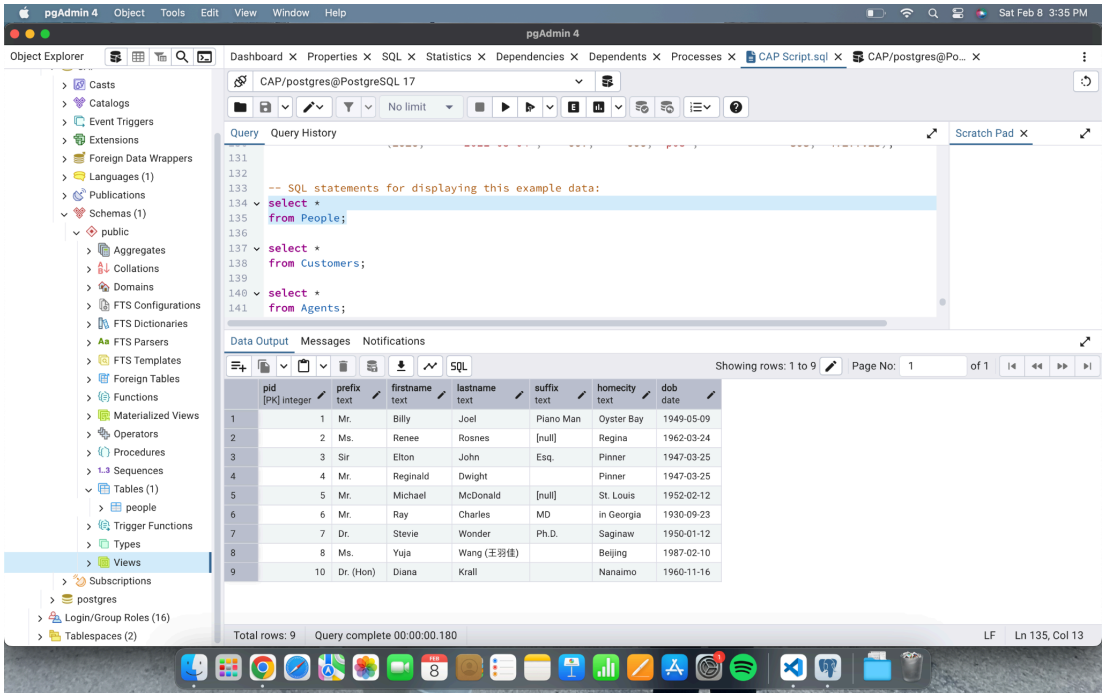
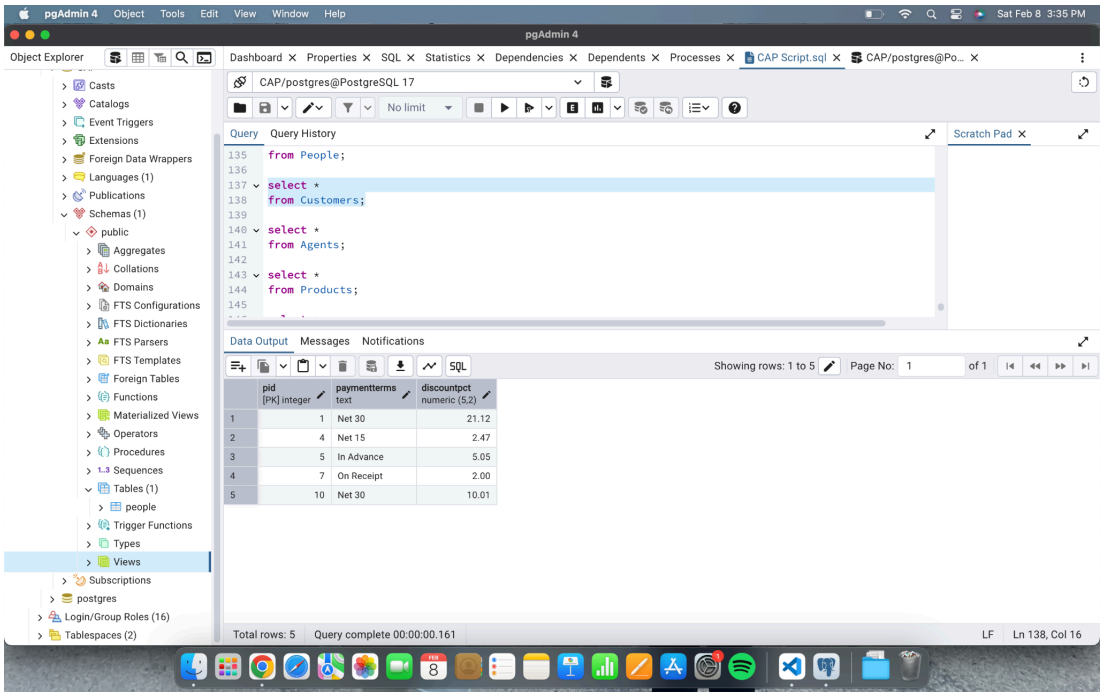
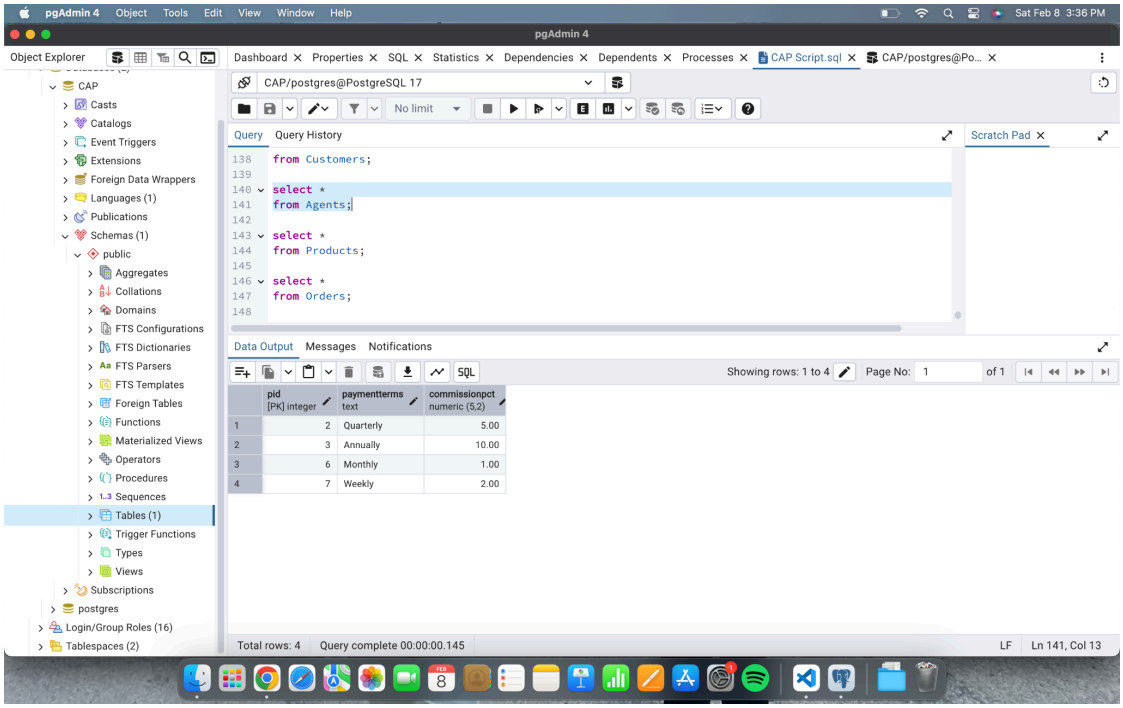
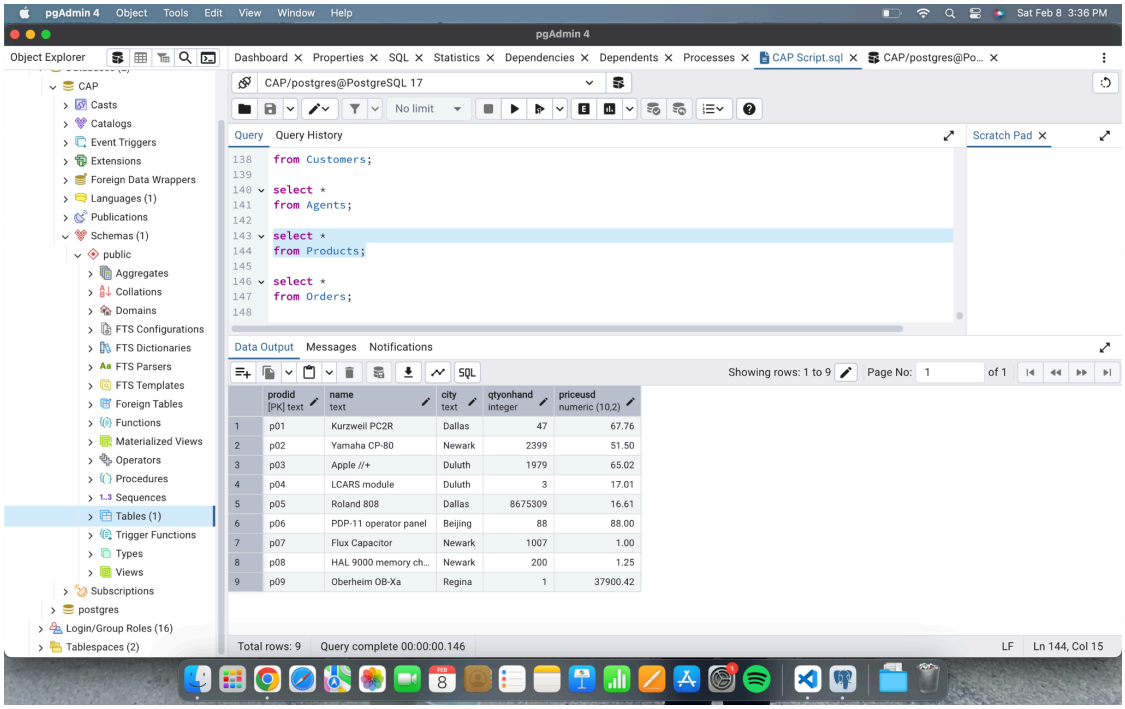
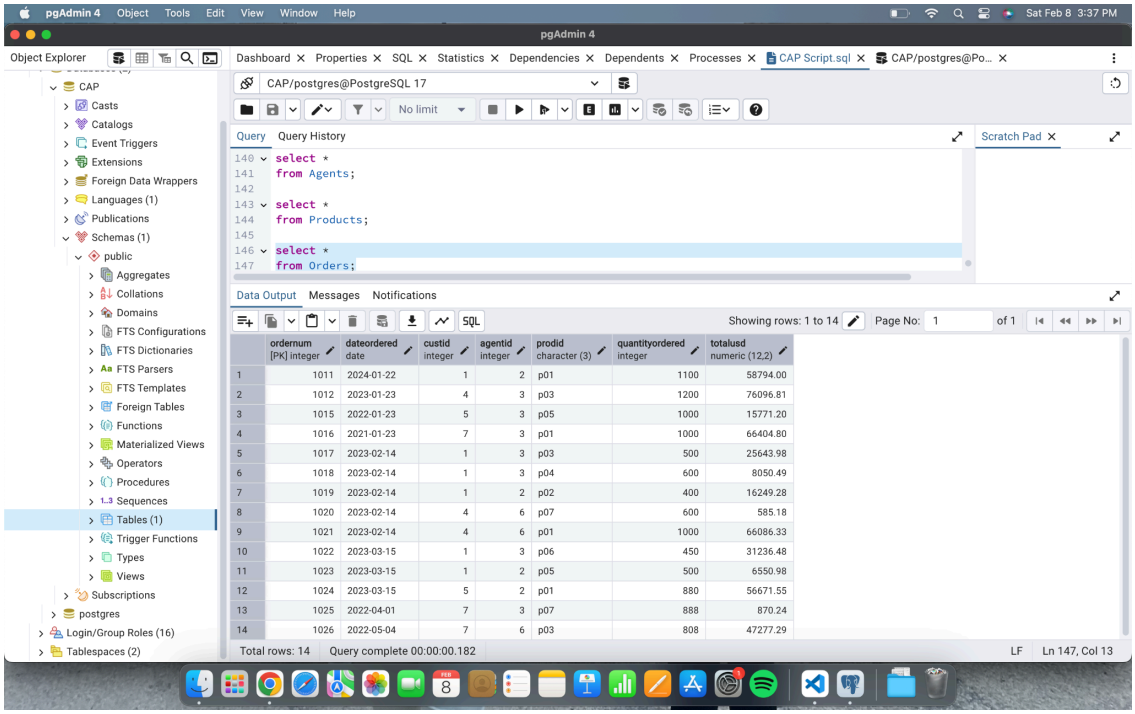
1)


pgAdmin 4
Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
CAP Script.sql
CAP/postgres@PostgreSQL 17
Query
Query History
Scratch Pad
-- SQL statements for displaying this example data:
select *
from People;
select *
from Customers;
select *
from Agents;
Data Output
Messages
Notifications
Showing rows: 1 to 9
Page No: 1
of 1
pid [PK] integer
prefix text
firstname text
lastname text
suffix text
homecity text
dob date
1 1 Mr. Billy Joel Piano Man Oyster Bay 1949-05-09
2 2 Ms. Renee Rosnes [null] Regina 1962-03-24
3 3 Sir Elton John Esq. Pinner 1947-03-25
4 4 Mr. Reginald Dwight Pinner 1947-03-25
5 5 Mr. Michael McDonald [null] St. Louis 1952-02-12
6 6 Mr. Ray Charles MD in Georgia 1930-09-23
7 7 Dr. Stevie Wonder Ph.D. Saginaw 1950-01-12
8 8 Ms. Yuja Wang (王羽佳) Beijing 1987-02-10
9 10 Dr. (Hon) Diana Krall Nanaimo 1960-11-16
Total rows: 9
Query complete 00:00:00.180
LF
Ln 135, Col 13


pgAdmin 4
Object Explorer
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
CAP Script.sql
CAP/postgres@PostgreSQL 17
Query
Query History
Scratch Pad
from People;
select *
from Customers;
select *
from Agents;
select *
from Products;
Data Output
Messages
Notifications
Showing rows: 1 to 5
Page No: 1
of 1
pid [PK] integer
paymentterms text
discountpct numeric (5,2)
1 1 Net 30 21.12
2 4 Net 15 2.47
3 5 In Advance 5.05
4 7 On Receipt 2.00
5 10 Net 30 10.01
Total rows: 5
Query complete 00:00:00.161
LF
Ln 138, Col 16


pgAdmin 4
Object Explorer
CAP/postgres@PostgreSQL 17
Query
Query History
Scratch Pad
from Customers;
select *
from Agents;
select *
from Products;
select *
from Orders;
Data Output
Messages
Notifications
Showing rows: 1 to 4
Page No: 1 of 1
pid [PK] integer
paymentterms text
commissionpct numeric (5,2)
1 2 Quarterly 5.00
2 3 Annually 10.00
3 6 Monthly 1.00
4 7 Weekly 2.00
Total rows: 4
Query complete 00:00:00.145
Ln 141, Col 13


pgAdmin 4
Object Explorer
CAP/postgres@PostgreSQL 17
Query
Query History
Scratch Pad
from Customers;
select *
from Agents;
select *
from Products;
select *
from Orders;
Data Output
Messages
Notifications
Showing rows: 1 to 9
Page No: 1 of 1
prodid [PK] text
name text
city text
qtyonhand integer
priceusd numeric (10,2)
1 p01 Kurzweil PC2R Dallas 47 67.76
2 p02 Yamaha CP-80 Newark 2399 51.50
3 p03 Apple //+ Duluth 1979 65.02
4 p04 LCARS module Duluth 3 17.01
5 p05 Roland 808 Dallas 8675309 16.61
6 p06 PDP-11 operator panel Beijing 88 88.00
7 p07 Flux Capacitor Newark 1007 1.00
8 p08 HAL 9000 memory ch... Newark 200 1.25
9 p09 Oberheim OB-Xa Regina 1 37900.42
Total rows: 9
Query complete 00:00:00.146
Ln 144, Col 15



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

These particular keys go down in order, from super key, to candidate key, then to primary key. Super keys are any column / set of columns that uniquely identify every row in a given table. This is the most broad key, in that it will take in any attribute that uniquely identifies a row, so in the CAP people table for example, it allows pid, first name, last name, and dob to be super keys. Candidate keys will narrow down these super keys, as they are defined to be a minimal super key. This allows us to take away first and last name and keep pid and dob, as these attributes have more of a chance to be unique identifiers, as names can certainly repeat. Primary keys narrow down the candidate key to just one, so in this case, we would choose pid since it is guaranteed to be different in each row; making this the most specific key out of these three.

3. 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.

As mentioned in class, many data types that we already know of exist in SQL, such as integers and “text,” which upon some quick research, is a variation of a string. We also mentioned in class that there will be some new data types that we haven’t seen yet, such as date / timestamps. These data types allow for certain columns to have proper assignment, where one column can only contain one specific data type. Without data types, the data in our databases cannot be properly stored.

Now, let's create an example. Say we are looking at a table of various NBA players, titled just "NBA Players" where the fields include: pid, first name, last name, current team, years on contract, salary, and date of first game played (YYYY-MM-DD). The fields with text as its data type would be first name, last name, and current team. There would be two integer fields with years on contract and pid, one decimal field for their salary (as some salaries may go down to a specific cent), and one date data type for their first game played. For this specific example, I am going to say that each field is not nullable, as every player in the NBA automatically has these associations just by being a player in the league, which will hold true in this table of players.

4. Explain the following relational "rules" with examples and reasons why they are important.
a. The "first normal form" rule

Definition - There can be no multi-valued attributes or values with internal structure at any intersection of a row and column of a table (no repeating groups or fields) -> All values must be **Atomic**, in that they cannot be subdivided (i.e. don't particularly want first and last name in the same column, instead have one for first name and one for last).

This rule is important because a convoluted attribute can be really confusing, and it is just bad design. From the James Bond example in class, the skills column contained multiple as opposed to just one, and this is a violation. Splitting them up into "skill1" and "skill2" simplifies it and fixes this violation, since the multi-valued attribute has been dealt with. However, this can still be improved, where we can create a new table referring to just the skills, and the violation will most definitely have been removed.

Referring to the beloved CAP database, the People table is a great example of a table that does not violate this rule, as there are no repeating groups or fields, and each value is atomic. Each attribute is uniquely defined, and we have a primary key of pid to ensure that each row is uniquely defined with no repeats. Say there happens to be a row with two people that have the same first and last name, same date of birth, same home city, suffix, and prefix. These two rows will still be unique, as their pid's will be different, signifying that they are different people.

b. The "access rows by content only" rule

Definition - We can only ask for (query) data by what is there, never by where it is. This means that we need to be very specific in what we are looking for, as our tables are ultimately sets, and have no intrinsic order. For example (using my NBA table), we cannot ask "what is the current team in the second row?" Instead, we should ask "What is the current for pid 2?"

This rule is important as we (should) want to be as specific as possible when searching what we want to look for. By asking the question that we should ask, we get right to the point, and we don't have to incorrectly think that our table has an intrinsic order. Just by asking for data specifically regarding pid 2, we can get anything we may need from it.

To build off of the above example, the correct way access our rows is the outline for a basic SQL query, so translate to it, we would say "SELECT currentTeam | FROM NBA Players | WHERE pid = 2;" overall demonstrating that this language properly follows this rule, enforcing you to create your queries the correct way (at least in Postgres).

c. The “all rows must be unique” rule

The definition is practically in the name, in that every row truly must be unique. If two rows were identical, we would not be able to identify their differences.

This is why something like a pid, or in our Marist case, a CWID is necessary, in that it establishes uniqueness, and would allow us to see what subtypes a certain person or other object is part of. For example, in the CAP database, we can see that Stevie Wonder is both a customer and an agent. Without the use of pid, we probably would not be able to point that out, and it can create confusion in that we could think that the customer and agent are two different people. Using the same example from part a, uniqueness is super necessary with the use of an ID. As mentioned, we could have two people with the same first name, last name, DOB, and so on, but because an ID is present, the rows automatically become unique (further demonstrating the importance of primary keys).

Overall, this row cannot be overlooked, as a duplicate row is a big mistake in practice.