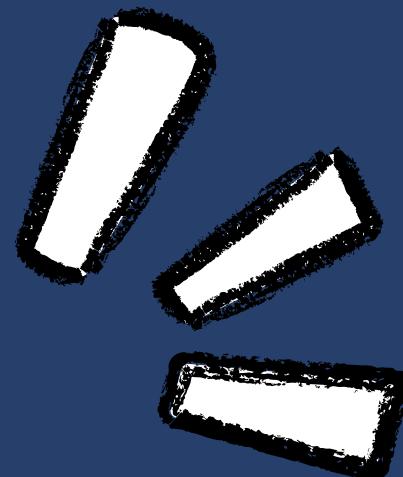


# **BIG DATA ANALYTICS: SALICYL SALES DASHBOARD**



oooo



# QUERY



## Soal 1 \*:

Dari 2 query ini, mana yang bekerja lebih baik? Jelaskan mengapa.

- A. SELECT \* FROM pelanggan WHERE SUBSTR(alamat, 1, 3) = Mat;
- B. SELECT \* FROM pelanggan WHERE alamat LIKE 'Mat%'

*\*disclaimer: soal ini tidak terkait dengan data source*

**Jawaban:** B. SELECT \* FROM pelanggan WHERE alamat LIKE 'Mat%'

### Alasan:

Query A menggunakan fungsi SUBSTR() untuk mengambil tiga karakter pertama dari kolom alamat. Query ini akan mengembalikan hasil yang benar jika alamat pelanggan dimulai dengan "Mat". Namun, query ini akan mengembalikan hasil yang salah jika alamat pelanggan hanya berisi "Mat".

Query B menggunakan operator LIKE untuk mencari nilai "Mat" di awal kolom alamat. Query ini akan mengembalikan hasil yang benar untuk semua alamat pelanggan yang dimulai dengan "Mat", termasuk alamat yang hanya berisi "Mat".

# QUERY



## Soal 2 \*:

Anggap kita memiliki tabel pelanggan dengan kolom: id, nama, tanggal\_lahir, alamat. Bagaimana cara yang lebih tepat dalam menulis query untuk mendapatkan data pelanggan yang tanggal\_lahir nya ada di antara 2000-01-01 sampai 2008-12-31? Pilihlah salah satu jawaban dan berikan alasannya.

- 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: soal ini tidak terkait dengan data source

**Jawaban:** B. SELECT \* FROM pelanggan WHERE tanggal\_lahir BETWEEN '2000-01-01' AND '2008-12-31'

### Alasan:

- Operator BETWEEN lebih tepat digunakan untuk kondisi perbandingan dua nilai tanggal. Operator ini akan mengembalikan nilai true jika nilai tanggal memenuhi kondisi lebih besar sama dengan dan kurang dari sama dengan.
- Operator >= dan <= juga bisa digunakan untuk kondisi perbandingan dua nilai tanggal. Namun, operator BETWEEN lebih efisien karena hanya perlu melakukan satu kali perbandingan.

# MENENTUKAN PK



## Soal 3 \*:

Tentukan primary key dari table penjualan. jelaskan alasannya

### Jawaban:

Primary key dari table penjualan yaitu id\_invoice sebab pada kolom id\_invoice setiap baris memiliki nilai key yang berbeda (unik) dan tidak ada duplikat kemudian tidak ditemukan nilai yang null (kosong). pada data source yang diberikan terdapat 2 kolom yang menjadi pilihan yang tepat diantaranya id\_invoice dan id\_customer. yang membedakan antara 2 kolom ini yaitu bahwasanya pada kolom id\_customer telah dipakai sebagai primary key dari table pelanggan. Oleh sebab itu, kolom id\_invoice menjadi pilihan tepat untuk primary key dari table penjualan.

The screenshot shows a Jupyter Notebook cell with the following SQL code:

```
-- database: c:\Users\ASUS\Documents\Bah Untitled-1
-- database: c:\Users\ASUS\Documents\Bahasa Pemrograman>SQL>kimiaFarma-sales.db
SELECT id_invoice, COUNT(*) AS jumlah_duplicate
FROM penjualan
GROUP BY id_invoice HAVING COUNT(*) > 1;
```

Below the code, there is a table with columns 'id\_invoice' and 'jumlah\_duplicate'. The table has 15 rows, with the first row showing 'id\_invoice' as 1 and 'jumlah\_duplicate' as 1.

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
df_penjualan.info()
```

The output shows the structure of the 'df\_penjualan' DataFrame:

#	Column	Non-Null Count	Dtype	
0	id_distributor	350	non-null	object
1	id_cabang	350	non-null	object
2	id_invoice	350	non-null	object
3	tanggal	350	non-null	datetime64[ns]
4	id_customer	350	non-null	object
5	id_barang	350	non-null	object
6	jumlah_barang	350	non-null	float64
7	unit	350	non-null	object
8	harga	350	non-null	float64
9	mata_uang	350	non-null	object
10	brand_id	350	non-null	object
11	lini	350	non-null	object
12	Unnamed: 12	0	non-null	float64
13	Unnamed: 13	6	non-null	object
14	Unnamed: 14	5	non-null	object

Additional details from the output:

- dtypes: datetime64[ns](1), float64(3), object(11)
- memory usage: 41.1+ KB

# DESIGN DATAMART



## Soal 4 \*:

Buatlah design datamart (Terdiri dari tabel base, dan tabel aggregate). Upload file query dalam gdrive mu (pastikan dapat diakses public). Lalu masukkan linknya di tabel di bawah, dan cantumkan juga screenshot query nya (jika lebih dari 1 file, maka masing masing file di-screenshot)

NAMA FILE	LINK
<b>TABLE BASE (BASE_TABLE)</b>	<a href="#"><u>LINK</u></a>
<b>TABLE AGGREGATE (AGGREGATE_SALESDATE)</b>	<a href="#"><u>LINK</u></a>
<b>TABLE AGGREGATE (AGGREGATE_CUSTOMER)</b>	<a href="#"><u>LINK</u></a>
<b>TABLE AGGREGATE (AGGREGATE_BARANG)</b>	<a href="#"><u>LINK</u></a>

# TABLE BASE “BASE\_TABLE”

```
base_table.sql X

C: > Users > ASUS > Documents > Big Data Analyst_kimia Farma > Final Task > base_table.sql
    Connect | Connect and Open Panel
1 -- database: c:\Users\ASUS\Documents\Bahasa Pemrograman\SQL\kimiaFarma-sales.db
2
3 -- Use the button in the top right corner to run the entire file.
4 -- By ADLI FIQRULLAH
5
6 CREATE TABLE base_sales AS
7     SELECT
8         DATE(tanggal) AS tanggal,
9             id_invoice,
10            penjualan.id_customer,
11            nama AS nama_customer,
12            cabang_sales,
13            "group",
14            id_barang,
15            nama_barang,
16            unit,
17            harga,
18            jumlah_barang,
19            harga * jumlah_barang AS total_harga
20        FROM penjualan
21        JOIN pelanggan ON pelanggan.id_customer = penjualan.id_customer
22        JOIN barang ON barang.kode_barang = penjualan.id_barang;
```

# TABLE BASE “BASE\_TABLE”



COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>TANGGAL</b>	<b>DATE</b>	<b>TANGGAL TRANSAKSI</b>	<b>PENGUBAHAN DATA TYPE MENJADI DATE</b>
<b>ID_INVOICE</b>	<b>VARCHAR</b>	<b>ID INVOICE</b>	-
<b>ID_CUSTOMER</b>	<b>VARCHAR</b>	<b>ID CUSTOMER</b>	-
<b>NAMA_CUSTOMER</b>	<b>VARCHAR</b>	<b>NAMA CUSTOMER</b>	<b>PENGUBAHAN “NAMA” MENJADI “NAMA_CUSTOMER”</b>

# TABLE BASE “BASE\_TABLE”

COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>CABANG_SALES</b>	<b>VARCHAR</b>	<b>WILAYAH CABANG</b>	-
<b>GROUP</b>	<b>VARCHAR</b>	<b>TIPE CABANG</b>	-
<b>ID_BARANG</b>	<b>VARCHAR</b>	<b>ID BARANG</b>	-
<b>NAMA_BARANG</b>	<b>VARCHAR</b>	<b>NAMA BARANG</b>	-



# TABLE BASE “BASE\_TABLE”

COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>UNIT</b>	<b>VARCHAR</b>	<b>KEMASAN BARANG</b>	-
<b>HARGA</b>	<b>DOUBLE</b>	<b>HARGA PERBARANG</b>	-
<b>JUMLAH_BARANG</b>	<b>DOUBLE</b>	<b>JUMLAH PEMBELIAN</b>	-
<b>TOTAL_HARGA</b>	<b>DOUBLE</b>	<b>TOTAL HARGA</b>	<b>HASIL DARI HARGA * JUMLAH_BARANG</b>

# TABLE AGGREGATE “AGGREGATE\_SALESDATE”

```
C: > Users > ASUS > Documents > Big Data Analyst_kimia Farma > Final Task > aggregate_salesDate.sql
    ⌂ Connect | ⌂ Connect and Open Panel
1  -- database: c:\Users\ASUS\Documents\Bahasa Pemrograman\SQL\kimiaFarma-sales.db
2
3  -- Use the ⌂ button in the top right corner to run the entire file.
4  -- By ADLI FIQRULLAH
5
6  CREATE TABLE aggregate_salesDate AS
7      SELECT
8          tanggal,
9          SUM(jumlah_barang) AS total_barang,
10         SUM(total_harga) AS total_penjualan
11     FROM base_sales
12    GROUP BY tanggal;
```



# TABLE AGGREGATE “AGGREGATE\_SALESDATE”

COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>TANGGAL</b>	<b>DATE</b>	<b>TANGGAL TRANSAKSI</b>	-
<b>TOTAL BARANG</b>	<b>DOUBLE</b>	<b>TOTAL BARANG TERJUAL</b>	<b>TOTAL PENJUMLAHAN DARI JUMLAH_BARANG</b>
<b>TOTAL PENJUALAN</b>	<b>DOUBLE</b>	<b>TOTAL PENJUALAN</b>	<b>TOTAL PENJUMLAHAN DARI TOTAL_HARGA</b>

# TABLE AGGREGATE “AGGREGATE\_CUSTOMER”



```
C: > Users > ASUS > Documents > Big Data Analyst_kimia Farma > Final Task > aggregate_customer.sql
    ⌂ Connect | ⌂ Connect and Open Panel
1   -- database: c:\Users\ASUS\Documents\Bahasa Pemrograman\SQL\kimiaFarma-sales.db
2
3   -- Use the ▶ button in the top right corner to run the entire file.
4   -- By ADLI FIQRULLAH
5
6   CREATE TABLE aggregate_customer AS
7       SELECT
8           nama_customer,
9           cabang_sales,
10          "group",
11          SUM(total_harga) AS total_penjualan
12      FROM base_sales
13      GROUP BY nama_customer;
```

# TABLE AGGREGATE “AGGREGATE\_CUSTOMER”



COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>NAMA_CUSTOMER</b>	VARCHAR	<b>NAMA CUSTOMER</b>	-
<b>CABANG SALES</b>	VARCHAR	<b>WILAYAH CABANG</b>	-
<b>GROUP</b>	VARCHAR	<b>TIPE CABANG</b>	-
<b>TOTAL PENJUALAN</b>	DOUBLE	<b>TOTAL PENJUALAN</b>	<b>TOTAL PENJUALAN DARI TOTAL_HARGA</b>

# TABLE AGGREGATE “AGGREGATE\_BARANG”



```
C:\> Users > ASUS > Documents > Big Data Analyst_kimia Farma > Final Task > aggregate_barang.sql
    ⌂ Connect | ⌂ Connect and Open Panel
1   -- database: c:\Users\ASUS\Documents\Bahasa Pemrograman\SQL\kimiaFarma-sales.db
2
3   -- Use the ⌂ button in the top right corner to run the entire file.
4   -- By ADLI FIQRULLAH
5
6   CREATE TABLE aggregate_barang AS
7       SELECT
8           id_barang,
9           nama_barang,
10          unit,
11          SUM(jumlah_barang) AS barang_terjual,
12          SUM(total_harga) AS jumlah_penjualan
13      FROM base_sales
14      GROUP BY id_barang;
```

# TABLE AGGREGATE “AGGREGATE\_BARANG”



COLUMN	DATA TYPE	DESCRIPTION	TRANSFORMATION
<b>ID_BARANG</b>	<b>VARCHAR</b>	<b>ID BARANG</b>	-
<b>NAMA_BARANG</b>	<b>VARCHAR</b>	<b>NAMA BARANG</b>	-
<b>UNIT</b>	<b>VARCHAR</b>	<b>KEMASAN BARANG</b>	-
<b>BARANG_TERJUAL</b>	<b>DOUBLE</b>	<b>JUMLAH PEMBELIAN</b>	<b>TOTAL PENJUMLAHAN DARI JUMLAH_BARANG</b>
<b>JUMLAH PENJUALAN</b>	<b>DOUBLE</b>	<b>TOTAL PENJUALAN</b>	<b>TOTAL PENJUMLAHAN DARI TOTAL_HARGA</b>

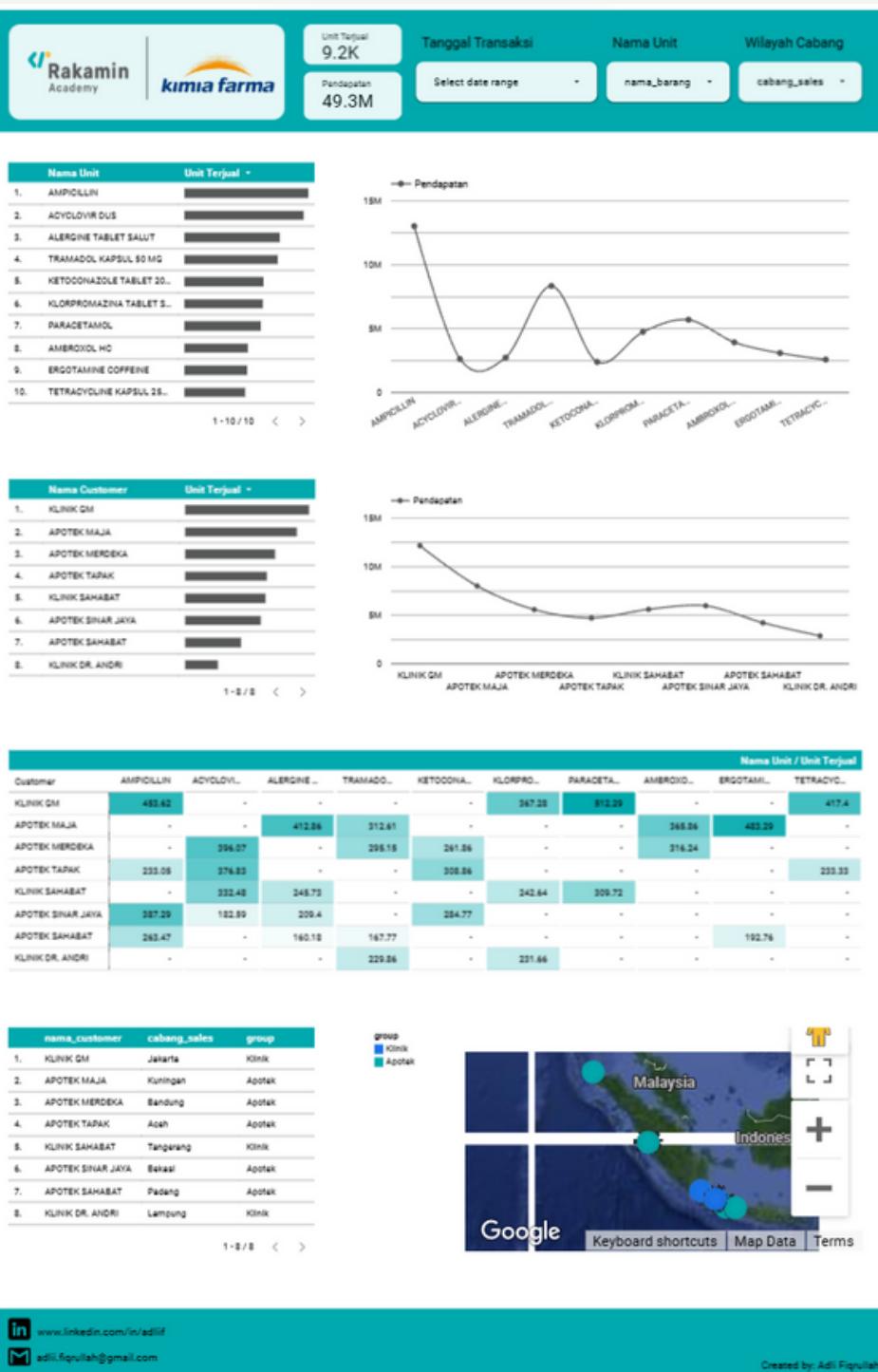
# DATA VISUALIZATION



## Soal 5 \*:

Buatlah data visualisasinya, dan cantumkan linknya di bawah (pastikan bisa diakses publik). Lalu cantumkan juga screenshot visualisasinya

**LINK**



# ADDITIONAL COMPLEMENTARY DATA



## Soal 6 \*:

Dari data yang tersedia, menurut kamu untuk melengkapi analisis nya apakah diperlukan data lain juga? jika iya, sebutkan data apa yang kamu maksud dan mengapa memerlukan data tersebut

### Jawaban:

Menurut pendapat saya pribadi, perlu menambahkan beberapa data seperti pengeluaran yang dibutuhkan (pengiriman barang, modal pembuatan barang) dengan tujuan untuk dapat menghitung laba yang diperoleh. Dengan demikian, kita sebagai data analyst dapat membuat keputusan apakah perusahaan mengalami keuntungan ataupun mengalami kerugian.

Selain itu, informasi ini dapat juga digunakan untuk membuat strategi penjualan dan pemasaran yang lebih efektif. Misalnya, perusahaan dapat membuat suatu diskon kepada produk yang dihasilkan dengan tujuan untuk meningkatkan keuntungan dan menarik banyak konsumen nantinya. Oleh karena itu, saya menyarankan agar data pengeluaran untuk ditambahkan ke dalam data source.



# TERIMA KASIH



[www.linkedin.com/in/adliif](https://www.linkedin.com/in/adliif)



[adlii.fiqrullah@gmail.com](mailto:adlii.fiqrullah@gmail.com)

