Database Lab 2 Report

Course: Database Management
Lab Number: *Lab 2*
Date: *2025-01-29*
Name: *Ryan Munger*

1. Objective

- More practice getting around in the PostgreSQL and pgAdmin environments.
- Familiarize yourself with the CAP database data.
- Get more easy lab points.

2. Lab Setup

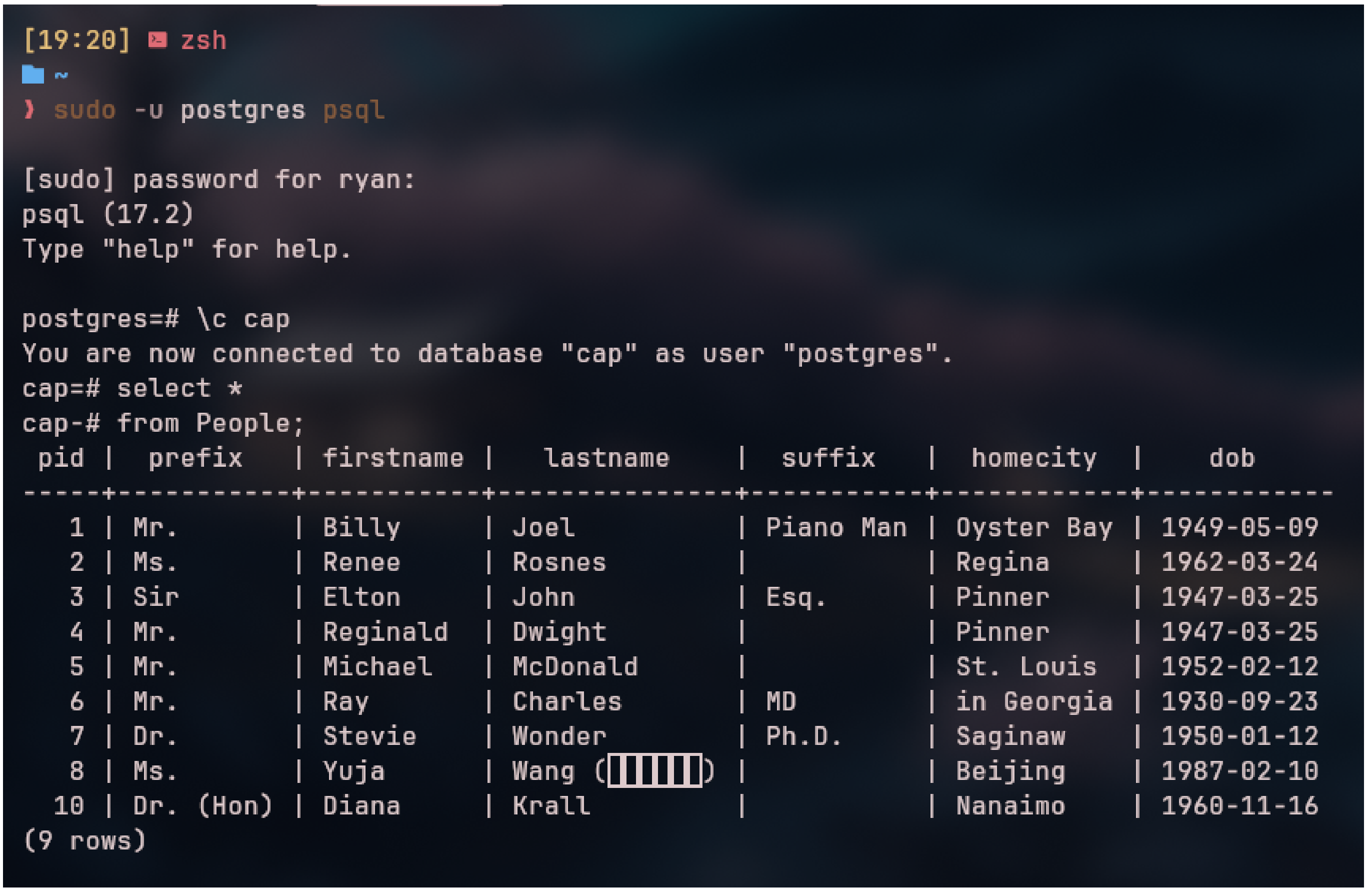
Create the CAP database in PostgreSQL.

3. Procedure

Part 1: *Querying CAP*

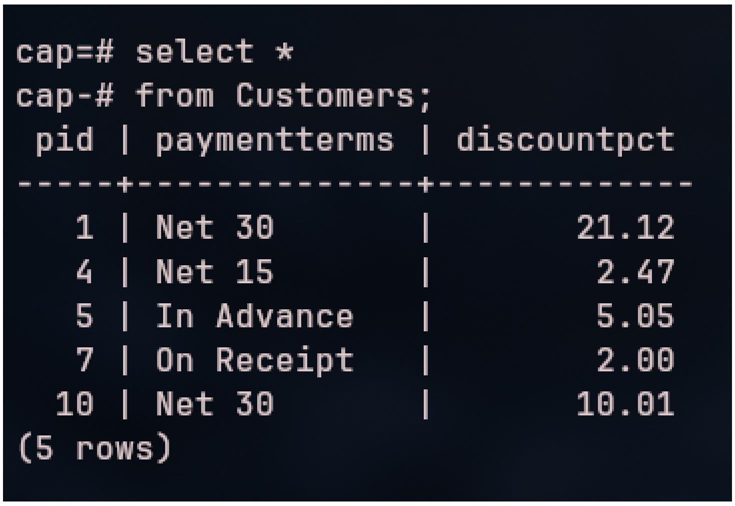
Execute the following queries (one at a time), take a screenshot, and verify its correctness.

```
select *
from People;
```



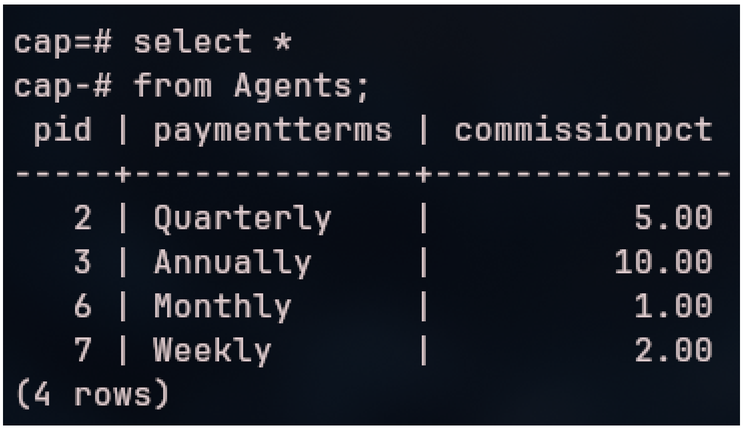
```
[19:20] zsh
~
❯ sudo -u postgres psql

[sudo] password for ryan:
psql (17.2)
Type "help" for help.

postgres=# \c cap
You are now connected to database "cap" as user "postgres".
cap=# select *
cap-# from People;
 pid |   prefix    | firstname |    lastname     |   suffix    |  homecity   |    dob
-----+-------------+-----------+-----------------+-------------+-------------+------------
   1 | Mr.         | Billy     | Joel            | Piano Man | Oyster Bay | 1949-05-09
   2 | Ms.         | Renee     | Rosnes          |             | Regina      | 1962-03-24
   3 | Sir         | Elton     | John            | Esq.        | Pinner      | 1947-03-25
   4 | Mr.         | Reginald  | Dwight          |             | Pinner      | 1947-03-25
   5 | Mr.         | Michael   | McDonald        |             | St. Louis   | 1952-02-12
   6 | Mr.         | Ray       | Charles         | MD          | in Georgia | 1930-09-23
   7 | Dr.         | Stevie    | Wonder          | Ph.D.       | Saginaw     | 1950-01-12
   8 | Ms.         | Yuja      | Wang (▯▯▯▯▯▯) |             | Beijing     | 1987-02-10
  10 | Dr. (Hon) | Diana     | Krall           |             | Nanaimo     | 1960-11-16
(9 rows)
```

```
select *
from Customers;
```



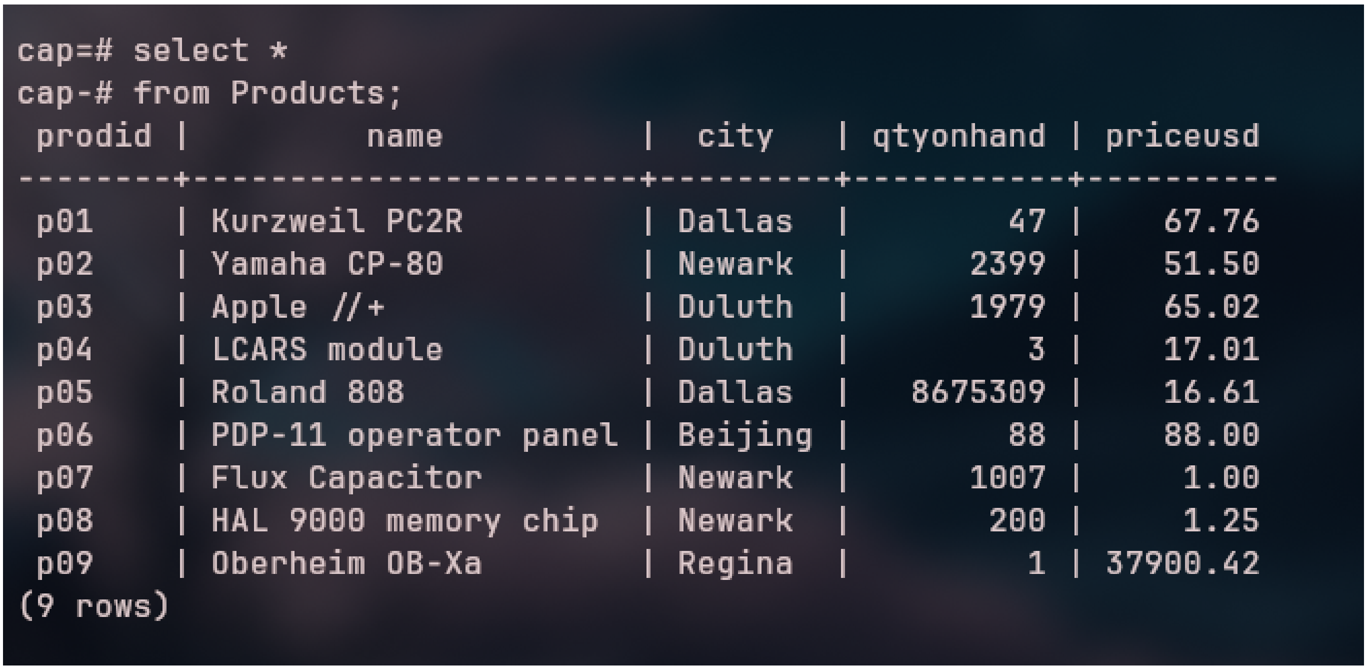
```
cap=# select *
cap-# from Customers;
 pid | paymentterms | discountpct
-----+--------------+-------------
   1 | Net 30       |       21.12
   4 | Net 15       |        2.47
   5 | In Advance   |        5.05
   7 | On Receipt   |        2.00
  10 | Net 30       |       10.01
(5 rows)
```

```
select *
from Agents;
```



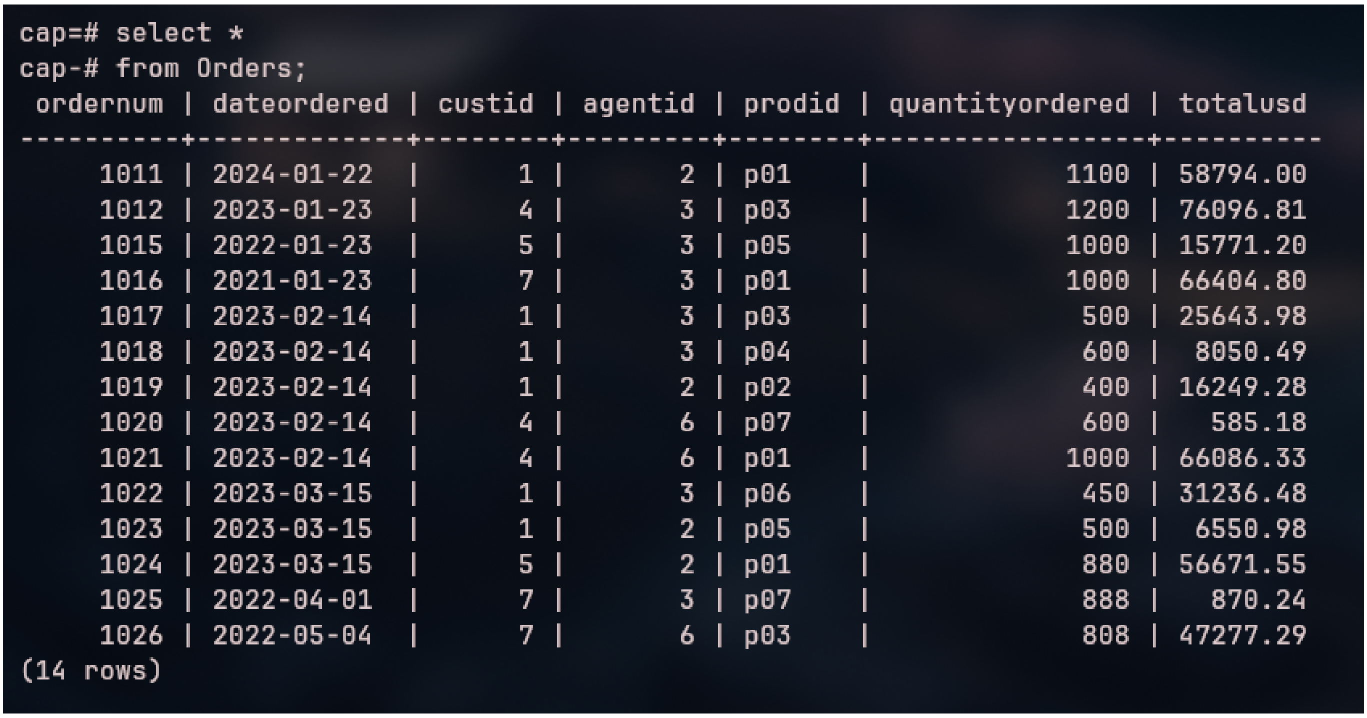
```
cap=# select *
cap-# from Agents;
 pid | paymentterms | commissionpct
-----+--------------+---------------
   2 | Quarterly    |          5.00
   3 | Annually     |         10.00
   6 | Monthly      |          1.00
   7 | Weekly       |          2.00
(4 rows)
```

```
select *
from Products;
```



```
cap=# select *
cap-# from Products;
 prodid |          name          |  city   | qtyonhand | priceusd
--------+------------------------+---------+-----------+----------
 p01    | Kurzweil PC2R          | Dallas  |        47 |    67.76
 p02    | Yamaha CP-80           | Newark  |      2399 |    51.50
 p03    | Apple //+              | Duluth  |      1979 |    65.02
 p04    | LCARS module           | Duluth  |         3 |    17.01
 p05    | Roland 808             | Dallas  |   8675309 |    16.61
 p06    | PDP-11 operator panel  | Beijing |        88 |    88.00
 p07    | Flux Capacitor         | Newark  |      1007 |     1.00
 p08    | HAL 9000 memory chip   | Newark  |       200 |     1.25
 p09    | Oberheim OB-Xa         | Regina  |         1 | 37900.42
(9 rows)
```

```
select *
from Orders
```



```
cap=# select *
cap-# from Orders;
 ordernum | dateordered | custid | agentid | prodid | quantityordered | totalusd
----------+-------------+--------+---------+--------+-----------------+----------
     1011 | 2024-01-22  |      1 |       2 | p01    |            1100 | 58794.00
     1012 | 2023-01-23  |      4 |       3 | p03    |            1200 | 76096.81
     1015 | 2022-01-23  |      5 |       3 | p05    |            1000 | 15771.20
     1016 | 2021-01-23  |      7 |       3 | p01    |            1000 | 66404.80
     1017 | 2023-02-14  |      1 |       3 | p03    |             500 | 25643.98
     1018 | 2023-02-14  |      1 |       3 | p04    |             600 |  8050.49
     1019 | 2023-02-14  |      1 |       2 | p02    |             400 | 16249.28
     1020 | 2023-02-14  |      4 |       6 | p07    |             600 |   585.18
     1021 | 2023-02-14  |      4 |       6 | p01    |            1000 | 66086.33
     1022 | 2023-03-15  |      1 |       3 | p06    |             450 | 31236.48
     1023 | 2023-03-15  |      1 |       2 | p05    |             500 |  6550.98
     1024 | 2023-03-15  |      5 |       2 | p01    |             880 | 56671.55
     1025 | 2022-04-01  |      7 |       3 | p07    |             888 |   870.24
     1026 | 2022-05-04  |      7 |       6 | p03    |             808 | 47277.29
(14 rows)
```

Part 2: *Keys*

Explain the distinctions among the terms primary key, candidate key, and superkey.

A superkey is any field or set of fields in a table that can uniquely identify every row in said table. It often has unnecessary attributes.

A candidate key is a minimal superkey. Like the superkey, it can uniquely identify each row in the table. "Minimal superkey" means that it is the superkey with a small set of columns/attributes required to ensure the uniqueness.

The primary key is the candidate key selected to provide uniqueness. The primary key is the most minimal candidate and can be used for table relations. The primary keys should not be null nor should they be defined by the user. We should use artificial keys like CWID wherever possible.

Summary: superkey is any set of fields that ensures uniqueness, candidate key is minimal superkey, and primary key is the best candidate key.

Part 3: *Data Types*

Write a short essay on data types. Select a topic for which you might create a table. Name the table and list its fields (columns). For each field, give its data type and whether or not it is nullable

Data types good. Essay done. Despite the obvious value of data types, we need to appreciate that they restrict the kind of data that can be stored in each column of a table. The data in each column fits specific format and is comparable. Common data types in databases include integers, floating-point numbers, strings, dates, and booleans.
Topic: Job applicants (its always on my mind...)

Applicants: (This isn't a DB table its a reverse table describing the columns)

Field Name	Data Type	Nullable	Description

Field Name	Data Type	Nullable	Description
`applicant_id`	`INT`	No	Unique applicant identifier (Primary Key)
`first_name`	`STRING`	No	First name
`last_name`	`STRING`	No	Last name
`email`	`STRING`	No	Email
`phone_number`	`STRING`	Yes	Phone - not required
`position_applied`	`STRING`	No	Position applied for
`application_date`	`DATE`	No	Date application was submitted
`cover_letter`	`TEXT`	Yes	Cover letter - optional

Part 4: *Relational Rules*

1. First Normal Form
    1. This rule dictates that there are to be no multi-valued attributes or values with internal structure at any intersection of a row and a column in a table (original description... definitely not the slides talking). All values at the intersection must be atomic (cannot be divided further). An example of this would be a column for email that has someone's home and work emails both listed separated by a comma. This field is not atomic, and should be divided into "home_email" and "work_email" attributes.
2. Access Rows by Content Only
    1. This rule states that data should be queried based on *what it is* (content) rather than *where is is* (row/col). For example, querying "What is the first name of pid 007?" is valid because this is content based and will work no matter how the table is ordered. However, asking, "What is the name in the first row?" references an arbitrary and temporary table location. As new data is added, the table is ordered, or defragmentation occurs, the table rows will shift positions. Your location based query will no longer work (or even worse - return a different record!).
3. All Rows must be Unique
    1. This rule declares (without trumpets) that all rows need to be distinguishable from every other row. The uniqueness of rows is essential to data integrity and querying. For example, if two rows were the same, it would be unclear which row a query actually pertained to. This rule is not as heavily enforced, as we saw in class. However, restrictions such as primary keys will often prevent duplicate rows from being created.