SumUp Case Study Questions

```sql
--Q1: Top 10 stores per transacted amount
-- I understood it as the top 10 stores with the highest transaction amount in the single transaction.
SELECT
        store_name,
        amount as amount
FROM
       `inspiring-ring-382618.sumup.dim_transactions`
WHERE
-- only consider accepted transactions, i.e. successful transactions only
 status = 'accepted'
order by
       2 desc
limit
       10

--Q2: Top 10 products sold
SELECT
        product_name,
        COUNT(transaction_id) AS total_sales
FROM
       `inspiring-ring-382618.sumup.dim_transactions`
WHERE status = 'accepted' -- only consider accepted transactions
GROUP BY
   product_name
ORDER BY
   total_sales DESC
LIMIT 10;

--Q3: Average transacted amount per store typology and country
SELECT
        store_typology,
        store_country,
        ROUND(AVG(amount),2) AS avg_transacted_amount
```

```
FROM
       `inspiring-ring-382618.sumup.dim_transactions`
WHERE
       status = 'accepted'
GROUP BY
       store_country, store_typology
ORDER BY
       1,2

--Q4: Percentage of transactions per device type
--Percentage of transactions per device type = (Number of transactions with a specific device type / Total number of transactions) * 100

 Select
        device_type,
        ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (), 2) AS percentage
FROM
 `inspiring-ring-382618.sumup.dim_transactions`
GROUP BY
       device_type
ORDER BY
       device_type

--Q5: Average time for a store to perform its 5 first transactions
-- First I have numbered the transactions using row_number for every store in the
--ascending order. Then I will take the time difference in hours and days between the
--first and the 5th transaction datetime. For the store id's if the difference is null
--then it means the store has not yet encountered the 5th transaction.
WITH dim_transaction AS(
SELECT
        store_id,
        transaction_happend_at,
        ROW_NUMBER() OVER(PARTITION BY store_id ORDER BY transaction_happend_at) AS
       store_transaction_number
FROM
       `inspiring-ring-382618.sumup.dim_transactions`
)
```

```sql
SELECT
 store_id,
 TIMESTAMP_DIFF(MAX(CASE WHEN store_transaction_number = 5 THEN transaction_happend_at
END),
                MIN(CASE WHEN store_transaction_number = 1 THEN
transaction_happend_at END),
                HOUR) AS total_hours_for_first_five_transactions,
   TIMESTAMP_DIFF(MAX(CASE WHEN store_transaction_number = 5 THEN
transaction_happend_at end),
                MIN(CASE WHEN store_transaction_number = 1 THEN
transaction_happend_at END),
                DAY) AS number_of_days_for_first_five_transactions

FROM
       dim_transaction
WHERE
       store_transaction_number in (1, 5)
GROUP BY
       store_id
ORDER BY
       1
```

**** The dataset supplied with this test contains only sample data. Your design and implementation should scale for larger volumes of data (millions to billions records)***

The volume of data in the dim_transaction table increases rapidly every business hour as we operate in 35 markets worldwide. I would like to materialize it as an incremental table.

Even though we use the best strategy in storing(Partitioning, clustering etc.) that data in the data warehouse, The performance will be significantly affected as we are joining it with two other tables in my solution i.e. dim_stores and raw_devices(no.of. Tables to be joined may increase as well depending on the requirements). I have designed the pipeline in such a way that raw data is extracted every 2 hours and dumped into the data lake and my source models will be scheduled to pull the new records and the same will be appended in the dim_transactions every 6 or 12 hours. Since the transaction data is very critical/important data, to ensure no data is lost while appending this table I have used *pre_hook* which drops the previous 6 hours data and reinserts it in the current run. If business aims to visualize these numbers in the BI tool in the near real-time then storing, query optimization is a must thing to be taken care about.