DATABASE DESIGN AND DATABASE PROGRAMMIN PROJECT WORK REPORT

Recording Business Transactions of Breakfast Food Stalls

Amna Ariria, Fahtahul Fahmi, Lisana Sidka Alia, Rahmad Ramadhan Laska
Group 2 Database Design and Database Programming with SQL Training
Fresh Graduate Academy Digitalent with Riau University

A. PENDAHULUAN

Kuliner saat ini menjadi bisnis yang berkembang sangat pesat di Indonesia. Bisnis yang dilaksanakan dapat berskala besar maupun kecil. Banyak tempat makan yang telah dibuka dengan menyajikan berbagai macam menu dengan makanan unggulan sebagai ciri khas sebuah tempat makan. Pemanfaatan teknologi informasi dalam bidang kuliner menjadi salah satu strategi untuk meningkatkan efisiensi dan efektifitas transaksi bisnis seperti pemesanan, penjualan, dan pelaporan.

Pengelolaan sistem teknologi informasi tersebut yang dapat mengelola seluruh informasi bisnis membutuhkan sistem yang terintegrasi untuk dapat menyimpan dan menampilkan pelayanan dari tempat makan. Sistem basis data atau database merupakan salah satu improvisasi dan sangat berguna untuk mengatur informasi bisnis yang berlaku.

Database dapat menyimpan informasi pelanggan, pemesanan, produk, pembayaran, dan dapat dikumpulkan dalam sebuah laporan online (report). Penambahan database yang mudah diakses dan mudah diperbaharui dapat mempersonalisasi laporan penjualan dengan mengakses catatan transaksi yang lalu. Seluruh informasi dapat diakses oleh penjual dalam menyiapkan pesanan dan pelanggan dalam mengakses menu pemesanan maupun pembayaran.

B. BUSINESS AND MISSION

Catatan bisnis yang akan dianalisa scenario pelaksanaan yang menjadi bahan rujukan pembangunan desain database adalah industri kuliner berskala kecil yaitu warung makan keluarga. Misi yang dicapai pada akhir pengerjaan proyek adalah menyediakan layanan yang dipersonalisasi untuk merekap catatan transaksi bisnis oleh klien.

C. BUSINESS REQUIREMENT (PROJECT SCENARIO)

Berperan sebagai team konsultasi database yang mengkhususkan diri dalam mengembangkan database untuk industri makanan dan produksi. Team baru saja mendapatkan kontrak untuk mengembangkan model data untuk sistem aplikasi database kepada seorang klien penjual makanan sarapan pagi untuk menyimpan seluruh transaksi bisnis yan berlaku dari tahap pemesanan hingga pembayaran. Sistem dapat menyimpan data informasi produk yang dijual oleh klien, informasi pelanggan seperti nama, Alamat, dan no. telpon, informasi pemesanan berupa produk yang dipesan, kuantiti pemesanan, dan jenis pengiriman, serta informasi pembayaran seperti metode yang digunakan dan total pembayaran. Pada akhirnya, basis data (desain database) ini akan digunakan untuk melacak catatan pelaksanaan penjualan oleh klien berdasarkan data yang terkumpul.

D. HASIL KESIMPULAN ATURAN SKENARIO BISNIS

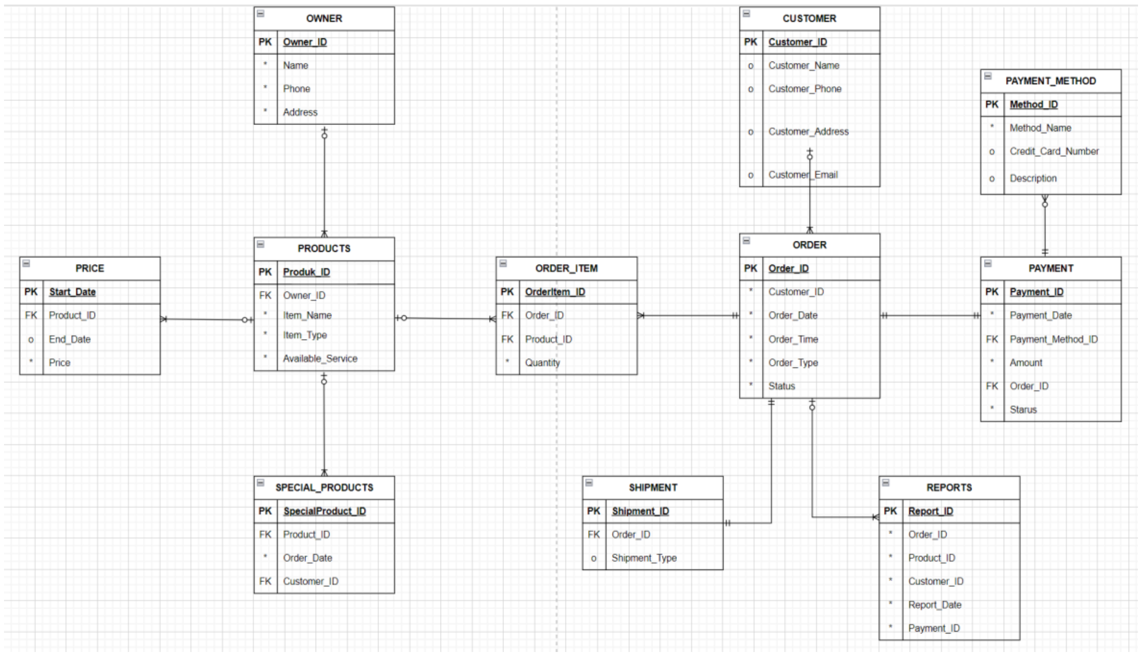
Subjek Interview Proyek Kelompok	: Warung Nasi Uduk Asyifa Sarapan Pagi.
Narasumber	: Siti Makmuria.
Lokasi Usaha	: Jalan Mahang 3, Pandau Jaya, Kec. Siak Hulu, Kab. Kampar, Riau.

Pelaksanaan Interview : Minggu, 11 Agustus 2024, Pukul 13.00 WIB s.d. Selesai.

Kesimpulan : Pemilik merupakan pemegan utama scenario bisnis sebagai pihak produksi dan penjual. Penjual menjual makanan yang dibagi berdasarkan jenis pemesanan oleh pelanggan. Pemesanan dibagi menjadi dua yaitu pemesanan menu regular dan pemesanan menu khusus. Menu regular disediakan 35 porsi setiap harinya. Pesanan menu khusus tidak dapat dilakukan lebih dari satu kali dihari yang sama. Pemesanan dapat dilakukan secara online maupun secara langsung. Pemesanan dapat dilakukan berdasarkan jadwal waktu penjualan. Pesanan dikirim dengan jasa kurir online atau melalui jasa kurir restoran. Pembayaran dilakukan setelah pesanan selesai dikirim dan dapat dilakukan secara tunai, transfer bank, dan e-wallet.

E. RELATIONAL MODEL DESIGN

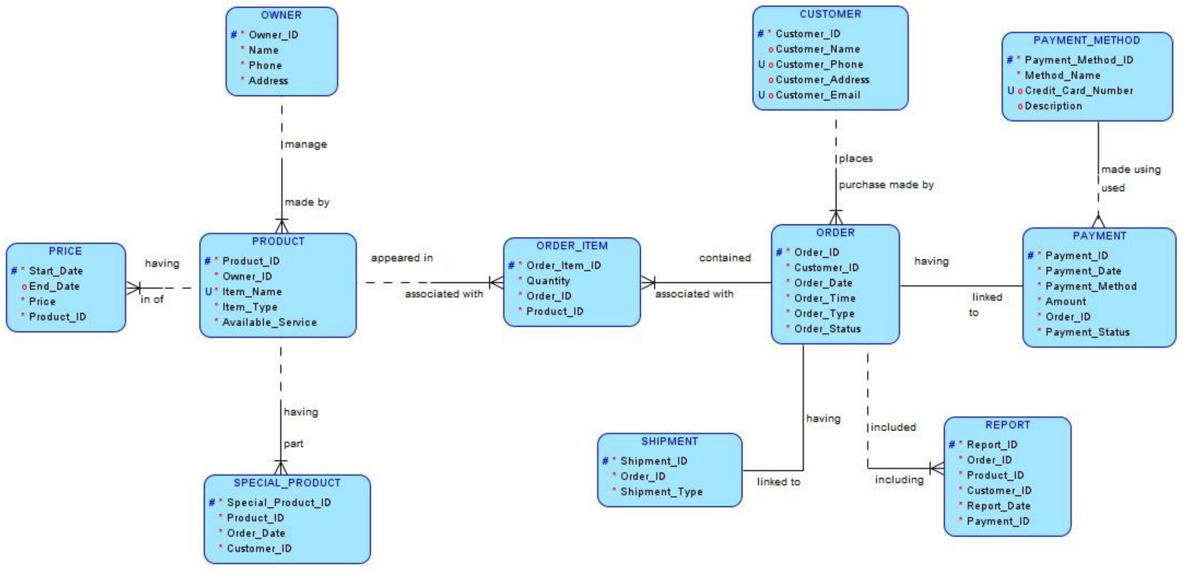
Hasil relational model diakses melalui penggambaran flowchart yang menjelaskan proses normalization dalam pembangunan ERD. Berikut hasil sampel mapping to relational dalam scenario bisnis proyek:



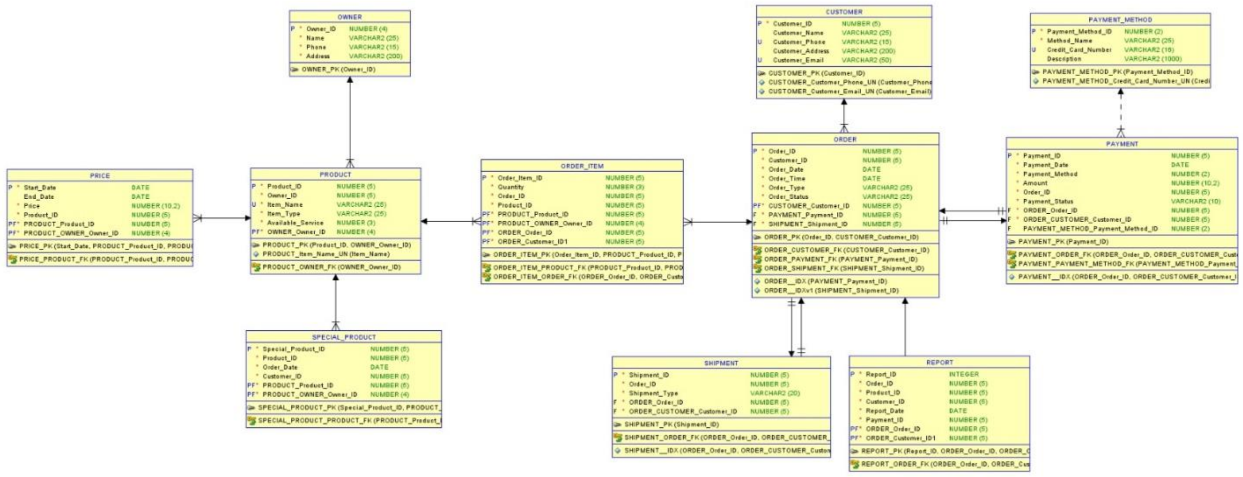
Gambar 1. Hasil Sampel Mapping to Relational Business Scenario.

F. SAMPLE TABLE

Sample table dibangun dalam SQL Modeller Version 23 dengan 2 jenis hasil sampel, yaitu Logical Model dan Relational Model. Berikut hasil Logical Model dan Relational Model dari aturan dan scenario model yang dibangun.



Gambar 2. Logical Model



Gambar 3. Relational Model Table

G. ASSUMPTION DAN RECOMMENDATION

Assumption dan Recommendation merupakan poin penjelasan lebih lanjut untuk pengembangan dari sebuah desain database yang akan dilanjutkan oleh klien. Berikut hasil Assumption dan Recommendation dari Business Requirement dalam pengerjaan proyek, yaitu:

1. Bisnis diekspansi hingga penerimaan staff sehingga memerlukan database dalam penanganan management employee yang dapat mentracking posisi, jadwal, dan gaji.
2. Promosi dan diskon akan ditambahkan dengan menginteragsi data pada order.
3. Klien ingin menerapkan sistem feedback yang dikumpulkan melakui review pelanggan terhadap menu atau produk yang disajikan.

Dan klien dapat mengembangkan sistem loyalty untuk pelanggan setia dengan sistem poin atau hadiah.

H. CREATE TABLE DAN ADD CONSTRAINT

Create Table dan Add Constraint merupakan hasil dari penyusunan script sql berdasarkan nilai Table Instance Chart pada database design. Hasil Create Table dan Add Constraint sebagai berikut,

Tabel 1. Script Create Table dan Add Constraint

TABLE	CREATE TABLE	ADD CONSTRAINT
OWNERS	**/* Tabel Owners */** CREATE TABLE OWNERS (Owner_ID NUMBER(4), Name VARCHAR2(25), Phone VARCHAR2(15), Address VARCHAR2(200));	/* Constraint OWNERS */ ALTER TABLE OWNERS ADD CONSTRAINT owners_ow_id_pk PRIMARY KEY (Owner_ID); ALTER TABLE OWNERS MODIFY(Name CONSTRAINT owners_nm_nn NOT NULL, Phone CONSTRAINT owners_phn_nn NOT NULL, Address CONSTRAINT owners_addrss_nn NOT NULL);
CUSTOMERS	**/* Tabel Customers */** CREATE TABLE CUSTOMERS (Customer_ID NUMBER(5), Customer_Name VARCHAR2(25), Customer_Phone VARCHAR2(15), Customer_Address VARCHAR2(200), Customer_Email VARCHAR2(50));	/* Constraint CUSTOMERS */ ALTER TABLE CUSTOMERS ADD CONSTRAINT customers_cstmr_id_pk PRIMARY KEY (Customer_ID); ALTER TABLE CUSTOMERS ADD CONSTRAINT cstmr_cstmr_phn_uk UNIQUE (Customer_Phone); ALTER TABLE CUSTOMERS ADD CONSTRAINT cstmr_cstmr_ml_uk UNIQUE (Customer_Email);
PRODUCTS	**/* Tabel Products */** CREATE TABLE PRODUCTS (Product_ID NUMBER(5), Owner_ID NUMBER(5), Item_Name VARCHAR2(25), Item_Type VARCHAR2(25), Available_Service NUMBER(3));	ALTER TABLE PRODUCTS ADD CONSTRAINT pr_product_id_pk PRIMARY KEY (Product_ID); ALTER TABLE PRODUCTS ADD CONSTRAINT pr_ow_id_fk FOREIGN KEY(Owner_ID) REFERENCES OWNERS(Owner_ID) ON DELETE CASCADE; ALTER TABLE PRODUCTS MODIFY (owner_id CONSTRAINT pr_owner_id_nn NOT NULL, item_name CONSTRAINT pr_item_name_nn NOT NULL, item_type CONSTRAINT pr_item_typee_nn NOT NULL, available_service CONSTRAINT pr_avb_service_nn NOT NULL); ALTER TABLE PRODUCTS ADD CONSTRAINT item_type_chk CHECK (item_type IN ('Regular', 'Special')); ALTER TABLE PRODUCTS ADD CONSTRAINT pr_itm_nm_uk UNIQUE (Item_Name);

PRICES	**/* Tabel Prices */** CREATE TABLE PRICES (Start_Date DATE DEFAULT SYSDATE, End_Date DATE, Price NUMBER(10,2), Product_ID NUMBER(5));	ALTER TABLE PRICES ADD CONSTRAINT price_start_date_pk PRIMARY KEY (Start_Date); ALTER TABLE PRICES ADD CONSTRAINT price_product_id_fk FOREIGN KEY(Product_ID) REFERENCES PRODUCTS(Product_ID); ALTER TABLE PRICES MODIFY (price CONSTRAINT price_price_nn NOT NULL, product_id CONSTRAINT price_product_id_nn NOT NULL); ALTER TABLE PRICES ADD CONSTRAINT end_date_chk CHECK (End_Date > Start_Date);
ORDERS	**/* Tabel Order */** CREATE TABLE ORDERS (Order_ID NUMBER(5), Customer_ID NUMBER(5), Order_Date TIMESTAMP DEFAULT SYSTIMESTAMP, Order_Type VARCHAR2(25), Order_Status VARCHAR2(25));	ALTER TABLE orders ADD CONSTRAINT orders_id_pk PRIMARY KEY (order_id); ALTER TABLE orders ADD CONSTRAINT orders_cust_id_fk FOREIGN KEY (customer_id) REFERENCES customers (customer_id) ON DELETE CASCADE; ALTER TABLE orders MODIFY (customer_id CONSTRAINT orders_cust_id_nn NOT NULL, order_date CONSTRAINT orders_date_nn NOT NULL, order_type CONSTRAINT orders_type_nn NOT NULL, order_status CONSTRAINT orders_status_nn NOT NULL); ALTER TABLE orders ADD CONSTRAINT orders_type_chk CHECK (order_type IN ('Online', 'In-Person')); ALTER TABLE orders ADD CONSTRAINT orders_status_chk CHECK (order_status IN ('Delivered', 'Accepted', 'Cancelled'));
PAYMENT _METHODS	**/* Tabel Payment */** CREATE TABLE PAYMENTS (payment_id NUMBER(5), payment_date DATE DEFAULT SYSDATE, payment_method_id NUMBER(2), amount NUMBER(10,2), order_id NUMBER(5), payment_status VARCHAR2(10));	/* Constraint PAYMENT */ ALTER TABLE payments ADD CONSTRAINT payments_id_pk PRIMARY KEY (payment_id); ALTER TABLE payments ADD CONSTRAINT payments_method_id_fk FOREIGN KEY (payment_method_id) REFERENCES payment_methods (payment_method_id) ON DELETE CASCADE; ALTER TABLE payments

		ADD CONSTRAINT payments_order_id_fk FOREIGN KEY (order_id) REFERENCES orders (order_id) ON DELETE CASCADE; ALTER TABLE payments MODIFY (payment_date CONSTRAINT payments_date_nn NOT NULL, payment_method_id CONSTRAINT payments_method_id_nn NOT NULL, amount CONSTRAINT payments_amount_nn NOT NULL, order_id CONSTRAINT payments_order_id_nn NOT NULL, payment_status CONSTRAINT payments_status_nn NOT NULL); ALTER TABLE payments ADD CONSTRAINT payments_status_chk CHECK (payment_status IN ('Completed', 'Failed', 'Pending'));
PAYMENTS	**/* Tabel Payment_Method */** CREATE TABLE PAYMENT_METHODS (payment_method_id NUMBER(2), method_name VARCHAR2(25), credit_card_number VARCHAR2(16), description VARCHAR2(1000));	/* Constraint PAYMENT_METHODS */ ALTER TABLE payment_methods ADD CONSTRAINT pay_methods_id_pk PRIMARY KEY (payment_method_id); ALTER TABLE payment_methods MODIFY (method_name CONSTRAINT pay_method_name_nn NOT NULL); ALTER TABLE payment_methods ADD CONSTRAINT pay_methods_ccn_uk UNIQUE (credit_card_number);
ORDER_ITEMS	**/* Tabel Order_Item */** CREATE TABLE ORDER_ITEMS (Order_Item_ID NUMBER(5), Order_ID NUMBER(5), Product_ID NUMBER(5), Quantity NUMBER(3));	ALTER TABLE ORDER_ITEMS ADD CONSTRAINT order_item_id_pk PRIMARY KEY (Order_Item_ID); ALTER TABLE ORDER_ITEMS ADD CONSTRAINT order_id_fk FOREIGN KEY(Order_ID) REFERENCES ORDERS(Order_ID) ON DELETE CASCADE; ALTER TABLE ORDER_ITEMS MODIFY (order_id CONSTRAINT oi_order_id_nn NOT NULL, product_id CONSTRAINT oi_product_id_nn NOT NULL, quantity CONSTRAINT oi_quantity_nn NOT NULL);
SPECIAL _PRODUCTS	**/* Tabel Special_Product */** CREATE TABLE SPECIAL_PRODUCTS (Special_Product_ID NUMBER(5), Product_ID NUMBER(5),	ALTER TABLE SPECIAL_PRODUCTS ADD CONSTRAINT special_products_pk PRIMARY KEY (Special_Product_ID); ALTER TABLE SPECIAL_PRODUCTS

	Order_Date DATE DEFAULT SYSDATE, Customer_ID NUMBER(5));	ADD CONSTRAINT sp_product_id_fk FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID) ON DELETE CASCADE ; ALTER TABLE SPECIAL_PRODUCTS ADD CONSTRAINT sp_customer_id_fk FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID) ON DELETE CASCADE; ALTER TABLE SPECIAL_PRODUCTS MODIFY (Product_ID CONSTRAINT sp_pi_nn NOT NULL, Order_Date CONSTRAINT sp_oi_nn NOT NULL, Customer_id CONSTRAINT sp_ci_nn NOT NULL);
SHIPMENTS	**/* Tabel Shipment */** CREATE TABLE SHIPMENTS (Shipment_ID NUMBER(2), Order_ID NUMBER(5), Shipment_Type VARCHAR2(20));	ALTER TABLE SHIPMENTS ADD CONSTRAINT shipment_id_pk PRIMARY KEY (Shipment_ID); ALTER TABLE SHIPMENTS ADD CONSTRAINT sp_order_id_fk FOREIGN KEY (Order_ID) REFERENCES ORDERS (Order_ID) ON DELETE CASCADE; ALTER TABLE SHIPMENTS ADD CONSTRAINT chk_shipment_type CHECK (Shipment_Type IN ('Courier', 'In-House')); ALTER TABLE SHIPMENTS MODIFY (Order_ID CONSTRAINT s_oi_nn NOT NULL, Shipment_Type CONSTRAINT s_st_nn NOT NULL);
REPORTS	**/* Tabel Report */** CREATE TABLE REPORTS (Report_ID NUMBER(5), Order_ID NUMBER(5), Product_ID NUMBER(5), Customer_ID NUMBER(5), Report_Date DATE DEFAULT SYSDATE, Payment_ID NUMBER(5));	ALTER TABLE REPORTS ADD CONSTRAINT reports_id_pk PRIMARY KEY (Report_ID); ALTER TABLE REPORTS ADD CONSTRAINT rp_order_id_fk FOREIGN KEY (Order_ID) REFERENCES ORDERS(Order_ID) ON DELETE CASCADE; ALTER TABLE REPORTS ADD CONSTRAINT rp_product_id_fk FOREIGN KEY (Product_ID) REFERENCES PRODUCTS (Product_ID) ON DELETE CASCADE; ALTER TABLE REPORTS ADD CONSTRAINT rp_customer_id_fk FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (Customer_ID) ON DELETE CASCADE; ALTER TABLE REPORTS ADD CONSTRAINT rp_payment_id_fk FOREIGN KEY (Payment_id) REFERENCES PAYMENTS(Payment_id) ON DELETE CASCADE; ALTER TABLE REPORTS

		MODIFY (Order_ID CONSTRAINT rp_oi_nn NOT NULL, Product_ID CONSTRAINT rp_pi_nn NOT NULL, Customer_ID CONSTRAINT rp_ci_nn NOT NULL, Report_Date CONSTRAINT rp_rp_nn NOT NULL, Payment_id CONSTRAINT rp_pyi_nn NOT NULL);

I. CREATE VIEW

Create view digunakan untuk mempermudah penampilan data yang akan diinformasikan dan mempermudahkan dengan aksesbilitas dengan tabel yang lain. Hasil Create View sebagai berikut.

Tabel 2. Create View

View Name	Create View	TARGET

View_Menus	CREATE OR REPLACE VIEW view_menus AS
	SELECT
	    P.Product_ID AS "ID",
	    P.Item_Name AS "Product",
	    PR.Price AS "Price",
	    P.Available_Service AS "Available Service"
	FROM
	    PRODUCTS P
	JOIN
	    PRICES PR ON P.Product_ID = PR.Product_ID;
		Public for user with role customer to read the information about the menu of the stall sells.

View_Order	CREATE OR REPLACE VIEW view_orders AS
	SELECT
	    O.Order_ID AS "ID",
	    O.Order_Type AS "Type",
	    TO_CHAR(O.Order_Date, 'YYYY-MM-DD') AS "Order Date",
	    O.Customer_ID AS "Customer ID",
	    P.Item_Name AS "Item Name",
	    PR.Price AS "Price",
	    OI.Quantity AS "Quantity",
	    PAY.amount AS "Total"
	FROM
	    ORDERS O
	JOIN
	    ORDER_ITEMS OI ON O.Order_ID = OI.Order_ID
	JOIN
	    PRODUCTS P ON OI.Product_ID = P.Product_ID
	JOIN
	    PRICES PR ON P.Product_ID = PR.Product_ID
	JOIN
	    PAYMENTS PAY ON O.Order_ID = PAY.order_id;
		Public for user with role customer to read and insert an order and the information of all orders.

View_Payments	CREATE OR REPLACE VIEW view_payments AS
	SELECT
	    PAY.payment_id AS "Payment ID",
	    TO_CHAR(PAY.Payment_Date, 'YYYY-MM-DD') AS "Date",
	    PAY.order_id AS "Order ID",
	    O.Customer_id,
	    PM.method_name AS "Method",
	    PM.credit_card_number AS "Credit Card Number",
	    PAY.payment_status AS "Payment Status"
		Restricted view for the information of all payment that already query in database.

	FROM
	    PAYMENT_METHODS PM
	JOIN
	    PAYMENTS PAY ON PM.payment_method_id = PAY.payment_method_id
	JOIN
	    ORDERS O ON PAY.Order_id = O.Order_id
	JOIN
	    customers C ON O.Customer_ID = C.Customer_ID;	

View_Special_Orders	CREATE VIEW view_special_orders AS
	SELECT
	    SP.Special_Product_ID AS "ID",
	    OI.Order_ID AS "Order ID",
	    P.Item_Name AS "Item Name",
	    OI.Quantity AS "Quantity",
	TO_CHAR(SP.Order_Date, 'YYYY-MM-DD') AS "Date"
	FROM
	    SPECIAL_PRODUCTS SP
	JOIN
	    ORDER_ITEMS OI ON SP.Product_ID = OI.Product_ID
	JOIN
	    PRODUCTS P ON OI.Product_ID = P.Product_ID;
		Public for user with role customer to add some order in special condition.

View_Reports	CREATE OR REPLACE VIEW view_reports
	AS SELECT report_id, report_date, order_id, customer_id
	FROM reports;
	
	SELECT * FROM view_reports
	ORDER BY order_id;
		To get information about all the transaction and activity for the owner.

View_Sales_Per_Day	CREATE OR REPLACE VIEW view_sales_per_day AS
	SELECT
	    TO_CHAR(payment_date, 'YYYY-MM-DD') AS sale_date,
	    COUNT(payment_id) AS total_transactions,
	    SUM(amount) AS total_sales
	FROM
	    payments
	GROUP BY TO_CHAR(payment_date, 'YYYY-MM-DD');
	
	SELECT * FROM view_sales_per_day
	ORDER BY sale_date;
		To get information of sales per day buy the owner.

J. CREATE SEQUENCE

Create Sequence memiliki fungsi sebagai auto increment dalam sebuah data yang berurut yang disimpan dalam sequence yang dibangun. Hasil Create Sequence dari database yang dibangun yaitu:

Tabel 3. Create Sequence

SEQUENCE	SCRIPT

OWNER_ID	CREATE SEQUENCE owner_id_sq
	    INCREMENT BY 1
	    START WITH 1000
	    MAXVALUE 9999

	NOCYCLE NOCACHE;
CUSTOMER_ID	CREATE SEQUENCE customer_id_sq INCREMENT BY 1 START WITH 1 MAXVALUE 99999 NOCYCLE NOCACHE;
ORDER_ID	CREATE SEQUENCE order_id_sq INCREMENT BY 1 START WITH 10 MAXVALUE 99999 CYCLE NOCACHE;
ORDER_ITEM_ID	CREATE SEQUENCE order_item_id_sq INCREMENT BY 1 START WITH 100 MAXVALUE 99999 CYCLE NOCACHE;
PRODUCT_ID	CREATE SEQUENCE product_id_sq INCREMENT BY 1 START WITH 10 MAXVALUE 99 NOCYCLE NOCACHE;
PAYMENT_ID	CREATE SEQUENCE payment_id_sq INCREMENT BY 1 START WITH 1000 MAXVALUE 9999 CYCLE NOCACHE;
SPECIAL_PRODUCT_ID	CREATE SEQUENCE special_product_id_sq INCREMENT BY 1 START WITH 1 MAXVALUE 99999 NOCYCLE NOCACHE;
SHIPMENT_ID	CREATE SEQUENCE shipment_id_seq INCREMENT BY 1 START WITH 1 MAXVALUE 99 NOCYCLE NOCACHE;
REPORT_ID	CREATE SEQUENCE report_id_seq INCREMENT BY 1 START WITH 10000 MAXVALUE 99999 NOCACHE NOCYCLE;

K. ADD DATA TO TABLE

Add Data to Table menggunakan clausa INSERT dan dapat dilakukan modifikasi, update, dan delete berdasarakan kebutuhan. Berikut rincian script yang disusun untuk inputan dalam database yang telah disusun.

Tabel 4. Add Data to Table

TABEL	INSERT

OWNER	INSERT INTO OWNERS (Owner_ID, Name, Phone, Address)
	VALUES (owner_id_sq.NEXTVAL, 'Siti Makmuria', '082187659870', 'Jalan Mahang 3, Pandau Jaya, Kec. Siak Hulu, Kab.Kampar, Riau');

CUSTOMERS	INSERT INTO CUSTOMERS (Customer_ID)
	VALUES (customer_id_sq.NEXTVAL);
	
	INSERT INTO CUSTOMERS (Customer_ID, Customer_Name, Customer_Phone, Customer_Address, Customer_Email)
	VALUES (customer_id_sq.NEXTVAL, 'Rahmad Ramadhan', '081312345678', 'Jl.Mahang', 'Rrama@gmail.com');
	
	INSERT INTO CUSTOMERS (Customer_ID, Customer_Name, Customer_Phone, Customer_Address, Customer_Email)
	VALUES (customer_id_sq.NEXTVAL, 'Lisana Sidka', '081323456781', 'Jl.Tebing', 'Lsidk@gmail.com');
	
	INSERT INTO CUSTOMERS (Customer_ID)
	VALUES (customer_id_sq.NEXTVAL);
	
	INSERT INTO CUSTOMERS (Customer_ID)
	VALUES (customer_id_sq.NEXTVAL);
	
	INSERT INTO CUSTOMERS (Customer_ID, Customer_Name, Customer_Phone, Customer_Address, Customer_Email)
	VALUES (customer_id_sq.NEXTVAL, 'Fattahul Fahmi', '081334567812', 'Jl.Ujung', 'Ffahm@gmail.com');
	
	INSERT INTO CUSTOMERS (Customer_ID, Customer_Name, Customer_Phone, Customer_Address, Customer_Email)
	VALUES (customer_id_sq.NEXTVAL, 'Amna Ariria', '081345678123', 'Jl.Siak', 'Aarir@gmail.com');

PRODUCTS	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Nasi Uduk', 'Special', 35);
	
	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Mie Ayam', 'Regular', 35);
	
	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Seblak', 'Regular', 35);
	
	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Lontong Sayur', 'Regular', 35);

	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Somai', 'Regular', 35);
	
	INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service)
	VALUES (product_id_sq.NEXTVAL, 1004, 'Cilok', 'Regular', 35);

PRICES	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 10000, 12);
	
	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 10000, 14);
	
	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 15000, 15);
	
	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 10000, 16);
	
	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 5000, 17);
	
	INSERT INTO PRICES (Start_Date, End_Date, Price, Product_ID)
	VALUES (SYSDATE, NULL, 5000, 18);

ORDERS	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 1, SYSTIMESTAMP, 'Online', 'Delivered');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 2, SYSTIMESTAMP, 'In-Person', 'Accepted');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 3, SYSTIMESTAMP, 'Online', 'Cancelled');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 4, SYSTIMESTAMP, 'In-Person', 'Delivered');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 5, SYSTIMESTAMP, 'Online', 'Accepted');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 6, SYSTIMESTAMP, 'In-Person', 'Cancelled');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 7, SYSTIMESTAMP, 'Online', 'Delivered');

	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 8, SYSTIMESTAMP, 'In-Person', 'Accepted');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 9, SYSTIMESTAMP, 'Online', 'Cancelled');
	
	INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)
	VALUES (order_id_sq.NEXTVAL, 10, SYSTIMESTAMP, 'In-Person', 'Delivered');

PAYMENTS	INSERT INTO PAYMENTS (payment_id, payment_method_id, order_id, payment_status) VALUES (payment_id_sq.NEXTVAL, 2, 12, 'Completed');
	
	INSERT INTO PAYMENTS (payment_id, payment_method_id, order_id, payment_status) VALUES (payment_id_sq.NEXTVAL, 1, 13, 'Completed');
	
	INSERT INTO PAYMENTS (payment_id, payment_method_id, order_id, payment_status) VALUES (payment_id_sq.NEXTVAL, 3, 14, 'Completed');

PAYMENT_METHODS	INSERT INTO PAYMENT_METHODS (payment_method_id, method_name, credit_card_number, description)
	VALUES (1, 'Cash', 'N/A', 'Pembayaran dilakukan secara tunai');
	
	INSERT INTO PAYMENT_METHODS (payment_method_id, method_name, credit_card_number, description)
	VALUES (2, 'Bank Transfer', '1234567890', 'Pembayaran dilakukan melalui transfer bank');
	
	INSERT INTO PAYMENT_METHODS (payment_method_id, method_name, credit_card_number, description)
	VALUES (3, 'E-Wallet', '4567890123', 'Pembayaran melalui E-Wallet seperti GoPay, OVO, dll.');

ORDER_ITEMS	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 12, 12, 23);
	
	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 12, 12, 23);
	
	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 13, 13, 15);
	
	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 14, 14, 8);
	
	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 15, 15, 8);
	
	INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity)
	VALUES (order_item_id_sq.NEXTVAL, 16, 16, 10);

SPECIAL_PRODUCTS	INSERT INTO SPECIAL_PRODUCTS (Special_Product_ID, Product_ID, Order_Date, Customer_ID)
	VALUES (special_product_id_sq.NEXTVAL, 12, SYSDATE, 1);
	
	INSERT INTO SPECIAL_PRODUCTS (Special_Product_ID, Product_ID, Order_Date, Customer_ID)
	VALUES (special_product_id_sq.NEXTVAL, 14, SYSDATE, 2);
	
	INSERT INTO SPECIAL_PRODUCTS (Special_Product_ID, Product_ID, Order_Date, Customer_ID)
	VALUES (special_product_id_sq.NEXTVAL, 15, SYSDATE, 3);
	
	INSERT INTO SPECIAL_PRODUCTS (Special_Product_ID, Product_ID, Order_Date, Customer_ID)
	VALUES (special_product_id_sq.NEXTVAL, 16, SYSDATE, 4);

SHIPMENTS	INSERT INTO SHIPMENTS (Shipment_ID, Order_ID, Shipment_Type)
	VALUES (shipment_id_seq.NEXTVAL, 12, 'In-House');
	
	INSERT INTO SHIPMENTS (Shipment_ID, Order_ID, Shipment_Type)
	VALUES (shipment_id_seq.NEXTVAL, 17, 'In-House');
	
	INSERT INTO SHIPMENTS (Shipment_ID, Order_ID, Shipment_Type)
	VALUES (shipment_id_seq.NEXTVAL, 19, 'In-House');

REPORTS	INSERT INTO REPORTS (Report_ID, Order_ID, Product_ID, Customer_ID, Report_Date, Payment_ID)
	VALUES (report_id_seq.NEXTVAL, 12, 12, 1, SYSDATE, 1001)

Noted: diperlukan TRIGGER untuk menampilkan nilai amount pada PAYMENTS dengan script sebagai berikut.

TRIGGER for Amount in Payment Table

```
create or replace TRIGGER trg_calculate_payment_amount
BEFORE INSERT ON PAYMENTS
 FOR EACH ROW
DECLARE
    v_price NUMBER(10,2);
    v_quantity NUMBER(3);
BEGIN
    -- Get the quantity from ORDER_ITEMS
    SELECT oi.Quantity
    INTO v_quantity
    FROM ORDER_ITEMS oi
    WHERE oi.Order_ID = :NEW.order_id;

    -- Get the price from PRICES
    SELECT p.Price
    INTO v_price
    FROM PRICES p
    JOIN ORDER_ITEMS oi ON p.Product_ID = oi.Product_ID
    WHERE oi.Order_ID = :NEW.order_id
    AND p.Start_Date <= SYSDATE
    AND (p.End_Date IS NULL OR p.End_Date >= SYSDATE)
    AND ROWNUM = 1; -- Assuming there's one price for the product

    -- Calculate and set the amount
    :NEW.amount := v_price * v_quantity;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        :NEW.amount := 0; -- Handle case where no price or quantity is found
   END;
```

L. CREATE INDEX

Tabel 5. Create Index

TABLE	INDEX	CREATE INDEX
ORDERS	customer_id & Orders	CREATE INDEX idx_orders_customer_id ON ORDERS (Customer_ID);
PRICES	Price & Product_ID	CREATE INDEX idx_prices_product_id ON PRICES (Product_ID);
PRODUCTS	PRODUCT_ID & Item_name	CREATE INDEX idx_pr_id_it_nm ON PRODUCTS (Product_ID, Item_Name);

ORDER_ITEMS	Order_ID & Order_Item
	
	Product_ID & Order_Items
		CREATE INDEX idx_order_items_order_id ON ORDER_ITEMS (Order_ID);
		
		CREATE INDEX idx_order_items_product_id ON ORDER_ITEMS (Product_ID);

PAYMENTS	Payment_ID & Method Name
	
	Order_ID & Payments
		CREATE INDEX idx_payments_method_id ON PAYMENTS (Payment_Method_ID);
		
		CREATE INDEX idx_payments_order_id ON PAYMENTS (Order_ID);

REPORTS	Report_date	CREATE INDEX idx_rp_date ON REPORTS(Report_Date);
SPECIAL_PRODUCTS	Product_ID & Special_Products	CREATE INDEX idx_sp_product_id ON SPECIAL_PRODUCTS (Product_ID);

M. CREATE SYNONYM

Tabel 6. Create Synonym

SYNONYM	SCRIPT

VIEW_MENUS	CREATE PUBLIC SYNONYM kel2_vm
	FOR View_Menus;

VIEW_ORDERS	CREATE PUBLIC SYNONYM kel2_vo
	FOR View_Orders;

VIEW_PAYMENTS	CREATE SYNONYM kel2_vp
	FOR View_Payments;

VIEW_REPORTS	CREATE SYNONYM kel_2vr
	FOR View_REPORTS;

VIEW_SALES_PER_DAY	CREATE SYNONYM kel2_vspd
	FOR View_Sales_Per_Day;

VIEW_SPECIAL_ORDERS	CREATE PUBLIC SYNONYM kel2_vso
	FOR View_Special_Orders;

Noted: Terdapat kondisi GRANT untuk dapat mengaktifkan atau mengakses CREATE SYNONYM dengan penjabaran sebagai berikut.

```
SQL> connect system/system@XE (username database)
SQL> GRANT CREATE PUBLIC SYNONYM TO KEL2(username apex);
```

N. TESTING DATABASE

Tabel 7. Testing Database

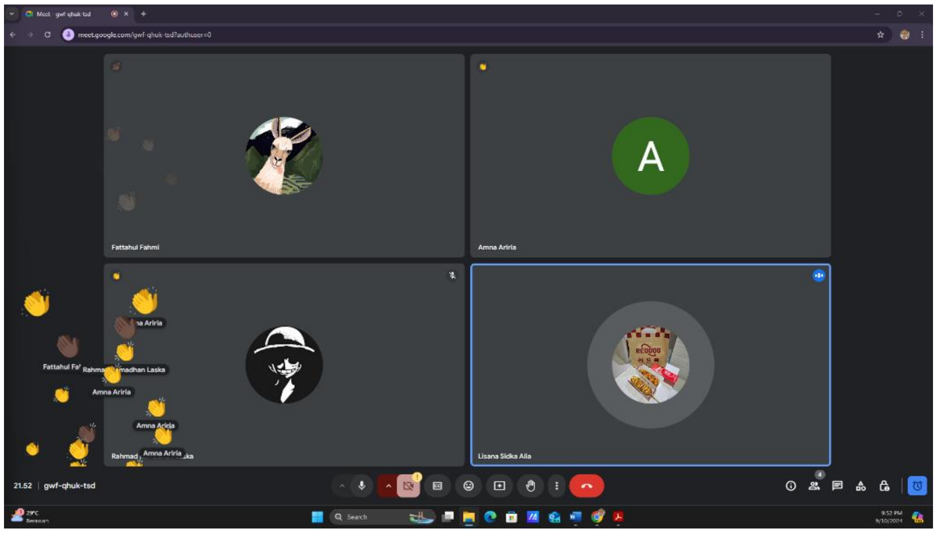
No.	Date	Test Description	Input	Expected Output	Result	Action
1.	13/09/2024	Confirmation Can enter NULL values in the Customer table except Customer_ID	`INSERT INTO CUSTOMERS (Customer_ID) VALUES (customer_id_sq.NEXTVAL);`	`Row_Inserted`	`1 row(s) inserted.`	`None`
2.	13/09/2024	Confirm cannot enter identical values for Item_Name, CONSTRAINT UNIQUE (Item_Name)in Products table.	`INSERT INTO PRODUCTS (Product_ID, Owner_ID, Item_Name, Item_Type, Available_Service) VALUES (product_id_sq.NEXTVAL, 1004, 'Nasi Uduk', 'Special', 40);`	`Constraint Violated`	`ORA-00001: unique constraint (KEL2.PR_ITM_NM_UK) violated`	`None`
3.	13/09/2024	Confirm cannot set	`INSERT INTO PRICES (Start_Date,`	`Constraint Violated`	`ORA-02290: check`	`None`

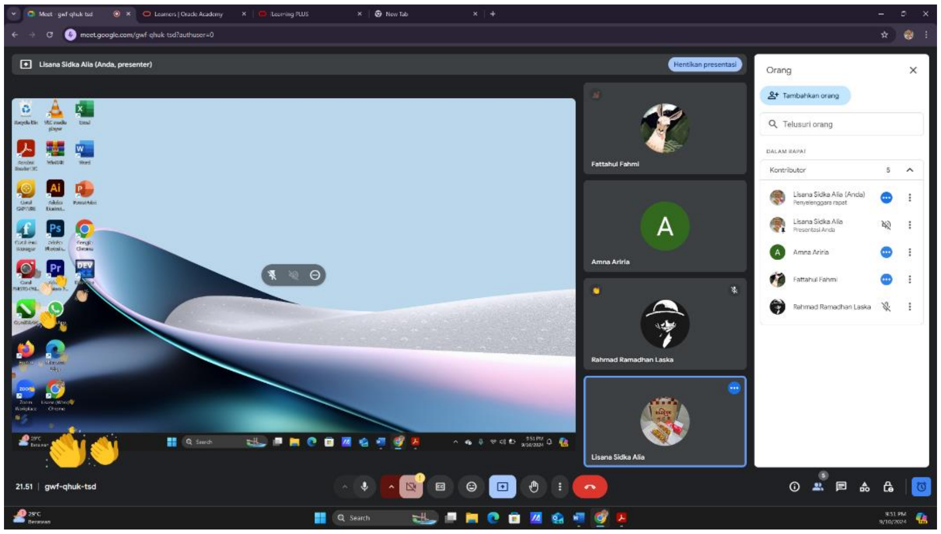
		input for End_Date before Start_Date, CONSTRAINT CHECK (End_Date > Start_Date);	`End_Date, Price, Product_ID) VALUES (SYSDATE, '12/09/2023', 5000, 18);`		constraint (KEL2.END_DATE_CHK) violated	
4.	13/09/2024	Confirm cannot set input for Order_Status if the input is not matched with the value of check constraint.	`INSERT INTO ORDERS (Order_ID, Customer_ID, Order_Date, Order_Type, Order_Status)` `VALUES (order_id_sq.NEXTVAL, 1, SYSTIMESTAMP, 'Online', 'Diantar');`	Check Constraint Violated	ORA-02290: check constraint (KELOMPOK_2.ORDERS_STATUS_CHK) violated	None
5.	13/09/2024	Confirm cannot set input for order_id if data is not in the parent table for Order_Items.	`INSERT INTO ORDER_ITEMS (Order_item_ID,Order_ID, Product_ID, Quantity) VALUES (order_item_id_sq.NEXTVAL, 30, 12, 23);`	Foreign key constraint violated	integrity constraint (ID_D885_SQL_S32.ORDER_ID_FK) violated - parent key not found	none
6.	12/09/2024	Confirm the TRIGGER for AMOUNT in PAYMENTS is working successfully and insert value for one row without violated	`INSERT INTO PAYMENTS (payment_id, payment_method_id, order_id, payment_status) VALUES (payment_id_sq.NEXTVAL, 2, 12, 'Completed');`	Row_Inserted	1 row(s) inserted.	none
7.	13/09/2024	Confirm the unique constraint in Method_Name for	`INSERT INTO PAYMENT_METHODS (payment_method_id,`	Unique Constraint Violated	ORA-00001: unique constraint (KELOM	none

		Payment_Mehtods is active	method_name, credit_card_number, description) VALUES (4, 'Cash', 'N/A', 'Pembayaran dilakukan secara tunai');		POK_2.PAY_METHODS_CCN_UK) violated	
8.	13/09/2024	Confirm the Special_Products cannot input null values	INSERT INTO SPECIAL_PRODUCTS (Special_Product_ID, Product_ID, Order_Date, Customer_ID) VALUES (special_product_id_sq.NEXTVAL, 12, NULL , 1);	Cannot insert NULL value	ORA-01400: cannot insert NULL into ("KELOMPOK_2"."SPECIAL_PRODUCTS"."ORDER_DATE")	none
9.	13/09/2024	Confirm the Primary Key of a table cannot be the same.	INSERT INTO SHIPMENTS (Shipment_ID, Order_ID, Shipment_Type) VALUES (1, 13, 'In-House');	Violated Unique Constraint in Primary Key Constraint	ORA-00001: unique constraint (KELOMPOK_2.SHIPMENT_ID_PK) violated	none
10.	13/09/2024	Confirm the Report is worked in a sequence.	INSERT INTO REPORTS (Report_ID, Order_ID, Product_ID, Customer_ID, Report_Date, Payment_ID) VALUES (report_id_seq.NEXTVAL, 13, 13, 3, SYSDATE, 1002);	Row inserted with row+1 auto increment	1 row(s) inserted. Report_ID = 10001	none

LAMPIRAN

Hasil Dokumentasi dan laporan kegaiatan kelompok





Link Hasil Diskusi :

https://docs.google.com/document/d/1D4OjEoWG1jtCTkyyACNNq0-wxEUkdLw-JQYmZG2RUS8/edit?usp=sharing

JOBDESK DATABASE PROGRAMMING GROUP 2

Recording Business Transactions of Breakfast Food Stalls

Fresh Graduate Academy Digitalent with Riau University

JOBDESK	AMNA	FAHTAHUL	LISANA	RAHMAD
1. Discussion	▨	▨	▨	▨
2. Notes		▨	▨	
3. ERD	▨	▨	▨	▨
4. Logical Model	▨			
5. Relational Model	▨			
6. Table OWNERS			▨	
7. Table CUSTOMERS			▨	
8. Table PRODUCT	▨			
9. Table PRICE	▨			
10. Table ORDER ITEMS	▨			
11. Table ORDERS				▨
12. Table PAYMENT				▨
13. Table PAYMENT METHODS				▨
14. Table SHIPMENTS		▨		
15. Table SPECIAL PRODUCTS		▨		
16. Table REPORTS		▨		
17. Constraint	▨	▨	▨	▨
18. View	▨	▨	▨	▨
19. Sequence		▨	▨	
20. Index		▨	▨	
21. Synonym	▨	▨	▨	
22. Scema	▨	▨		
23. Group Report			▨	
24. PPT				▨
25. SQL Grant Privillage		▨		
26. Testing Database		▨	▨	▨
27. Lead Group			▨	