SQL Article

Assignment title: **How to Connect PostgreSQL to Power BI Using Local PostgreSQL and Aiven.**

Write and publish a technical article on Dev.to or Medium explaining how to connect Power BI to PostgreSQL running locally and how to connect to PostgreSQL hosted on Aiven, including key steps, screenshots with brief explanations, and relevant code snippets or commands in code blocks. Once published, make sure the article is public, then copy the article URL and submit that link via the Google Forms link that has been shared with you.

SQL ASSIGNMENT QUESTIONS (HEALTHCARE DATASET)

1. Retrieve the list of all male patients who were born after 1990, including their patient ID, first name, last name, and date of birth.

2. Produce a report showing the ten most recent appointments in the system, ordered from the newest to the oldest.

3. Generate a report that shows all appointments along with the full names of the patients and doctors involved.

4. Prepare a list that shows all patients together with any treatments they have received, ensuring that patients without treatments also appear in the results.

5. Identify any treatments recorded in the system that do not have a matching appointment.

6. Create a summary that shows how many appointments each doctor has handled, ordered from the highest to the lowest count.

7. Produce a list of doctors who have handled more than twenty appointments, showing their doctor ID, specialization, and total appointment count.

8. Retrieve the details of all patients who have had appointments with doctors whose specialization is “Cardiology.”

9. Produce a list of patients who have at least one bill that remains unpaid.

10. Retrieve all bills whose total amount is higher than the average total amount for all bills in the system.

11. For each patient in the database, identify their most recent appointment and list it along with the patient's ID.

12. For every appointment in the system, assign a sequence number that ranks each patient's appointments from most recent to oldest.

13. Generate a report showing the number of appointments per day for October 2021, including a running total across the month.

14. Using a temporary query structure, calculate the average, minimum, and maximum total bill amount, and then return these values in a single result set.

15. Build a query that identifies all patients who currently have an outstanding balance, based on information from admissions and billing records.

16. Create a query that generates all dates from January 1 to January 15, 2021, and show how many appointments occurred on each of those dates.

17. Add a new patient record to the Patients table, providing appropriate information for all required fields.

18. Modify the appointments table so that any appointment with a NULL status is updated to show "Scheduled."

19. Remove all prescription records that belong to appointments marked as "Cancelled."

20. Create a stored procedure that adds a new record to the Doctors table.

The procedure should accept the doctor's ID, first name, last name, specialization, email, and phone number as input parameters.

After creating the procedure, call it using a set of sample doctor details to insert a new doctor into the database.

21. Create a stored procedure that records a new appointment and automatically performs validation before inserting.

The procedure should accept an appointment ID, patient ID, doctor ID, appointment date, status, and nurse ID.

Inside the procedure, implement the following logic:

* Verify that the patient exists in the Patients table.

* Verify that the doctor exists in the Doctors table.

* If either does not exist, prevent the insertion and return an error message.

* If both exist, insert the appointment into the Appointments table.

After creating the procedure, call it with sample data to demonstrate both a successful and a failed insertion attempt.

Importing Data ditectly into postgress

(for those who were not in class)

Prerequisites

1. Open DBeaver and connect to your PostgreSQL database.
2. Decide the **target schema** (e.g., `public` or `hospital`). You can create a schema called hospital.

Option A - Create a new table from CSV (Wizard)

1. In **Database Navigator**, expand your connection → expand your **database** → expand **Schemas** (e.g., `hospital`).
2. Right-click the **schema** you want (e.g., `hospital`) → **Import Data…**
3. In the import wizard:
    - **Source**: choose **CSV** → click **Next**.
    - **Files**: click **Add…**, select your `.csv`.
    - Check **Header** if the first row contains column names.
    - Set **Delimiter** (`,`), **Quote** (`"`), **Encoding** (`UTF-8`).
    - Click **Next**.
4. **Target**:
    - Choose **New** table.
    - Give it a name (e.g., `patients`).
    - Ensure the **Target container** is your schema (e.g., `hospital`).
    - Click **Next**.
5. **Columns mapping & data types** (this is where you fix types):
    - You'll see each CSV column mapped to a new table column.
    - For each column:
        - **Name**: rename to clean, lowercase, `snake_case` (e.g., `first_name`) to avoid quoting later.
        - **Data type**: click the type cell and change as needed:
            - IDs/keys: `text` (safe if they may contain letters/leading zeros)
            - Names/emails/phones: `text`
            - Integers: `integer` or `bigint`

        - Money/amounts: `numeric(12,2)` (or similar)
        - Dates: `date` (set the **date format** on the next page if needed)
        - Timestamps: `timestamp` (set a parse format if needed)
        - Booleans: `boolean` (`true/false`, `1/0`, or `Y/N`; adjust mapping if needed)
    - If a CSV column name has spaces/caps and you **don't** rename it, DBeaver will create `"QuotedNames"`; you'll need double quotes in SQL forever. It's better to rename to lowercase now.
    - Click **Next**.
6. **Format** page:
    - Confirm **Date format** (e.g., `yyyy-MM-dd`) if you used `date`/`timestamp`.
    - Set **Null string** (e.g., empty → treat empty as NULL; or if the CSV uses `NULL`, add it here).
    - Click **Next**.
7. **Import settings** (PostgreSQL):
    - Tick **Use bulk load** (this uses `COPY` behind the scenes; faster and safer).
    - Batch/commit defaults are fine; leave **Auto-commit** on.
    - Click **Finish** to run.
8. Verify:
    - Right-click the new table → **Read data** (or run `SELECT * FROM hospital.patients;`)

How to change data types (after or during import)

- **During the wizard** (Option A new table): on the **Columns** page, click the **Data type** cell for each column and pick the correct PostgreSQL type. Also set **date/timestamp formats** on the Format page.
- **After import**:
    - Right-click table → **Generate SQL** → **DDL** to see current types.
    - Use `ALTER TABLE` to adjust:
    - `ALTER TABLE hospital.patients`
    - `  ALTER COLUMN date_of_birth TYPE date USING date_of_birth::date,`
    - `  ALTER COLUMN phone_number TYPE text;`
    - For numeric text that includes commas/`$`, clean first (e.g., via staging table or using `replace()` in an UPDATE).

Referencing "quoted" columns with " " (identifiers)

- If a column/table was created with capitals or spaces (e.g., `"First Name"`, `"TotalAmount"`), you **must** always reference it with **double quotes** and exact case:
- `SELECT "First Name", "TotalAmount" FROM hospital."Bills";`
- Best practice: **avoid quoted identifiers**. Use lowercase `snake_case` at import time so you can write:
- `SELECT first_name, total_amount FROM hospital.bills;`

Creating relationships (foreign keys) optional here

For this assignment you **don't need** relationships, but if you ever do:

In DBeaver UI

1. Right-click the child table (e.g., `appointments`) → **Edit Table**.
2. Go to **Foreign Keys** → **New**.
3. Pick the **referenced table** (e.g., `patients`) and map `appointments.patientid` → `patients.patientid`.
4. Choose actions (**ON UPDATE/ON DELETE**) if needed → **Save**.

SQL

```
ALTER TABLE hospital.appointments
  ADD CONSTRAINT fk_appt_patient
  FOREIGN KEY (patientid)
  REFERENCES hospital.patients(patientid)
  ON UPDATE CASCADE
  ON DELETE RESTRICT;
```

Again, not necessary for this assignment—but ensure **data types match** across related columns (e.g., both `text` or both `uuid`).

Submission Instructions

- Push your work to a **public GitHub repo** with: `functions.sql`, `stored_procedures.sql`, `ctes.sql`, and a brief `README.md`.
- Publish a **blog post** on your article
- **Submit two links** (GitHub repo + blog post) via this form: https://forms.gle/roW5Khh4E2siZtPT7 by **23rd November 2025**
- Use **PostgreSQL**; ensure queries run and objects are named clearly.

Assignment 3: Group Work

Power BI Healthcare Analytics

Goal

Connect to the PostgreSQL healthcare dataset and produce a complete Power BI report: data import → cleaning → modeling → measures (named only) → dashboards → publish & refresh.

Data access (choose one)

- **Local PostgreSQL (DBeaver)**
  Server: `localhost:5432` (or `127.0.0.1:5432`)
  Database: your local DB
- **Aiven PostgreSQL**
  Server: `<your Aiven host>:<port>` (use the exact hostname/port from Aiven)
  Database: `defaultdb`
  Notes: install the Aiven CA certificate in Windows Trusted Root; use Database authentication; keep "Encrypt connection" on.

Part 1 Staging (recommended)

Create three **database views** (or equivalent SQL queries you'll import) so modeling is clean:

1. **Appointments Enriched**
   Includes: AppointmentID, PatientID, DoctorID, AppointmentDate, Status (normalized), patient first/last, doctor first/last, specialization, plus a pure date column extracted from AppointmentDate.
2. **Patient Balances**
   Aggregates per PatientID: Total Billed, Total Paid, Total Outstanding (from Admissions ↔ Bills).
3. **Doctor Monthly Metrics**
   Per month & DoctorID: total appointments, cancelled appointments, cancellation rate.

Part 2 Import & Cleaning (Power Query)

- Get Data → PostgreSQL → connect (per your option above).
- **Select only the three views** (or your equivalent queries), not all raw tables.
- In Power Query:
  - Rename columns to readable names (title case, spaces OK).
  - Set correct **data types**: dates to *Date*, amounts to *Decimal Number*, IDs to *Text*.
  - Trim/clean text, replace blanks with nulls where appropriate.
  - Create (if needed) a Year, Month, and Year-Month text column for easy visuals.
  - Close & Apply.

Part 3 Date table

Create a **Date** table covering the full range of your appointments.
You may create it in **Power Query** (preferred for this assignment) or via DAX if you already know how.
Mark it as the **Date table** in Model view.

Part 4 Data model

- Relationships:
  - **Date[Date] → Appointments Enriched[Appointment Date]** (single direction).
- Optional dimension links (if you import Patients/Doctors separately).
- Ensure relationship directions and cardinality are appropriate (avoid bi-directional filters unless required).

Part 5 Measures

Create the calculations needed to answer these business questions clearly and reliably:

- Overall appointment activity and its change over time
- Cancellations and their proportion relative to total activity
- Year-to-date appointment performance compared with prior periods
- Total amounts billed, amounts paid, and current outstanding balances
- Distinct patient volume and typical billing per patient
- Monthly performance at the doctor level, including volume and cancellation behaviour

Document each calculation (name, purpose, and data fields used) in a short notes page or README.

Part 6 Report pages (outcomes only)

Design at least three report pages that collectively deliver:

Executive Overview

- A compact summary of the most important operational and financial outcomes
- A clear time-based view for trend monitoring
- Interactive filtering for period and clinical specialization

Appointments Analysis

- Day-level or month-level activity over time
- Comparisons across specializations and/or individual doctors

- A way to identify the highest and lowest performers based on your chosen criteria

Financials

- Period-based view of billed, paid, and outstanding amounts
- A patient-level view that highlights those with the largest balances
- An insight that shows how much of the total is concentrated among a subset of patients or periods

(Optional) Drill Detail

- Record-level detail (appointments joined to patient and doctor context) with filters for status and specialization

General requirements:

- Use a proper Date table and ensure it drives time analysis across the model.
- Ensure all visuals interact appropriately (cross-filter/cross-highlight).
- Provide clear titles, labels, and helpful tooltips.
- Keep page layout consistent and readable.

Part 7 Publish & Refresh

- Publish the `.pbix` to your workspace.
- If source is local PostgreSQL or Aiven:
  - Install **On-premises Data Gateway** on a machine that can reach the DB.
  - Add a **PostgreSQL data source** with the same Server/Database/User.
  - For Aiven, ensure the **CA certificate** is installed on the gateway machine.
  - Map the dataset to the gateway and configure **scheduled refresh**.

Deliverables

- **GitHub repo** (public):
  - `/sql/` folder with the view scripts or the exact SQL queries used for import
  - the `.pbix` file
  - a short `README.md` with connection notes (no passwords), data model diagram/screenshot, and any assumptions
- **Blog post**: brief write-up of connection, cleaning, modeling choices, and dashboard screenshots (no formulas)
- Submit both links to breebridgit5@gmail.com by **29th November 2025**

Notes & constraints

- Relationships beyond Date → Appointments are **not required** for this assignment, but **data types must be correct**.

- If you encounter Aiven SSL or connection-limit issues, follow the class notes (install Aiven CA; disable parallel table loading; or use Aiven's connection pooler).
- Keep column names student-friendly in Power BI (rename in Power Query if needed).