Final Project: Database Design and Implementation

Scenario

In this scenario, you have recently been hired as a Data Engineer by a New York-based coffee shop chain looking to expand nationally by opening several franchise locations. They want to streamline operations and revamp their data infrastructure as part of their expansion process.

Your job is to design their relational database systems for improved operational efficiencies and make it easier for their executives to make data-driven decisions.

Currently, their data resides in several systems: accounting software, supplier databases, point of sales (POS) systems, and even spreadsheets. You will review the data in all of these systems and design a central database to house all of the data. You will then create the database objects and load them with source data. Finally, you will create subsets of data your business partners require, export them, and load them into staging databases using several RDBMS.

Data used in this project

In this project, you will be working with a subset of data from the Coffee shop sample data.

You will use a modified version of the data for the project, so to succeed in the project, download the linked files when prompted in the instructions. You do not need to use any data from the source.

In your scenario, you will be working with data from the following sources:

- Staff information held in a spreadsheet at headquarters (HQ)
- Sales outlet information held in a spreadsheet at HQ
- Sales data output as a CSV file from the POS system in the sales outlets
- Customer data output as a CSV file from a custom customer relationship management system
- Product information maintained in a spreadsheet exported from your supplier's database

Objectives

After completing this lab, you will be able to:

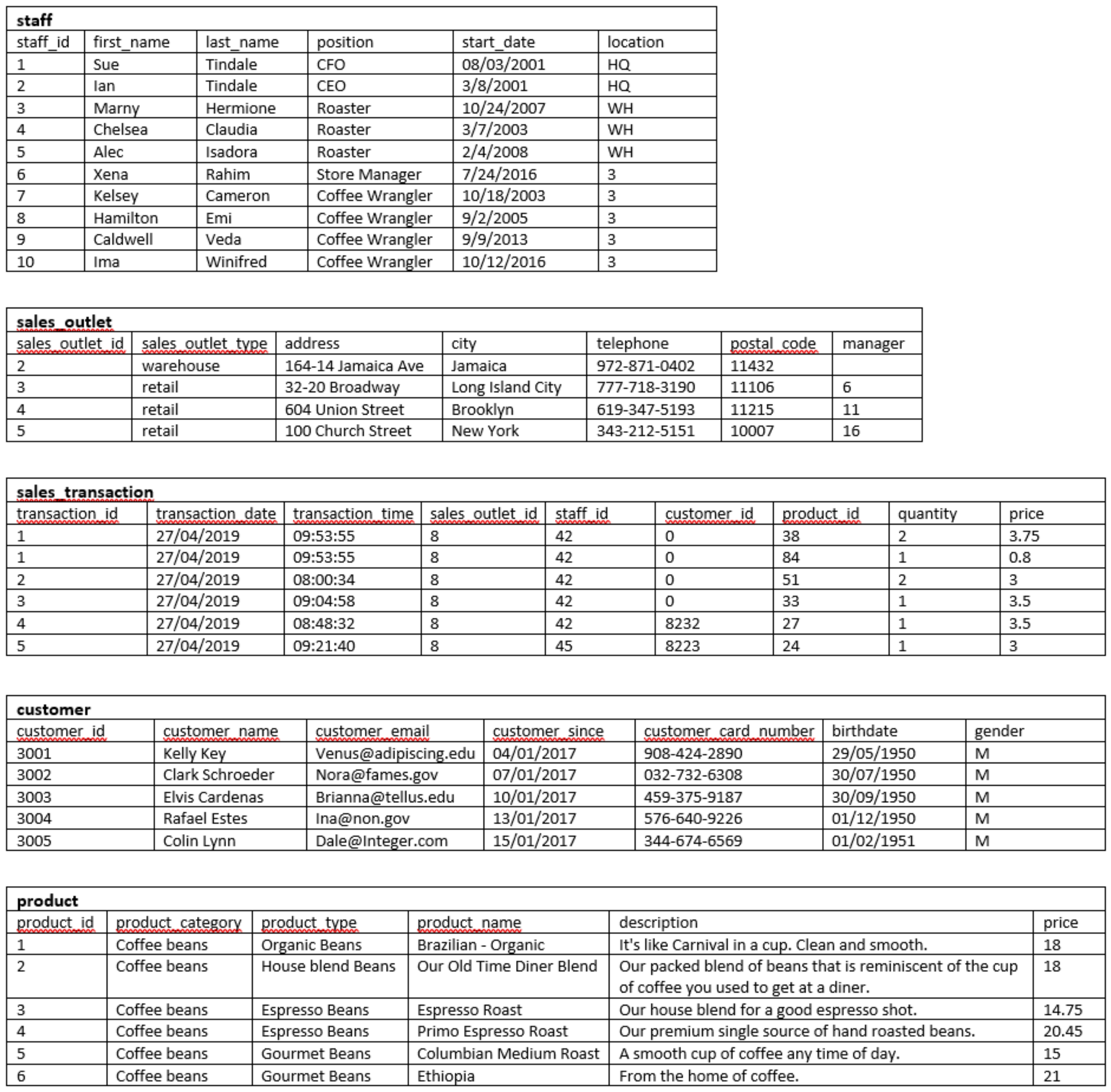
- Identify entities

- Identity attributes
- Create an entity relationship diagram (ERD) using the pgAdmin ERD tool
- Normalize tables
- Define keys and relationships
- Create database objects by generating and running the SQL script from the ERD tool
- Create a view and export the data
- Create a materialized view and export the data
- Import data into a MySQL database using phpMyAdmin GUI tool

Task 1: Identify entities

The first step when designing a new database is to review any existing data and identify the entities for your new system.

1. The following image shows sample data from each source you will be working with to design your new central database. Review the image and identify the entities you plan to create.



staff

staff_id	first_name	last_name	position	start_date	location
1	Sue	Tindale	CFO	08/03/2001	HQ
2	Ian	Tindale	CEO	3/8/2001	HQ
3	Marny	Hermione	Roaster	10/24/2007	WH
4	Chelsea	Claudia	Roaster	3/7/2003	WH
5	Alec	Isadora	Roaster	2/4/2008	WH
6	Xena	Rahim	Store Manager	7/24/2016	3
7	Kelsey	Cameron	Coffee Wrangler	10/18/2003	3
8	Hamilton	Emi	Coffee Wrangler	9/2/2005	3
9	Caldwell	Veda	Coffee Wrangler	9/9/2013	3
10	Ima	Winifred	Coffee Wrangler	10/12/2016	3

sales_outlet

sales_outlet_id	sales_outlet_type	address	city	telephone	postal_code	manager
2	warehouse	164-14 Jamaica Ave	Jamaica	972-871-0402	11432	
3	retail	32-20 Broadway	Long Island City	777-718-3190	11106	6
4	retail	604 Union Street	Brooklyn	619-347-5193	11215	11
5	retail	100 Church Street	New York	343-212-5151	10007	16

sales_transaction

transaction_id	transaction_date	transaction_time	sales_outlet_id	staff_id	customer_id	product_id	quantity	price
1	27/04/2019	09:53:55	8	42	0	38	2	3.75
1	27/04/2019	09:53:55	8	42	0	84	1	0.8
2	27/04/2019	08:00:34	8	42	0	51	2	3
3	27/04/2019	09:04:58	8	42	0	33	1	3.5
4	27/04/2019	08:48:32	8	42	8232	27	1	3.5
5	27/04/2019	09:21:40	8	45	8223	24	1	3

customer

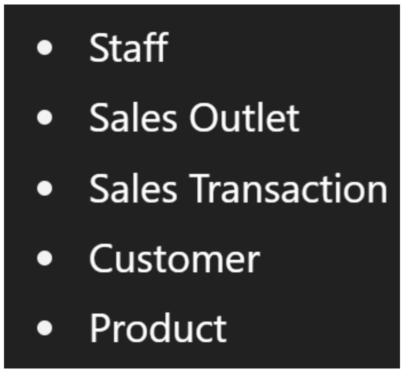
customer_id	customer_name	customer_email	customer_since	customer_card_number	birthdate	gender
3001	Kelly Key	Venus@adipiscing.edu	04/01/2017	908-424-2890	29/05/1950	M
3002	Clark Schroeder	Nora@fames.gov	07/01/2017	032-732-6308	30/07/1950	M
3003	Elvis Cardenas	Brianna@tellus.edu	10/01/2017	459-375-9187	30/09/1950	M
3004	Rafael Estes	Ina@non.gov	13/01/2017	576-640-9226	01/12/1950	M
3005	Colin Lynn	Dale@Integer.com	15/01/2017	344-674-6569	01/02/1951	M

product

product_id	product_category	product_type	product_name	description	price
1	Coffee beans	Organic Beans	Brazilian - Organic	It's like Carnival in a cup. Clean and smooth.	18
2	Coffee beans	House blend Beans	Our Old Time Diner Blend	Our packed blend of beans that is reminiscent of the cup of coffee you used to get at a diner.	18
3	Coffee beans	Espresso Beans	Espresso Roast	Our house blend for a good espresso shot.	14.75
4	Coffee beans	Espresso Beans	Primo Espresso Roast	Our premium single source of hand roasted beans.	20.45
5	Coffee beans	Gourmet Beans	Columbian Medium Roast	A smooth cup of coffee any time of day.	15
6	Coffee beans	Gourmet Beans	Ethiopia	From the home of coffee.	21

- Note: You can download a copy of this image or open it in another browser tab for reference later in the lab.

Make a list of the entities you have identified.

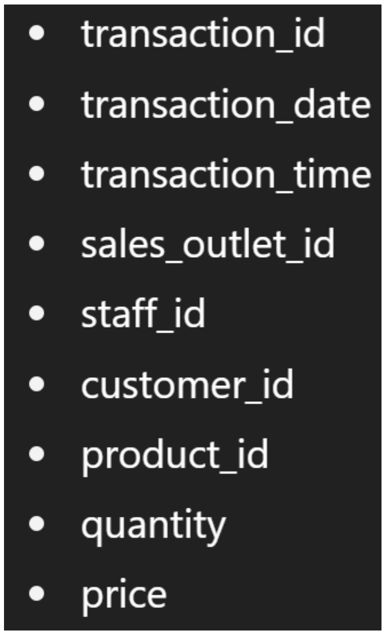


- Staff
- Sales Outlet
- Sales Transaction
- Customer
- Product

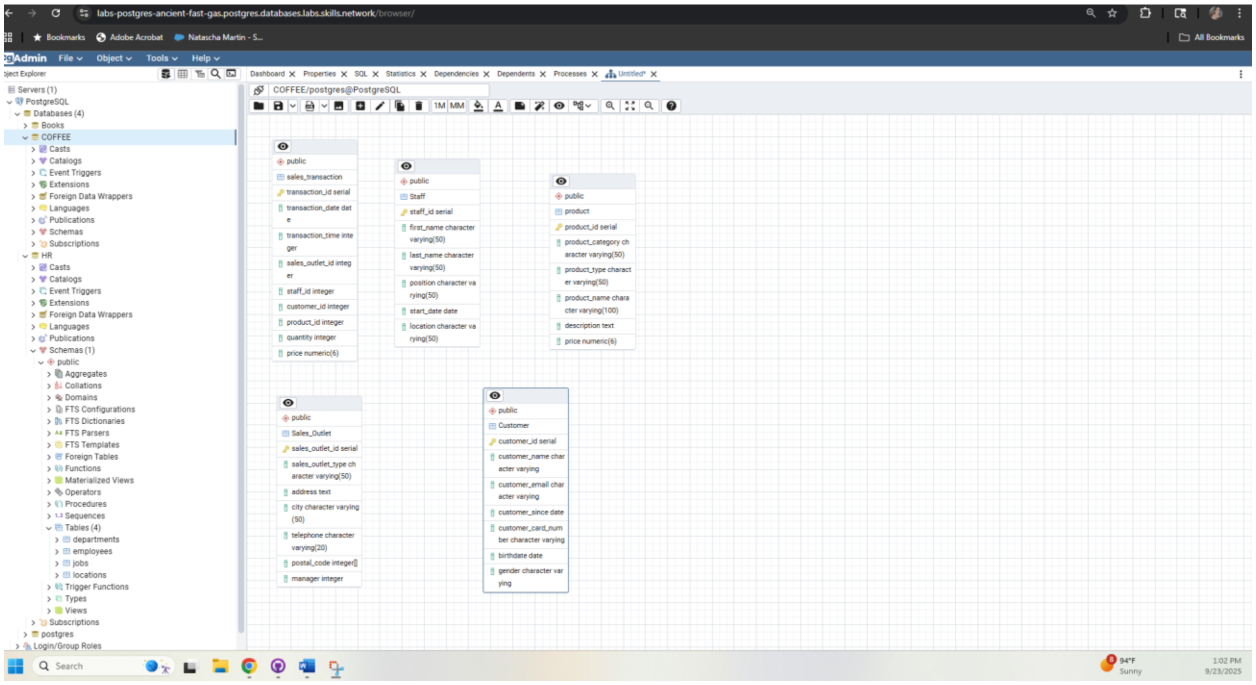
Task 2: Identify attributes

In this task, you will identify the attributes of one of the entities you plan to create.

1. Using the information from the sample data in the image from Task 1, identify the entity's attributes that will store the sales transaction data.
2. Make a list of the sales transaction attributes that you identified.



- transaction_id
- transaction_date
- transaction_time
- sales_outlet_id
- staff_id
- customer_id
- product_id
- quantity
- price

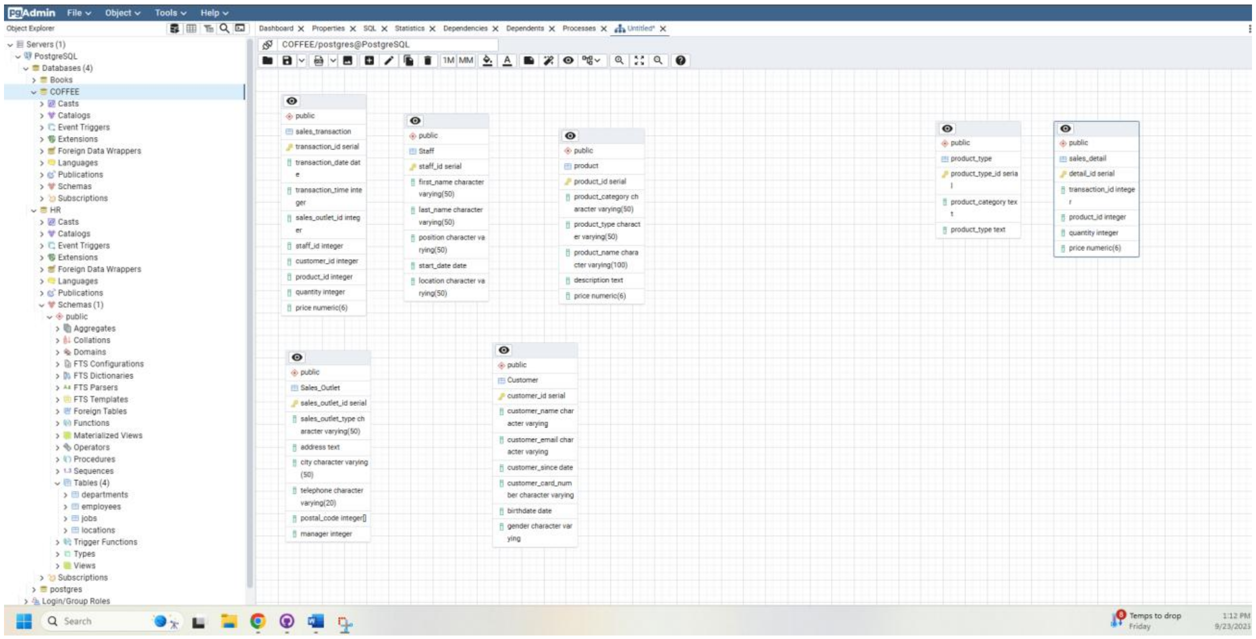


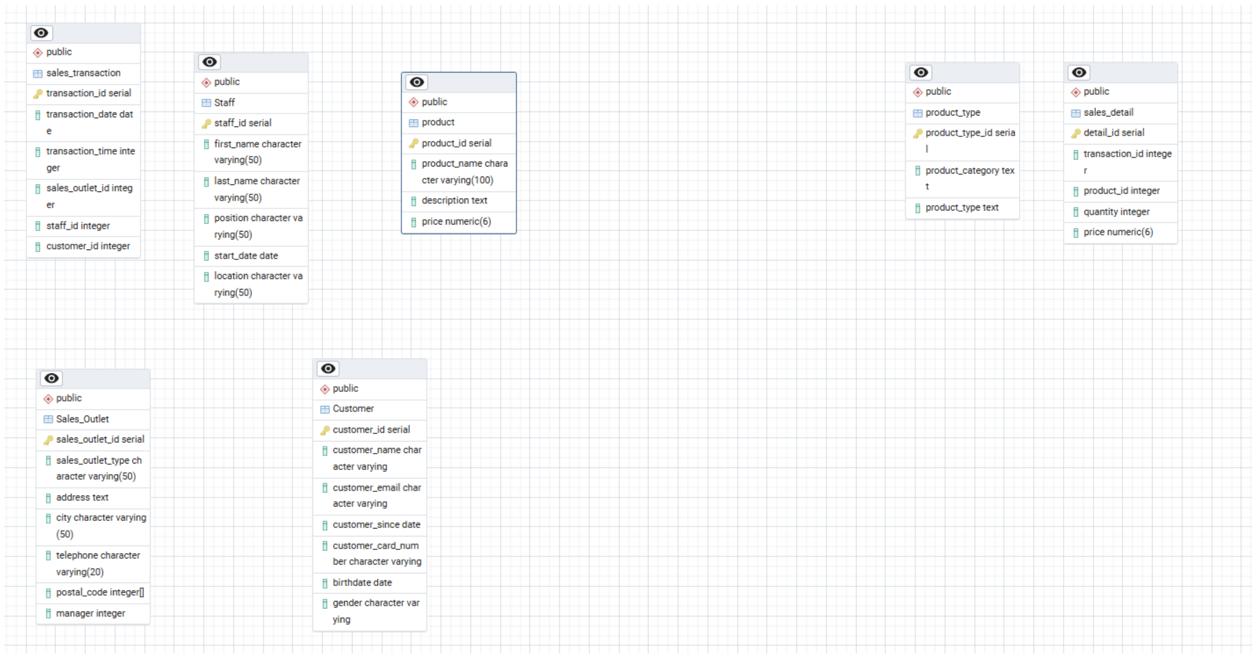
Task 4: Normalize tables

When reviewing your ERD, you notice it does not conform to the second normal form. In this task, you will normalize some of the tables within the database.

1. Review the data in the sales transaction table. Note that the transaction id column does not contain unique values because some transactions include multiple products.
2. Determine which columns should be stored in a separate table to remove the repeating rows and to put this table into second normal form.
3. Add a new table named sales_detail to the ERD, define the columns in the new table, and delete the moved columns from the sales transaction table, leaving a matching column in each of the two tables to create a relationship between them later.
4. Take a screenshot of your ERD and save it as Task4A.png or Task4A.jpg.
5. Review the data in the product table. Note that the product category and product type columns contain redundant data.
6. Determine which columns should be stored in a separate table to reduce redundant data and to put this table into a second normal form.

7. Add a new table named product_type to the ERD, define the columns in the new table, and delete the moved columns from the product table, leaving a matching column in each of the two tables to create a relationship between them later.

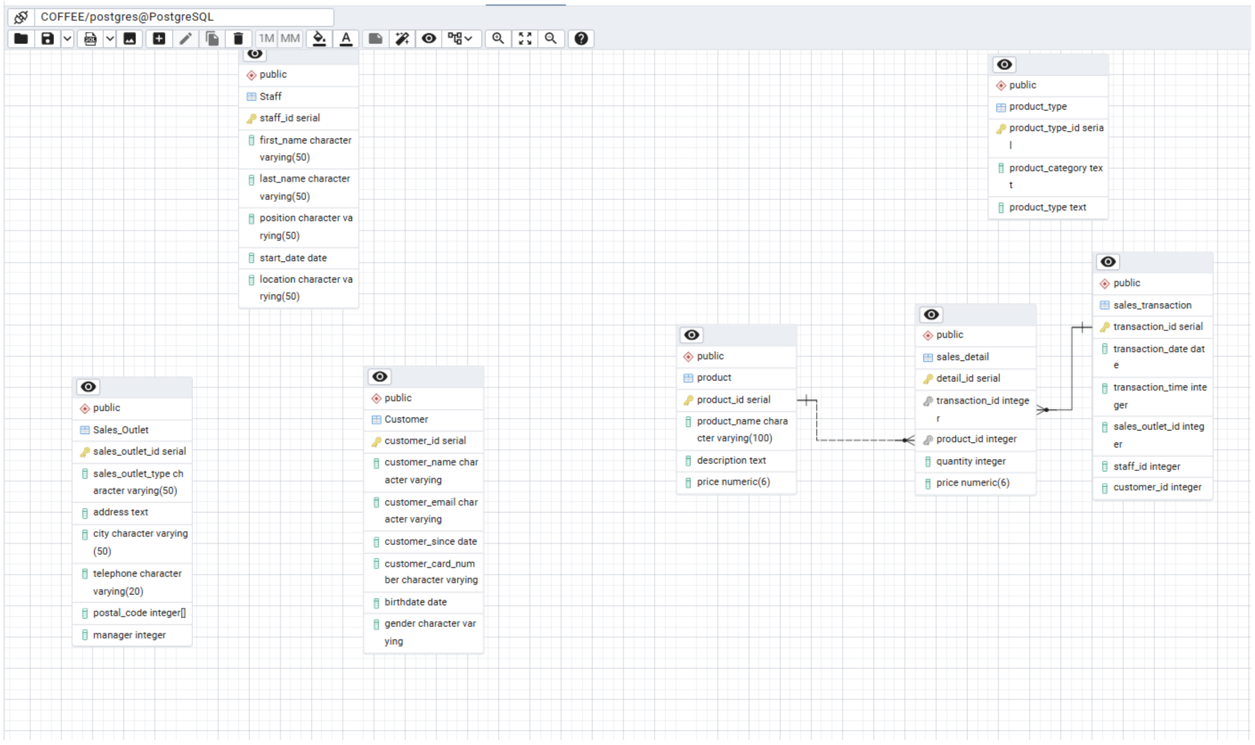


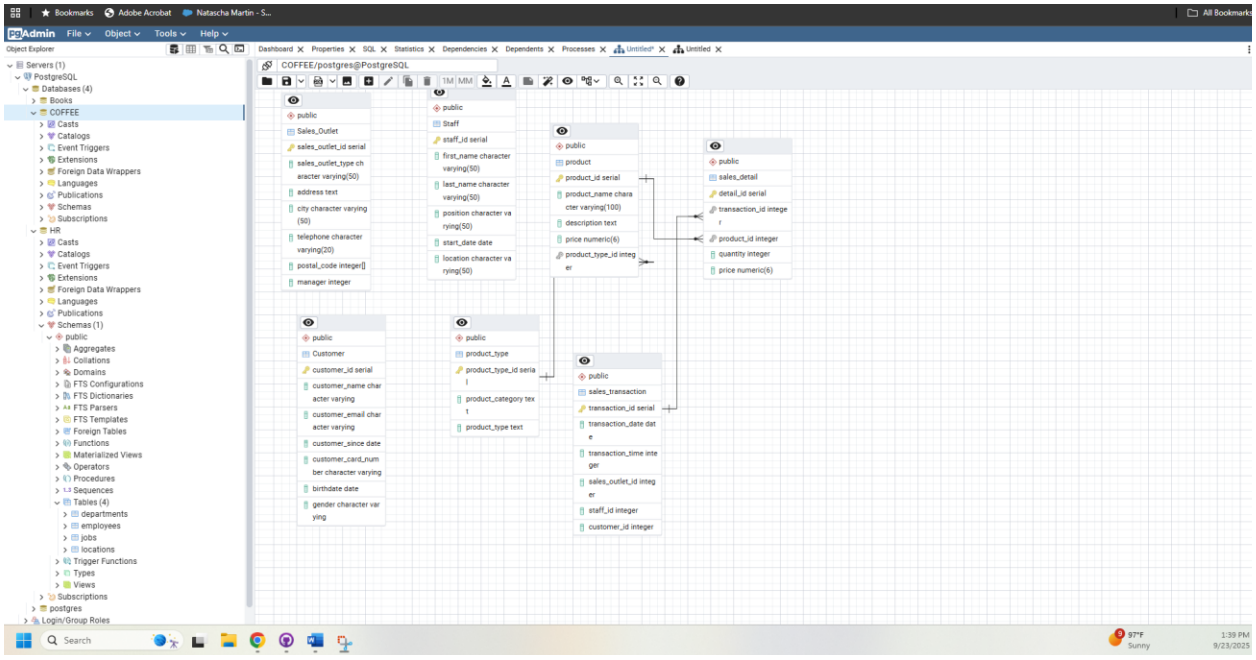


Task 5: Define keys and relationships

After normalizing your tables, you can define their primary keys and relationships between the tables in your ERD.

1. Identify an appropriate column in each table to be a primary key and create the primary keys in the tables in your entity-relationship diagram (ERD).
2. Take a screenshot of your ERD and save it as Task5A.png or Task5A.jpg.
3. Identify the relationships between the following pairs of tables and then create the relationships in your ERD:
   - sales_detail to sales_transaction
   - sales_detail to product
   - product to product_type





Task 6: Create database objects by generating and running the SQL script from the ERD tool

Now that your design is complete, you will generate an SQL script from your ERD, which you can use to create your database schema. For this project, you will then use a given SQL script to ensure that you can load the sample data into the schema. Finally, you will load the existing data from various sources into your new database schema.

1. Use the Generate SQL functionality in the ERD tool to create an SQL script from your ERD.

2. Download the following GeneratedScript.sql file to your local computer.

    - GeneratedScript.sql

3. In pgAdmin, open the query tool, upload and open the GeneratedScript.sql file from your local computer, and then run the script to create the tables defined in the ERD. Verify that the tables exist in the COFFEE database\'s public schema now.

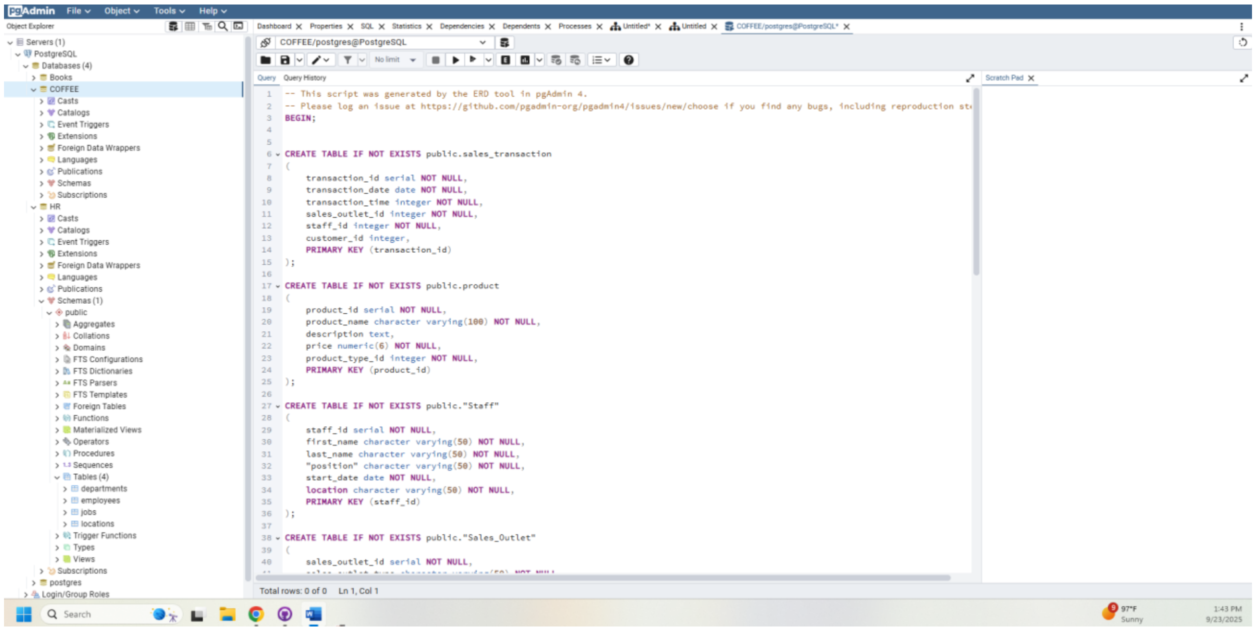
4. Take a screenshot of the tables shown in the tree-view pane on the left side of the page and save it as Task6A.png or Task6A.jpg.

5. Download the following CoffeeData.sql file to your local computer.

    - CoffeeData.sql

6. In pgAdmin, open another instance of the Query tool, upload and open the CoffeeData.sql file from your local computer, and then run the script to populate the tables you just created.

7. In pgAdmin, view the first 100 rows of the sales_detail table.



```
-- This script was generated by the ERD tool in pgAdmin 4.
-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction st
BEGIN;


CREATE TABLE IF NOT EXISTS public.sales_transaction
(
    transaction_id serial NOT NULL,
    transaction_date date NOT NULL,
    transaction_time integer NOT NULL,
    sales_outlet_id integer NOT NULL,
    staff_id integer NOT NULL,
    customer_id integer,
    PRIMARY KEY (transaction_id)
);

CREATE TABLE IF NOT EXISTS public.product
(
    product_id serial NOT NULL,
    product_name character varying(100) NOT NULL,
    description text,
    price numeric(6) NOT NULL,
    product_type_id integer NOT NULL,
    PRIMARY KEY (product_id)
);

CREATE TABLE IF NOT EXISTS public."Staff"
(
    staff_id serial NOT NULL,
    first_name character varying(50) NOT NULL,
    last_name character varying(50) NOT NULL,
    "position" character varying(50) NOT NULL,
    start_date date NOT NULL,
    location character varying(50) NOT NULL,
    PRIMARY KEY (staff_id)
);

CREATE TABLE IF NOT EXISTS public."Sales_Outlet"
(
    sales_outlet_id serial NOT NULL,
```

Total rows: 0 of 0   Ln 1, Col 1

-- This script was generated by the ERD tool in pgAdmin 4.

-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.

BEGIN;

CREATE TABLE IF NOT EXISTS public.sales_transaction

(

  transaction_id serial NOT NULL,

  transaction_date date NOT NULL,

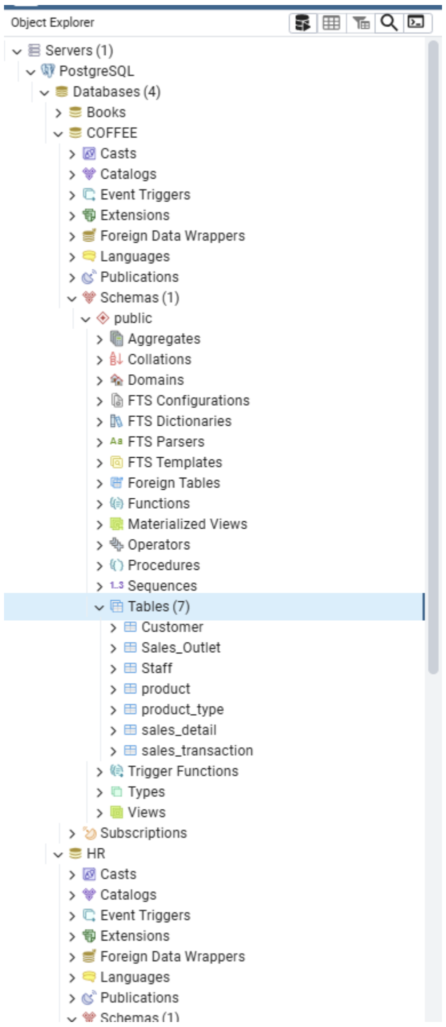
  transaction_time integer NOT NULL,

  sales_outlet_id integer NOT NULL,

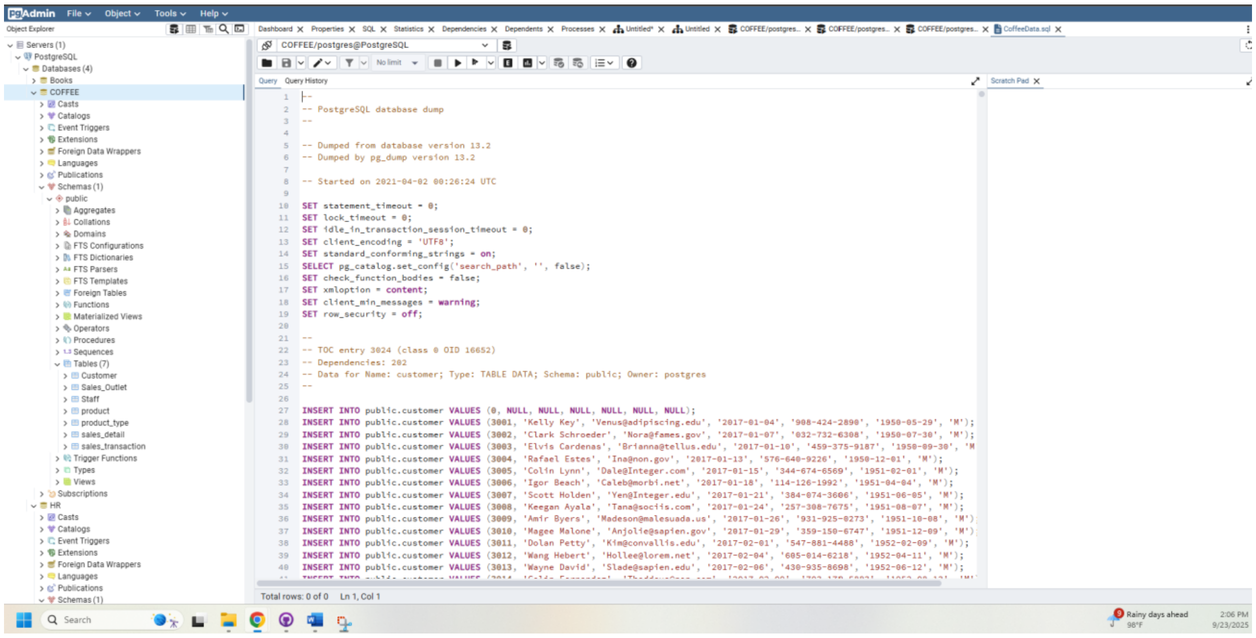
  staff_id integer NOT NULL,

```
    customer_id integer,
    PRIMARY KEY (transaction_id)
);

CREATE TABLE IF NOT EXISTS public.product
(
    product_id serial NOT NULL,
    product_name character varying(100) NOT NULL,
    description text,
    price numeric(6) NOT NULL,
    product_type_id integer NOT NULL,
    PRIMARY KEY (product_id)
);

CREATE TABLE IF NOT EXISTS public."Staff"
(
    staff_id serial NOT NULL,
    first_name character varying(50) NOT NULL,
    last_name character varying(50) NOT NULL,
    "position" character varying(50) NOT NULL,
    start_date date NOT NULL,
    location character varying(50) NOT NULL,
    PRIMARY KEY (staff_id)
);

CREATE TABLE IF NOT EXISTS public."Sales_Outlet"
```

```
(
    sales_outlet_id serial NOT NULL,
    sales_outlet_type character varying(50) NOT NULL,
    address text NOT NULL,
    city character varying(50) NOT NULL,
    telephone character varying(20) NOT NULL,
    postal_code integer[] NOT NULL,
    manager integer,
    PRIMARY KEY (sales_outlet_id)
);

CREATE TABLE IF NOT EXISTS public."Customer"
(
    customer_id serial NOT NULL,
    customer_name character varying NOT NULL,
    customer_email character varying NOT NULL,
    customer_since date NOT NULL,
    customer_card_number character varying NOT NULL,
    birthdate date NOT NULL,
    gender character varying NOT NULL,
    PRIMARY KEY (customer_id)
);

CREATE TABLE IF NOT EXISTS public.sales_detail
(
    detail_id serial NOT NULL,
    transaction_id integer NOT NULL,
    product_id integer NOT NULL,
    quantity integer NOT NULL,
    price numeric(6) NOT NULL,
    PRIMARY KEY (detail_id)
);

CREATE TABLE IF NOT EXISTS public.product_type
(
    product_type_id serial NOT NULL,
    product_category text NOT NULL,
    product_type text NOT NULL,
    PRIMARY KEY (product_type_id)
);

ALTER TABLE IF EXISTS public.product
    ADD FOREIGN KEY (product_type_id)
    REFERENCES public.product_type (product_type_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public.sales_detail
    ADD FOREIGN KEY (transaction_id)
    REFERENCES public.sales_transaction (transaction_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;



ALTER TABLE IF EXISTS public.sales_detail
    ADD FOREIGN KEY (product_id)
    REFERENCES public.product (product_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;

END;
```



```
Object Explorer
Servers (1)
  PostgreSQL
    Databases (4)
      Books
      COFFEE
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
          public
            Aggregates
            Collations
            Domains
            FTS Configurations
            FTS Dictionaries
            FTS Parsers
            FTS Templates
            Foreign Tables
            Functions
            Materialized Views
            Operators
            Procedures
            Sequences
            Tables (7)
              Customer
              Sales_Outlet
              Staff
              product
              product_type
              sales_detail
              sales_transaction
            Trigger Functions
            Types
            Views
        Subscriptions
      HR
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
```



```
--
-- PostgreSQL database dump
--


-- Dumped from database version 13.2
-- Dumped by pg_dump version 13.2

-- Started on 2021-04-02 00:26:24 UTC

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

--
-- TOC entry 3024 (class 0 OID 16652)
-- Dependencies: 202
-- Data for Name: customer; Type: TABLE DATA; Schema: public; Owner: postgres
--

INSERT INTO public.customer VALUES (0, NULL, NULL, NULL, NULL, NULL, NULL);
INSERT INTO public.customer VALUES (3001, 'Kelly Key', 'Venus@adipiscing.edu', '2017-01-04', '908-424-2890', '1950-05-29', 'M');
INSERT INTO public.customer VALUES (3002, 'Clark Schroeder', 'Nora@fames.gov', '2017-01-07', '032-732-6308', '1950-07-30', 'M');
INSERT INTO public.customer VALUES (3003, 'Elvis Cardenas', 'Brianna@tellus.edu', '2017-01-10', '459-375-9187', '1950-09-30', 'M
INSERT INTO public.customer VALUES (3004, 'Rafael Estes', 'Ina@non.gov', '2017-01-13', '576-640-9226', '1950-12-01', 'M');
INSERT INTO public.customer VALUES (3005, 'Colin Lynn', 'Dale@Integer.com', '2017-01-15', '344-674-6569', '1951-02-01', 'M');
INSERT INTO public.customer VALUES (3006, 'Igor Beach', 'Caleb@morbi.net', '2017-01-18', '114-126-1992', '1951-04-04', 'M');
INSERT INTO public.customer VALUES (3007, 'Scott Holden', 'Yen@Integer.edu', '2017-01-21', '384-074-3606', '1951-06-05', 'M');
INSERT INTO public.customer VALUES (3008, 'Keegan Ayala', 'Tana@sociis.com', '2017-01-24', '257-308-7675', '1951-08-07', 'M');
INSERT INTO public.customer VALUES (3009, 'Amir Byers', 'Madeson@malesuada.us', '2017-01-26', '931-925-0273', '1951-10-08', 'M')
INSERT INTO public.customer VALUES (3010, 'Magee Malone', 'Anjolie@sapien.gov', '2017-01-29', '359-150-6747', '1951-12-09', 'M')
INSERT INTO public.customer VALUES (3011, 'Dolan Petty', 'Kim@convallis.edu', '2017-02-01', '547-881-4488', '1952-02-09', 'M');
INSERT INTO public.customer VALUES (3012, 'Wang Hebert', 'Hollee@lorem.net', '2017-02-04', '605-014-6218', '1952-04-11', 'M');
INSERT INTO public.customer VALUES (3013, 'Wayne David', 'Slade@sapien.edu', '2017-02-06', '430-935-8698', '1952-06-12', 'M');
```

Total rows: 0 of 0   Ln 1, Col 1

Task 7: Create a view and export the data

The external payroll company has requested a list of employees and the locations at which they work. This list should not include the CEO or CFO who owns the company. In this task, you will create a view in your PostgreSQL database that returns this information and export the results to a CSV file.

1. In your COFFEE database, create a new view named staff_locations_view using the following SQL:

    1. 1
    2. 2
    3. 3
    4. 4
    5. 5
    6. 6

vii.SELECT staff.staff_id,

viii.staff.first_name,

ix.staff.last_name,

x.staff.location

xi.FROM staff

xii.WHERE "position" NOT IN ('CEO', 'CFO');

Copied!Wrap Toggled!

2. View all the rows returned from the view.
3. Save the query results to a file named staff_locations_view.csv on your local computer.

Project Summary – Coffee Shop Database Design & Implementation

The goal of this project was to design, normalize, implement, and populate a database for a coffee shop chain expanding nationally. I created the COFFEE database in PostgreSQL/pgAdmin and built the base tables: Staff, Sales_Outlet, Customer, Product, and Sales_Transaction.

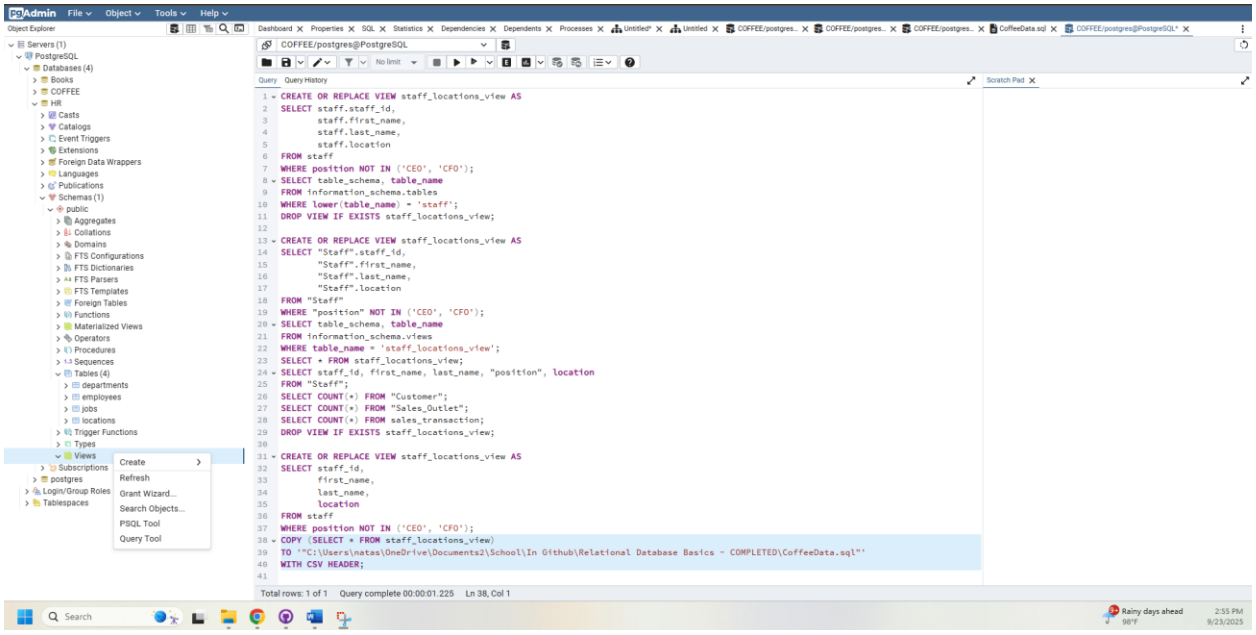
For normalization, I split Sales_Transaction into sales_transaction (transaction metadata) and sales_detail (line items), and I split Product into product (details) and product_type (categories). Primary keys were added to all tables and foreign keys were defined between sales_detail and sales_transaction, sales_detail and product, and product and product_type. I generated the schema from the ERD, ran the SQL script in pgAdmin, and used CoffeeData.sql to populate the tables. I confirmed that Customer, Sales_Outlet, and sales_transaction had rows of data.

The biggest issue was with the staff data. The ERD created a table named "Staff" with a capital S in quotes. The CoffeeData.sql script inserted data into staff in lowercase. That meant the table I thought I should be using ("Staff") was completely empty, and the lowercase staff table was the one that had the rows. Because of this mismatch, my initial attempts to build the staff_locations_view returned no data. I spent over 45 minutes trying to figure it out, running different queries, checking information_schema.tables and information_schema.views, refreshing pgAdmin over and over, and even dropping and recreating the view multiple times. At first the view wouldn't show up in the tree, then it showed up but returned zero rows. Eventually, the problem was confirmed: I was pointing to the wrong version of the table. The fix was to stop using "Staff" and recreate the view against lowercase staff, which finally showed the employees.

I tried to export the view results to CSV, but PostgreSQL would not write to my Windows OneDrive path. That caused more wasted time because COPY kept failing. I learned the fix would be to export through the pgAdmin GUI instead of COPY, but I did not finish that step.

At that point my patience was gone. I completed Tasks 1–6, and I partially completed Task 7 by creating the view and getting it to return rows after fixing the Staff vs staff problem. I did not export the CSV and I did not move forward with Task 8 (materialized view), Task 9 (import staff_locations_view.csv into MySQL), Task 10 (import product_info_m_view.csv into MySQL), or Task 11 (optional Db2 import).

Final status: schema created, normalization and keys done, data loaded, staff_locations_view created and tested, but no exports or imports completed.



```sql
CREATE OR REPLACE VIEW staff_locations_view AS
SELECT staff.staff_id,
       staff.first_name,
       staff.last_name,
       staff.location
FROM staff
WHERE position NOT IN ('CEO', 'CFO');
SELECT table_schema, table_name
FROM information_schema.tables
WHERE lower(table_name) = 'staff';
DROP VIEW IF EXISTS staff_locations_view;

CREATE OR REPLACE VIEW staff_locations_view AS
SELECT "Staff".staff_id,
       "Staff".first_name,
       "Staff".last_name,
       "Staff".location
FROM "Staff"
WHERE "position" NOT IN ('CEO', 'CFO');
SELECT table_schema, table_name
FROM information_schema.views
WHERE table_name = 'staff_locations_view';
SELECT * FROM staff_locations_view;
SELECT staff_id, first_name, last_name, "position", location
FROM "Staff";
SELECT COUNT(*) FROM "Customer";
SELECT COUNT(*) FROM "Sales_Outlet";
SELECT COUNT(*) FROM sales_transaction;
DROP VIEW IF EXISTS staff_locations_view;

CREATE OR REPLACE VIEW staff_locations_view AS
SELECT staff_id,
       first_name,
       last_name,
       location
FROM staff
WHERE position NOT IN ('CEO', 'CFO');
COPY (SELECT * FROM staff_locations_view)
TO '"C:\Users\natas\OneDrive\Documents2\School\In Github\Relational Database Basics - COMPLETED\CoffeeData.sql"'
WITH CSV HEADER;
```

```
Total rows: 1 of 1    Query complete 00:00:01.225    Ln 38, Col 1
```