

Riantama Putra
18224061

IF2040 Pemodelan Basis Data **Semester I 2025/2026**

Praktikum Pengganti

Dipersiapkan oleh:
Asisten Laboratorium Basis Data

Waktu Pelaksanaan:
Jum'at, 7 November 2025 – 16.10 - 17.50 WIB
Durasi: 100 Menit

KETENTUAN

Peserta kuliah IF2040 akan mengerjakan praktikum 2 mengenai *Advanced SQL*. Silakan mengunduh *docker* yang sudah disediakan pada *folder* praktikum untuk mendapatkan *database* yang akan digunakan.

Peserta dapat memanfaatkan *file* yang terdapat pada folder **Untuk Peserta** dan **Dokumentasi MySQL** dalam pengerjaan soal praktikum.

Deliverables yang harus dikumpulkan untuk Praktikum ini adalah **file .pdf** berisikan lembar kerja praktikum ini (daftar *query* beserta *screenshot* eksekusi tiap *query*) dengan format nama: **K<Kelas>_P3_<NIM>.pdf**

Contoh: K01_P3_18224000.pdf

Isi dari dokumen adalah sebagai berikut.

- a. Untuk setiap soal:
 - i. Perintah atau langkah yang dilakukan untuk menyelesaikan persoalan. Sediakan semua kode dalam bentuk teks.
 - ii. *Query* pengecekan dalam bentuk teks (jika diminta pada soal).
 - iii. Hasil perintah / *output* dari *query* tersebut.

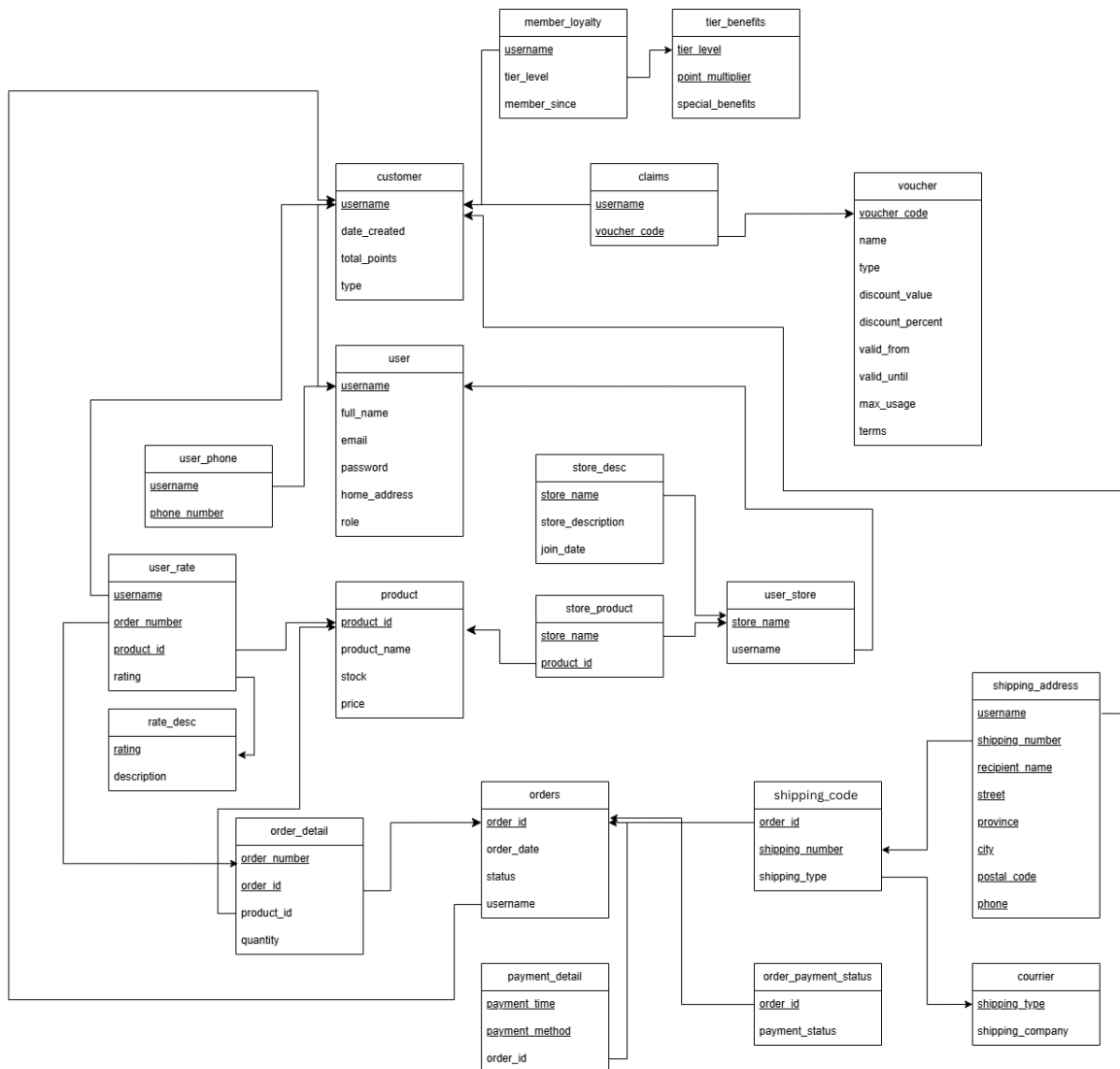
Deliverables dikumpulkan pada pranala berikut.

<https://s.hmif.dev/SubmitPraktikumIF2040>

Keterlambatan pengumpulan *deliverables* akan mengakibatkan pengurangan nilai. **Segala tindak kecurangan akan ditindaklanjuti dan diikuti konsekuensi serius.**

~ Selamat mengerjakan! ♥ ~

I. Skema Basis Data



- Voucher = (**voucher_code**, name, type, discount_value, discount_percent, valid_from, valid_until, max_usage, terms)
- Product = (**product_id**, product_name, stock, price)
- User = (**username**, full_name, email, password, home_address, role)
- User_phone = (**username**, **phone_number**)
- User_store = (**store_name**, username)
- Store_desc = (**store_name**, store_description, join_date)
- Order_detail = (**order_number**, **order_id**, product_id, quantity)

- Shipping_address = (username, shipping_number, recipient_name, street, city, province, postal_code, phone)
- Claims = (username, voucher_code)
- Store_product = (store_name, product_id)
- Payment_detail = (payment_time, payment_method, order_id)
- Order_payment_status = (order_id, payment_status)
- Shipping_code = (order_id, shipping_number, shipping_type)
- Courier = (shipping_type, shipping_company)
- Order = (order_id, order_date, status, username)
- User_rate = (username, order_number, product_id, rating)
- Rate_desc = (rating, description)
- Member_loyalty = (username, tier_level, point_multiplier, member_since, special_benefits)
- Tier_benefits = (tier_level, point_multiplier, special_benefits)
- Customer = (username, date_created, total_points, type)

II. Hint Umum

Pembatasan pada hasil dapat dilakukan dengan fungsi LIMIT(offset, row count) dengan offset dimulai dari 0 seperti indeks pada umumnya dan row count akan mewakili berapa banyak row yang diinginkan.

Contoh:

LIMIT 10 akan menunjukkan 10 row teratas.

LIMIT 2, 5 akan menunjukkan 5 row dimulai dari row ke-3

III. Soal

- Untuk soal yang hasil querynya panjang, SS minimal meliputi 5 baris awal termasuk query DAN 5 baris terakhir termasuk jumlah row
- Kerjakan bagian 1 terlebih dahulu sebelum berlanjut ke bagian 2

BAGIAN I - DB Querying

1. Tampilkan username pelanggan Shopee dengan type = 'loyalty' dan total_harga_pengiriman (total harga produk dari pesanan yang sudah dikirim), yang memiliki minimal 2 pesanan berstatus "shipped".

Query	<pre> SELECT c.username, sum(p.price*od.quantity) AS total_harga_pengiriman FROM customer c JOIN orders o ON c.username = o.username JOIN order_detail od ON o.order_id = od.order_id JOIN product p ON od.product_id = p.product_id JOIN shipping_code sc ON o.order_id = sc.order_id WHERE c.type = 'loyalty' AND o.status = 'shipped' GROUP BY c.username HAVING COUNT(o.order_id) >= 2; </pre>
-------	--

SS	<pre> +-----+-----+ username total_harga_pengiriman +-----+-----+ Cahya389 1702 Fani698 1892 Hendri894 1306 Krisna643 4842 Miko165 1772 Miko538 578 Shinta880 2377 Sonny68 1294 Umi810 1404 Vino955 2256 Wahyu375 3797 +-----+-----+ 11 rows in set (0.001 sec) </pre>
----	---

2. Tampilan **username customer (alfabetik), tipe customer, dan kode voucher** yang pernah digunakan, di mana total poin customer berkisar dari **2000 hingga 5000** (inklusif) serta kode voucher diawali dengan 'VCHR' dan memiliki discount_value kurang dari rata-rata discount_value secara keseluruhan

Query	<pre> SELECT c.username, c.type, v.voucher_code FROM customer c JOIN claims cl ON c.username = cl.username JOIN voucher v ON cl.voucher_code = v.voucher_code WHERE c.total_points BETWEEN 2000 AND 5000 AND v.voucher_code like 'VCHR%' AND v.discount_value < (SELECT AVG(discount_value) FROM voucher v) GROUP BY c.username, c.type, v.voucher_code; </pre>
-------	---

SS	<pre> +-----+-----+-----+ username type voucher_code +-----+-----+-----+ Irfan874 basic VCHR148 Tia623 basic VCHR131 Yuda613 basic VCHR749 +-----+-----+-----+ 3 rows in set (0.001 sec) </pre>
----	--

3. Buatlah query SQL untuk menampilkan total pengeluaran setiap pelanggan pada setiap store selama tahun 2024!

HINT:

- Hanya perhitungkan *order* yang memiliki status *completed*

Query	<pre> SELECT c.username, us.store_name, SUM(od.quantity * p.price) AS total_spending FROM orders o JOIN order_detail od ON o.order_id = od.order_id JOIN product p ON od.product_id = p.product_id JOIN customer c ON o.username = c.username JOIN user_store us ON c.username = us.username WHERE o.status = 'completed' AND o.order_date BETWEEN '2024-01-01' AND '2024-12-31' GROUP BY c.username, us.store_name; </pre>
-------	---

SS	<pre> +-----+-----+-----+ username store_name total_spending +-----+-----+-----+ Bambang755 Seni & Kerajinan 14 Beby464 Sentuhan Mewah 160 Clara550 Sinarnya Abadi 78 Fikri947 Solusi Gadget 512 Hendri894 Sportivo 3527 Miko842 Toko Amanah 910 Putri216 Toko Segar 339 Putri859 Tontonan Favorit 246 Surya893 Trend Elektronik 174 +-----+-----+-----+ 9 rows in set (0.001 sec) </pre>
----	---

Bagian 2

Kerjakan bagian ini setelah selesai mengerjakan bagian 1. Terdapat 3 file SS yang diminta pada bagian ini, yaitu sebelum, saat, dan sesudah melakukan modifikasi.

Pada bagian II, SS minimal meliputi hasil query seleksi sebelum modifikasi atau pembuatan view, SS saat melakukan modifikasi atau pembuatan view, dan SS seleksi setelah modifikasi atau pembuatan view. SS saat pastikan terdapat tulisan 'Query OK' dan berapa rows yang dimodifikasi

Query seleksi digunakan untuk menghasilkan 'SS sebelum' dan 'SS sesudah' sehingga terlihat perbedaan sebelum dan sesudah modifikasi

- Buatlah query pembuatan *view* dengan nama **store_performance** yang berisikan **store_name**, total pembelian item pada setiap store sebagai **total_item**, dan total order pada setiap store sebagai **total_order**. Setelah berhasil membuat **store_performance**, tampilkan isinya!

HINT:

Cara pembuatan view:

```
CREATE VIEW <nama view> AS
    <Query Pembuatan>
;
```

Query Seleksi	<pre>SELECT sd.store_name, SUM(od.quantity) as total_item, COUNT(DISTINCT o.order_id) as total_order FROM store_product sp JOIN order_detail od ON sp.product_id = od.product_id JOIN orders o ON od.order_id = o.order_id JOIN store_desc sd ON sp.store_name = sd.store_name GROUP BY sd.store_name;</pre>
SS Query Seleksi Sebelum Modifikasi	<pre>+-----+-----+-----+ store_name total_item total_order +-----+-----+-----+ Aksesoris Cantik 7 2 Alat Dapur 6 3 Alat Kantor 3 2 Art & Craft 8 2 Bahagia Store 6 2 Travel Gear 8 3 Trend Center 1 1 Trend Elektronik 15 4 Trend Optik 3 2 Tulisanku 20 6 +-----+-----+-----+ 85 rows in set (0.002 sec)</pre>
SS Query Seleksi Setelah Modifikasi	<pre>MariaDB [shopee]> SELECT * FROM store_performance; +-----+-----+-----+ store_name total_item total_order +-----+-----+-----+ Aksesoris Cantik 7 2 Alat Dapur 6 3 Alat Kantor 3 2 Art & Craft 8 2 Bahagia Store 6 2 </pre>

	<pre> Travel Gear 8 3 Trend Center 1 1 Trend Elektronik 15 4 Trend Optik 3 2 Tulisanku 20 6 +-----+-----+ 85 rows in set (0.002 sec) </pre>
Query Modifikasi	<pre> CREATE VIEW store_performance AS SELECT sd.store_name, SUM(od.quantity) as total_item, COUNT(DISTINCT o.order_id) as total_order FROM store_product sp JOIN order_detail od ON sp.product_id = od.product_id JOIN orders o ON od.order_id = o.order_id JOIN store_desc sd ON sp.store_name = sd.store_name GROUP BY sd.store_name; </pre>
SS Saat Modifikasi	<pre> MariaDB [shopee]> CREATE VIEW store_performance AS -> SELECT sd.store_name, SUM(od.quantity) as total_item, COUNT(DISTINCT o.order_id) as total_order -> FROM store_product sp -> JOIN order_detail od ON sp.product_id = od.product_id -> JOIN orders o ON od.order_id = o.order_id -> JOIN store_desc sd ON sp.store_name = sd.store_name -> GROUP BY sd.store_name; Query OK, 0 rows affected (0.011 sec) </pre>

5. Tim *marketing* Shopee baru saja meluncurkan program Shopee Loyalty yang membagi *customer* ke dalam beberapa level: **Silver**, **Gold**, dan **Platinum** berdasarkan total poin yang mereka kumpulkan dari transaksi. Namun, data ini belum masuk ke *database*. Sebagai *data analyst*, kamu diminta untuk menambahkan kolom baru yang bernama **loyalty_level** dengan tipe data VARCHAR(20) lalu mengisi nilainya secara otomatis berdasarkan jumlah poin yang ada sebagai berikut:

- Jika total poin $\geq 8000 \rightarrow$ Platinum
- Jika total poin $\geq 5000 \rightarrow$ Gold
- Selain itu, masuk ke dalam Silver

Tampilkan hasilnya dengan menampilkan semua atribut dalam tabel Customer terurut menurun berdasarkan total poin pelanggan.

HINT:

- Cukup menuliskan ' ≥ 5000 ' pada query, tidak perlu menuliskan BETWEEN 5000 and 7999 karena SQL secara otomatis memasukkan ke dalam prioritas

Query Seleksi	select * from customer;
SS Query Seleksi Sebelum Modifikasi	<pre> MariaDB [shopee]> select * from customer -> ; +-----+-----+-----+-----+ username date_created total_points type +-----+-----+-----+-----+ Agus140 2010-12-23 11:02:02 3874 loyalty Agus178 2019-05-14 21:00:33 5190 loyalty Agus190 2019-12-04 19:38:50 6824 basic Agus563 2018-05-27 03:04:25 5892 loyalty Agus746 2019-01-04 11:06:33 1369 loyalty Zaki772 2019-08-21 22:57:03 614 loyalty Zaky872 2018-04-07 13:49:24 4555 basic Zamzam388 2020-11-01 20:03:21 6270 loyalty Zulfikar236 2014-04-13 00:25:04 9956 loyalty Zulfikar875 2021-03-20 01:05:43 1905 loyalty +-----+-----+-----+-----+ 357 rows in set (0.001 sec) </pre>
SS Query Seleksi Setelah Modifikasi	<pre> MariaDB [shopee]> select * from customer; +-----+-----+-----+-----+-----+ username date_created total_points type loyalty_level +-----+-----+-----+-----+-----+ Agus140 2010-12-23 11:02:02 3874 loyalty Silver Agus178 2019-05-14 21:00:33 5190 loyalty Gold Agus190 2019-12-04 19:38:50 6824 basic Gold Agus563 2018-05-27 03:04:25 5892 loyalty Gold Agus746 2019-01-04 11:06:33 1369 loyalty Silver Zaki772 2019-08-21 22:57:03 614 loyalty Silver Zaky872 2018-04-07 13:49:24 4555 basic Silver Zamzam388 2020-11-01 20:03:21 6270 loyalty Gold Zulfikar236 2014-04-13 00:25:04 9956 loyalty Platinum Zulfikar875 2021-03-20 01:05:43 1905 loyalty Silver +-----+-----+-----+-----+-----+ 357 rows in set (0.000 sec) </pre>
Query Modifikasi	<pre> ALTER TABLE customer ADD COLUMN loyalty_level VARCHAR(20); UPDATE customer SET loyalty_level = CASE WHEN total_points >= 8000 THEN 'Platinum' WHEN total_points >= 5000 THEN 'Gold' ELSE 'Silver' END; </pre>

SS Saat Modifikasi	<pre> MariaDB [shopee]> ALTER TABLE customer ADD COLUMN loyalty_level VARCHAR(20); Query OK, 0 rows affected (0.022 sec) Records: 0 Duplicates: 0 Warnings: 0 MariaDB [shopee]> UPDATE customer -> SET loyalty_level = CASE WHEN total_points >= 8000 THEN 'Platinum' -> WHEN total_points >= 5000 THEN 'Gold' -> ELSE 'Silver' -> END; Query OK, 357 rows affected (0.055 sec) Rows matched: 357 Changed: 357 Warnings: 0 </pre>
--------------------	--

6. Ody adalah salah satu pemilik toko di Shopee dengan nama 'Toko Idaman'. Sebagai pemilik yang baik, Ody ingin mencari tahu produk yang paling diminati di tokonya. Buatlah view dengan nama 'top_toko_idaman' yang dapat menampilkan ID produk, nama produk, dan jumlah produk yang terjual.

Query Seleksi	<pre> SELECT sp.product_id, p.product_name, SUM(od.quantity) AS total_sold FROM store_product sp JOIN product p ON sp.product_id = p.product_id JOIN order_detail od ON p.product_id = od.product_id JOIN orders o ON od.order_id = o.order_id JOIN user_store us ON o.username = us.username WHERE us.store_name = 'Toko Idaman' GROUP BY sp.product_id, p.product_name ORDER BY total_sold desc; </pre>
Query pembuatan view	<pre> CREATE VIEW top_toko_idaman AS SELECT sp.product_id, p.product_name, SUM(od.quantity) AS total_sold FROM store_product sp JOIN product p ON sp.product_id = p.product_id JOIN order_detail od ON p.product_id = od.product_id JOIN orders o ON od.order_id = o.order_id JOIN user_store us ON o.username = us.username WHERE us.store_name = 'Toko Idaman' GROUP BY sp.product_id, p.product_name ORDER BY total_sold DESC; </pre>

<p>SS Pembuatan View</p>	<pre>MariaDB [shopee]> CREATE VIEW top_toko_idaman AS -> SELECT sp.product_id, p.product_name, SUM(od.quantity) AS total_sold -> FROM store_product sp -> JOIN product p ON sp.product_id = p.product_id -> JOIN order_detail od ON p.product_id = od.product_id -> JOIN orders o ON od.order_id = o.order_id -> JOIN user_store us ON o.username = us.username -> WHERE us.store_name = 'Toko Idaman' -> GROUP BY sp.product_id, p.product_name -> ORDER BY total_sold DESC; Query OK, 0 rows affected (0.009 sec)</pre>
<p>SS Hasil view</p>	<pre>MariaDB [shopee]> select * from top_toko_idaman; Empty set (0.001 sec)</pre>



Selamat sudah menyelesaikan praktikum PBD ^_^
Salam Basis Data
~ Asisten