Analisis Penjualan Iphone di Indonesia (data simulasi kaggle.com)
Oleh Giffary Soeltoni

Pendahuluan
Proyek ini bertujuan untuk menganalisis data penjualan produk Iphone di Indonesia dengan fokus pada kinerja revenue, unit terjual, dan distribusi pasar. Analisis dilakukan menggunakan PostgreSQL untuk query data dan Tableau untuk visualisasi.

Sumber data
Data ini adalah simulasi dari https://www.kaggle.com/datasets/gerhardien/iphone-transactions-indonesia-market

Tujuan Analisis
1. Mengetahui penjualan Iphone perkota di Indonesia
2. Mengetahui model terlaris perkota di Indonesia
3. Melihat penjualan perbulan perkota
4. Mengidentifikasi market share per model

Tabel Metriks / KPI

Objective	Metrik / KPI	Definisi	Tujuan
1. Mengetahui penjualan iPhone per kota di Indonesia	Total Penjualan per Kota	Jumlah unit iPhone yang terjual di masing-masing kota	Membandingkan performa penjualan antar kota
2. Mengetahui model terlaris per kota di Indonesia	Model Terlaris per Kota	Model iPhone dengan penjualan tertinggi di setiap kota	Menentukan produk unggulan di setiap kota
3. Melihat penjualan per bulan per kota	Penjualan Bulanan per Kota	Jumlah unit iPhone yang terjual tiap bulan di masing-masing kota	Melihat tren penjualan per kota dari waktu ke waktu
4. Mengidentifikasi market share per model	Market Share per Model	Persentase kontribusi penjualan tiap model terhadap total penjualan	Mengukur dominasi model iPhone di pasar

Scope Analisis
Tools: PostgreSQL (data extraction), Tableau (visualisasi), GitHub (dokumentasi & sharing)

Rute pengerjaan:
1. Menentukan metriks
2. Cleaning data dengan PostgreSQL
3. Query data dengan PostgreSQL
4. Visualisasi di Tableau
5. Pembahasan

Cleaning Data
1. Cek data hilang/missing value/null

A. Table customers
Query

```sql
-- cek null tabel customers
select count(*) as total_rows,
count(*) filter(where customer_id is null) as null_customer_id,
count(*) filter(where name is null) as null_name,
count(*) filter(where email is null) as null_email,
count(*) filter(where address is null) as null_address,
count(*) filter(where phone_number is null) as null_phone_number,
count(*) filter(where age is null) as null_age,
count(*) filter(where city is null) as null_city,
count(*) filter(where preferred_product_id is null) as null_preferred_product_id
from iphone_indonesia_market.customers;
```

Hasil

total_rows	null_customer_id	null_name	null_email	null_address	null_phone_number	null_age	null_city	null_preferred_product_id
50000	0	0	0	0	0	0	0	0

B. Table Products

Query

```sql
-- cek null tabel products
select count(*) as table_rows,
count(*) filter(where product_id is null) as null_product_id,
count(*) filter(where product_name is null) as null_product_name,
count(*) filter(where category is null) as null_category,
count(*) filter(where brand is null) as null_brand,
count(*) filter(where price is null) as null_price,
count(*) filter(where description is null) as null_description,
count(*) filter(where discount is null) as null_discount,
count(*) filter(where storage is null) as null_storage,
count(*) filter(where color is null) as null_color,
count(*) filter(where release_year is null) as null_release_year,
count(*) filter(where years_since_release is null) as null_year_since_release,
count(*) filter(where sales_factor is null) as null_sales_factor,
count(*) filter(where preferred_product_id is null) as null_preferred_product_id
from iphone_indonesia_market.products;
```

Hasil

table_rows	null_product_id	null_product_name	null_category	null_brand	null_price	null_description	null_discount	null_storage	null_color	null_release_year	null_year_since_release	null_sales_factor	null_preferred_product_id
12	0	0	0	0	0	0	0	0	0	0	0	0	0

C. Tabel transaction_details

Query

```sql
-- cek null transaction_details
select count(*) as total_rows,
count(*) filter(where transaction_id is null) as null_transaction_id,
count(*) filter(where product_id is null) as null_product_id,
count(*) filter(where quantity is null) as null_quantity,
count(*) filter(where unit_price is null) as null_unit_price,
count(*) filter(where discount is null) as null_discount,
count(*) filter(where total is null) as null_total
from iphone_indonesia_market.transaction_details;
```

Hasil

total_rows	null_transaction_id	null_product_id	null_quantity	null_unit_price	null_discount	null_total
200000	0	0	0	0	0	0

D. Tabel transactions

Query

```sql
-- cek null transactions
select count(*) as total_row,
count(*) filter(where transaction_id is null) as null_transaction_id,
count(*) filter(where customer_id is null) as null_customer_id,
count(*) filter(where transaction_date is null) as null_transaction_date,
count(*) filter(where total_amount is null) as null_total_amount,
count(*) filter(where payment_method is null) as null_payment_method,
count(*) filter(where shipping_method is null) as null_shipping_method,
count(*) filter(where delivery_time is null) as null_delivery_time,
count(*) filter(where coupon_code is null) as null_coupon_code,
count(*) filter(where city is null) as null_city,
count(*) filter(where product_id is null) as null_product_id,
count(*) filter(where sales_factor is null) as null_sales_factor
from iphone_indonesia_market.transactions;
```

Hasil

total_row	null_transaction_id	null_customer_id	null_transaction_date
100000	0	0	0

null_total_amount	null_payment_method	null_shipping_method	null_delivery_time
0	0	0	0

null_coupon_code	null_city	null_product_id	null_sales_factor
0	0	0	0

2. Cek duplikat

A. Tabel customers

```
-- cek duplicate tabel customers
select
customer_id,
name,
email,
address,
phone_number,
age,
city,
preferred_product_id,
count(*) as jumlah
from customers
group by
customer_id,
name,
email,
address,
phone_number,
age,
city,
preferred_product_id
having count(*) > 1;
```

Hasil

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
customer_id	name	email	address	phone_number	age	city	preferred_product_id	jumlah

B. Tabel Products

```
-- cek duplicate table products
select
product_id,
product_name,
category,
brand,
price,
description,
discount,
storage,
color,
release_year,
years_since_release,
sales_factor,
preferred_product_id,
count(*) as jumlah
from products
group by
product_id,
product_name,
category,
brand,
price,
description,
discount,
storage,
color,
release_year,
years_since_release,
sales_factor,
preferred_product_id
having count(*)>1;
```

Hasil

Column1	Column2	Column3	Column4
product_id	product_name	category	brand

Column5	Column6	Column7	Column8	Column9
price	description	discount	storage	color

Column10	Column11	Column12	Column13	Column14
release_year	years_since_release	sales_factor	preferred_product_id	jumlah

C. Tabel transaction_details

```
--cek duplicate table transaction_details (ADA DUPLICATE)
select*from transaction_details;
select
transaction_id,
product_id,
quantity,
unit_price,
discount,
total,
count(*) as jumlah
from transaction_details
group by
transaction_id,
product_id,
quantity,
unit_price,
discount,
total
having count(*)>1;
```

Hasil (ada duplikat 191 data, data yang di sreenshot hanya 5)

transaction_id	product_id	quantity	unit_price	discount	total	jumlah
2146	9	2	12000000	5	22800000	2
5359	9	3	10000000	15	25500000	2
35366	9	3	12000000	15	30600000	2
75186	4	2	7000000	15	11900000	2
54505	3	4	12000000	5	45600000	2

Hasilnya ada duplikat, jadi kita akan delete duplikatnya dengan create table baru tanpa duplikat

```
--hapus duplicate di transaction_details dengan buat table baru
create table transaction_details_clean as
select distinct*
from transaction_details;
```

Hapus tabel lama transaction_details

```
--hapus table transaction_details karena ada duplicate
drop table transaction_details;
```

Ubah nama tabel transaction_details_clean menjadi transaction_details

```
--ubah kembali nama transaction_details_clean menjadi transaction_details
alter table transaction_details_clean
rename to transaction_details;
```

D. Tabel transactions

```
--cek duplicate table transactions
select
transaction_id,
customer_id,
transaction_date,
total_amount,
payment_method,
shipping_method,
delivery_time,
coupon_code,
city,
product_id,
sales_factor,
count(*) as jumlah
from transactions
group by
transaction_id,
customer_id,
transaction_date,
total_amount,
payment_method,
shipping_method,
delivery_time,
coupon_code,
city,
product_id,
sales_factor
having count(*)>1;
```

Hasil

Column1	Column2	Column3	Column4
transaction_id	customer_id	transaction_date	total_amount

Column5	Column6	Column7	Column8	Column9
payment_method	shipping_method	delivery_time	coupon_code	city

Column10	Column11	Column12
product_id	sales_factor	jumlah

3. Cek duplikat

A. Tabel customers

- Tidak ada outliers yang perlu di cek

B. Tabel products

- Tidak ada outliers yag perlu di cek

C. Tabel transaction_details

```sql
--outlier transaction_details
select * from transaction_details;
select max(quantity)
from transaction_details;

select max(unit_price)
from transaction_details;

select total
from transaction_details
where not
--dimana bukan (quantity x unit_price x discount)/100
SELECT
    quantity,
    unit_price,
    discount,
    total,
    (quantity * unit_price) - ((quantity * unit_price * discount) / 100.0) AS seharusnya,
    total - ((quantity * unit_price) - ((quantity * unit_price * discount) / 100.0)) AS selisih
FROM transaction_details
WHERE total <> (quantity * unit_price) - ((quantity * unit_price * discount) / 100.0);
```

Hasil

Column1	Column2	Column3	Column4	Column5	Column6
quantity	unit_price	discount	total	seharusnya	selisih

D. Tabel transactions

- tidak ada outliers yang perlu di cek

4. Cek kesalahan eja/typo

A. Tabel customers, kolom city

```sql
--cek typo table customers dengan cara membuat group setiap isi kolom city
SELECT city, COUNT(*) AS jumlah
FROM customers
GROUP BY city
ORDER BY jumlah ASC;
```

Hasil

city	jumlah
Denpasar	2425
Makassar	2455
Surabaya	2475
Medan	2483
Yogyakarta	5069
Bandung	9920
Jakarta	25173

Nama kota tidak ada yang salah eja

B. Cek typo tabel products, kolom products_id

```sql
--cek typo table products
-- kolom products_id
SELECT product_id, COUNT(*) AS jumlah
FROM products
GROUP BY product_id
ORDER BY jumlah ASC;
```

Hasil

product_id	jumlah
2	1
9	1
4	1
11	1
12	1
10	1
7	1
3	1
6	1
5	1
1	1
8	1

Cek tabel products, kolom product_name

```
-- kolom products_name
SELECT product_name, COUNT(*) AS jumlah
FROM products
GROUP BY product_name
ORDER BY jumlah ASC;
```

Hasil

product_name	jumlah
iPhone 14 Pro Max	1
iPhone 13	1
iPhone 15	1
iPhone SE (3rd Gen)	1
iPhone 14	1
iPhone 13 Pro	1
iPhone 15 Pro	1
iPhone 14 Plus	1
iPhone 13 Pro Max	1
iPhone 15 Pro Max	1
iPhone 15 Plus	1
iPhone 14 Pro	1

C. Tabel transation_details tidak ada

D. Tabel transations, kolom city

```
SELECT city, COUNT(*) AS jumlah
FROM transactions
GROUP BY city
ORDER BY jumlah ASC;
```

Hasil

city	jumlah
Makassar	4928
Denpasar	4935
Surabaya	4942
Medan	5053
Yogyakarta	10163

Bandung	19806
Jakarta	50173

5. Cek tipe data

A. Tabel customers

```sql
--wrong data type table customers
--check:
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'customers';

-- ubah type data
-- Age jadi integer
ALTER TABLE customers
ALTER COLUMN age TYPE integer
USING age::integer;

-- Preferred product id jadi integer
ALTER TABLE customers
ALTER COLUMN preferred_product_id TYPE integer
USING preferred_product_id::integer;
```

B. Tabel products

```sql
--wrong data type table products
--check:
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'products';

-- 1. product_id → integer
ALTER TABLE products
ALTER COLUMN product_id TYPE INT
USING product_id::INT;

-- 2. category → varchar(50)
ALTER TABLE products
ALTER COLUMN category TYPE VARCHAR(50);

-- 3. brand → varchar(50)
ALTER TABLE products
ALTER COLUMN brand TYPE VARCHAR(50);

-- 4. price → numeric(10,2)
ALTER TABLE products
ALTER COLUMN price TYPE NUMERIC(10,2);

-- 5. description → text
ALTER TABLE products
ALTER COLUMN description TYPE TEXT;

-- 6. discount → numeric(5,2)
ALTER TABLE products
ALTER COLUMN discount TYPE NUMERIC(5,2);

-- 7. storage → varchar(20)
ALTER TABLE products
ALTER COLUMN storage TYPE VARCHAR(20);
```

```sql
-- 8. color → varchar(30)
ALTER TABLE products
ALTER COLUMN color TYPE VARCHAR(30);

-- 9. preferred_product_id → integer
ALTER TABLE products
ALTER COLUMN preferred_product_id TYPE INT
USING preferred_product_id::INT;

-- 10. Drop kolom redundant years_since_release
ALTER TABLE products DROP COLUMN years_since_release;
```

C. Tabel transaction_details

```sql
--cek wrong data type table transaction_details
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'transaction_details';
-- 1. product_id → INT (biar bisa join dengan products.product_id)
ALTER TABLE transaction_details
ALTER COLUMN product_id TYPE INT
USING product_id::INT;

-- 2. unit_price → numeric(10,2)
ALTER TABLE transaction_details
ALTER COLUMN unit_price TYPE NUMERIC(10,2);

-- 3. discount → numeric(10,2)
ALTER TABLE transaction_details
ALTER COLUMN discount TYPE NUMERIC(10,2);

-- 4. total → numeric(12,2)
ALTER TABLE transaction_details
ALTER COLUMN total TYPE NUMERIC(12,2);
```

D. Tabel transactions

```sql
--cek wrong data type table transactions
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'transactions';
-- total_amount jadi NUMERIC(12,2)
UPDATE transactions
SET delivery_time = regexp_replace(delivery_time, '[^0-9]', '', 'g')
WHERE delivery_time ~ '[^0-9]';

ALTER TABLE transactions
ALTER COLUMN total_amount TYPE NUMERIC(12,2);

-- delivery_time jadi INTEGER
ALTER TABLE transactions
ALTER COLUMN delivery_time TYPE INTEGER
USING delivery_time::INTEGER;

-- product_id jadi INT biar konsisten dengan tabel products
ALTER TABLE transactions
ALTER COLUMN product_id TYPE INT
USING product_id::INT;
```

6. Cek kolom yang tidak perlu

```sql
--irrelevant data (hapus kolom yg tidak dipakai)
 -- 1. table customers
 ALTER TABLE customers
 DROP COLUMN address,
 DROP COLUMN email,
 drop column phone_number,
 drop column preferred_product_id;
--2. table products
 ALTER TABLE products
 DROP COLUMN category,
 DROP COLUMN brand,
 drop column description,
 drop column color,
 drop column sales_factor,
 drop column preferred_product_id;
--3. table transaction_details tidak ada
 --4. table transactions
 ALTER TABLE transactions
 drop column coupon_code,
 drop column delivery_time,
 drop column sales_factor;
```

7. Hubungan antar kolom

```sql
--Primary Key Uniqueness Pastikan transaction_id, customer_id, product_id nggak ada duplikat.
SELECT transaction_id, COUNT(*)
FROM transactions
GROUP BY transaction_id
HAVING COUNT(*) > 1;

-- Foreign Key Consistency cari transaksi dengan customer_id yang tidak ada di customers
SELECT t.transaction_id, t.customer_id
FROM transactions t
LEFT JOIN customers c ON t.customer_id = c.customer_id
WHERE c.customer_id IS NULL;

--Referential Integrity antar tabel
SELECT td.transaction_id
FROM transaction_details td
LEFT JOIN transactions t ON td.transaction_id = t.transaction_id
WHERE t.transaction_id IS NULL;

--Business Rule Validation age di customers harus > 0 dan wajar (misalnya < 120). price di products t
SELECT * FROM products WHERE price < 0;
SELECT * FROM customers WHERE age <= 0 OR age > 120;
SELECT * FROM transaction_details WHERE discount < 0 OR discount > 100;
```

Query Metriks

Total unit terjual

```sql
--1. total unit terjual
SELECT SUM(td.quantity) AS total_unit_terjual
FROM transaction_details td
INNER JOIN products p
    ON td.product_id = p.product_id
WHERE p.product_name ILIKE 'iPhone%';
```

Hasil

total_unit_terjual
499660

Penjelasan:
Jumlah unit terjual adalh sebanyak 499.660 unit

1. Mengetahui penjualan Iphone perkota di Indonesia

```sql
SELECT t.city, SUM(td.quantity) AS total_unit_terjual
FROM transaction_details td
INNER JOIN transactions t
    ON td.transaction_id = t.transaction_id
GROUP BY t.city

UNION ALL

SELECT 'TOTAL', SUM(td.quantity)
FROM transaction_details td
INNER JOIN transactions t
    ON td.transaction_id = t.transaction_id

ORDER BY total_unit_terjual DESC;
```

Hasil

city	total_unit_terjual
TOTAL	499660
Jakarta	249814
Bandung	99625
Yogyakarta	50890
Surabaya	25081
Medan	25044
Denpasar	24772
Makassar	24434

Penjelasan

· Jakarta mendominasi pasar iPhone di Indonesia, menyumbang setengah dari seluruh penjualan nasional.
· Bandung & Yogyakarta berada di posisi kedua dan ketiga, menunjukkan minat iPhone cukup tinggi di kota besar dan kota pelajar.
· Surabaya, Medan, Denpasar, Makassar punya angka penjualan relatif mirip (sekitar 24–25 ribu unit), jauh di bawah tiga kota teratas.
· Ada gap yang sangat besar antara Jakarta dan kota lainnya → potensi strategi marketing bisa lebih difokuskan ke kota selain Jakarta agar lebih merata.

2. Mengetahui model terlaris perkota di Indonesia

Query

```sql
WITH ranked AS (
    SELECT
        t.city,
        p.product_name AS model_iphone,
        SUM(td.quantity) AS total_unit_terjual,
        RANK() OVER (PARTITION BY t.city ORDER BY SUM(td.quantity) DESC) AS rnk
    FROM transaction_details td
    INNER JOIN transactions t ON td.transaction_id = t.transaction_id
    INNER JOIN products p ON td.product_id = p.product_id
    GROUP BY t.city, p.product_name
)
SELECT city, model_iphone, total_unit_terjual
FROM ranked
WHERE rnk = 1

UNION ALL

-- Baris total seluruh kota
SELECT 'TOTAL', NULL, SUM(total_unit_terjual)
FROM ranked;
```

Hasil

city	model_iphone	total_unit_terjual
Bandung	iPhone 14	8573
Denpasar	iPhone 14	2107
Jakarta	iPhone 15 Plus	21408
Makassar	iPhone 14 Pro Max	2174
Medan	iPhone 15 Pro	2171
Surabaya	iPhone 15 Pro	2245
Yogyakarta	iPhone SE (3rd Gen)	4317

Penjelasan

- **Kota dengan penjualan tertinggi: Jakarta (21.408 unit, iPhone 15 Plus).**
- **Kota dengan penjualan terendah: Denpasar (2.107 unit, iPhone 14).**
- **Model bervariasi per kota, artinya setiap kota punya preferensi berbeda untuk model iPhone.**
- **iPhone 15 Series (15 Plus & 15 Pro) terlihat cukup dominan di beberapa kota besar (Jakarta, Medan, Surabaya).**

3. Melihat penjualan perbulan di Indonesia

Query

```sql
SELECT
    COALESCE(TO_CHAR(DATE_TRUNC('month', t.transaction_date), 'YYYY-MM'), 'TOTAL') AS bulan,
    SUM(td.quantity) AS total_unit_terjual
FROM transaction_details td
INNER JOIN transactions t
    ON td.transaction_id = t.transaction_id
GROUP BY ROLLUP(DATE_TRUNC('month', t.transaction_date))
ORDER BY
    CASE WHEN DATE_TRUNC('month', t.transaction_date) IS NULL THEN 2 ELSE 1 END,
    bulan;
```

Hasil contoh 5 baris

bulan	total_unit_terjual
2021-12	3793
2022-01	13715
2022-02	12912
2022-03	14678
2022-04	13471
2022-05	14010
2022-06	14176
2022-07	13922

Penjelasan

· Kenaikan awal signifikan menunjukkan adanya momentum (mungkin peluncuran produk baru atau promosi besar di 2022).

· Stabilitas tinggi dari 2022–2024 → pasar iPhone sudah mapan dengan permintaan konsisten.

· Fluktuasi kecil bulanan (±500–1.000 unit) wajar, mungkin dipengaruhi siklus promo, musim belanja, atau peluncuran seri baru.

· Penurunan di akhir 2024 (10.366 unit) perlu diperhatikan, bisa menandakan:

- Pergeseran tren konsumen ke merek/produk lain.

- Menunggu peluncuran model baru.

- Faktor ekonomi/musim.

4. Mengidentifikasi market share per model

· Generasi terbaru (iPhone 15 Series) → memiliki kontribusi signifikan (15 Plus, Pro, Pro Max, dan 15 biasa) → total jika digabung lebih besar dibanding seri lama.

· Model "Plus" (iPhone 15 Plus & 14 Plus) punya posisi kuat → artinya konsumen di Indonesia cukup menyukai varian dengan layar lebih besar, tapi lebih murah dibanding seri Pro.

· Market share yang merata menunjukkan tidak ada model yang benar-benar mendominasi → preferensi konsumen sangat beragam, bergantung pada budget dan fitur yang dibutuhkan.

· Produk lama (iPhone 13 Series & iPhone SE 3rd Gen) masih punya porsi signifikan, artinya ada segmen pasar yang memilih harga lebih terjangkau.

·