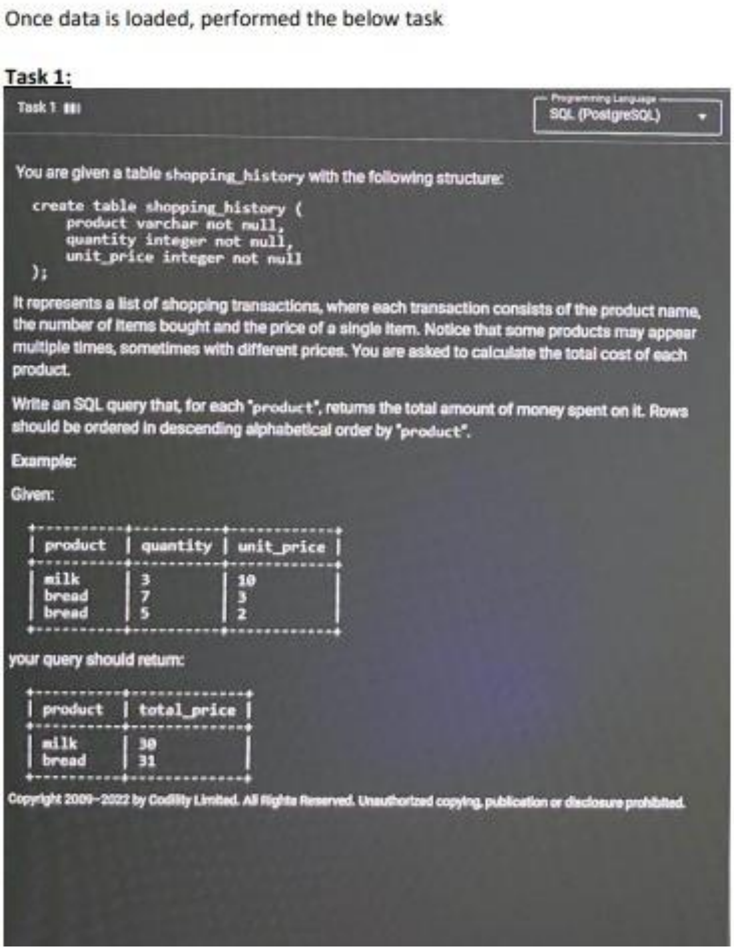
FSDA ASSIGNMENT 1:



Once data is loaded, performed the below task

Task 1:

Task 1

Programming Language: SQL (PostgreSQL)

You are given a table shopping_history with the following structure:

```
create table shopping_history (
    product varchar not null,
    quantity integer not null,
    unit_price integer not null
);
```

It represents a list of shopping transactions, where each transaction consists of the product name, the number of items bought and the price of a single item. Notice that some products may appear multiple times, sometimes with different prices. You are asked to calculate the total cost of each product.

Write an SQL query that, for each "product", returns the total amount of money spent on it. Rows should be ordered in descending alphabetical order by "product".

Example:

Given:

```
+---------+----------+------------+
| product | quantity | unit_price |
+---------+----------+------------+
| milk    | 3        | 10         |
| bread   | 7        | 3          |
| bread   | 5        | 2          |
+---------+----------+------------+
```

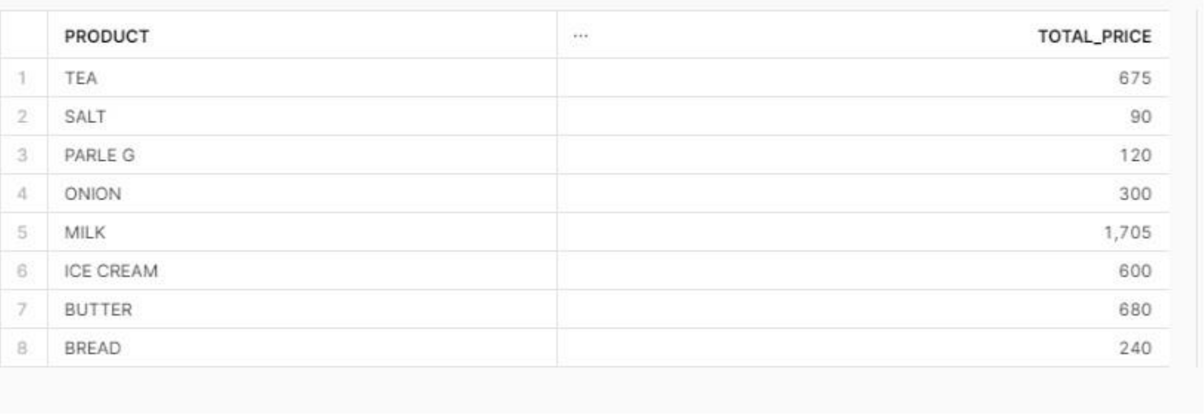
your query should return:

```
+---------+-------------+
| product | total_price |
+---------+-------------+
| milk    | 30          |
| bread   | 31          |
+---------+-------------+
```

Copyright 2009–2022 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

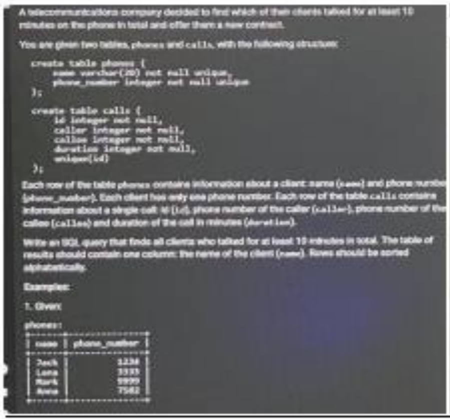
--CALCULATE TOTAL PRICE

```
SELECT PRODUCT,SUM(quantity*unit_price) AS TOTAL_PRICE from
RS_SHOPPING_HISTORY group by PRODUCT ORDER BY PRODUCT DESC;
```



	PRODUCT	...	TOTAL_PRICE
1	TEA		675
2	SALT		90
3	PARLE G		120
4	ONION		300
5	MILK		1,705
6	ICE CREAM		600
7	BUTTER		680
8	BREAD		240

Task 2:



A telecommunications company decided to find which of their clients talked for at least 10 minutes on the phone in total and offer them a new contract.

You are given two tables, phones and calls, with the following structure:

```
create table phones (
    name varchar(20) not null unique,
    phone_number integer not null unique
);

create table calls (
    id integer not null,
    caller integer not null,
    callee integer not null,
    duration integer not null,
    unique(id)
);
```

Each row of the table phones contains information about a client: name (name) and phone number (phone_number). Each client has only one phone number. Each row of the table calls contains information about a single call: id (id), phone number of the caller (caller), phone number of the callee (callee) and duration of the call in minutes (duration).

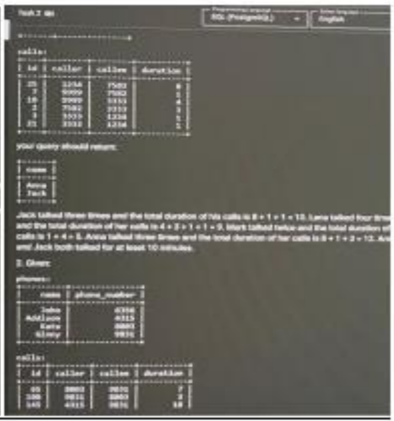
Write an SQL query that finds all clients who talked for at least 10 minutes in total. The table of results should contain one column: the name of the client (name). Rows should be sorted alphabetically.

Examples:

1. Given:

phones:

name	phone_number
Jack	1234
Lena	3333
Mark	9999
Anna	7582



calls:

id	caller	callee	duration
25	1234	7582	8
7	9999	7582	1
18	9999	3333	4
2	7582	3333	3
3	3333	1234	1
21	3333	1234	1

your query should return:

name
Anna
Jack

Jack talked three times and the total duration of his calls is 8 + 1 + 1 = 10. Lena talked four times and the total duration of her calls is 4 + 3 + 1 + 1 = 9. Mark talked twice and the total duration of his calls is 1 + 4 = 5. Anna talked three times and the total duration of her calls is 8 + 1 + 3 = 12. Anna and Jack both talked for at least 10 minutes.

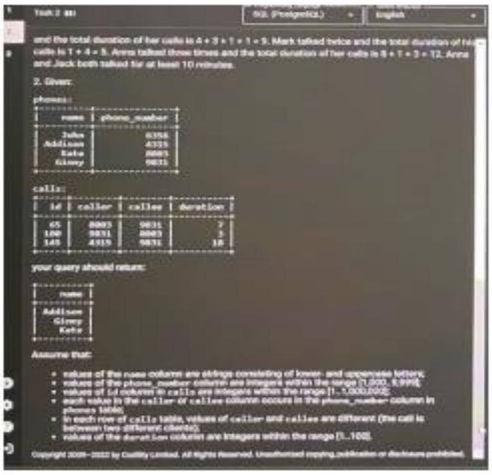
2. Given:

phones:

name	phone_number
John	6356
Addison	4315
Kate	8003
Ginny	9831

calls:

id	caller	callee	duration
65	8003	9831	7
100	9831	8003	3
145	4315	9831	18



and the total duration of her calls is 4 + 3 + 1 + 1 = 9. Mark talked twice and the total duration of his calls is 1 + 4 = 5. Anna talked three times and the total duration of her calls is 8 + 1 + 3 = 12. Anna and Jack both talked for at least 10 minutes.

2. Given:

phones:

name	phone_number
John	6356
Addison	4315
Kate	8003
Ginny	9831

calls:

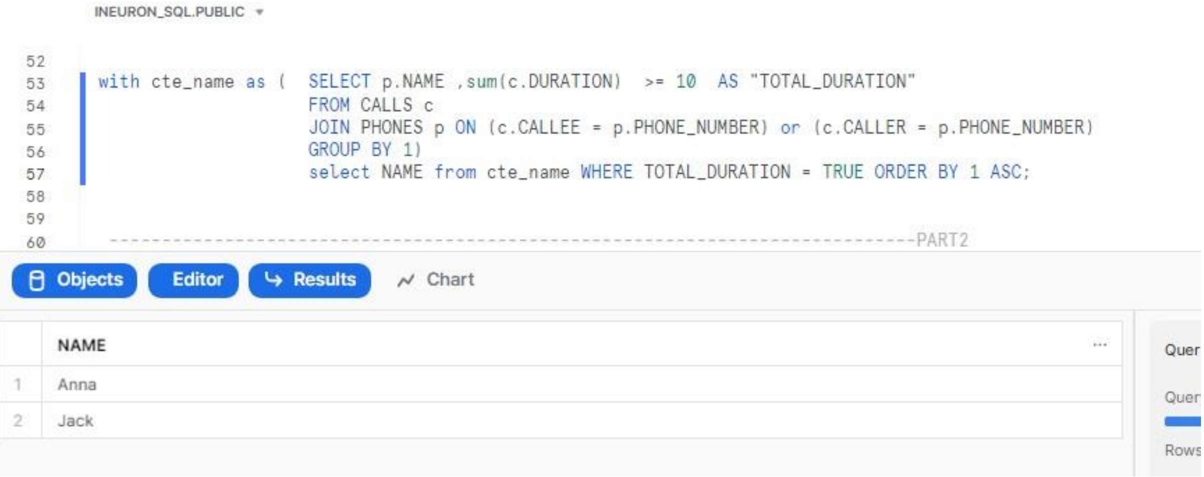
id	caller	callee	duration
65	8003	9831	7
100	9831	8003	3
145	4315	9831	18

your query should return:

name
Addison
Ginny
Kate

Assume that:

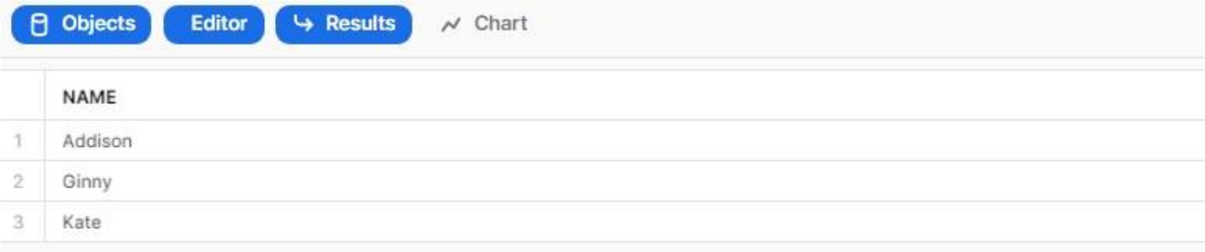
- values of the name column are strings consisting of lower- and uppercase letters;
- values of the phone_number column are integers within the range [1,000..9,999];
- values of id column in calls are integers within the range [1..1,000,000];
- each value in the caller or callee column occurs in the phone_number column in phones table;
- in each row of calls table, values of caller and callee are different (the call is between two different clients);
- values of the duration column are integers within the range [1..100].



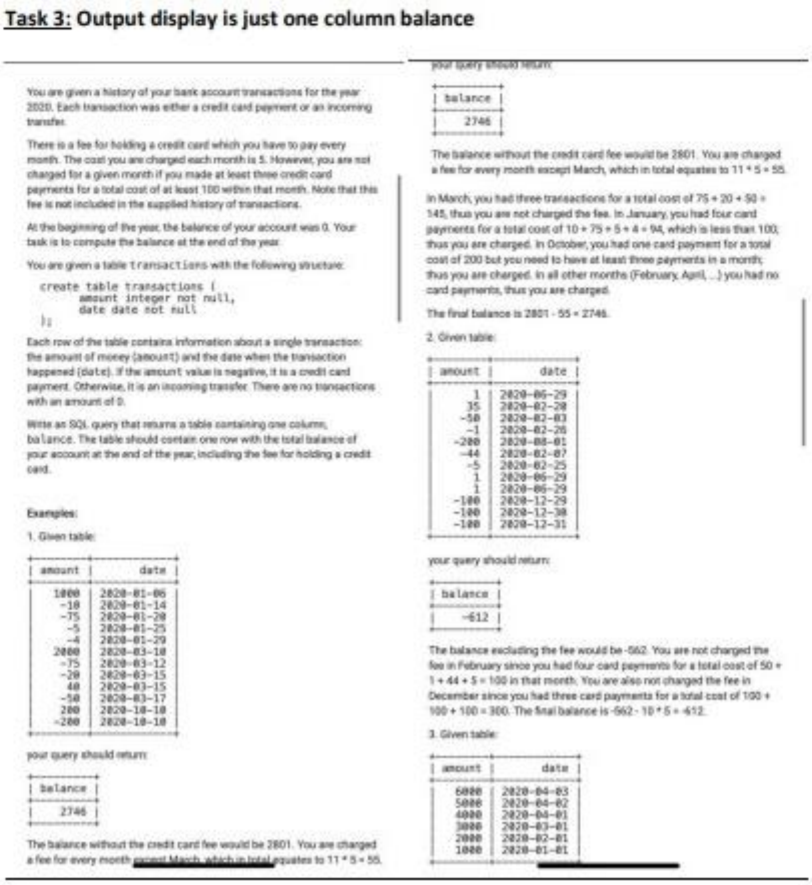
```
with cte_name as (  SELECT p.NAME ,sum(c.DURATION)  >= 10  AS "TOTAL_DURATION"
                    FROM CALLS c
                    JOIN PHONES p ON (c.CALLEE = p.PHONE_NUMBER) or (c.CALLER = p.PHONE_NUMBER)
                    GROUP BY 1)
                    select NAME from cte_name WHERE TOTAL_DURATION = TRUE ORDER BY 1 ASC;


  ------------------------------------------------------------------------------PART2
```

	NAME
1	Anna
2	Jack



	NAME
1	Addison
2	Ginny
3	Kate



Task 3: Output display is just one column balance

You are given a history of your bank account transactions for the year 2020. Each transaction was either a credit card payment or an incoming transfer.

There is a fee for holding a credit card which you have to pay every month. The cost you are charged each month is 5. However, you are not charged for a given month if you made at least three credit card payments for a total cost of at least 100 within that month. Note that this fee is not included in the supplied history of transactions.

At the beginning of the year, the balance of your account was 0. Your task is to compute the balance at the end of the year.

You are given a table `transactions` with the following structure:

```
create table transactions (
      amount integer not null,
      date date not null
);
```

Each row of the table contains information about a single transaction: the amount of money (`amount`) and the date when the transaction happened (`date`). If the `amount` value is negative, it is a credit card payment. Otherwise, it is an incoming transfer. There are no transactions with an amount of 0.

Write an SQL query that returns a table containing one column, `balance`. The table should contain one row with the total balance of your account at the end of the year, including the fee for holding a credit card.

Examples:

1. Given table:

amount	date
1000	2020-01-06
-10	2020-01-14
-75	2020-01-20
-5	2020-01-25
-4	2020-01-29
2000	2020-03-10
-75	2020-03-12
-20	2020-03-15
40	2020-03-15
-50	2020-03-17
200	2020-10-10
-200	2020-10-10

your query should return:

balance
2746

The balance without the credit card fee would be 2801. You are charged a fee for every month except March, which in total equates to 11 * 5 = 55.

your query should return:

balance
2746

The balance without the credit card fee would be 2801. You are charged a fee for every month except March, which in total equates to 11 * 5 = 55.

In March, you had three transactions for a total cost of 75 + 20 + 50 = 145, thus you are not charged the fee. In January, you had four card payments for a total cost of 10 + 75 + 5 + 4 = 94, which is less than 100, thus you are charged. In October, you had one card payment for a total cost of 200 but you need to have at least three payments in a month; thus you are charged. In all other months (February, April, ...) you had no card payments, thus you are charged.

The final balance is 2801 - 55 = 2746.

2. Given table:

amount	date
1	2020-06-29
35	2020-02-20
-50	2020-02-03
-1	2020-02-26
-200	2020-08-01
-44	2020-02-07
-5	2020-02-25
1	2020-06-29
1	2020-06-29
-100	2020-12-29
-100	2020-12-30
-100	2020-12-31

your query should return:

balance
-612

The balance excluding the fee would be -562. You are not charged the fee in February since you had four card payments for a total cost of 50 + 1 + 44 + 5 = 100 in that month. You are also not charged the fee in December since you had three card payments for a total cost of 100 + 100 + 100 = 300. The final balance is -562 - 10 * 5 = -612.

3. Given table:

amount	date
6000	2020-04-03
5000	2020-04-02
4000	2020-04-01
3000	2020-03-01
2000	2020-02-01
1000	2020-01-01

This one query work in all part of task3

```
With D as(
select Count(distinct(to_char('MMMM'))) as Mon_Count from transactions where Amount < 0
group by to_char('MMMM')
Having Sum(Amount) <= -100 and Count(Amount) >=3),
tm as(
select (12 - zeroifnull(sum(D.Mon_Count)))*5 as ded from  D)
,tr as(
Select sum(amount) as ff
from transactions)
Select tr.ff - tm.ded from tr,tm;
```