Combining Multiple Tables Using UNION

Business problem: Combining 2 Product Tables into 1

The chief growth officer is focused on reducing churn, or the number of customers that don't renew their annual subscription. She is planning to launch multiple product experiments and marketing campaigns throughout 2023 to drive users to renew their subscriptions. She's first conducting research to understand the potential impact that the churn initiative can have on the business. She reaches out to your team and asks when all active subscriptions are going to expire.

Because of data modeling limitations, your company was prevented from putting both products in the same table, so there are currently separate tables for each product, `subscriptionsproduct1` and `subscriptionsproduct2`.

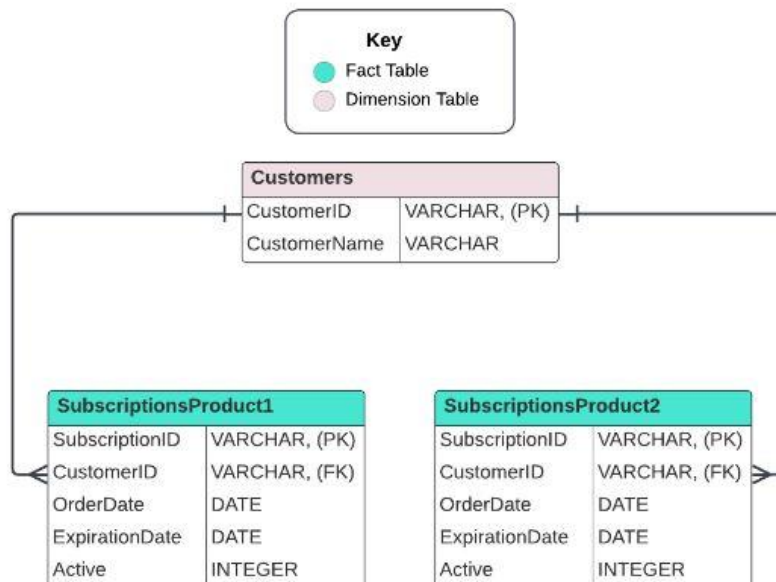Sometimes, even if it's not the most optimal solution, data can be stored in multiple places due to:

- Data coming from different data sources
- Poor ETL and data modeling
- Technical migrations that don't align with legacy systems
- Company or product acquisitions or mergers

Even though you don't need to segment by product, you do need to include BOTH products in your yearly counts. UNION will be the best tool to use to combine both products into one table to fulfill the chief growth officer's request.

Task:

Count the number of active subscriptions, `active = 1`, that will expire in each year.

Subscriptions Data Model

Key
- Fact Table
- Dimension Table

Customers

CustomerID	VARCHAR, (PK)
CustomerName	VARCHAR

SubscriptionsProduct1

SubscriptionID	VARCHAR, (PK)
CustomerID	VARCHAR, (FK)
OrderDate	DATE
ExpirationDate	DATE
Active	INTEGER

SubscriptionsProduct2

SubscriptionID	VARCHAR, (PK)
CustomerID	VARCHAR, (FK)
OrderDate	DATE
ExpirationDate	DATE
Active	INTEGER

```
1    -- SQL code by John Uzoma
2    With all_subscriptions as(
3
4    --UNION subscriptions tables here
5        select subscriptionid, expirationdate from subscriptionsproduct1
6        where active = 1
7        union all
8        select subscriptionid, expirationdate from subscriptionsproduct2
9        where active = 1
10   )
11   select
12       date_trunc('year', expirationdate) as exp_year,
13       count(*) as subscriptions
14   from
15       all_subscriptions
16   group by
17       date_trunc('year', expirationdate)
18
```

Result


```
------------------------------------------
| EXP_YEAR                 | SUBSCRIPTIONS |
------------------------------------------
| 2023-01-01 00:00:00.0 | 5             |
| 2024-01-01 00:00:00.0 | 2             |
------------------------------------------
```