```
-- DROPPING TABLES BEFORE CREATING IF EXISTS --
DROP TABLE IF EXISTS Workfor;
DROP TABLE
DROP TABLE IF EXISTS Own;
DROP TABLE
DROP TABLE IF EXISTS Stockedby;
DROP TABLE
DROP TABLE IF EXISTS Orderitems;
DROP TABLE
DROP TABLE IF EXISTS Orders;
DROP TABLE
DROP TABLE IF EXISTS Products;
DROP TABLE
DROP TABLE IF EXISTS Stores;
DROP TABLE
DROP TABLE IF EXISTS Vehicles;
DROP TABLE
DROP TABLE IF EXISTS Shoppers;
DROP TABLE
DROP TABLE IF EXISTS Customers;
DROP TABLE
DROP TABLE IF EXISTS UserPhone;
DROP TABLE
DROP TABLE IF EXISTS Users;
DROP TABLE

-- Entity tables below: --
CREATE TABLE Users (
      user_id VARCHAR(100) NOT NULL PRIMARY KEY,
      email VARCHAR(100),
      first_name VARCHAR(100) NOT NULL,
      last_name VARCHAR(100) NOT NULL
);
CREATE TABLE
CREATE TABLE UserPhone (
      user_id VARCHAR(100) REFERENCES Users(user_id),
      kind VARCHAR(20)
      CHECK(kind IN ('HOME', 'OFFICE', 'MOBILE')) NOT NULL,
      number VARCHAR(20) NOT NULL,
      PRIMARY KEY(user_id, number)
);
CREATE TABLE
CREATE TABLE Customers (
      user_id VARCHAR(100) REFERENCES Users(user_id),
      PRIMARY KEY(user_id)
);
CREATE TABLE
CREATE TABLE Shoppers (
      user_id VARCHAR(100) REFERENCES Users(user_id),
      capacity INTEGER,
      PRIMARY KEY(user_id)
);
CREATE TABLE
CREATE TABLE Vehicles (
      state VARCHAR(15) NOT NULL,
      license_plate VARCHAR(20) NOT NULL,
      year INTEGER,
      model VARCHAR(50),
```

```
      make VARCHAR(50) NOT NULL,
      color VARCHAR(50) NOT NULL,
      PRIMARY KEY(state, license_plate)
);
CREATE TABLE
CREATE TABLE Stores (
      store_id VARCHAR(10) PRIMARY KEY,
      name VARCHAR(100) NOT NULL,
      street VARCHAR(100) NOT NULL,
      city VARCHAR(100) NOT NULL,
      state VARCHAR(100) NOT NULL,
      zip_code INTEGER NOT NULL,
      phone VARCHAR(100) NOT NULL,
      categories TEXT NOT NULL
);
CREATE TABLE
CREATE TABLE Products (
      product_id VARCHAR(50) PRIMARY KEY,

      category VARCHAR(100)
      CHECK(category IN ('Baby Care', 'Beverages', 'Bread & Bakery',
      'Breakfast & Cereal', 'Canned Goods & Soups', 'Condiments & Spice & Bake',
      'Cookies & Snacks & Candy', 'Dairy & Eggs & Cheese', 'Deli',
      'Frozen Foods', 'Fruits & Vegetables', 'Grains & Pasta & Sides',
      'Meat & Seafood', 'Paper & Cleaning & Home', 'Personal Care & Health',
      'Pet Care')) NOT NULL,

      name VARCHAR(100) NOT NULL,
      description VARCHAR(100) NOT NULL,
      list_price NUMERIC
);
CREATE TABLE
CREATE TABLE Orders (
      order_id VARCHAR(100) PRIMARY KEY,
      total_price NUMERIC,
      time_placed TIMESTAMP NOT NULL,
      pickup_time TIMESTAMP,
      customer_id VARCHAR(100) REFERENCES Customers(user_id) ON DELETE CASCADE,
      shopper_id VARCHAR(100) REFERENCES Shoppers(user_id) ON DELETE CASCADE,
      state VARCHAR(100) NOT NULL,
      license_plate VARCHAR(20) NOT NULL,
      FOREIGN KEY (state, license_plate) REFERENCES Vehicles(state, license_plate)
ON DELETE CASCADE,
      store_id VARCHAR(10) REFERENCES Stores(store_id) ON DELETE CASCADE,
      time_fulfilled TIMESTAMP
);
CREATE TABLE
CREATE TABLE OrderItems (
      item_id VARCHAR(50),
      qty INTEGER NOT NULL,
      selling_price NUMERIC NOT NULL,
      order_id VARCHAR(100) REFERENCES Orders(order_id) ON DELETE CASCADE,
      product_id VARCHAR(100) REFERENCES Products(product_id) ON DELETE CASCADE,
      PRIMARY KEY(item_id, order_id)
);
CREATE TABLE

-- Relationship tables below: --
CREATE TABLE StockedBy (
```

```
      product_id VARCHAR(100) REFERENCES Products(product_id) ON DELETE CASCADE,
      store_id VARCHAR(100) REFERENCES Stores(store_id) ON DELETE CASCADE,
      qty INTEGER NOT NULL,
      PRIMARY KEY(product_id, store_id)
);
CREATE TABLE
CREATE TABLE Own (
      state VARCHAR(100) NOT NULL,
      license_plate VARCHAR(20) NOT NULL,
      FOREIGN KEY (state, license_plate) REFERENCES Vehicles(state, license_plate)
ON DELETE CASCADE,
      user_id VARCHAR(100) REFERENCES Users(user_id) ON DELETE CASCADE,
      PRIMARY KEY(state, license_plate, user_id)
);
CREATE TABLE
CREATE TABLE WorkFor (
      store_id VARCHAR(100) REFERENCES Stores(store_id) ON DELETE CASCADE,
      shopper_id VARCHAR(100) REFERENCES Shoppers(user_id) ON DELETE CASCADE,
      PRIMARY KEY(store_id, shopper_id)
);
CREATE TABLE

-- Data Loading --
copy users from '/Users/rew/Downloads/ShopALot.com Data/users.csv' delimiter ','
CSV HEADER;
COPY 5000
copy UserPhone from '/Users/rew/Downloads/ShopALot.com Data/phones.csv' delimiter
',' CSV HEADER;
COPY 5563
copy customers from '/Users/rew/Downloads/ShopALot.com Data/customers.csv'
delimiter ',' CSV HEADER;
COPY 5000
copy shoppers from '/Users/rew/Downloads/ShopALot.com Data/shoppers.csv' delimiter
',' CSV HEADER;
COPY 1000
copy vehicles from '/Users/rew/Downloads/ShopALot.com Data/vehicles.csv' delimiter
',' CSV HEADER;
COPY 5000
copy stores from '/Users/rew/Downloads/ShopALot.com Data/stores.csv' delimiter ','
CSV HEADER;
COPY 400
copy products from '/Users/rew/Downloads/ShopALot.com Data/products.csv' delimiter
',' CSV HEADER;
COPY 541
copy orders from '/Users/rew/Downloads/ShopALot.com Data/orders.csv' delimiter ','
CSV HEADER;
COPY 20000
copy orderitems from '/Users/rew/Downloads/ShopALot.com Data/orderitems.csv'
delimiter ',' CSV HEADER;
COPY 28646
copy stockedby from '/Users/rew/Downloads/ShopALot.com Data/stockedby.csv'
delimiter ',' CSV HEADER;
COPY 15701
copy own from '/Users/rew/Downloads/ShopALot.com Data/own.csv' delimiter ',' CSV
HEADER;
COPY 5559
copy workfor from '/Users/rew/Downloads/ShopALot.com Data/workfor.csv' delimiter
',' CSV HEADER;
COPY 1085
```

```
--Query Answers--
-- Problem A --
SELECT (SELECT COUNT(*) FROM stores) as stores_count,
      (SELECT COUNT(*) FROM customers) as customer_count,
      (SELECT COUNT(*) FROM products) as products_count;

 stores_count | customer_count | products_count
--------------+----------------+----------------
          400 |           5000 |            541
(1 row)

-- Problem B --
SELECT sp.user_id as Shoppers
FROM Shoppers sp, Stores s, Workfor w
WHERE sp.user_id = w.shopper_id
      and s.store_id = w.store_id
      and sp.capacity > 4
      and s.city = 'Seattle'
      and s.state = 'WA';

 shoppers
----------
 A75S3
(1 row)

-- Problem C --
SELECT s.store_id, s.name, COUNT(DISTINCT pr.product_id) as product_count
FROM Orderitems oi, Products pr, Stores s, Stockedby sb
WHERE sb.product_id = pr.product_id
      and sb.store_id = s.store_id
      and pr.product_id = oi.product_id
      and oi.selling_price < pr.list_price
GROUP BY s.store_id
ORDER BY COUNT(DISTINCT pr.product_id) DESC
LIMIT 10;

 store_id |           name            | product_count
----------+---------------------------+---------------
 FIB83    | Cracker Barrel Stores Inc |           487
 GBNTS    | Quick Chek Food Stores    |           484
 KGVM4    | Mapco                     |           483
 6Q67U    | Beasley Enterprises Inc   |           482
 VHNP8    | Express Mart Stores       |           482
 WTMYV    | Cubbys                    |           481
 HJI4K    | Maay Convenient Inc       |           480
 KCNOQ    | Express Mart Stores       |           480
 3MGOV    | Dandy Mini Mart           |           478
 XWYDM    | Irving Oil Corp           |           478
(10 rows)

-- Problem D --
SELECT o.order_id, o.total_price, o.time_placed
FROM orderitems oi, orders o
WHERE oi.order_id = o.order_id
      and time_placed > TO_TIMESTAMP('2020-05-01', 'YYYY-MM-DD')
      and time_placed < TO_TIMESTAMP('2020-07-01', 'YYYY-MM-DD')
      and oi.qty > 25;
```

```
 order_id | total_price |     time_placed
----------+-------------+---------------------
 SW6PI    |      624.56 | 2020-05-05 09:54:58
 ZWLJS    |       147.0 | 2020-05-19 11:24:28
 5IZ2R    |       168.3 | 2020-06-09 02:42:38
 FSX1T    |      233.16 | 2020-06-13 08:05:36
(4 rows)

-- Problem E --
SELECT AVG(o.total_price)
FROM Orders o
WHERE o.time_placed >= TO_TIMESTAMP('2020-03-25', 'YYYY-MM-DD')
      and o.time_placed < TO_TIMESTAMP('2020-04-1', 'YYYY-MM-DD');

         avg
---------------------
 53.8707571801566580
(1 row)

-- Problem F --
SELECT s.store_id, s.name, category_list, array_length(category_list, 1)
FROM Stores s, string_to_array(s.categories,', ') AS category_list
WHERE s.zip_code = 44401;

 store_id |             name             |
category_list
| array_length
----------+------------------------------
+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------+--------------
 FNJXE    | Pump-N-Pantry Of NY          | {Beverages,Deli,"Frozen Foods","Fruits &
Vegetables","Grains & Pasta & Sides","Personal Care & Health","Baby Care","Bread &
Bakery","Breakfast & Cereal","Meat & Seafood","Pet Care"} |           11
 FX0RG    | 6-Twelve Convenient-Mart Inc | {"Baby Care","Bread & Bakery","Canned
Goods & Soups","Frozen Foods","Fruits & Vegetables","Meat & Seafood"}
|            6
(2 rows)

-- Problem G --
SELECT s.store_id, s.name, category_list
FROM Stores s, UNNEST(string_to_array(s.categories,', ')) AS category_list
WHERE s.zip_code = 44401
ORDER BY s.store_id;

 store_id |             name             |     category_list
----------+------------------------------+-----------------------
 FNJXE    | Pump-N-Pantry Of NY          | Beverages
 FNJXE    | Pump-N-Pantry Of NY          | Deli
 FNJXE    | Pump-N-Pantry Of NY          | Frozen Foods
 FNJXE    | Pump-N-Pantry Of NY          | Fruits & Vegetables
 FNJXE    | Pump-N-Pantry Of NY          | Grains & Pasta & Sides
 FNJXE    | Pump-N-Pantry Of NY          | Personal Care & Health
 FNJXE    | Pump-N-Pantry Of NY          | Baby Care
 FNJXE    | Pump-N-Pantry Of NY          | Bread & Bakery
 FNJXE    | Pump-N-Pantry Of NY          | Breakfast & Cereal
 FNJXE    | Pump-N-Pantry Of NY          | Meat & Seafood
 FNJXE    | Pump-N-Pantry Of NY          | Pet Care
 FX0RG    | 6-Twelve Convenient-Mart Inc | Baby Care
```

```
 FX0RG    | 6-Twelve Convenient-Mart Inc | Bread & Bakery
 FX0RG    | 6-Twelve Convenient-Mart Inc | Canned Goods & Soups
 FX0RG    | 6-Twelve Convenient-Mart Inc | Frozen Foods
 FX0RG    | 6-Twelve Convenient-Mart Inc | Fruits & Vegetables
 FX0RG    | 6-Twelve Convenient-Mart Inc | Meat & Seafood
(17 rows)

-- Problem H --
SELECT cat.category_list as categories, cat.stores_count, pro.minimum, pro.maximum,
pro.average
FROM (SELECT category_list, COUNT(s.store_id) as stores_count
            FROM Stores s, UNNEST(string_to_array(s.categories, ', ')) AS
category_list
            GROUP BY category_list
            ORDER BY COUNT(s.store_id) DESC
            LIMIT 5) as cat,
      (SELECT pr.category, min(pr.list_price) as minimum, max(pr.list_price) as
maximum, avg(pr.list_price) as average
            FROM Products pr
            GROUP BY pr.category
            ) as pro
where pro.category = cat.category_list
ORDER BY cat.stores_count DESC;

       categories        | stores_count | minimum | maximum |       average
-------------------------+--------------+---------+---------+---------------------
 Canned Goods & Soups    |          254 |    0.99 |    4.29 |  2.4960606060606061
 Pet Care                |          250 |    0.99 |   28.29 | 14.8733333333333333
 Grains & Pasta & Sides  |          249 |    0.99 |   21.99 |  4.3429411764705882
 Paper & Cleaning & Home |          246 |    1.99 |   19.99 |  8.6305405405405405
 Baby Care               |          245 |    0.99 |   32.99 |  6.9014285714285714
(5 rows)

-- Problem I --
EXPLAIN ANALYZE SELECT COUNT(*) AS orders_greater_than_650
FROM Orders o
WHERE o.total_price > 650;
                                                      QUERY PLAN
-------------------------------------------------------------------------------------
-------------------------
 Aggregate  (cost=270.05..270.06 rows=1 width=8) (actual time=5.522..5.522 rows=1
loops=1)
   ->  Seq Scan on orders o  (cost=0.00..268.61 rows=576 width=0) (actual
time=1.059..5.516 rows=4 loops=1)
         Filter: (total_price > '650'::numeric)
         Rows Removed by Filter: 19996
 Planning Time: 0.045 ms
 Execution Time: 5.545 ms
(6 rows)

/*The query scanned every column in Orders entity and filtered out 19996 rows that
did not satisfy the condition, and the execution time is 5.545 ms.*/

-- Problem J --
SELECT COUNT(o.order_id)
FROM Orders o, (SELECT oi.order_id, SUM(oi.selling_price * oi.qty) AS price
                        FROM Orderitems oi
                        GROUP BY oi.order_id) AS oi1
WHERE o.order_id = oi1.order_id and o.total_price != oi1.price;
```

```
 count
-------
     5
(1 row)

UPDATE Orders o
SET total_price = offending_orders.price
FROM (SELECT o.order_id, o.total_price, oi1.price as price
           FROM Orders o, (SELECT oi.order_id, SUM(oi.selling_price * oi.qty) AS
price
                                    FROM Orderitems oi
                                    GROUP BY oi.order_id) AS oi1
           WHERE o.order_id = oi1.order_id and o.total_price != oi1.price) AS
Offending_orders
WHERE o.order_id = Offending_orders.order_id;

UPDATE 5

-- Problem K --
CREATE INDEX total_price
ON Orders(total_price);
CREATE INDEX

-- RUN QUERY I WITH EXPLAIN ANALYZE --
EXPLAIN ANALYZE SELECT COUNT(*) AS orders_greater_than_650
FROM Orders o
WHERE o.total_price > 650;

                                                                    QUERY PLAN
-------------------------------------------------------------------------------------------
-------------------------------------------
 Aggregate  (cost=478.96..478.97 rows=1 width=8) (actual time=0.054..0.055 rows=1
loops=1)
   ->  Bitmap Heap Scan on orders o  (cost=131.96..462.29 rows=6667 width=0)
(actual time=0.045..0.050 rows=4 loops=1)
         Recheck Cond: (total_price > '650'::numeric)
         Heap Blocks: exact=4
         ->  Bitmap Index Scan on total_price  (cost=0.00..130.29 rows=6667
width=0) (actual time=0.027..0.027 rows=4 loops=1)
               Index Cond: (total_price > '650'::numeric)
 Planning Time: 0.235 ms
 Execution Time: 0.092 ms
(8 rows)

/* The running time after creating the index is 0.045 ms, which is much faster than
before. I think the main reason is that the query doesn't scan every column in the
Order entity, but only the subset (total_price).*/

-- Problem L --
SELECT s.state, s.city, s.zip_code, count(s.store_id)
FROM Stores s
GROUP BY ROLLUP(s.state, s.city, s.zip_code)
ORDER BY (s.state, s.city, s.zip_code) DESC
LIMIT 20;

 state |     city     | zip_code | count
-------+--------------+----------+-------
       |              |          |   400
```

```
 WV    |              |          |     8
 WV    | Sistersville |          |     1
 WV    | Sistersville |    26175 |     1
 WV    | Lerona       |          |     1
 WV    | Lerona       |    25971 |     1
 WV    | Huntington   |          |     1
 WV    | Huntington   |    25774 |     1
 WV    | Glenwood     |          |     1
 WV    | Glenwood     |    25520 |     1
 WV    | Follansbee   |          |     1
 WV    | Follansbee   |    26037 |     1
 WV    | Elkhorn      |          |     1
 WV    | Elkhorn      |    24831 |     1
 WV    | Charleston   |          |     1
 WV    | Charleston   |    25311 |     1
 WV    | Bomont       |          |     1
 WV    | Bomont       |    25030 |     1
 WI    |              |          |     9
 WI    | Ripon        |          |     1
(20 rows)

-- Problem M (extra credit) --
SELECT pr.product_id, pr.name, pr.category, pr.list_price,
            DENSE_RANK() OVER (PARTITION BY pr.category ORDER BY pr.list_price
DESC) AS ranks
FROM Products pr
LIMIT 10;

 product_id |                                              name
| category  | list_price | ranks
------------
+-------------------------------------------------------------------------------
---+-----------+------------+-------
 Y7KB7      | Pull-Ups Training Pants Learning Designs 4T-5T Day & Night Disney
Pixar - 56 Count  | Baby Care |      32.99 |     1
 84G67      | Pampers Swaddlers Diapers Size 1 Newborn - 96 Count
| Baby Care |      26.99 |     2
 9S30I      | Pampers Baby Dry Diapers Super Pack Size 5 - 78 Count
| Baby Care |      26.99 |     2
 Y5ZMA      | Signature Care Diapers Leakage Protection Size 6 35 Lb Plus - 64
Count              | Baby Care |      19.49 |     3
 2AT8T      | Huggies Refreshing Clean Baby Wipes Refill - 352 Count
| Baby Care |      17.99 |     4
 9DA7J      | Pampers Complete Clean Baby Wipes Baby Fresh Scent 7 Pack - 504 Count
| Baby Care |      17.99 |     4
 M3QZZ      | PediaSure Grow & Gain Kids Nutritional Shake Ready To Drink Chocolate
- 6-8 Fl. Oz. | Baby Care |      12.99 |     5
 WFT9G      | Pedialyte Electrolyte Powder Strawberry Lemonade - 6-0.6 Oz
| Baby Care |      11.99 |     6
 ZKYBU      | Pedialyte Electrolyte Solution Ready To Drink Grape - 33.8 Fl. Oz.
| Baby Care |       7.99 |     7
 Y56S6      | Pedialyte Electrolyte Solution Ready To Drink Strawberry - 33.8 Fl.
Oz.            | Baby Care |       7.99 |     7
(10 rows)
```