Question:

Our Data Analyst needs to compute a `fulfillment promised date` for each of the orderline deliveries based on the Service Level Agreements (SLA) of the logistics provider selected for the delivery.

The fulfillment promised date is computed based on the fulfillment creation date plus the number of promised working days for delivery provided by the SLA of the logistics provider.

The `working_days` table contains the working days of the world up to 2030.

Part of the query he is using is as follows:

```
select
...
sum(is_working_day::integer order by working_days.date) as
number_of_work_days

...
from order_line
left join working_days
on order_line.fulfillment_creation_date < working_days.date
```

He is facing query timeout issues consistently when computing the provider KPIs. What are the steps you would take to help him with this problem?

<u>Answer:</u>

Based on his query, it is likely his intention is to find the cumulative sum for each of the order_line fulfilment promised date for each of the orderline_devlieries, i.e order_id.

However, he applied a window function of sum order by the working_day_date to compute a cumulative sum for each of the working_day_date. It is likely he will be taking this cumulative sum and adding it to the fulfil_promised_date in other part of the projection (not shown in the query).

If that is the case, i will advise him to do the following:

First step. I will help to build a working day table:

```
Create table working_days_table (
Date date,
working_day int
)
```

I will materialize this table as a physical table as such tables will be repeatedly being used by other end users in the company.

Secondly, i will make the following changes to the query:

```
select  order_line.fulfillment_creation_date,
future_working_days.number_of_work_days -
working_days.number_of_work_days as fulfilment_days


from (select distinct fulfillment_creation_date from order_line) order_line
Inner join working_days_table working_days
on order_line.fulfillment_creation_date = working_days.date

Inner join working_days_table future_working_days
on order_line.fulfillment_creation_date < working_days.date
```

By removing useless left joins (since every date will have a number_of_working days) and considering that in a order_line there will be many same fulfilment creation date for different order_id, there is no point trying to replicate the cumulative sum for all the different order_id.

Thirdly, I will use common table expression and wrap the query in the 2nd stage

```
WITH fulfilment_promised_date AS (
   SELECT order_line.fulfillment_creation_date
         ,future_working_days.number_of_work_days -
working_days.number_of_work_days AS fulfilment_days
FROM (
         SELECT DISTINCT fulfillment_creation_date
         FROM order_line
         ) order_line
INNER JOIN working_days_table working_days ON
order_line.fulfillment_creation_date = working_days.DATE
INNER JOIN working_days_table future_working_days ON
order_line.fulfillment_creation_date < working_days.DATE
)

select … from order_line
Inner join fulfilment_promised_date prom_date
on order_line.fulfillment_creation_date =
prom_date.fulfillment_creation_date
```

If performance of the fulfilment_create_date is bad, I may consider creating a partition on fulfillment_creation_date table.

This will allow each orderline record to have a fulfilment_days computed from prom_Date.

However, the end result of what the DA wants to do is unlikely to be just this. They will want to find the distribution of the possible promised_fulfilment days and etc as there is a logistics limitation to how many orders the company can try to deliver in a day due to manpower crunch. Hence, I would continue to understand the exact problem statement that they are trying to build and suggest a more complete optimization solution. (As it is common for DA/ BI to be trapped in false dichotomy in their design of their pipelines)