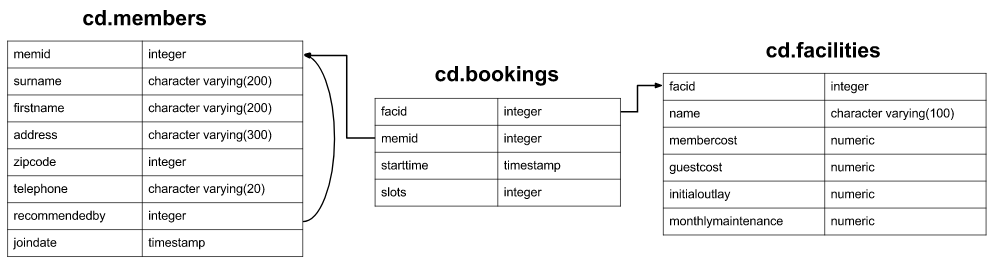
August’2022

Week 1

Problem 2:

Produce a list of the total number of slots booked per facility in the month of September 2012. Produce an output table consisting of facility id and slots, sorted by the number of slots.



**SELECT A.facid as facid, SUM(B.slots) as Total\_slots**

**FROM cd.facilities A**

**JOIN cd.bookings B**

**ON A.facid = B.facid**

**WHERE DATE(starttime) BETWEEN '2012-09-01' AND '2012-09-30'**

**GROUP BY 1**

**ORDER BY 2**

Problem 3:

How can you produce a list of all members who have used a tennis court? Include in your output the name of the court, and the name of the member formatted as a single column. Ensure no duplicate data, and order by the member name followed by the facility name.

**SELECT DISTINCT(CONCAT(A.firstname,' ',A.surname)) as member**

**,C.name as facility**

**FROM cd.members A**

**JOIN cd.bookings B**

**ON A.memid = B.memid**

**JOIN cd.facilities C**

**ON C.facid = B.facid**

**WHERE C.name LIKE 'Tennis Court%'**

**ORDER BY 1,2**

### Week 2:

### Problem 1:

### List the teachers who have NULL for their department.

### Table

### teacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **d** | **dept** | **name** | **phone** | **mobile** |
| 101 | 1 | Shrivell | 2753 | 07986 555 1234 |
| 102 | 1 | Throd | 2754 | 07122 555 1920 |
| 103 | 1 | Splint | 2293 |  |
| 104 |  | Spiregrain | 3287 |  |
| 105 | 2 | Cutflower | 3212 | 07996 555 6574 |
| 106 |  | Deadyawn | 3345 |  |
| ... | | | | |

|  |  |
| --- | --- |
| dept | |
| **id** | **name** |
| 1 | Computing |
| 2 | Design |
| 3 | Engineering |
| ... | |

### SELECT A.name as teacher\_name

### FROM teacher A

### LEFT JOIN dept B

### ON A.dept = B.id

### WHERE A.dept is NULL

### -------------------------------------------------------------------------------------------------

### Problem 2:

Find the result of subtracting the timestamp '2012-07-30 01:00:00' from the timestamp '2012-08-31 01:00:00'

**SELECT timestamp '2012-08-31 01:00:00' - timestamp '2012-07-30 01:00:00' as interval**

### Problem 3:

Return a count of bookings for each month, sorted by month

**SELECT date\_trunc('month' ,starttime),count(\*)**

**FROM cd.bookings**

**GROUP BY 1**

**ORDER BY 1**

**October-2022**

**Week 1**

1. Produce a list of facilities with more than 1000 slots booked. Produce an output table consisting of facility id and slots, sorted by facility id.

Table Names: cd. members, cd. Bookings, cd.facilities

SELECT

facid,

SUM(slots) total\_slots

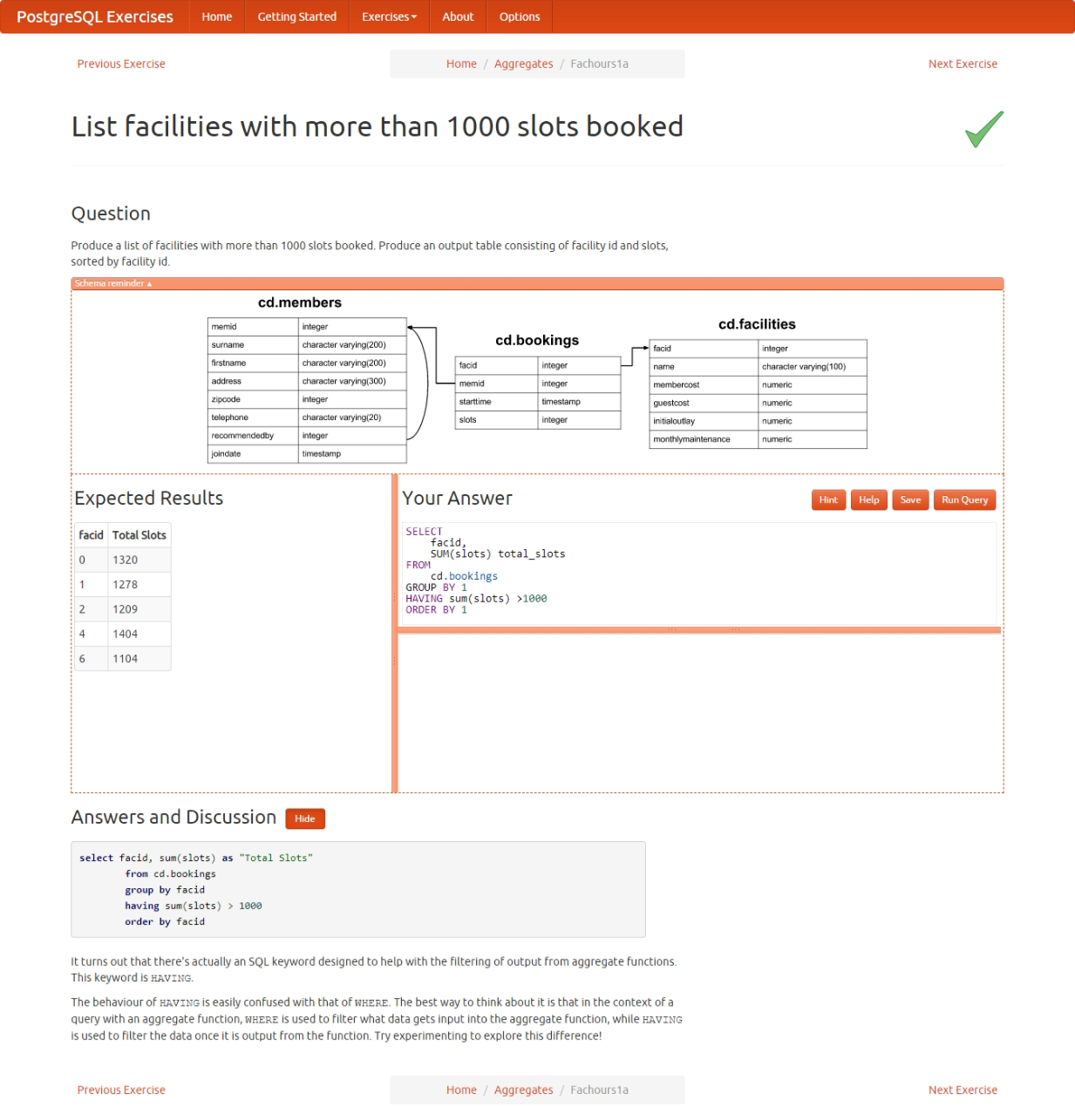
FROM

cd.bookings

GROUP BY 1

HAVING sum(slots) >1000

ORDER BY 1



1. Produce a list of facilities along with their total revenue. The output table should consist of facility name and revenue, sorted by revenue. Remember that there's a different cost for guests and members!

SELECT

B.name,

SUM(A.slots \* (

CASE

WHEN A.memid = 0 THEN B.guestcost

ELSE B.membercost

END)) as revenue

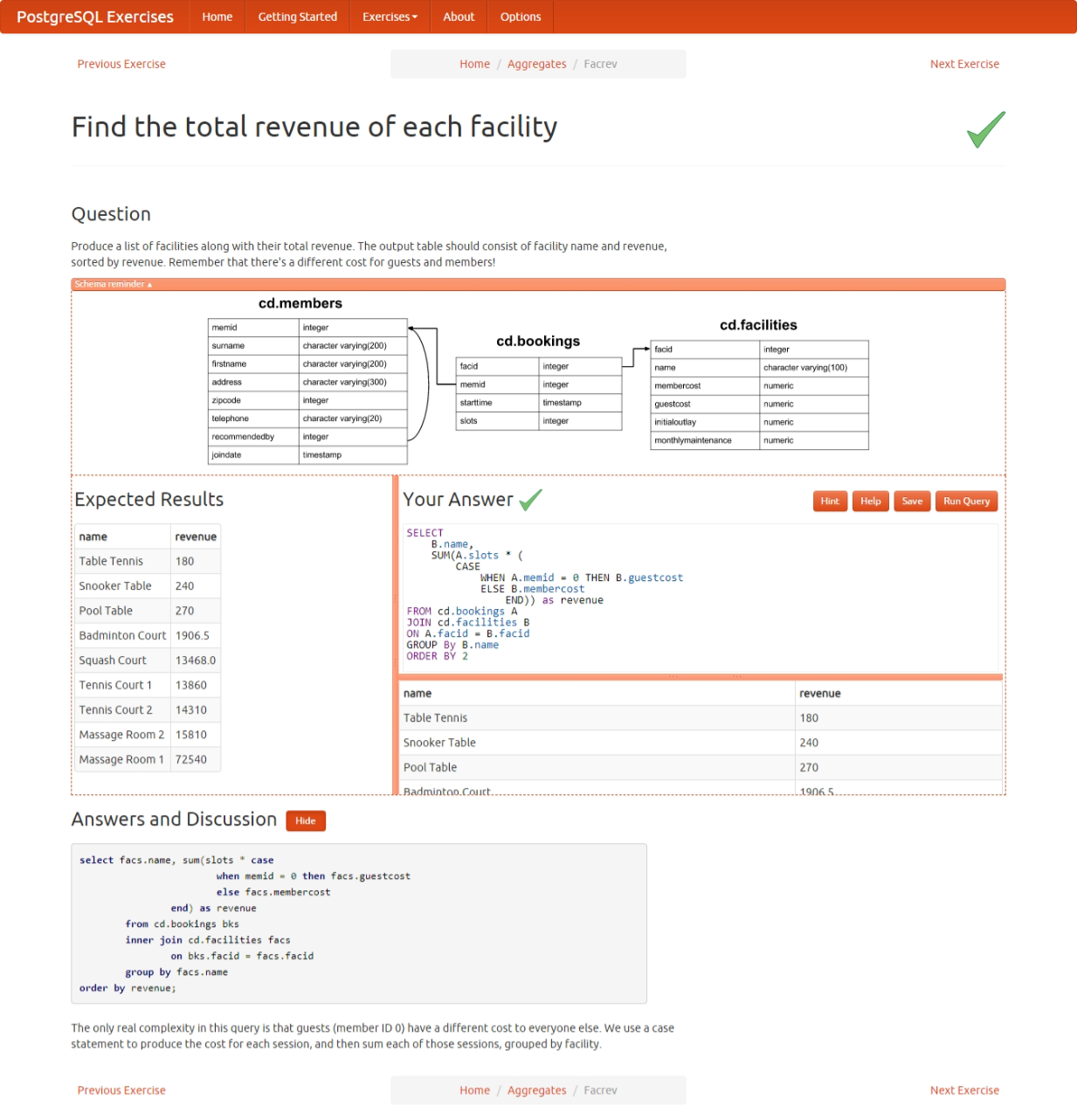
FROM cd.bookings A

JOIN cd.facilities B

ON A.facid = B.facid

GROUP By B.name

ORDER BY 2



1. The telephone numbers in the database are very inconsistently formatted. You'd like to print a list of member ids and numbers that have had '-','(',')', and ' ' characters removed. Order by member id.

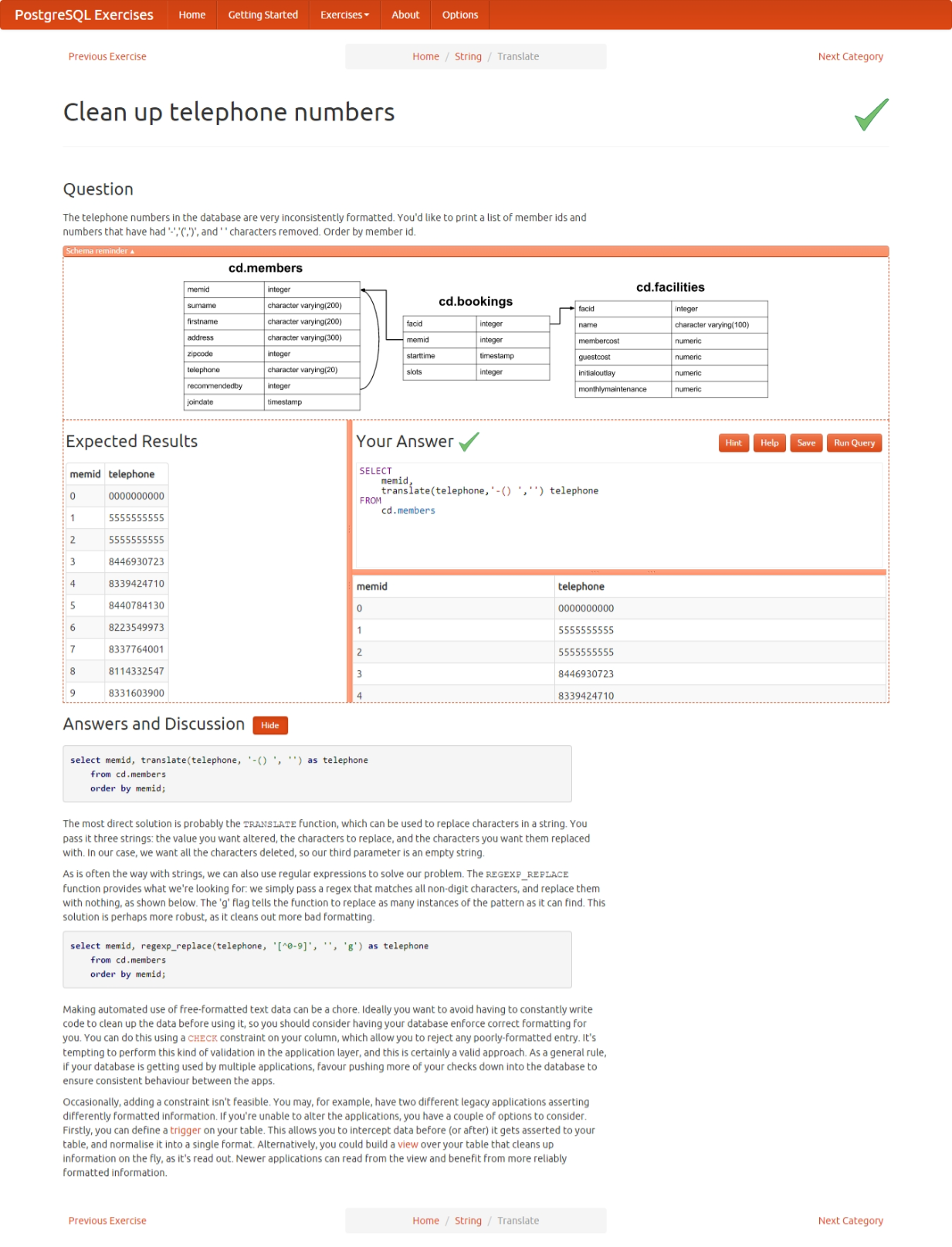
SELECT

memid,

translate(telephone,'-() ','') as telephone

FROM

cd.members

****

Week 27 SQL Challenge

Table: Stocks

+---------------+---------+

| Column Name | Type |

+---------------+---------+

| stock\_name | varchar |

| operation | enum |

| operation\_day | int |

| price | int |

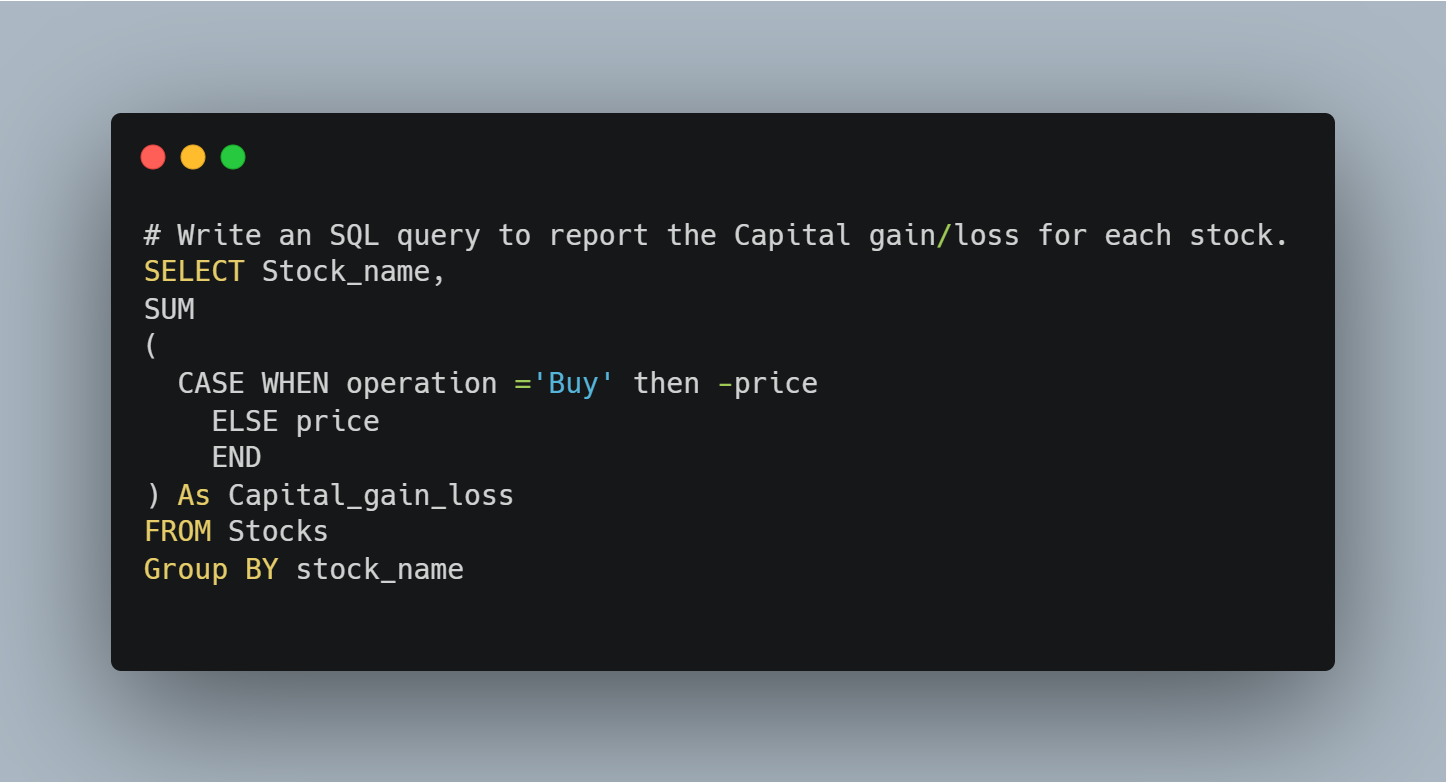
+---------------+---------+

(stock\_name, operation\_day) is the primary key for this table.

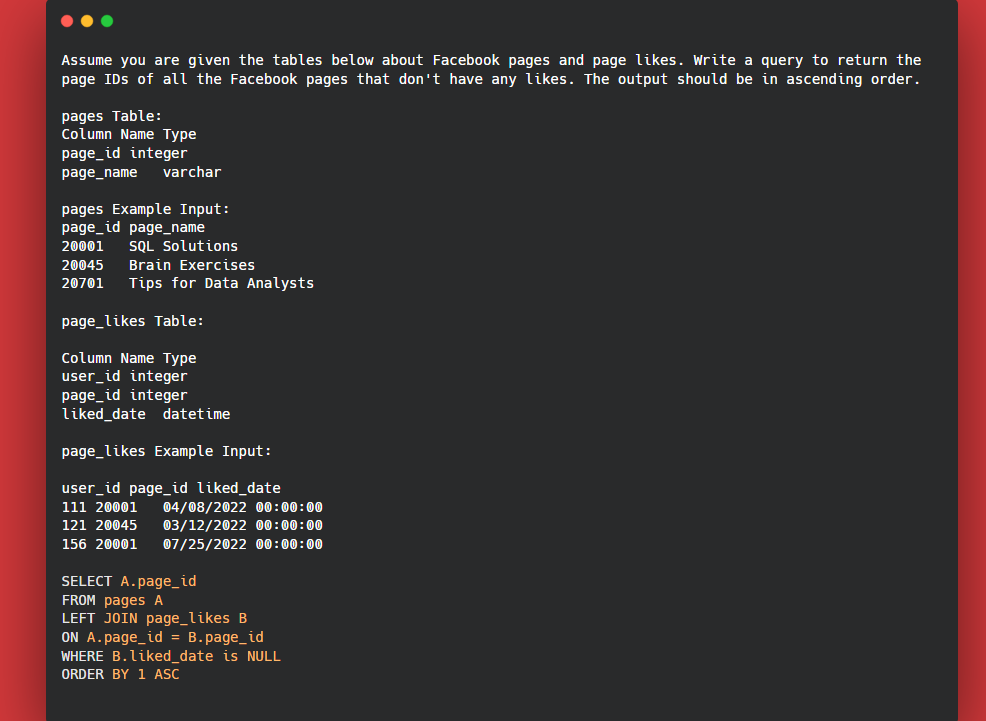
The operation column is an ENUM of type ('Sell', 'Buy')

Each row of this table indicates that the stock which has stock\_name had an operation on the day operation\_day with the price.

It is guaranteed that each 'Sell' operation for a stock has a corresponding 'Buy' operation in a previous day. It is also guaranteed that each 'Buy' operation for a stock has a corresponding 'Sell' operation in an upcoming day.



week 28 of our weekly SQL challenge



week 29 of our weekly SQL challenge

Robinhood trades

You are given the tables below containing information on Robinhood trades and users. Write a query to list the top three cities that have the most completed trade orders in descending order.

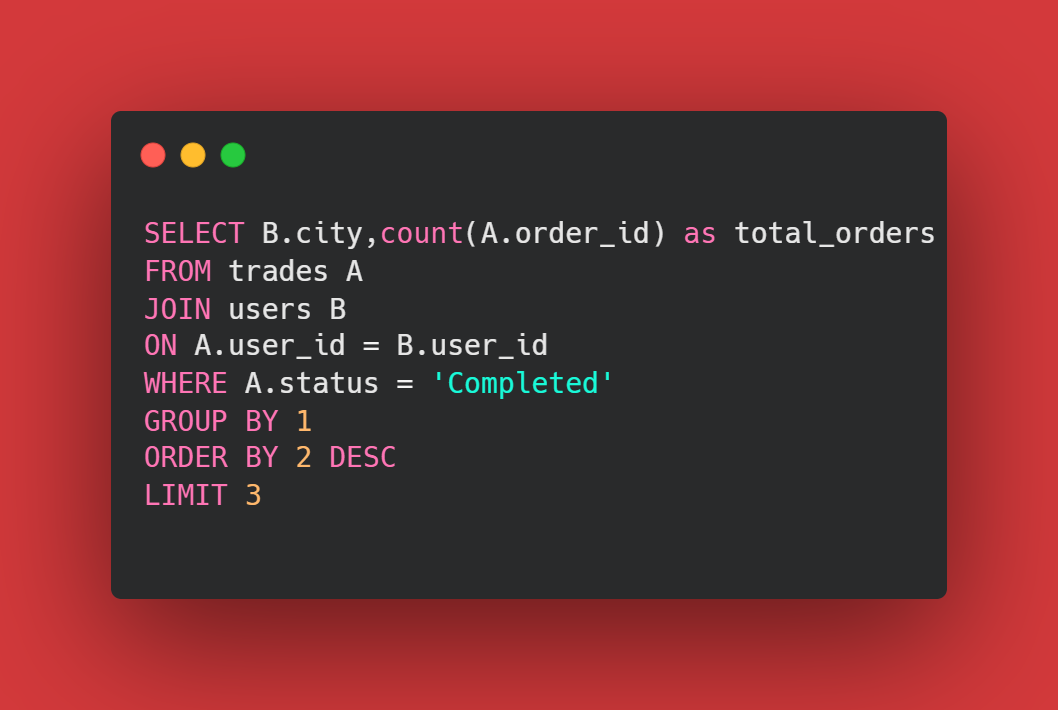
Output the city and number of orders.

**trades Example Input:**

| **order\_id** | **user\_id** | **price** | **quantity** | **status** | **timestamp** |
| --- | --- | --- | --- | --- | --- |
| 100101 | 111 | 9.80 | 10 | Cancelled | 08/17/2022 12:00:00 |
| 100102 | 111 | 10.00 | 10 | Completed | 08/17/2022 12:00:00 |
| 100259 | 148 | 5.10 | 35 | Completed | 08/25/2022 12:00:00 |
| 100264 | 148 | 4.80 | 40 | Completed | 08/26/2022 12:00:00 |
| 100305 | 300 | 10.00 | 15 | Completed | 09/05/2022 12:00:00 |
| 100400 | 178 | 9.90 | 15 | Completed | 09/09/2022 12:00:00 |
| 100565 | 265 | 25.60 | 5 | Completed | 12/19/2022 12:00:00 |

**users Example Input:**

| **user\_id** | **city** | **email** | **signup\_date** |
| --- | --- | --- | --- |
| 111 | San Francisco | [rrok10@gmail.com](mailto:rrok10@gmail.com) | 08/03/2021 12:00:00 |
| 148 | Boston | [sailor9820@gmail.com](mailto:sailor9820@gmail.com) | 08/20/2021 12:00:00 |
| 178 | San Francisco | [harrypotterfan182@gmail.com](mailto:harrypotterfan182@gmail.com) | 01/05/2022 12:00:00 |
| 265 | Denver | [shadower\_@hotmail.com](mailto:shadower_@hotmail.com) | 02/26/2022 12:00:00 |
| 300 | San Francisco | [houstoncowboy1122@hotmail.com](mailto:houstoncowboy1122@hotmail.com) | 06/30/2022 12:00:00 |



Twitter:

The table below contains information about tweets over a given period of time. Calculate the 3-day rolling average of tweets published by each user for each date that a tweet was posted. Output the user id, tweet date, and rolling averages rounded to 2 decimal places.

**Important Assumptions**:

Rows in this table are consecutive and ordered by date.

Each row represents a different day

A day that does not correspond to a row in this table is not counted. The most recent day is the next row above the current row.

**Note: Rolling average is a metric that helps us analyze data points by creating a series of averages based on different subsets of a dataset. It is also known as a moving average, running average, moving mean, or rolling mean.**

tweets Example Input:

| **tweet\_id** | **user\_id** | **tweet\_date** |
| --- | --- | --- |
| 214252 | 111 | 06/01/2022 12:00:00 |
| 739252 | 111 | 06/01/2022 12:00:00 |
| 846402 | 111 | 06/02/2022 12:00:00 |
| 241425 | 254 | 06/02/2022 12:00:00 |
| 137374 | 111 | 06/04/2022 12:00:00 |
|  |  |  |

1. Let’s find the number of tweets on each day by each user.
2. To find rolling average for 3 days, we have to use AVG() windows function for each user subset with dates in order long with **ROWS BETWEEN 2 PRECEDING AND CURRENT ROW** to calculate average for 3days.



Week 38 :

# International Call Percentage [Verizon SQL Interview Question]

A phone call is considered an international call when the person calling is in a different country than the person receiving the call.

What percentage of phone calls are international? Round the result to 1 decimal.

Assumption:  caller\_id in phone\_info table refers to both the caller and receiver.

Tables :

phone\_calls

| **Column Name** | **Type** |
| --- | --- |
| caller\_id | integer |
| receiver\_id | integer |
| call\_time | timestamp |

phone\_info

| **Column Name** | **Type** |
| --- | --- |
| caller\_id | integer |
| country\_id | integer |
| network | integer |
| phone\_number | string |

1. Identify caller country and receiver country
2. Count international calls (**caller country! = receiver country**) and total calls
3. Find the % of international calls (**round (100.0\*(total calls/international calls**))

Step 1 :

1. Identify caller country and receiver country

SELECT caller.country\_id as caller\_country,

receiver.country\_id as receiver\_country

FROM phone\_calls calls

LEFT JOIN phone\_info caller

ON calls.caller\_id = caller.caller\_id

LEFt JOIN phone\_info receiver

ON calls.receiver\_id = receiver.caller\_id

Step 2:

1. Identifying international calls count and total count

SELECT

SUM (CASE WHEN caller.country\_id <> receiver.country\_id THEN 1 END) as international\_calls

,count(caller.caller\_id)

FROM phone\_calls calls

LEFT JOIN phone\_info caller

ON calls.caller\_id = caller.caller\_id

LEFt JOIN phone\_info receiver

ON calls.receiver\_id = receiver.caller\_id

Step 3: Calculating international calls percentage

SELECT

ROUND(100.0\*SUM(CASE

WHEN caller.country\_id <> receiver.country\_id THEN 1

ELSE NULL

END) /count(caller.caller\_id),1) as Internationa\_calls\_pct

FROM phone\_calls calls

LEFT JOIN phone\_info caller

ON calls.caller\_id = caller.caller\_id

LEFt JOIN phone\_info receiver

ON calls.receiver\_id = receiver.caller\_id

# Supercloud Customer [Microsoft SQL Interview Question]

A Microsoft Azure Supercloud customer is a company which buys at least 1 product from each product category.

Write a query to report the company ID which is a Supercloud customer.

As of 5 Dec 2022, data in the *customer\_contracts* and *products* tables were updated.

**customer\_contracts Table:**

| **Column Name** | **Type** |
| --- | --- |
| customer\_id | integer |
| product\_id | integer |
| amount | integer |

**customer\_contracts Example Input:**

| **customer\_id** | **product\_id** | **amount** |
| --- | --- | --- |
| 1 | 1 | 1000 |
| 1 | 3 | 2000 |
| 1 | 5 | 1500 |
| 2 | 2 | 3000 |
| 2 | 6 | 2000 |

1. Combine both the tables to pull customer id and prod. Category into single table
2. Count distinct of prod.category ,group by cust.id
3. Condition having Count distinct of prod.category(combined table) = count distinct of product category

SELECT cust.customer\_id supercloud\_customer

FROM customer\_contracts cust

LEFT join products prod

ON cust.product\_id = prod.product\_id

GROUP BY 1

HAVING count(distinct(prod.product\_category)) =

(SELECT count(distinct(product\_category)) FROM products)