebook\_ID\_fkey" FOREIGN KEY ("ebook\_ID") REFERENCES public."Ebooks" ("ebook\_ID") MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public."Ebook\_Genre" (

"ebook\_ID" INTEGER NOT NULL,

"genre\_ID" INTEGER NOT NULL,

CONSTRAINT "Ebook\_Genre\_pkey" PRIMARY KEY ("ebook\_ID", "genre\_ID"),

CONSTRAINT "Ebook\_Genre\_ebook\_ID\_fkey" FOREIGN KEY ("ebook\_ID") REFERENCES public."Ebooks" ("ebook\_ID") MATCH SIMPLE,

CONSTRAINT "Ebook\_Genre\_genre\_ID\_fkey" FOREIGN KEY ("genre\_ID") REFERENCES public."Genre" ("genre\_ID") MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public."Ebooks" (

"ebook\_ID" SERIAL NOT NULL,

"ebookTitle" VARCHAR COLLATE pg\_catalog."default" NOT NULL,

"author\_ID" INTEGER NOT NULL,

"ebookPrice" MONEY NOT NULL,

"genre\_ID" INTEGER NOT NULL,

CONSTRAINT "Ebooks\_pkey" PRIMARY KEY ("ebook\_ID")

);

CREATE TABLE IF NOT EXISTS public."Genre" (

"genre\_ID" SERIAL NOT NULL,

"genreName" VARCHAR COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Genre\_pkey" PRIMARY KEY ("genre\_ID")

);

CREATE TABLE IF NOT EXISTS public."User" (

"user\_ID" SERIAL NOT NULL,

"username" VARCHAR COLLATE pg\_catalog."default" NOT NULL,

"emailHome" VARCHAR COLLATE pg\_catalog."default",

"emailWork" VARCHAR COLLATE pg\_catalog."default",

"emailSchool" VARCHAR COLLATE pg\_catalog."default",

CONSTRAINT "User\_pkey" PRIMARY KEY ("user\_ID")

);

CREATE TABLE IF NOT EXISTS public."User\_Author" (

"user\_ID" INTEGER NOT NULL,

"author\_ID" INTEGER NOT NULL,

CONSTRAINT "User\_Author\_pkey" PRIMARY KEY ("user\_ID", "author\_ID"),

CONSTRAINT "User\_Author\_author\_ID\_fkey" FOREIGN KEY ("author\_ID") REFERENCES public."Author" ("author\_ID") MATCH SIMPLE,

CONSTRAINT "User\_Author\_user\_ID\_fkey" FOREIGN KEY ("user\_ID") REFERENCES public."User" ("user\_ID") MATCH SIMPLE

);

CREATE TABLE IF NOT EXISTS public."User\_Genre" (

"user\_ID" INTEGER NOT NULL,

"genre\_ID" INTEGER NOT NULL,

CONSTRAINT "User\_Genre\_pkey" PRIMARY KEY ("user\_ID", "genre\_ID"),

CONSTRAINT "User\_Genre\_genre\_ID\_fkey" FOREIGN KEY ("genre\_ID") REFERENCES public."Genre" ("genre\_ID") MATCH SIMPLE,

CONSTRAINT "User\_Genre\_user\_ID\_fkey" FOREIGN KEY ("user\_ID") REFERENCES public."User" ("user\_ID") MATCH SIMPLE

);

END;

**Exercise 4:**

The "**Author**" table represents authors who have written ebooks. It includes attributes such as author ID, first name, and last name. Each author can be associated with multiple ebooks, which are captured in the "**Ebook-Author**" table. This table establishes a many-to-many relationship between ebooks and authors. Additionally, ebooks belong to specific genres, as indicated in the “**Genre**”, and "**Ebook\_Genre**" tables, which establishes another many-to-many relationship. The "**Ebooks**" table contains information about each ebook, including its title, price, and genre. “**Users**” can interact with ebooks and authors, as depicted in the "**User\_Author**" table, which records relationships between users and authors based on their interactions. Similarly, users can have preferences for specific genres, as recorded in the "**User\_Genre**" table, establishing associations between users and genres.

A diagram of a computer

Description automatically generated

**Exercise 5:**

**Create Database:**

CREATE DATABASE exercise5

WITH

OWNER = postgres

ENCODING = 'UTF8'

LOCALE\_PROVIDER = 'libc'

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

BEGIN;

CREATE TABLE IF NOT EXISTS public.class (

class\_id serial PRIMARY KEY,

class\_name character varying NOT NULL,

class\_grade character varying NOT NULL

);

CREATE TABLE IF NOT EXISTS public.student (

student\_id serial PRIMARY KEY,

student\_name character varying NOT NULL

);

CREATE TABLE IF NOT EXISTS public.university (

university\_id serial PRIMARY KEY,

university\_name character varying NOT NULL

);

CREATE TABLE IF NOT EXISTS public.class\_student (

class\_id integer NOT NULL,

student\_id integer NOT NULL,

PRIMARY KEY (class\_id, student\_id),

FOREIGN KEY (class\_id) REFERENCES public.class (class\_id) ON DELETE CASCADE,

FOREIGN KEY (student\_id) REFERENCES public.student (student\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS public.student\_university (

student\_id integer NOT NULL,

university\_id integer NOT NULL,

PRIMARY KEY (student\_id, university\_id),

FOREIGN KEY (student\_id) REFERENCES public.student (student\_id) ON DELETE CASCADE,

FOREIGN KEY (university\_id) REFERENCES public.university (university\_id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS public.class\_university (

class\_id integer NOT NULL,

university\_id integer NOT NULL,

PRIMARY KEY (class\_id, university\_id),

FOREIGN KEY (class\_id) REFERENCES public.class (class\_id) ON DELETE CASCADE,

FOREIGN KEY (university\_id) REFERENCES public.university (university\_id) ON DELETE CASCADE

);

END;

**Exercise 5:**

In Exercise 5, the database schema consists of three main tables: "**classes**", "**students**", and "**university**". The "classes" table represents academic classes offered by universities and includes attributes such as class ID, class name, and class grade. Each class can have multiple students enrolled, which is captured in the "students" table. The "university" table represents universities and includes attributes like university ID and name. Both students and classes are associated with universities through foreign key constraints, indicating the university to which they belong. Specifically, each student is associated with a university they attend, and each class is associated with the university where it is offered. This schema allows for the tracking of students enrolled in classes at various universities.A diagram of a computer program

Description automatically generated with medium confidence

**Problem #2:**

**Business Scenario:**

Newfie Nook is an ecommerce platform dedicated to celebrating the culture, flavors, and craftsmanship of Newfoundland and Labrador. The platform offers a curated selection of products made by Newfoundlanders and Labradorians, ranging from iconic treats to unique artisanal creations. Vendors control the products they list and receive notifications for sales.

Entities for Database Creation:

* Products:
  + Attributes: Product ID, Product Name, Description, Price, Image, Category, Vendor ID
  + Description: Represents the products available for purchase on the platform. Includes both NL treats (e.g., Purity Products, Pineapple Crush, Caramel Logs) and artisanal creations (e.g., handcrafted sea glass jewelry, traditional knitwear).
* Customers:
  + Attributes: Customer ID, First Name, Last Name, Email, Address, City, Province, Postal Code, Phone Number
  + Description: Represents individuals who create accounts on the platform to make purchases. Stores customer information for order processing and communication purposes.
* Customer Accounts:
  + Attributes: Account ID, Customer ID, Username, Password, Payment Information
  + Description: Stores login credentials and links customers to their accounts, enabling them to access personalized features and track their order history.
* Vendors:
  + Attributes: Vendor ID, Vendor Name, Email, Address, Phone Number
  + Description: Represents the artisans and businesses from Newfoundland and Labrador who supply products to Newfie Nook. Stores vendor information for inventory management and communication purposes.
* Vendor Accounts:
  + Attributes: Account ID, Vendor ID, Username, Password
  + Description: Stores login credentials for vendors, allowing them to manage their product listings, update inventory, and track sales performance.
* Vendor Bio Info:
  + Attributes: Vendor ID, Vendor Name, Vendor Description, Vendor Contact Information, Vendor Social Media Addresses, Vendor Website Information
* Order Information:
  + Attributes: Invoice ID, Date, Product IDs, Product Name, Product Prices, Vendor ID, Vendor Name, Customer ID, Customer Name, Customer Address, Shipping Information

Business Rules:

1. Customers must create an account to make purchases on Newfie Nook.
2. Each product is associated with a specific vendor who supplies it.
3. Customers can add products to their shopping cart and proceed to checkout for payment.
4. Vendors can manage their product listings, update inventory levels, and view sales reports through their vendor accounts.
5. Newfie Nook ensures secure payment processing and timely delivery of orders to customers.
6. The platform promotes local craftsmanship and supports NL artisans by showcasing their products to a wider audience.
7. Newfie Nook provides customer support services to address inquiries, resolve issues, and enhance the shopping experience for users.

**Problem 2:**

BEGIN;

CREATE TABLE IF NOT EXISTS public."Customer\_Order"

(

"customer\_ID" integer NOT NULL,

"invoice\_ID" integer NOT NULL,

CONSTRAINT "Customer\_Order\_pkey" PRIMARY KEY ("invoice\_ID", "customer\_ID")

);

CREATE TABLE IF NOT EXISTS public."Product\_Order"

(

"product\_ID" integer NOT NULL,

invoice\_id integer NOT NULL,

CONSTRAINT "Product\_Order\_pkey" PRIMARY KEY ("product\_ID", invoice\_id)

);

CREATE TABLE IF NOT EXISTS public."Vendor\_Products"

(

"vendor\_ID" integer NOT NULL,

"product\_ID" integer NOT NULL,

CONSTRAINT "Vendor\_Products\_pkey" PRIMARY KEY ("vendor\_ID", "product\_ID")

);

CREATE TABLE IF NOT EXISTS public.customer\_accounts

(

account\_id serial NOT NULL,

customer\_id integer NOT NULL,

username character varying(50) COLLATE pg\_catalog."default" NOT NULL,

password character varying(50) COLLATE pg\_catalog."default" NOT NULL,

payment\_information text COLLATE pg\_catalog."default",

CONSTRAINT customer\_accounts\_pkey PRIMARY KEY (account\_id)

);

CREATE TABLE IF NOT EXISTS public.customers

(

customer\_id serial NOT NULL,

first\_name character varying(50) COLLATE pg\_catalog."default" NOT NULL,

last\_name character varying(50) COLLATE pg\_catalog."default" NOT NULL,

email character varying(100) COLLATE pg\_catalog."default" NOT NULL,

address character varying(255) COLLATE pg\_catalog."default" NOT NULL,

city character varying(100) COLLATE pg\_catalog."default" NOT NULL,

province character varying(100) COLLATE pg\_catalog."default" NOT NULL,

postal\_code character varying(20) COLLATE pg\_catalog."default" NOT NULL,

phone\_number character varying(20) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT customers\_pkey PRIMARY KEY (customer\_id)

);

CREATE TABLE IF NOT EXISTS public.order\_information

(

invoice\_id serial NOT NULL,

date date NOT NULL,

product\_ids text COLLATE pg\_catalog."default" NOT NULL,

product\_names text COLLATE pg\_catalog."default" NOT NULL,

product\_prices text COLLATE pg\_catalog."default" NOT NULL,

vendor\_id integer NOT NULL,

vendor\_name character varying(100) COLLATE pg\_catalog."default" NOT NULL,

customer\_id integer NOT NULL,

customer\_name character varying(100) COLLATE pg\_catalog."default" NOT NULL,

customer\_address character varying(255) COLLATE pg\_catalog."default" NOT NULL,

shipping\_information text COLLATE pg\_catalog."default",

CONSTRAINT order\_information\_pkey PRIMARY KEY (invoice\_id)

);

CREATE TABLE IF NOT EXISTS public.products

(

product\_id serial NOT NULL,

product\_name character varying(255) COLLATE pg\_catalog."default" NOT NULL,

description text COLLATE pg\_catalog."default",

price numeric(10, 2) NOT NULL,

image character varying(255) COLLATE pg\_catalog."default",

category character varying(100) COLLATE pg\_catalog."default",

vendor\_id integer NOT NULL,

CONSTRAINT products\_pkey PRIMARY KEY (product\_id)

);

CREATE TABLE IF NOT EXISTS public.vendor\_accounts

(

account\_id serial NOT NULL,

vendor\_id integer NOT NULL,

username character varying(50) COLLATE pg\_catalog."default" NOT NULL,

password character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT vendor\_accounts\_pkey PRIMARY KEY (account\_id)

);

CREATE TABLE IF NOT EXISTS public.vendor\_bio\_info

(

vendor\_id integer NOT NULL,

vendor\_description text COLLATE pg\_catalog."default",

vendor\_contact\_information text COLLATE pg\_catalog."default",

vendor\_social\_media\_addresses text COLLATE pg\_catalog."default",

vendor\_website\_information text COLLATE pg\_catalog."default",

CONSTRAINT vendor\_bio\_info\_pkey PRIMARY KEY (vendor\_id)

);

CREATE TABLE IF NOT EXISTS public.vendors

(

vendor\_id serial NOT NULL,

vendor\_name character varying(100) COLLATE pg\_catalog."default" NOT NULL,

email character varying(100) COLLATE pg\_catalog."default" NOT NULL,

address character varying(255) COLLATE pg\_catalog."default" NOT NULL,

phone\_number character varying(20) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT vendors\_pkey PRIMARY KEY (vendor\_id)

);

ALTER TABLE IF EXISTS public."Customer\_Order"

ADD CONSTRAINT "invoice\_ID" FOREIGN KEY ("invoice\_ID")

REFERENCES public.order\_information (invoice\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Customer\_Order"

ADD FOREIGN KEY ("customer\_ID")

REFERENCES public.customers (customer\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Product\_Order"

ADD FOREIGN KEY ("product\_ID")

REFERENCES public.products (product\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Product\_Order"

ADD FOREIGN KEY ("product\_ID")

REFERENCES public.products (product\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Product\_Order"

ADD FOREIGN KEY (invoice\_id)

REFERENCES public.order\_information (invoice\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Vendor\_Products"

ADD CONSTRAINT "vendor\_ID" FOREIGN KEY ("vendor\_ID")

REFERENCES public.vendors (vendor\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public."Vendor\_Products"

ADD FOREIGN KEY ("product\_ID")

REFERENCES public.products (product\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public.customer\_accounts

ADD CONSTRAINT customer\_accounts\_customer\_id\_fkey FOREIGN KEY (customer\_id)

REFERENCES public.customers (customer\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public.vendor\_accounts

ADD CONSTRAINT vendor\_accounts\_vendor\_id\_fkey FOREIGN KEY (vendor\_id)

REFERENCES public.vendors (vendor\_id) MATCH SIMPLE

ALTER TABLE IF EXISTS public.vendor\_bio\_info

ADD CONSTRAINT vendor\_bio\_info\_vendor\_id\_fkey FOREIGN KEY (vendor\_id)

REFERENCES public.vendors (vendor\_id) MATCH SIMPLE

CREATE INDEX IF NOT EXISTS vendor\_bio\_info\_pkey

ON public.vendor\_bio\_info(vendor\_id);

END;

**Problem 2:**

A diagram of a computer

Description automatically generated with medium confidence