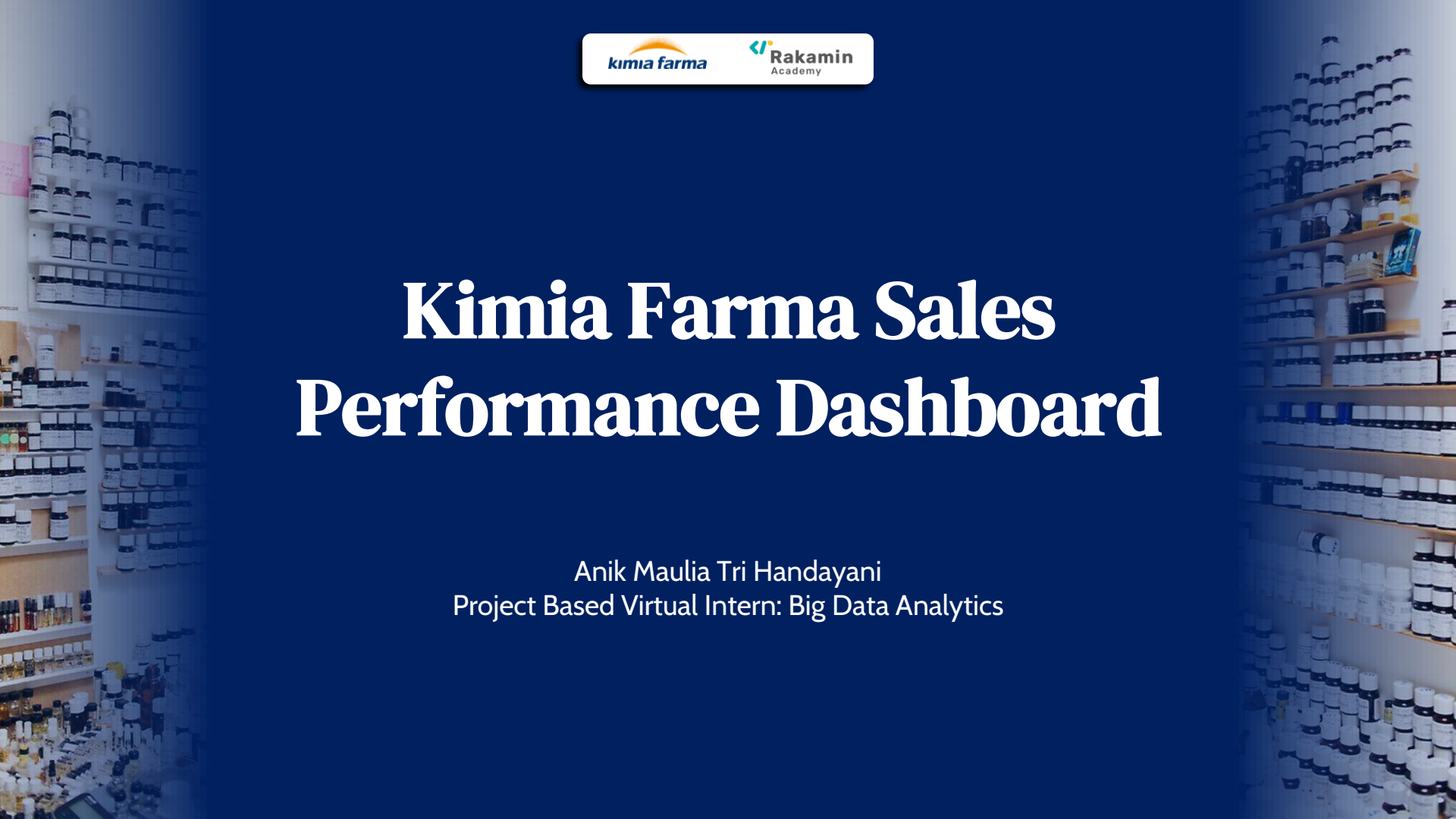


Kimia Farma Sales Performance Dashboard

Anik Maulia Tri Handayani
Project Based Virtual Intern: Big Data Analytics

Our Company

Kimia Farma is the first pharmaceutical company in Indonesia, founded by the Dutch East Indies government in 1817. Over the years, the company has transformed into an integrated healthcare services provider in Indonesia. Presently, Kimia Farma is focusing on visualizing its product sales performance through a dashboard for a thorough assessment and identification of areas for improvement or optimization. This initiative is undertaken to support ongoing efforts in enhancing and developing their pharmaceutical business.

Table of contents

01 Query Analysis

02 Define Primary Key

03 Design Datamart

04 Data Visualization

01

Query Analysis

Question 1

Which of the two queries works better? Explain why.

(a) SELECT * FROM pelanggan WHERE SUBSTR(alamat, 1, 3) = Mat;
(b) SELECT * FROM pelanggan WHERE alamat LIKE ‘Mat%’

*disclaimer: this question is not related to a specific data source

Answer: Query (b) works better.

Reason: Query (b) is more efficient in terms of processing. The alamat LIKE 'Mat%' condition allows for a more direct and optimized matching process, as it searches for addresses that start with "Mat" without the need to extract the first three characters. In contrast, query (a) involves a lengthier process with SUBSTR(alamat, 1, 3) = 'Mat' where it first extracts the first three characters of each address and then performs the comparison. This makes query (b) more streamlined and potentially faster in execution, especially when dealing with large datasets.

Question 2

Assume we have a pelanggan table with columns: id, nama, tanggal_lahir, Alamat. What is more appropriate way to write a query to retrieve customer data whose tanggal_lahir is between 2000–01–01 and 2008–12–31? Choose one answer and provide the reason.

(a) SELECT * FROM pelanggan WHERE tanggal_lahir >= '2000–01–01' AND tanggal_lahir <= '2008–12–31'

(b) SELECT * FROM pelanggan WHERE tanggal_lahir BETWEEN '2000–01–01' AND ' 2008–12–31'

*disclaimer: this question is not related to a specific data source

Answer: Query (b) works better.

Reason: Using BETWEEN in query (b) is a more suitable and concise approach for filtering tanggal_lahir within a specific range. It simplifies the condition and enhances the readability of the query. This option is more straightforward, making it an efficient way to retrieve customer data with tanggal_lahir between 2000–01 –01 and 2008–12–31.

02

Define Primary Key

Define Primary Key: penjualan Table

The primary key of the **penjualan** table is the **id_invoice** column. This is because the id_invoice has a unique value for each sale; every transaction will have a different id_invoice.

While the id_customer column also has a unique value for each row in the penjualan table, it is not the primary key but rather a foreign key from the pelanggan table**.** The uniqueness of the id_customer in the penjualan table is presumed to be because each customer makes only one purchase, recording a unique id.

03

Design Datamart

Design Datamart

The datamart I created is focused on sales performance, involving three tables: penjualan (sales), pelanggan (customer), and barang (product), utilizing only the relevant columns from each table. The objective is to analyze sales performance for potential improvements and optimizations.

No.	Nama File	Link
1.	SQL Query Datamart Base_Final Task PBI BDA Kimia Farma	Click here
2.	SQL Query Datamart Aggregate_Final Task PBI BDA Kimia Farma	Click here

Table Base: sales_performance

Query | Query History

```sql
-- Create datamart base: sales_performance
CREATE TABLE sales_performance AS(
    SELECT
        pn.tanggal,
        pn.id_distributor,
        pn.id_customer,
        pn.unit AS kemasan,
        pn.harga,
        pn.jumlah_barang,
        (pn.harga * pn.jumlah_barang) AS pendapatan,
        CONCAT(pl.nama, '_', pl.cabang_sales) AS company_cabang,
        b.nama_barang
    FROM penjualan_table pn
    LEFT JOIN pelanggan_table pl ON pn.id_customer = pl.id_customer
    LEFT JOIN barang_table b ON pn.id_barang = b.kode_barang
    ORDER BY pn.tanggal
);

-- Check the datamart base table
SELECT * FROM sales_performance;
```

Base Table: sales_performance

Column	data_type	Description	Transformation
tanggal	Date	Transaction date	-
id_distributor	Character Varying/ String	Distributor ID	-
id_customer	Character Varying/ String	Customer ID	-
kemasan	Character Varying/ String	Product packaging	Change data 'DUS' to 'TABLET'
harga	Double Precision/ Float	Prduct price	-
jumlah_barang	Integer	Total quantity purchased	-
pendapatan	Double Precision/ Float	Revenue from customer transactions	Multiply price with quantity
company_cabang	Character Varying/ String	Name of the company branch and its location	Concatenate name of the company branch with its location
nama_barang	Character Varying/ String	Product name	-

Aggregate Table: sales_aggregate

Query | Query History

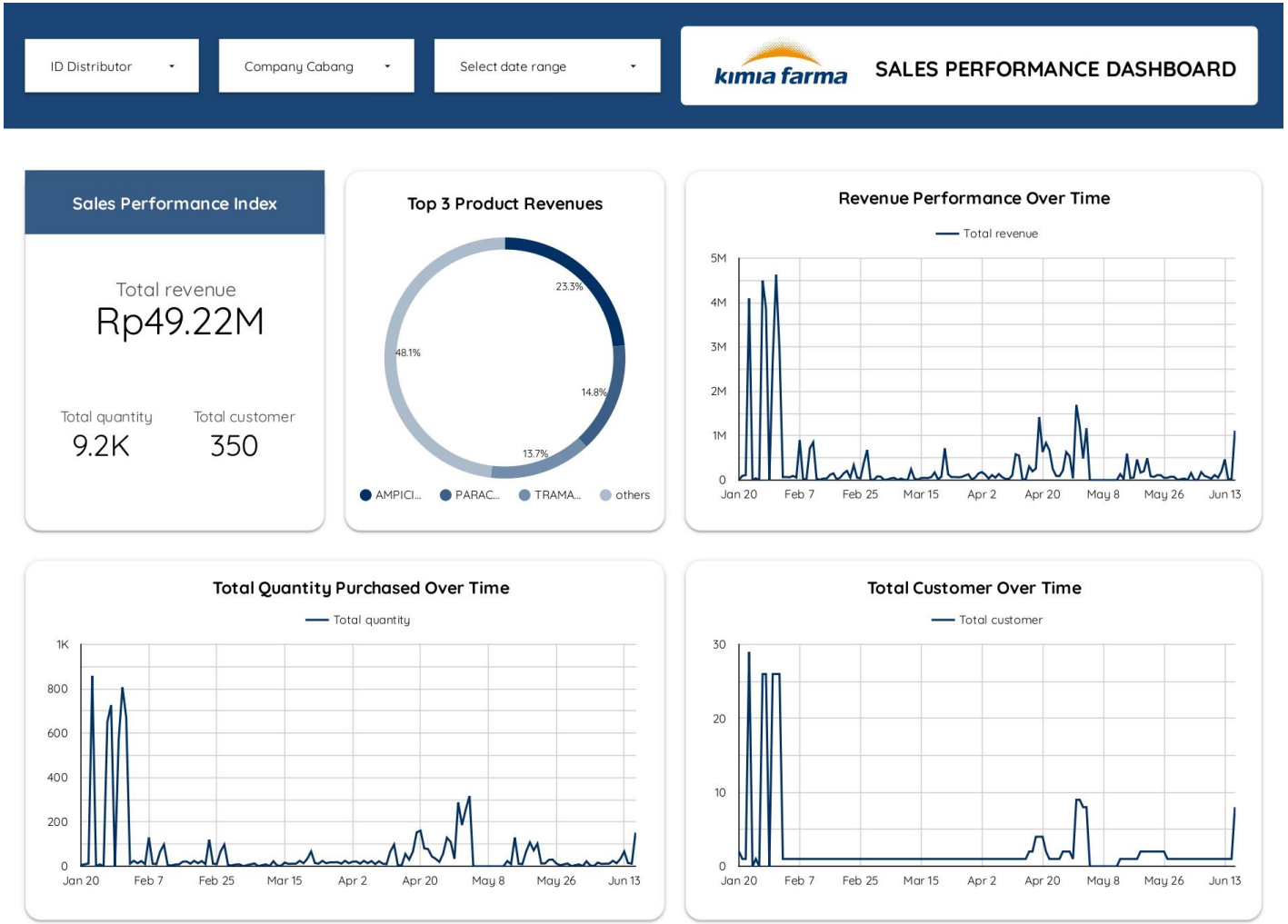
```sql
-- Create datamart aggregate: sales_aggregate
CREATE TABLE sales_aggregate AS(
    SELECT
        nama_barang,
        company_cabang,
        COUNT(tanggal) AS total_transaksi,
        COUNT(DISTINCT id_distributor) AS total_distributor,
        COUNT(DISTINCT id_customer) AS total_customer,
        SUM(pendapatan) AS total_pendapatan
    FROM sales_performance
    GROUP BY nama_barang, company_cabang;
);

-- Check the datamart aggregat table
SELECT * FROM sales_aggregat;
```

Base Table: sales_performance

Column	data_type	Description	Transformation
nama_barang	Character Varying/ String	Product name	-
company_cabang	Character Varying/ String	Name of the company branch and its location	Already done during the creation of the base datamart
total_transaksi	Integer	Total transactions	Calculating the total transactions conducted for each product at each branch of the company
total_distributor	Integer	Total distributor	Calculating the total distributors for each product at each branch of the company
total_customer	Integer	Total customer	Calculating the total customers for each product at each branch of the company
total_pendapatan	Double Precision/ Float	Total revenues	Calculating the total revenue for each product at each branch of the company

04

Data Visualization



For more detail, click here

Additional Complementary Data

The available data is sufficient for the analysis I conducted, but perhaps the analysis would be even better if the data were provided in larger quantities, and if explanations for each table and each column were included to minimize any potential misinterpretations.