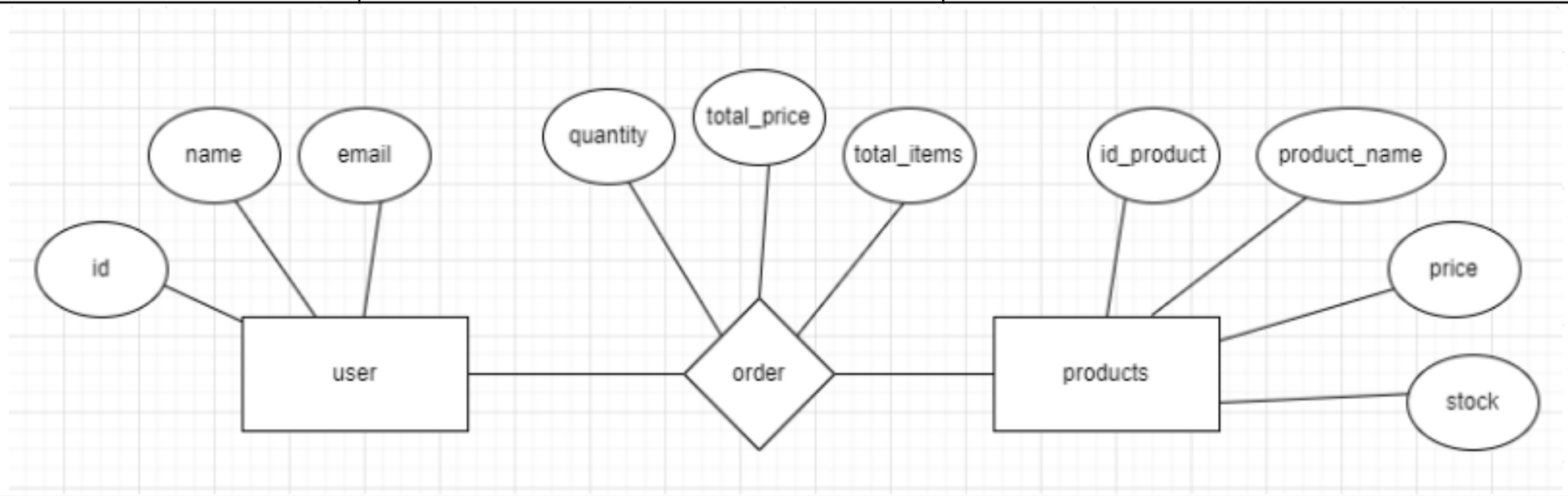
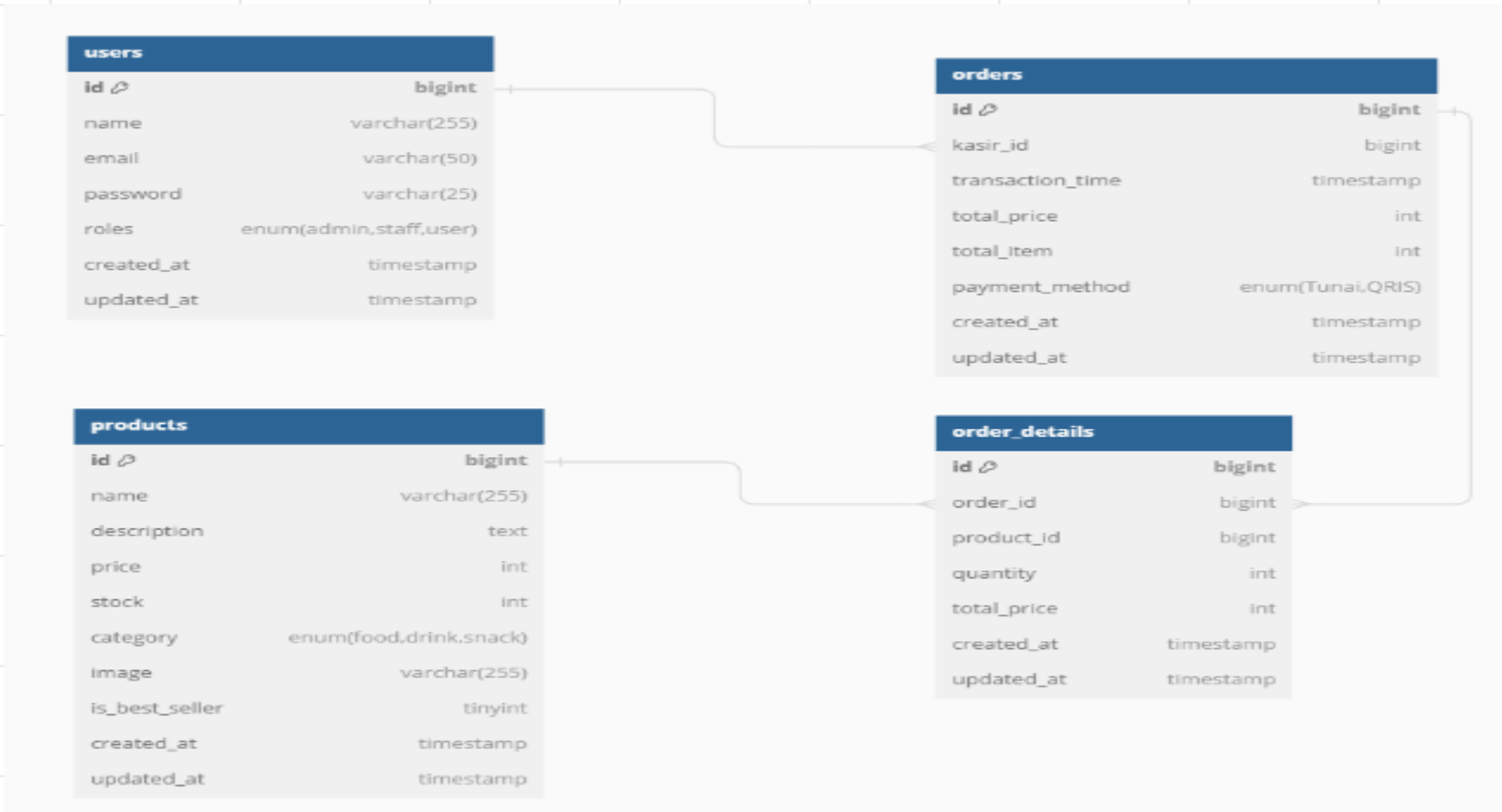
TUGAS KELOMPOK UJI KOMPETENSI BERBASIS INDUSTRI BIDANG SISTEM BASIS DATA

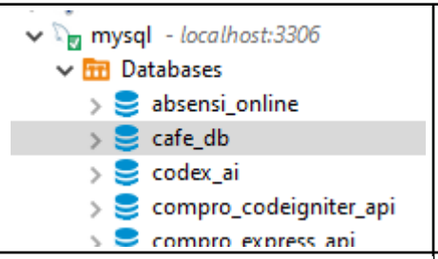
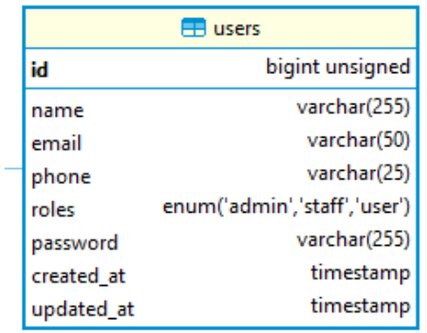
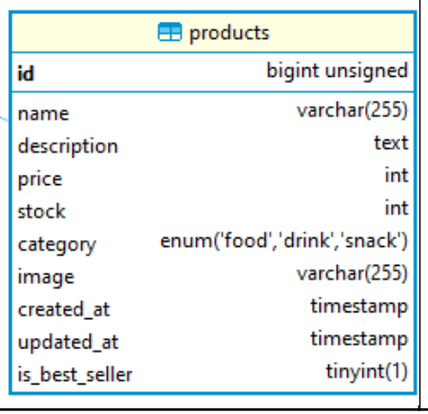
Kelompok : **99**
DMBS : **MariaDB**
Tema : **Aplikasi Order Coffee Shop**
Anggota Kelompok :

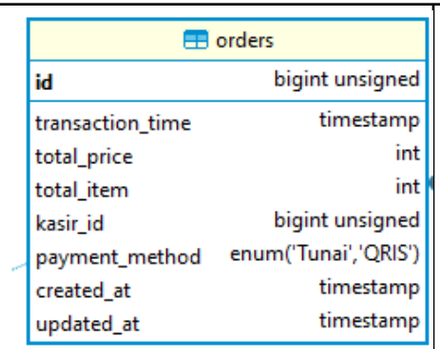
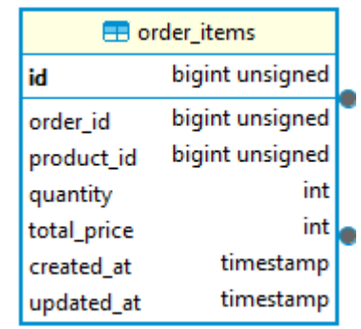
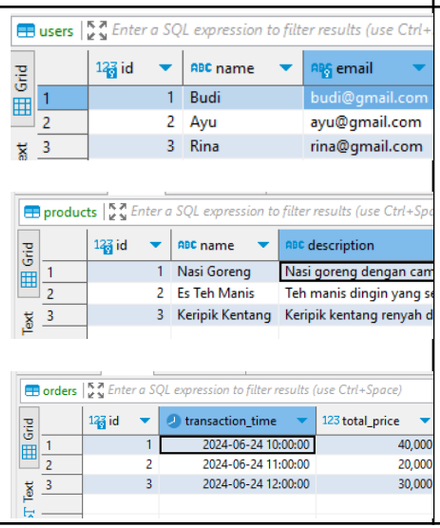
1. Ilham Maulana (14230018)
2. Ilham Maulana (14230019)
3. Ilham Maulana (14230020)
4. Ilham Maulana (14230021)
5. Ilham Maulana (14230022)

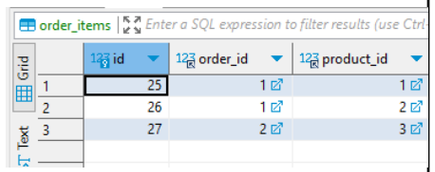
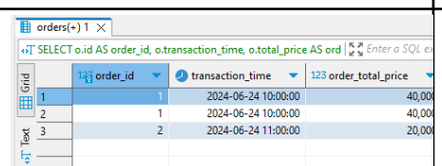
.oO Semoga Sukses Oo.

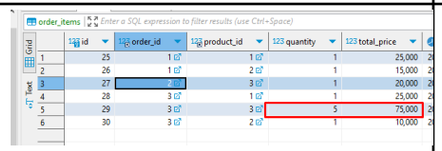
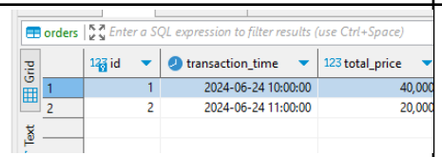
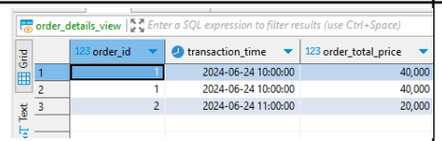
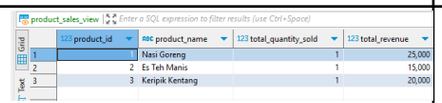
No	Nama Query	Hasil Gambar	Query Sql	Penjelasan Query
	ERD			

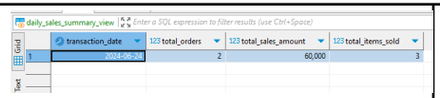
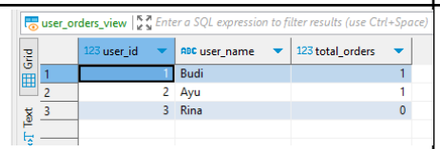
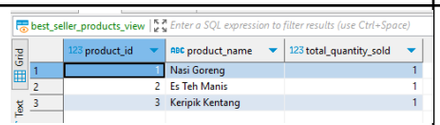
	ER			
No	**Nama Query**	**Hasil Gambar**	**Query Sql**	**Penjelasan Query**

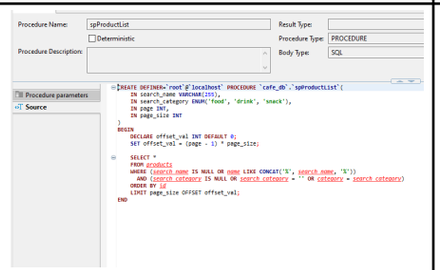
1	CREATE DATABASE		CREATE DATABASE cafe_db2; USE cafe_db;	Membuat database baru dan memilih database
2	CREATE TABLE		CREATE TABLE `users` (`id` bigint unsigned NOT NULL AUTO_INCREMENT, `name` varchar(255) NOT NULL, `email` varchar(50) NOT NULL, `phone` varchar(25) DEFAULT NULL, `roles` enum('admin','staff','user') NOT NULL DEFAULT 'user', `password` varchar(255) NOT NULL, `created_at` timestamp NULL DEFAULT NULL, `updated_at` timestamp NULL DEFAULT NULL, PRIMARY KEY (`id`), UNIQUE KEY `users_email_unique` (`email`));	Membuat Tabel users
3	CREATE TABLE		CREATE TABLE `products` (`id` bigint unsigned NOT NULL AUTO_INCREMENT, `name` varchar(255) NOT NULL, `description` text, `price` int NOT NULL DEFAULT '0', `stock` int NOT NULL DEFAULT '0', `category` enum('food','drink','snack') NOT NULL, `image` varchar(255) DEFAULT NULL, `created_at` timestamp NULL DEFAULT NULL, `updated_at` timestamp NULL DEFAULT NULL, `is_best_seller` tinyint(1) NOT NULL DEFAULT '0', PRIMARY KEY (`id`));	Membuat Tabel products

4	CREATE TABLE		CREATE TABLE `orders` (`id` bigint unsigned NOT NULL AUTO_INCREMENT, `transaction_time` timestamp NOT NULL, `total_price` int NOT NULL, `total_item` int NOT NULL, `kasir_id` bigint unsigned NOT NULL, `payment_method` enum('Tunai','QRIS') NOT NULL, `created_at` timestamp NULL DEFAULT NULL, `updated_at` timestamp NULL DEFAULT NULL, PRIMARY KEY (`id`), KEY `orders_kasir_id_foreign` (`kasir_id`), CONSTRAINT `orders_kasir_id_foreign` FOREIGN KEY (`kasir_id`) REFERENCES `users` (`id`));	Membuat Tabel orders
5	CREATE TABLE		CREATE TABLE `order_items` (`id` bigint unsigned NOT NULL AUTO_INCREMENT, `order_id` bigint unsigned NOT NULL, `product_id` bigint unsigned NOT NULL, `quantity` int NOT NULL, `total_price` int NOT NULL, `created_at` timestamp NULL DEFAULT NULL, `updated_at` timestamp NULL DEFAULT NULL, PRIMARY KEY (`id`), KEY `order_items_order_id_foreign` (`order_id`), KEY `order_items_product_id_foreign` (`product_id`), CONSTRAINT `order_items_order_id_foreign` FOREIGN KEY (`order_id`) REFERENCES `orders` (`id`), CONSTRAINT `order_items_product_id_foreign` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`));	Membuat Tabel order_items
6	INSERT		INSERT INTO `users` (name, email, phone, roles, password, created_at, updated_at) VALUES ('Budi', 'budi@gmail.com', '085644445555', 'user', SHA2('budi123', 256), NOW(), NOW()), ('Ayu', 'ayu@gmail.com', '085655556666', 'user', SHA2('ayu123', 256), NOW(), NOW()), ('Rina', 'rina@gmail.com', '085666667777', 'user', SHA2('rina123', 256), NOW(), NOW()); INSERT INTO `products` (name, description, price, stock, category, image, created_at, updated_at, is_best_seller) VALUES ('Nasi Goreng', 'Nasi goreng dengan campuran sayuran dan daging ayam.', 25000, 50, 'food', 'nasi_goreng.jpg', NOW(), NOW(), 1), ('Es Teh Manis', 'Teh manis dingin yang segar.', 5000, 100, 'drink', 'es_teh_manis.jpg', NOW(), NOW(), 0),	Menambahkan data kedalam tabel users, products, orders dan order_items

			('Keripik Kentang', 'Keripik kentang renyah dengan rasa gurih.', 15000, 30, 'snack', 'keripik_kentang.jpg', NOW(), NOW(), 0); INSERT INTO `orders` (transaction_time, total_price, total_item, kasir_id, payment_method, created_at, updated_at) VALUES ('2024-06-24 10:00:00', 40000, 2, 1, 'Tunai', NOW(), NOW()), ('2024-06-24 11:00:00', 20000, 1, 2, 'QRIS', NOW(), NOW()), ('2024-06-24 12:00:00', 30000, 3, 1, 'Tunai', NOW(), NOW()); INSERT INTO `order_items` (order_id, product_id, quantity, total_price, created_at, updated_at) VALUES (1, 1, 1, 25000, NOW(), NOW()), (1, 2, 1, 15000, NOW(), NOW()); INSERT INTO `order_items` (order_id, product_id, quantity, total_price, created_at, updated_at) VALUES (2, 3, 1, 20000, NOW(), NOW()); INSERT INTO `order_items` (order_id, product_id, quantity, total_price, created_at, updated_at) VALUES (3, 1, 1, 25000, NOW(), NOW()), (3, 3, 1, 15000, NOW(), NOW()), (3, 2, 1, 10000, NOW(), NOW());	
7	SELECT		SELECT o.id AS *order_id*, o.transaction_time, o.total_price AS *order_total_price*, o.total_item AS *order_total_item*, u.name AS *kasir_name*, o.payment_method, p.name AS *product_name*, p.description AS *product_description*, p.price AS *product_price*, oi.quantity, oi.total_price AS *item_total_price* FROM order_items *oi* LEFT JOIN orders *o* ON *oi*.order_id = *o*.id LEFT JOIN users *u* ON *o*.kasir_id = *u*.id LEFT JOIN products *p* ON *oi*.product_id = *p*.id;	Menampilkan isi tabel order_details dan relasi ke tabel orders, users dan products

8	UPDATE		UPDATE order_items oi JOIN products p ON oi.product_id = p.id SET oi.quantity = 5, oi.total_price = 5 * p.price, oi.updated_at = NOW() WHERE oi.order_id = 3 AND oi.product_id = 3;	Merubah quantity=5 dan total_price pada tabel order_items yang berelasi ke tabel products
9	DELETE		DELETE FROM order_items WHERE order_id = 3; DELETE FROM orders WHERE id = 3;	Menghapus order_id=3 pada tabel orders dan order_items
10	CREATE VIEW		CREATE VIEW order_details_view AS SELECT a.id AS order_id, a.transaction_time, a.total_price AS order_total_price, a.total_item AS order_total_item, b.name AS kasir_name, a.payment_method, d.name AS product_name, d.description AS product_description, d.price AS product_price, c.quantity, c.total_price AS item_total_price FROM orders a LEFT JOIN users b ON a.kasir_id = b.id LEFT JOIN order_items c ON a.id = c.order_id LEFT JOIN products d ON c.product_id = d.id;	Membuat view yang isinya menampilkan data detail order
11	CREATE VIEW		CREATE VIEW product_sales_view AS SELECT p.id AS product_id, p.name AS product_name, SUM(oi.quantity) AS total_quantity_sold, SUM(oi.total_price) AS total_revenue FROM products p LEFT JOIN order_items oi ON p.id = oi.product_id GROUP BY p.id, p.name;	Membuat view yang isinya menampilkan data sales berdasarkan product

12	CREATE VIEW		CREATE VIEW daily_sales_summary_view AS SELECT DATE(a.transaction_time) AS transaction_date, COUNT(a.id) AS total_orders, SUM(a.total_price) AS total_sales_amount, SUM(a.total_item) AS total_items_sold FROM orders a GROUP BY DATE(a.transaction_time);	
13	CREATE VIEW		CREATE VIEW user_orders_view AS SELECT u.id AS user_id, u.name AS user_name, COUNT(o.id) AS total_orders FROM users u LEFT JOIN orders o ON u.id = o.kasir_id GROUP BY u.id, u.name;	
14	CREATE VIEW		CREATE VIEW best_seller_products_view AS SELECT p.id AS product_id, p.name AS product_name, SUM(oi.quantity) AS total_quantity_sold FROM products p LEFT JOIN order_items oi ON p.id = oi.product_id GROUP BY p.id, p.name ORDER BY SUM(oi.quantity) DESC;	

15	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spProductList`(
			IN search_name VARCHAR(255),
			IN search_category ENUM('food', 'drink', 'snack'),
			IN page INT,
			IN page_size INT
			)
			BEGIN
			DECLARE offset_val INT DEFAULT 0;
			SET offset_val = (page - 1) * page_size;
			
			SELECT *
			FROM products
			WHERE (search_name IS NULL OR name LIKE CONCAT('%', search_name, '%'))
			AND (search_category IS NULL OR search_category = '' OR category = search_category)
			ORDER BY id
			LIMIT page_size OFFSET offset_val;
			END;	

16	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spProductDetail`(
			  IN product_id BIGINT UNSIGNED
			)
			BEGIN
			  SELECT *
			  FROM products
			  WHERE id = product_id;
			END	

17	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spProductAdd`(
			  IN p_name VARCHAR(255),
			  IN p_description TEXT,
			  IN p_price INT,
			  IN p_stock INT,
			  IN p_category ENUM('food', 'drink', 'snack'),
			  IN p_image VARCHAR(255)
			)
			BEGIN
			  DECLARE result_code INT;
			  DECLARE error_message VARCHAR(1000);
			  -- Check if name or category is empty
			  IF p_name = '' OR p_name IS NULL OR p_category = '' OR p_category IS NULL THEN
			  SET result_code = 39999;
			  SET error_message = 'Semua kolom (nama, kategori) harus diisi.';
			  SELECT result_code, error_message;
			  ELSE
			  -- Check if product name already exists
			  IF EXISTS (SELECT 1 FROM products WHERE name = p_name LIMIT 1) THEN
			  SET result_code = 39998;
			  SET error_message = 'Nama produk sudah ada dalam database.';
			  SELECT result_code, error_message;
			  ELSE
			  -- Insert new product
			  INSERT INTO products (name, description, price, stock, category, image, created_at, updated_at)
			  VALUES (p_name, p_description, p_price, p_stock, p_category, p_image, NOW(), NOW());
			  SET result_code = 1;
			  SET error_message = '';
			  SELECT result_code, error_message;
			  END IF;
			  END IF;
			END	

18	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spProductEdit`(
			IN p_id BIGINT UNSIGNED,
			IN p_name VARCHAR(255),
			IN p_description TEXT,
			IN p_price INT,
			IN p_stock INT,
			IN p_category ENUM('food', 'drink', 'snack'),
			IN p_image VARCHAR(255)
			)
			begin
			DECLARE result_code INT;
			DECLARE error_message VARCHAR(1000);
			DECLARE product_count INT;
			-- Check if the product exists
			SELECT COUNT(*) INTO product_count
			FROM products
			WHERE id = p_id;
			IF product_count = 0 THEN
			SET result_code = 39999;
			SET error_message = 'Produk dengan ID yang diberikan tidak ditemukan.';
			SELECT result_code, error_message;
			ELSE
			IF p_name = '' OR p_name IS NULL OR p_category = '' OR p_category IS NULL THEN
			SET result_code = 39999;
			SET error_message = 'Semua kolom (nama, kategori) harus diisi.';
			SELECT result_code, error_message;
			ELSE
			UPDATE products
			SET name = p_name,
			description = p_description,
			price = p_price,
			stock = p_stock,
			category = p_category,
			image = p_image,
			updated_at = NOW()
			WHERE id = p_id;
			
			SET result_code = 1;
			SET error_message = '';
			SELECT result_code, error_message;
			END IF;
			END IF;
			END;	

19	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spProductStockAdd`(
			IN p_product_id BIGINT UNSIGNED,
			IN p_add_stock INT
			)
			BEGIN
			DECLARE result_code INT;
			DECLARE error_message VARCHAR(1000);
			DECLARE current_stock INT;
			-- Check if the product exists
			IF NOT EXISTS (SELECT 1 FROM products WHERE id = p_product_id) THEN
			SET result_code = 0;
			SET error_message = 'Product ID does not exist';
			ELSE
			-- Get the current stock
			SELECT stock INTO current_stock FROM products WHERE id = p_product_id;
			-- Add the new stock
			UPDATE products SET stock = current_stock + p_add_stock, updated_at = NOW() WHERE id = p_product_id;
			SET result_code = 1;
			SET error_message = '';
			END IF;
			SELECT result_code, error_message;
			END	

20	RUN PROCEDURE		CALL spProductList('', null, 1, 10); CALL spProductList('', 'food', 1, 10); CALL spProductDetail(1); CALL spProductAdd(null, 'Deskripsi Produk', 10000, 50, 'food', 'gambar.jpg'); CALL spProductAdd('', 'Deskripsi Produk', 10000, 50, 'food', 'gambar.jpg'); CALL spProductAdd('Nama Produk2', 'Deskripsi Produk', 10000, 50, 'food', 'gambar.jpg'); CALL spProductEdit(12, '', 'Nama Produk', 12000, 60, 'drink', 'gambar_baru.jpg'); CALL spProductEdit(11, null, 'Deskripsi Produk Baru', 12000, 60, 'drink', 'gambar_baru.jpg'); CALL spProductEdit(11, '', 'Deskripsi Produk Baru', 12000, 60, 'drink', 'gambar_baru.jpg'); CALL spProductEdit(14, 'Nama Produk', 'Nama Produk', 12000, 60, 'drink', 'gambar_baru.jpg'); CALL spProductDelete(1); CALL spProductDelete(13);CALL spProductStockAdd(1, 10); CALL spProductStockAdd(1, 10);	

21	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spOrderAdd`(IN p_transaction_time TIMESTAMP, IN p_total_price INT, IN p_total_item INT, IN p_kasir_id BIGINT UNSIGNED, IN p_payment_method ENUM('Tunai', 'QRIS')) begin DECLARE result_code INT; DECLARE error_message VARCHAR(1000); INSERT INTO orders (transaction_time, total_price, total_item, kasir_id, payment_method, created_at, updated_at) VALUES (p_transaction_time, p_total_price, p_total_item, p_kasir_id, p_payment_method, NOW(), NOW()); SET result_code = 1; SET error_message = ''; SELECT result_code, error_message; END	

22	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spOrderItemAdd`(
			IN p_order_id BIGINT UNSIGNED,
			IN p_product_id BIGINT UNSIGNED,
			IN p_quantity INT,
			IN p_total_price INT
			)
			BEGIN
			DECLARE result_code INT;
			DECLARE error_message VARCHAR(1000);
			DECLARE product_stock INT;
			-- Check if the order exists
			IF NOT EXISTS (SELECT 1 FROM orders WHERE id = p_order_id) THEN
			SET result_code = 0;
			SET error_message = 'Order ID does not exist';
			
			-- Check if the product exists
			ELSEIF NOT EXISTS (SELECT 1 FROM products WHERE id = p_product_id) THEN
			SET result_code = 0;
			SET error_message = 'Product ID does not exist';
			
			-- Check if the stock is sufficient
			ELSE
			SELECT stock INTO product_stock FROM products WHERE id = p_product_id;
			IF product_stock < p_quantity THEN
			SET result_code = 0;
			SET error_message = 'Insufficient stock';
			ELSE	

			-- Insert the order item INSERT INTO order_items (order_id, product_id, quantity, total_price, created_at, updated_at) VALUES (p_order_id, p_product_id, p_quantity, p_total_price, NOW(), NOW()); -- Update the product stock UPDATE products SET stock = stock - p_quantity WHERE id = p_product_id; SET result_code = 1; SET error_message = ''; END IF; END IF; SELECT result_code, error_message; END	
23			CALL spOrderAdd('2024-06-24 14:00:00', 50000, 3, 1, 'Tunai'); CALL spOrderItemAdd(9,1, 1, 10000); CALL spProductStockAdd(1, 10);	

24	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spRegister`(
			IN p_name VARCHAR(255),
			IN p_email VARCHAR(50),
			IN p_phone VARCHAR(25),
			IN p_roles ENUM('admin','staff','user'),
			IN p_password VARCHAR(255)
			)
			BEGIN
			DECLARE result_code INT;
			DECLARE error_message VARCHAR(1000);
			-- Check if the email already exists
			IF EXISTS (SELECT 1 FROM users WHERE email = p_email) THEN
			SET result_code = 0;
			SET error_message = 'Email already exists';
			ELSE
			-- Insert the new user with hashed password
			INSERT INTO users (name, email, phone, roles, password, created_at, updated_at)
			VALUES (p_name, p_email, p_phone, p_roles, SHA2(p_password, 256), NOW(), NOW());
			SET result_code = 1;
			SET error_message = '';
			END IF;
			SELECT result_code, error_message;
			END	

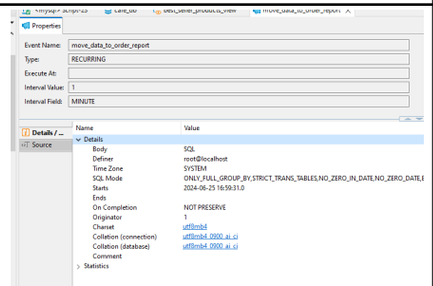
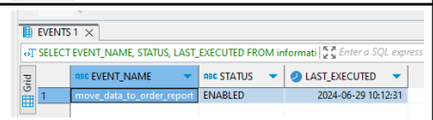
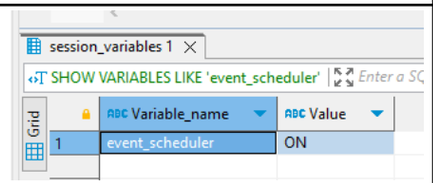
25	CREATE PROCEDURE		CREATE PROCEDURE `cafe_db`.`spLogin`(
			IN p_email VARCHAR(50),
			IN p_password VARCHAR(255)
			)
			BEGIN
			DECLARE result_code INT;
			DECLARE error_message VARCHAR(1000);
			DECLARE user_id BIGINT UNSIGNED;
			DECLARE user_name VARCHAR(255);
			DECLARE user_roles ENUM('admin','staff','user');
			-- Check if the email and hashed password match
			IF EXISTS (SELECT 1 FROM users WHERE email = p_email AND password = SHA2(p_password, 256)) THEN
			-- Get user details
			SELECT id, name, roles INTO user_id, user_name, user_roles FROM users WHERE email = p_email AND password = SHA2(p_password, 256);
			
			SET result_code = 1;
			SET error_message = '';
			
			SELECT result_code, error_message;
			SELECT user_id, user_name, user_roles;
			ELSE
			SET result_code = 0;
			SET error_message = 'Invalid email or password';
			SELECT result_code, error_message;
			END IF;
			
			END	

26	RUN PROCEDURE		select * from users; CALL spRegister('John Doe', 'johndoe@example.com', '123456789', 'user', 'budi123'); CALL spLogin('johndoe@example.com', 'budi123');	
27	CREATE TRIGGER		CREATE TRIGGER trg_after_insert_order_items AFTER INSERT ON order_items FOR EACH ROW BEGIN CALL spOrderTotalUpdate(NEW.order_id); END	

28	CREATE TRIGGER		CREATE TRIGGER trg_after_update_order_items AFTER UPDATE ON order_items FOR EACH ROW BEGIN CALL spOrderTotalUpdate(NEW.order_id); END	
29	CREATE TRIGGER		CREATE TRIGGER trg_after_delete_order_items AFTER DELETE ON order_items FOR EACH ROW BEGIN CALL spOrderTotalUpdate(OLD.order_id); END	

30	CREATE TABLE		CREATE TABLE order_report (
			   order_id BIGINT UNSIGNED NOT NULL,
			   transaction_time TIMESTAMP NOT NULL,
			   order_total_price INT NOT NULL,
			   order_total_item INT NOT NULL,
			   kasir_name VARCHAR(255) NOT NULL,
			   payment_method ENUM('Tunai', 'QRIS') NOT NULL,
			   product_name VARCHAR(255) NOT NULL,
			   product_description TEXT,
			   product_price INT NOT NULL,
			   quantity INT NOT NULL,
			   item_total_price INT NOT NULL,
			   INDEX idx_transaction_time (transaction_time),
			   INDEX idx_kasir_name (kasir_name),
			   INDEX idx_payment_method (payment_method),
			   INDEX idx_product_name (product_name)
			);	

31	CREATE PROCEDURE		CREATE PROCEDURE`cafe_db`.`spOrderReportAdd`()
			BEGIN
			     -- Insert data into order_report if it doesn't already exist
			   INSERT INTO order_report (
			       order_id,
			       transaction_time,
			       order_total_price,
			       order_total_item,
			       kasir_name,
			       payment_method,
			       product_name,
			       product_description,
			       product_price,
			       quantity,
			       item_total_price
			   )	

			SELECT v.order_id, v.transaction_time, v.order_total_price, v.order_total_item, v.kasir_name, v.payment_method, v.product_name, v.product_description, v.product_price, v.quantity, v.item_total_price FROM cafe_db.order_details_view v WHERE NOT EXISTS (SELECT 1 FROM order_report r WHERE r.order_id = v.order_id AND r.product_name = v.product_name AND r.quantity = v.quantity); END	
32	CREATE EVENT		CREATE EVENT IF NOT EXISTS move_data_to_order_report ON SCHEDULE EVERY 1 MINUTE STARTS CURRENT_TIMESTAMP ON COMPLETION NOT PRESERVE ENABLE DO CALL spOrderReportAdd();	
33	SELECT		SELECT EVENT_NAME, STATUS, LAST_EXECUTED FROM information_schema.EVENTS WHERE EVENT_SCHEMA = 'cafe_db' AND EVENT_NAME = 'move_data_to_order_report';	
34	SET GLOBAL		SET GLOBAL event_scheduler = ON; SHOW VARIABLES LIKE 'event_scheduler'; SELECT NOW();	

.oO Semoga Sukses Oo.