**Dokumentasi**

**ERD  
Sebuah gambar berisi teks, cuplikan layar, diagram, Rencana

Deskripsi dibuat secara otomatis**

* Table Customer memiliki hubungan one to many ke table Sale
* Table Sale memiliki hubungan one to many ke SaleDetail
* Table Staff memiliki hubungan one to many ke table Sale
* Table Staff memiliki hubungan one to many ke table Purchase
* Table Product memiliki hubungan one to many ke table SaleDetail
* Table Product memiliki hubungan one to many ke table ProductCategory
* Table Product memiliki hubungan one to many ke table PurchaseDetail
* Table Vendor memiliki hubungan one to many ke table Purchase
* Table Purchase memiliki hubungan one to many ke table PurchaseDetail

**DDL**

|  |  |
| --- | --- |
| **Deskripsi** | **Syntax** |
| Membuat database bernama JigiBoxZ | CREATE DATABASE JigiBoxZ |
| Syntax untuk menggunakan database JigiBoxZ | USE JigiBoxZ |
| Membuat table Customer dengan kolom CustomerID dengan constraint PRIMARY KEY dan Constraint CHECK untuk memastikan bahwa kolom CustomerID memiliki format ‘CU’ diikuti dengan 3 digit dibelakang,  kemudian untuk kolom CustomerName sebagai kolom nama pelanggan dengan tipe data varchar (panjangnya 100) dan tidak boleh kosong,  kolom CustomerGender sebagai jenis kelamin pelanggan dengan constraint CHECK untuk memastikan kolom ini hanya diisi dengan ‘Male’ dan ‘Female’,  kolom CustomerEmail sebagai email pelanggan dengan tipe data varchar dan tidak boleh kosong,  kolom CustomerDOB sebagai tanggal lahir pelanggan dengan tipe data date dan memiliki constraint CHECK untuk memastikan tanggal lahir pelanggan harus sebelum tahun 2000,  kolom CustomerAddress sebagai alamat pelanggan dengan tipe data varchar dan tidak boleh kosong. | CREATE TABLE Customer (  CustomerID CHAR(5) PRIMARY KEY CHECK (CustomerID LIKE 'CU[0-9][0-9][0-9]'),  CustomerName VARCHAR (100) NOT NULL,  CustomerGender VARCHAR (10) NOT NULL CHECK (CustomerGender IN ('Male', 'Female')),  CustomerEmail VARCHAR(100) NOT NULL,  CustomerDOB DATE NOT NULL CHECK (YEAR(CustomerDOB) < 2000),  CustomerAddress VARCHAR (255) NOT NULL  ) |
| Membuat table Staff  kolom StaffId sebagai id staff dengan constraint PRIMARY KEY dan constraint CHECK untuk memastikan idstaff harus mengikuti format ‘ST’ dengan 3 digit dibelakangnya,  Kolom StaffName sebagai nama staff dengan tipe data varchar dan tidak boleh kosong,  Kolom StaffEmail sebagai email staff dengan tipe data varchar dan tidak boleh kosong,  Kolom StaffGender sebagai kolom jenis kelamin staff dan memiliki constraint CHECK untuk memastikan kolom ini hanya di isi dengan ‘Male’ dan ‘Female’,  Kolom StaffDOB sebagai kolom tanggal lahir staff dengan tipe data date dan memiliki constraint CHECK untuk memastikan tanggal lahir harus sebelum tahun 2000,  Kolom StaffAddress sebagai alamat staff dengan tipe data varchar dan tidak boleh kosong. | CREATE TABLE Staff (  StaffID CHAR(5) PRIMARY KEY CHECK (StaffID LIKE 'ST[0-9][0-9][0-9]'),  StaffName VARCHAR(100) NOT NULL,  StaffEmail VARCHAR(100) NOT NULL,  StaffGender VARCHAR(10) NOT NULL CHECK (StaffGender IN ('Male', 'Female')),  StaffDOB DATE NOT NULL CHECK (YEAR(StaffDOB) < 2000),  StaffAddress VARCHAR(255) NOT NULL  ) |
| Membuat table Vendor  Kolom VendorID sebagai id vendor dengan constraint PRIMARY KEY dan CHECK untuk memastikan kolom ini mengikuti format ‘VR’ diikuti dengan 3 digit dibelakang,  Kolom VendorName sebagai nama vendor dengan tipe data varchar dan tidak boleh kosong,  Kolom VendorGender sebagai kolom jenis kelamin vendor dan memiliki constrain CHECK untuk memastikan kolom ini hanya diisi ‘Male’ dan ‘Female’,  Kolom VendorEmail dengan tipe data varchar dan tidak boleh kosong,  Kolom VendorDOB sebagai tangal lahir vendor dengan tipe data date dan tidak boleh kosong,  Kolom VendorAddress sebagai kolom alamat vendor dengan tipe data varchar dan tidak boleh kosong. | CREATE TABLE Vendor (  VendorID CHAR(5) PRIMARY KEY CHECK (VendorID LIKE 'VR[0-9][0-9][0-9]'),  VendorName VARCHAR(100) NOT NULL,  VendorGender VARCHAR(10) NOT NULL CHECK (VendorGender IN ('Male', 'Female')),  VendorEmail VARCHAR(100) NOT NULL,  VendorDOB DATE NOT NULL,  VendorAddress VARCHAR(255) NOT NULL  ) |
| Membuat table ProductCategory  Kolom CategoryId sebagai id category dengan constraint PRIMARY KEY dan CHECK untuk memastikan CategoryID mengikuti format ‘CT’ dan diikuti 3 digit dibelakangnya,  Kolom CategoryName dengan tipedata varchar dan memiliki constraint CHECK untuk memastikan CategoryName hanya di isi sesuai dengan kategori yang ada di dalam syntax. | CREATE TABLE ProductCategory (  CategoryID CHAR(5) PRIMARY KEY CHECK (CategoryID LIKE 'CT[0-9][0-9][0-9]'),  CategoryName VARCHAR(50) NOT NULL CHECK (CategoryName IN (  'Mobile Phones',  'Smartphones',  'Android Phones',  'IOS Devices',  'Budget Smartphones',  'Flagship Phones',  'Phone Accessories',  'Phone Cases',  'Wireless Earbuds',  'Mobile Phone Chargers'  ))  ) |
| Membuat table Product  Kolom ProductID sebagai id product dengan constraint PRIMARY KEY dan CHECK untuk memastikan kolom ini mengikuti format ‘PT’ dan diikuti 3 digit dibelakangnya,  Kolom CategoryID sebagai id category dengan constraint FOREIGN KEY yang merupakan kunci asing dari tabel ProductCategory,  Kolom ProductName sebagai nama produk dengan constraint CHECK untuk memastikan nama produk memiliki panjang karakter lebih dari 10,  Kolom ProductPrice sebagai harga produk dengan constraint CHECK yang memastikan harga produk diantara 1000 dan 2000 menggunakan BETWEEN,  Kolom ProductWeight sebagai berat produk dengan tipe data INT dan tidak boleh kosong. | CREATE TABLE Product (  ProductID CHAR(5) PRIMARY KEY CHECK (ProductID LIKE 'PT[0-9][0-9][0-9]'),  CategoryID CHAR(5) FOREIGN KEY REFERENCES ProductCategory(CategoryID),  ProductName VARCHAR(100) NOT NULL CHECK (LEN(ProductName) > 10),  ProductPrice INT NOT NULL CHECK (ProductPrice BETWEEN 1000 AND 2200),  ProductWeight INT NOT NULL  ) |
| Membuat table Sale  kolom SaleID sebagai id sale dengan constraint PRIMARY KEY dan CHECK untuk memastikan kolom ini mengikuti format ‘SH’ dan diikuti 3 digit dibelakangnya,  Kolom StaffID sebagai id staff dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Staff,  Kolom CustomerID sebagai id Customer dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Customer,  Kolom TransactionDate sebagai tanggal transaksi dengan tipe data date dan tidak boleh kosong. | CREATE TABLE Sale (  SaleID CHAR(5) PRIMARY KEY CHECK (SaleID LIKE 'SH[0-9][0-9][0-9]'),  StaffID CHAR(5) FOREIGN KEY REFERENCES Staff(StaffID),  CustomerID CHAR(5) FOREIGN KEY REFERENCES Customer(CustomerID),  TransactionDate DATE NOT NULL  ) |
| Membuat table SaleDetail yang merupakan detail dari table Sale  kolom SaleID sebagai id sale yang merupakan PRIMARY KEY dan FOREIGN KEY dari table Sale,  Kolom ProductID sebagai id product dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Product,  Kolom Quantity sebagai kuantitas penjualan dengan tipe data int dan tidak boleh kosong. | CREATE TABLE SaleDetail (  SaleID CHAR(5),  ProductID CHAR(5),  Quantity INT NOT NULL,  PRIMARY KEY (SaleID, ProductID),  FOREIGN KEY (SaleID) REFERENCES Sale(SaleID),  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)  ) |
| Membuat table Purchase  Kolom PurchaseID sebagai id purchase dengan constraint PRIMARY KEY dan CHECK untuk memastikan kolom ini mengikuti format ‘PH’ diikuti dengan 3 digit dibelakangnya,  Kolom StaffID sebagai id staff dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Staff,  Kolom VendorID sebagai id vendor dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Vendor,  Kolom PurchaseDate sebagai tanggal pembelian dengan tipe data date dan tidak boleh kosong. | CREATE TABLE Purchase (  PurchaseID CHAR(5) PRIMARY KEY CHECK (PurchaseID LIKE 'PH[0-9][0-9][0-9]'),  StaffID CHAR(5) FOREIGN KEY REFERENCES Staff(StaffID),  VendorID CHAR(5) FOREIGN KEY REFERENCES Vendor(VendorID),  PurchaseDate DATE NOT NULL  ) |
| Membuat table PurchaseDetail yang merupakan detail dari table Purchase  Kolom PurchaseID sebagai id purchase dengan constraint PRIMARY KEY dan FOREIGN KEY dari table Purchase,  Kolom ProductID sebagai id product dengan constraint FOREIGN KEY yang merupakan kunci asing dari table Product,  Kolom Quantity sebagai kuantitas pembelian dengan tipe data int dan tidak boleh kosong. | CREATE TABLE PurchaseDetail (  PurchaseID CHAR(5),  ProductID CHAR(5),  Quantity INT NOT NULL,  PRIMARY KEY (PurchaseID, ProductID),  FOREIGN KEY (PurchaseID) REFERENCES Purchase(PurchaseID),  FOREIGN KEY (ProductID) REFERENCES Product(ProductID)  ) |

DML

|  |  |
| --- | --- |
| **Deskiprsi** | **Syntax** |
| Memasukkan data ke table Customer dengan perintah ‘INSERT INTO’ diikuti nama table Customer.  Kolom yang diisi adalah CustomerID, CustomerName, CustomerGender, CustomerEmail, CustomerDOB, CustomerAddress.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Customer. | INSERT INTO Customer (CustomerID, CustomerName, CustomerGender, CustomerEmail, CustomerDOB, CustomerAddress) VALUES  ('CU029','Mr. Budi Santoso', 'Male', 'budi.santoso@example.com', '1986-09-19', 'Jl. Merdeka No. 10'),  ('CU030','Mrs. Siti Rahmawati', 'Female', 'siti.rahmawati@example.com', '1999-04-28', 'Jl. Sudirman No. 15'),  ('CU031','Mrs. Kirana Dewi', 'Female', 'kirana.dewi@example.com', '1988-06-20', 'Jl. Gajah Mungkur No. 8'),  ('CU032','Mrs. Felia Anggraini', 'Female', 'felia.anggraini@example.com', '1999-12-28', 'Jl. Pahlawan No. 3'),  ('CU033','Mr. Oka Masidi', 'Male', 'oka.masidi@example.com', '1999-08-12', 'Jl. Manggis No. 5'),  ('CU034','Mrs. Mita Permata', 'Female', 'mita.permata@example.com', '1993-02-18', 'Jl. Mawar No. 24'),  ('CU035','Mrs. Indah Mulia ', 'Female', 'indah.Mulia@example.com', '1999-12-01', 'Jl. Flaminggo No. 7'),  ('CU036','Mrs. Nina Melati', 'Female', 'nina.melati@example.com', '1986-07-07', 'Jl. Melati Raya No. 12'),  ('CU037','Mr. Lukman Hakim', 'Male', 'lukman.hakim@example.com', '1984-10-05', 'Jl. Jati No. 62'),  ('CU038','Mr. Risky Ridho', 'Male', 'risky.ridho@example.com', '1999-09-19', 'Jl. Boulevard No. 65'); |
| Memasukkan data ke table Staff dengan perintah ‘INSERT INTO’ diikuti nama table Staff.  Kolom yang diisi adalah StaffID, StaffName, StaffEmail, StaffGender, StaffDOB, StaffAddress.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Staff. | INSERT INTO Staff (StaffID, StaffName, StaffEmail, StaffGender, StaffDOB, StaffAddress)  VALUES  ('ST001', 'Alice Grabriela', 'alice.Grabriela@example.com', 'Female', '1985-03-10', 'Jl. Cendana No. 11'),  ('ST002', 'Mursid Putra', 'Mursid.Putra@example.com', 'Male', '1975-12-25', 'Jl. Flamboyan No. 7'),  ('ST003', 'David gaget', 'David.gaget@example.com','Male', '1988-02-14', 'Jl. Teratai No. 4'),  ('ST004', 'Muthia Putri', 'Muthia.Putri@example.com','Female', '1992-08-19', 'Jl. Bunga Raya No. 12'),  ('ST005','Siti Aminah', 'siti.aminah@example.com', 'Female', '1980-01-15', 'Jl. Cendana No. 1'),  ('ST006','Ahmad Faisal', 'ahmad.faisal@example.com', 'Male', '1978-05-20', 'Jl. Merpati No. 3'),  ('ST007','Nurul Hidayah', 'nurul.hidayah@example.com', 'Female', '1992-06-25', 'Jl. Mawar No. 7'),  ('ST008','Citra Dewi', 'citra.dewi@example.com', 'Female', '1983-12-20', 'Jl. Teratai No. 6'),  ('ST009','Yudi Santoso', 'yudi.santoso@example.com', 'Male', '1989-08-14', 'Jl. Merdeka No. 12'),  ('ST010','Rahmat Hidayat', 'rahmat.hidayat@example.com', 'Male', '1988-09-09', 'Jl. Melati No. 9'); |
| Memasukkan data ke table Vendor dengan perintah ‘INSERT INTO’ diikuti nama table Vendor.  Kolom yang diisi adalah VendorID, VendorName, VendorGender, VendorEmail, VendorDOB, VendorAddress.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Vendor. | INSERT INTO Vendor (VendorID, VendorName, VendorGender, VendorEmail, VendorDOB, VendorAddress)  VALUES  ('VR001', 'Faisal Udin', 'Male', 'faisal.udin@example.com', '1970-01-01', 'Jl. Market Street'),  ('VR002', 'Fajar Sidiq', 'Male', 'fajar.sidiq@example.com', '1982-08-15', 'Jl. Commerce Lane'),  ('VR003', 'Gita Anjani', 'Female', 'gita.anjani@example.com', '1975-10-10', 'Jl. Business Park'),  ('VR004', 'Lina Marlina', 'Male', 'lina.marlina@example.com', '1983-05-05', 'Jl. Home Street'),  ('VR005','Hendra Purnama', 'Male', 'hendra.purnama@example.com', '1991-08-25', 'Jl. Flamboyan No. 6, Makassar'),  ('VR006','Yudistira Putra', 'Female', 'yudistira.putra@example.com', '1989-09-30', 'Jl. Cendana No. 4, Bogor'),  ('VR007', 'Indra Wijaya', 'Male', 'indra.wijaya@example.com', '1985-02-20', 'Jl. Kebon Jeruk No. 3, Jakarta'),  ('VR008', 'Maya Sari', 'Female', 'maya.sari@example.com', '1978-06-12', 'Jl. Mawar No. 8, Surabaya'),  ('VR009', 'Dian Permana', 'Male', 'dian.permana@example.com', '1992-12-05', 'Jl. Melati No. 5, Bandung'),  ('VR010', 'Rina Kusuma', 'Female', 'rina.kusuma@example.com', '1980-03-25', 'Jl. Anggrek No. 2, Semarang'); |
| Memasukkan data ke table ProductCategory dengan perintah ‘INSERT INTO’ diikuti nama table ProductCategory.  Kolom yang diisi adalah CategoryID dan CategoryName.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data ProductCategory. | INSERT INTO ProductCategory (CategoryID,CategoryName) VALUES  ('CT001','Mobile Phones'),  ('CT002','Smartphones'),  ('CT003','Android Phones'),  ('CT004','IOS Devices'),  ('CT005','Budget Smartphones'),  ('CT006','Flagship Phones'),  ('CT007','Phone Accessories'),  ('CT008','Phone Cases'),  ('CT009','Wireless Earbuds'),  ('CT010','Mobile Phone Chargers'); |
| Memasukkan data ke table Product dengan perintah ‘INSERT INTO’ diikuti nama table Product.  Kolom yang diisi adalah ProductID, CategoryID, ProductName, ProductPrice, ProductWeight.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Product. | INSERT INTO Product (ProductID, CategoryID, ProductName, ProductPrice, ProductWeight) VALUES  ('PT001','CT001','Samsung Galaxy A12', 1000, 160),  ('PT002','CT002','iPhone 12 Pro', 1500, 200),  ('PT003','CT003','OnePlus Nord', 1500, 140),  ('PT004','CT004','iPhone SE (2020)', 1700, 200),  ('PT005','CT005','Xiaomi Redmi 9A', 1500, 200),  ('PT006','CT006','Samsung Galaxy S21 Ultra', 2200, 200),  ('PT007','CT007','Anker PowerCore Portable Charger', 1400, 200),  ('PT008','CT008','Spigen Tough Armor Case', 2200, 80),  ('PT009','CT009','Apple AirPods Pro', 2200, 50),  ('PT010','CT010','Belkin BoostCharge USB-C Charger',1100, 70); |
| Memasukkan data ke table Sale dengan perintah ‘INSERT INTO’ diikuti nama table Sale.  Kolom yang diisi adalah SaleID, StaffID, CustomerID, TransactionDate.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Sale. | INSERT INTO Sale (SaleID, StaffID, CustomerID, TransactionDate) VALUES  ('SH001', 'ST001', 'CU029', '2023-01-11'),  ('SH002', 'ST002', 'CU030', '2023-02-02'),  ('SH003', 'ST003', 'CU031', '2023-04-22'),  ('SH004', 'ST004', 'CU032', '2023-04-24'),  ('SH005', 'ST005', 'CU033', '2023-06-05'),  ('SH006', 'ST006', 'CU034', '2023-08-06'),  ('SH007', 'ST007', 'CU035', '2024-01-01'),  ('SH008', 'ST008', 'CU036', '2024-01-02'),  ('SH009', 'ST009', 'CU037', '2024-04-03'),  ('SH010', 'ST010', 'CU038', '2024-05-04'),  ('SH011', 'ST001', 'CU029', '2024-05-05'),  ('SH012', 'ST002', 'CU030', '2024-06-06'),  ('SH013', 'ST003', 'CU031', '2024-07-06'),  ('SH014', 'ST004', 'CU032', '2024-07-07'),  ('SH015', 'ST005', 'CU033', '2024-07-07'); |
| Memasukkan data ke table SaleDetail dengan perintah ‘INSERT INTO’ diikuti nama table SaleDetail.  Kolom yang diisi adalah SaleID, ProductID, Quantity.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data SaleDetail. | INSERT INTO SaleDetail (SaleID, ProductID, Quantity) VALUES  ('SH001', 'PT001', 2),  ('SH002', 'PT003', 10),  ('SH003', 'PT002', 1),  ('SH004', 'PT004', 5),  ('SH005', 'PT005', 1),  ('SH006', 'PT006', 2),  ('SH007', 'PT007', 15),  ('SH008', 'PT008', 3),  ('SH009', 'PT009', 2),  ('SH010', 'PT010', 1),  ('SH011', 'PT005', 3),  ('SH012', 'PT003', 4),  ('SH013', 'PT010', 7),  ('SH014', 'PT009', 2),  ('SH015', 'PT005', 6),  ('SH010', 'PT001', 5),  ('SH009', 'PT007', 1),  ('SH008', 'PT004', 2),  ('SH009', 'PT010', 1),  ('SH001', 'PT003', 9),  ('SH002', 'PT002', 11),  ('SH014', 'PT008', 2),  ('SH009', 'PT005', 4),  ('SH011', 'PT010', 1),  ('SH013', 'PT003', 5); |
| Memasukkan data ke table Purchase dengan perintah ‘INSERT INTO’ diikuti nama table Purchase.  Kolom yang diisi adalah PurchaseID, StaffID, VendorID, PurchaseDate.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data Purchase. | INSERT INTO Purchase (PurchaseID, StaffID, VendorID, PurchaseDate) VALUES  ('PH001', 'ST001', 'VR001', '2024-05-01'),  ('PH002', 'ST002', 'VR002', '2024-06-02'),  ('PH003', 'ST003', 'VR003', '2024-06-03'),  ('PH004', 'ST004', 'VR004', '2024-07-04'),  ('PH005', 'ST005', 'VR005', '2024-07-05'),  ('PH006', 'ST006', 'VR006', '2024-08-06'),  ('PH007', 'ST007', 'VR003', '2024-08-01'),  ('PH008', 'ST008', 'VR006', '2024-09-02'),  ('PH009', 'ST009', 'VR005', '2024-09-03'),  ('PH010', 'ST010', 'VR007', '2024-10-04'),  ('PH011', 'ST002', 'VR008', '2024-10-05'),  ('PH012', 'ST004', 'VR009', '2024-11-06'),  ('PH013', 'ST008', 'VR010', '2024-11-01'),  ('PH014', 'ST010', 'VR002', '2024-11-21'),  ('PH015', 'ST003', 'VR005', '2024-11-24'); |
| Memasukkan data ke table PurchaseDetail dengan perintah ‘INSERT INTO’ diikuti nama table PurchaseDetail.  Kolom yang diisi adalah PurchaseID, ProductID, Quantity.  Perintah ‘VALUES’ untuk memasukkan nilai-nilai setiap data PurchaseDetail. | INSERT INTO PurchaseDetail (PurchaseID, ProductID, Quantity) VALUES  ('PH001', 'PT001', 1),  ('PH002', 'PT003', 3),  ('PH003', 'PT002', 7),  ('PH004', 'PT004', 5),  ('PH005', 'PT003', 9),  ('PH006', 'PT007', 3),  ('PH007', 'PT004', 4),  ('PH008', 'PT008', 7),  ('PH009', 'PT010', 2),  ('PH010', 'PT001', 4),  ('PH011', 'PT009', 5),  ('PH012', 'PT003', 4),  ('PH013', 'PT004', 2),  ('PH014', 'PT007', 7),  ('PH015', 'PT001', 3),  ('PH011', 'PT003', 1),  ('PH003', 'PT003', 2),  ('PH004', 'PT006', 6),  ('PH006', 'PT004', 8),  ('PH002', 'PT010', 1),  ('PH012', 'PT004', 4),  ('PH015', 'PT003', 9),  ('PH005', 'PT008', 2),  ('PH001', 'PT007', 2),  ('PH009', 'PT002', 4); |

SIMULATE TRANSACTION

-- Purchase transaction (Table : Staff, Vendor, Purchase, PurchaseDetail, Product, ProductCategory)

Sebuah gambar berisi teks, cuplikan layar, Font, nomor

Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks, cuplikan layar, Font, nomor

Deskripsi dibuat secara otomatis

Penjelasan : Untuk dapat menjalankan transaksi pembelian, table staff dan vendor harus mengisi data secara lengkap. Kemudian jika Staff membeli produk ke vendor maka table Purchase akan memiliki data kedua table dan tanggal transaksinya. Selanjutnya untuk detail pembelian akan berada di table PurchaseDetail yang berisi data produk dan kuantitas setiap produk yang dibeli oleh staff.

-- Sales Transaction (Table : Customer, Staff, Sale, SaleDetail, Product, ProductCategory)

Sebuah gambar berisi teks, cuplikan layar, Font, nomor

Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

Penjelasan : Untuk menjalankan transaksi penjualan, table customer dan staff harus mengisi data secara lengkap. Kemudian jika customer membeli produk ke staff maka table Sale akan memiliki data kedua table dan tanggal transaksinya. Selanjutnya untuk detail penjualan akan berada di table SaleDetail yang berisi data produk dan kuantitas setiap produk yang dibeli oleh customer.

QUERY 10 Soal

1. Display Customer Name (obtained from the CustomerName starts with ‘Mr./Mrs. ’), Staff Name (obtained from StaffID concat by ‘ - ’ and StaffName), and Total Transaction Made (obtained from total number of sales) for each sales date occurred in 2023 and Customer that born on 1999.

SELECT CustomerName AS [Customer Name],

CONCAT(st.StaffID, ' - ', StaffName) AS [Staff Name],

COUNT(SaleID) AS [Total Transaction Made]

FROM Sale sh

INNER JOIN Customer c ON sh.CustomerID = c.CustomerID

INNER JOIN Staff st ON sh.StaffID = st.StaffID

WHERE YEAR(TransactionDate) = 2023 AND

YEAR(CustomerDOB) = 1999 AND

(CustomerName LIKE 'Mr. %' OR CustomerName LIKE 'Mrs. %')

GROUP BY CustomerName, CONCAT(st.StaffID, ' - ', st.StaffName);

**Penjelasan :** Mengambil CustomerName, Staff ID yang digabung dengan ‘-‘ serta StaffName, SaleID dengan perintah ‘SELECT’.

CustomerName diberi alias [Customer Name], Kemudian Menggabungkan st.StaffID dengan ‘-‘ serta StaffName dengan fungsi CONCAT dan diberi alias [Staff Name].

‘FROM’ digunakan untuk mengambil data dari table yang ditentukan yaitu Sale , lalu ‘INNER JOIN’ digunakan untuk menggabungkan table Customer berdasarkan CustomerID dan table Staff berdasarkan StaffID.

Kemudian dengan menggunakan ‘WHERE’ untuk kondisi TransactionDate harus sama dengan 2023 dan CustomerDOB harus sama dengan tahun 1999 dan CustomerName harus seperti ‘Mr. %’ atau ‘Mrs. %’ . Formatnya harus diawali dengan ‘Mr. ‘ atau ‘Mrs. ‘ lalu diikuti dengan karakter apa saja ‘%’. Fungsi YEAR(TransactionDate) adalah untuk mengambil tahunnya saja dari kolom TransactionDate begitu juga dengan kolom CustomerDOB.  
  
Terakhir ‘GROUP BY’ digunakan untuk mengelompokkan hasil query berdasarkan CustomerName dan CONCAT(st.StaffID, ' - ', st.StaffName).

Jadi query ini menghasilkan jumlah total transaksi yang dilakukan oleh setiap pelanggan dengan nama pelanggan dan nama staff yang menanganinya.

2. Display the Sales ID (obtained by replacing 'SH' in the salesID with 'SalesID '), ProductName, and Total Product Sold (obtained from the sum of quantity) for each SalesID where the last three digits are even, and the quantity sold was between 1 and 5 (inclusive).

SELECT REPLACE(sh.SaleID, 'SH', 'SalesID ') AS [Sales ID],

ProductName,

SUM(Quantity) AS [Total Product Sold]

FROM Sale sh

INNER JOIN SaleDetail sd ON sh.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

WHERE RIGHT(sh.SaleID, 3) % 2 = 0 AND

Quantity BETWEEN 1 AND 5

GROUP BY REPLACE(sh.SaleID, 'SH', 'SalesID '), ProductName;  
  
Penjelasan : Query ini mengambil SalesID (dengan mengganti ‘SH’ menjadi ‘SalesID’ menggunakan fungsi REPLACE() ), ProductName, dan Total Product Sold (menggunakan fungsi SUM() untuk mengambil jumlah total quantity tiap produk) dari table Sale. Kemudian table Sale memiliki relasi ke table SaleDetail dan Product menggunakan ‘INNER JOIN’. Kemudian ‘WHERE’ digunakan untuk pengkondisian yang dimana ‘RIGHT(sh.SaleID, 3) % 2 = 0’ digunakan untuk menyeleksi SaleID yang 3 digit terakhirnya harus genap menggunakan fungsi RIGHT() dan kondisi ‘Quantity BETWEEN 1 AND 5’ digunakan untuk menyeleksi bahwa Quantity harus diantara 1 dan 5 menggunakan fungsi BETWEEN.

3. Display the PurchaseID, Staff Email (obtained by replacing the ‘@example.com’ with

‘@DigiBoxZ.com’), VendorName, Total Purchases Made (obtained from the total number of purchases), and Total Quantity (obtained from the sum of quantity) for each staff whose name length more than 5 characters and whose purchase quantity was between 1 and 5 (inclusive).

SELECT p.PurchaseID,

REPLACE(StaffEmail, '@example.com', '@DigiBoxZ.com') AS [Staff Email],

VendorName,

COUNT(p.PurchaseID) AS [Total Purchases Made],

SUM(Quantity) AS [Total Quantity]

FROM Purchase p

INNER JOIN Staff s ON p.StaffID = s.StaffID

INNER JOIN Vendor v ON p.VendorID = v.VendorID

INNER JOIN PurchaseDetail pd ON p.PurchaseID = pd.PurchaseID

WHERE LEN(StaffName) > 5

GROUP BY p.PurchaseID, REPLACE(StaffEmail, '@example.com', '@DigiBoxZ.com'), VendorName

HAVING SUM(Quantity) BETWEEN 1 AND 5;

Penjelasan : Query ini mengambil PurchaseID, StaffEmail (mengganti @example.com dengan @DigiBoxZ.com menggunakan fungsi REPLACE() ) dan diberi alias [Staff Email], VendorName, PurchaseID (untuk purchaseID yang ini adalah untuk total purchase made dengan menggunakan fungsi COUNT() untuk menghitung jumlah pembelian) . Query ‘SELECT’ diambil dari table Purchase menggunakan FROM, lalu memiliki relasi ke table Staff, Vendor, dan PurchaseDetail. Kemudian ‘WHERE’ untuk pengkondisian LEN(StaffName) > 5 digunakan untuk menyeleksi data yang panjang karakter StaffNamenya harus lebih dari 5 menggunakan fungsi LEN() untuk mencari panjang suatu string. Untuk ‘GROUP BY’ digunakan untuk mengelompokkan hasil query berdasarkan p.PurchaseID, REPLACE(StaffEmail, '@example.com', '@DigiBoxZ.com'), VendorName . Terakhir ‘HAVING’, sama seperti ‘WHERE’ bedanya HAVING digunakan setelah query dikelompokkan dengan GROUP BY , HAVING disini digunakan untuk memberi kondisi bahwa jumlah Quantity harus antara 1 dan 5 menggunakan fungsi SUM() dan operator BETWEEN.

4. Display the Sales ID (obtained by replacing the ‘SH’ with ‘SalesID ’), CustomerName, Sales Date (obtained from salesDate formatted as ‘MON DD, YYYY’), Sales Date Quarter (obtained from the quarter of the sales date), Total Product Sold (obtained from the sum of quantity), and Total Sales Transactions Made (obtained from total number of sales) for each sale that occurred in the 1st or 2nd quarter of 2023.

SELECT REPLACE(sh.SaleID, 'SH', 'SalesID ') AS [Sales ID],

CustomerName,

FORMAT(TransactionDate, 'MMM dd, yyyy') AS [Sales Date],

DATENAME(QUARTER, TransactionDate) AS [Sales Date Quarter],

SUM(Quantity) AS [Total Product Sold],

COUNT(sh.SaleID) AS [Total Sales Transactions Made]

FROM Sale sh

INNER JOIN Customer c ON sh.CustomerID = c.CustomerID

INNER JOIN SaleDetail sd ON sh.SaleID = sd.SaleID

WHERE DATENAME(QUARTER, TransactionDate) IN (1,2) AND YEAR(TransactionDate) = 2023

GROUP BY REPLACE(sh.SaleID, 'SH', 'SalesID '), CustomerName, TransactionDate, DATENAME(QUARTER, TransactionDate);  
  
Penjelasan : Query ini mengambil SaleID (dengan mengganti ‘SH’ menjadi ‘SalesID’ dan diberikan alias [Sales ID]), CustomerName, TransactionDate (dengan menggunkan fungsi format() ‘bulan hari, tahun’ , TransactionDate (untuk transactionDate yang ini adalah untuk mencari kuartal tahun dengan fungsi QUARTER() ), Quantity (menggunakan fungsi SUM() untuk mencari jumlah Quantity), dan SaleID (Menggunakan fungsi COUNT() untuk menghitung jumlah transaksi Sale) dari table Sale mengguakan ‘FROM’ , lalu table Sale memiliki relasi ke table Customer dan SaleDetail dengan menggunakan ‘INNER JOIN’.   
Kemudian ‘WHERE’ digunakan untuk pengkondisian DATENAME(QUARTER, TransactionDate) IN (1,2) kondisi ini digunakan untuk menyeleksi TransactionDate harus berada di kuartal 1 dan 2 menggunakan fungsi Datename dengan paramerter pertama QUARTER. Dan kondisi YEAR(TransactionDate) = 2023 digunakan untuk menyeleksi TransactionDate sama dengan 2023 dengan fungsi YEAR. Hasil query ini kemudian dikelompokkan berdasarkan ID Sale yang telah diubah, CustomerName, TransactionDate, dan kuartal dari tanggal transaksi.

5. Display Product ID (obtained from the last 3 digits of productID), Staff Name (obtained from replacing ‘ST’ of StaffID with ‘Staff’ and concatenating it with ‘ - ’ and StaffName), Product Name (obtained by replacing ‘PT’ with ‘Product’ and concatenating it with ‘ - ’ and

ProductName), Product Price, and Average Price (calculated from the average of ProductPrice) for each PurchaseID where the last three digits are odd and ProductPrice more than Average Price.

SELECT RIGHT(p.ProductID, 3) AS [Product ID],

CONCAT(REPLACE(st.StaffID, 'ST', 'StaffID'), ' - ', StaffName) AS [Staff Name],

CONCAT(REPLACE(ProductName, 'PT', 'Product'), ' - ', ProductName) AS [Product Name],

ProductPrice,

(SELECT AVG(ProductPrice) FROM Product) AS [Average Price]

FROM Product p

INNER JOIN SaleDetail sd ON p.ProductID = sd.ProductID

INNER JOIN Sale s ON sd.SaleID = s.SaleID

INNER JOIN Staff st ON s.StaffID = st.StaffID

WHERE RIGHT(p.ProductID, 3) % 2 = 1

GROUP BY RIGHT(p.ProductID, 3),

CONCAT(REPLACE(st.StaffID, 'ST', 'StaffID'), ' - ', StaffName),

CONCAT(REPLACE(ProductName, 'PT', 'Product'), ' - ', ProductName),

ProductPrice

HAVING ProductPrice > (SELECT AVG(ProductPrice) FROM Product);

Penjelasan : Query ini mengambil ProductID (menggunakan fungsi RIGHT untuk mengambil 3 digit terakhir id produk), StaffID (menggunakan fungsi concat untuk menggabungkan staffid yang sudah diganti dengan ‘ST’ menjadi ‘StaffID’ dengan ‘-‘ dan StaffName, ProductName (menggunakan concat() untuk menggabungkan productName dengan fungsi REPLACE() yang digunakna untuk mengganti ‘PT’, menjadi ‘Product’ dengan ‘-‘ dan ProductName, ProductPrice, dan alias subquery (SELECT AVG(ProductPrice) FROM Product)

Yang mengambil rata-rata productpricce dari table product. Query ini hanya memilih produk dengan tiga karakter terakhir ProductID yang bernilai ganjil. Data diambil dari tabel Product yang dihubungkan dengan tabel SaleDetail berdasarkan ProductID, kemudian tabel Sale berdasarkan SaleID, dan akhirnya tabel Staff berdasarkan StaffID. Query ini mengelompokkan hasil berdasarkan 'Product ID', 'Staff Name', 'Product Name', dan harga produk, serta hanya menampilkan produk dengan harga di atas rata-rata harga produk.

6. Display SalesID, Sales Date (obtained from salesDate formatted as ‘DD MON YYYY’),

ProductName, CategoryName, Quantity (obtained from concatenating Quantity with ‘ PCS’),

and Product Price (obtained from concatenating ProductPrice with ‘$ ’ in front of ProductPrice) for each sales occurred on 2023 and TotalPrice(obtained from total of all product price) subtract by ProductPrice must be more than 14000.

SELECT SaleID,

FORMAT(TransactionDate, 'dd MMM yyyy') AS [Sales Date],

ProductName,

CategoryName,

CONCAT(Quantity, ' PCS') AS [Quantity],

CONCAT('$ ', ProductPrice) AS [Product Price]

FROM (SELECT s.SaleID, TransactionDate, ProductName, CategoryName, Quantity, ProductPrice FROM Sale s

INNER JOIN SaleDetail sd ON s.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

INNER JOIN ProductCategory pc ON p.CategoryID = pc.CategoryID

WHERE YEAR(TransactionDate) = 2023 AND

(SELECT SUM(ProductPrice) FROM Product) - ProductPrice > 14000) AS Sales2023

Query ini mengambil data penjualan dari database untuk tahun 2023, menampilkan ID penjualan, tanggal transaksi dalam format 'dd MMM yyyy', nama produk, nama kategori, jumlah produk dalam format 'PCS', dan harga produk dalam format '$'. Query ini pertama-tama memilih data dari tabel Sale, SaleDetail, Product, dan ProductCategory, menghubungkannya berdasarkan SaleID, ProductID, dan CategoryID. Kemudian, query memfilter hanya transaksi dari tahun 2023 di mana selisih antara total harga semua produk dan harga produk individu lebih besar dari 14.000. Data hasil seleksi ini kemudian diolah untuk ditampilkan dengan format yang diinginkan dan diberi alias Sales2023, yang akhirnya digunakan untuk mengambil kolom yang diperlukan untuk ditampilkan dalam hasil akhir.

7. Display SalesID, Customer Email (obtained by replacing '@example.com' with

'@DigiBoxZ.com'), Sales Date (obtained from SalesDate formatted as 'MON DD, YYYY' ),

ProductName, ProductPrice, and Result of MaxPrice – MinPrice (obtained from maximum of

ProductPrice substract by minimum of ProductPrice) for every sales quantity is more than 4 and the product price is less than or equal to the Result of Max Price - Min Price.

SELECT s.SaleID,

REPLACE(CustomerEmail, '@example.com', '@DigiBoxZ.com') AS [Customer Email],

FORMAT(TransactionDate, 'MMM dd, yyyy') AS [Sales Date],

ProductName,

ProductPrice,

(SELECT MAX(ProductPrice) - MIN(ProductPrice) FROM Product) AS [Result of MaxPrice - MinPrice]

FROM Sale s

INNER JOIN Customer c ON s.CustomerID = c.CustomerID

INNER JOIN SaleDetail sd ON s.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

WHERE quantity > 4

GROUP BY s.SaleID,

REPLACE(CustomerEmail, '@example.com', '@DigiBoxZ.com'),

FORMAT(TransactionDate, 'MMM dd, yyyy'),

ProductName,

ProductPrice

HAVING ProductPrice <= (SELECT MAX(ProductPrice) - MIN(ProductPrice) FROM Product)

Penjelasan : Query ini menampilkan informasi penjualan dari database dengan beberapa kriteria. Pertama, query mengambil ID penjualan, mengubah email pelanggan dari '@example.com' menjadi '@DigiBoxZ.com', dan memformat tanggal transaksi menjadi 'MMM dd, yyyy'. Juga, query menampilkan nama produk dan harga produk. Di samping itu, query menghitung selisih antara harga tertinggi dan terendah dari semua produk dalam database, dan menamainya sebagai 'Result of MaxPrice - MinPrice'. Data diambil dari tabel Sale yang dihubungkan dengan tabel Customer berdasarkan CustomerID, dengan tabel SaleDetail berdasarkan SaleID, dan dengan tabel Product berdasarkan ProductID. Query ini memfilter hanya penjualan dengan jumlah produk lebih dari 4, kemudian mengelompokkan hasil berdasarkan ID penjualan, email pelanggan yang sudah diubah, tanggal transaksi, nama produk, dan harga produk. Hasil akhirnya adalah menampilkan produk yang harganya kurang atau sama dengan selisih antara harga maksimum dan minimum dari tabel Product.

8. Display SalesID, VIP Customer (obtained from 'VIP CUSTOMER', ‘ - ’ concatenated with the CustomerName in uppercase format), Month (obtained from the month of the SalesDate), Total Expenses (obtained by casting the total multiplication of ProductPrice and Quantity and adding '$ ' in front of Total Expenses) for every sales that the Sales Month occurred in February, April, or July, and the total multiplication of ProductPrice and Quantity is greater than the average total multiplication of ProductPrice and Quantity.

SELECT SaleID,

CONCAT('VIP CUSTOMER', ' - ', UPPER(CustomerName)) AS [VIP Customer],

MONTH(TransactionDate) AS [Month],

CONCAT('$ ', CAST(SUM(ProductPrice\*Quantity) AS varchar)) AS [Total Expenses]

FROM (SELECT s.SaleID, CustomerName, TransactionDate, ProductPrice, Quantity FROM Sale s

INNER JOIN Customer c ON s.CustomerID = c.CustomerID

INNER JOIN SaleDetail sd ON s.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

WHERE DATENAME(MONTH, TransactionDate) IN ('February', 'April', 'July') AND

ProductPrice\*Quantity > (SELECT AVG(ProductPrice\*Quantity)

FROM Product p

INNER JOIN SaleDetail sd ON p.ProductID = sd.ProductID)) AS MontlySales

GROUP BY SaleID,

CONCAT('VIP CUSTOMER', ' - ', UPPER(CustomerName)),

MONTH(TransactionDate);

Penjelasan : Query ini menampilkan data penjualan yang memenuhi kriteria tertentu, dengan beberapa manipulasi data untuk memberikan informasi tambahan. Query ini menampilkan ID penjualan, nama pelanggan yang diperlakukan sebagai pelanggan VIP (dengan menambahkan 'VIP CUSTOMER - ' di depan dan mengubah nama pelanggan menjadi huruf besar), bulan transaksi, dan total pengeluaran dalam format '$ '. Data diambil dari tabel Sale yang dihubungkan dengan tabel Customer berdasarkan CustomerID, tabel SaleDetail berdasarkan SaleID, dan tabel Product berdasarkan ProductID. Query difilter untuk hanya menyertakan penjualan yang terjadi pada bulan Februari, April, dan Juli, serta memiliki total harga (ProductPrice\*Quantity) lebih besar dari rata-rata total harga produk yang terjual. Data hasil seleksi ini diolah dalam subquery bernama MontlySales. Kemudian, hasil dari subquery ini dikelompokkan berdasarkan ID penjualan, nama pelanggan yang sudah dimodifikasi, dan bulan transaksi, serta menghitung total pengeluaran dengan mengalikan harga produk dengan jumlah yang terjual dan mengkonversinya menjadi varchar untuk ditampilkan dalam format yang diinginkan.

9. Create a view named ‘MostAndLessBoughtProductPerCustomer’ to display Sales ID (obtained from replacing ‘SH’ of SalesID with ‘Sales ID ’), ProductID (obtained from replacing ‘PT’ of ProductID with ‘Product ID ’), SalesDate (obtained from SalesDate formatted as ‘MON DD, YYYY’), Most Bought Product (obtained from maximum of Quantity), and Less Bought Product (obtained from minimum of Quantity) for every product that the product weight is more than 180 and the product price is more than 1500.p

CREATE VIEW MostAndLessBoughtProductPerCustomer AS

SELECT REPLACE(s.SaleID, 'SH', 'Sales ID ') AS [Sales ID],

REPLACE(p.ProductID, 'PT', 'Product ID ') AS [ProductID],

FORMAT(TransactionDate, 'MMM dd, yyyy') AS [SalesDate],

(SELECT MAX(Quantity) FROM SaleDetail WHERE ProductID = p.ProductID) AS [Most Bought Product],

(SELECT MIN(Quantity) FROM SaleDetail WHERE ProductID = p.ProductID) AS [Less Bought Product]

FROM Sale s

INNER JOIN SaleDetail sd ON s.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

WHERE ProductWeight > 180 AND ProductPrice > 1500

GROUP BY s.SaleID,

p.ProductID,

TransactionDate;

Penjelasan : Query ini membuat sebuah view bernama `MostAndLessBoughtProductPerCustomer` yang menyajikan data mengenai produk yang paling banyak dan paling sedikit dibeli oleh pelanggan berdasarkan kriteria tertentu. Dalam view ini, beberapa kolom ditampilkan: ID penjualan (dengan 'SH' diganti menjadi 'Sales ID '), ID produk (dengan 'PT' diganti menjadi 'Product ID '), tanggal transaksi dalam format 'MMM dd, yyyy', kuantitas maksimum dari produk yang dibeli, dan kuantitas minimum dari produk yang dibeli. Data diambil dari tabel Sale, SaleDetail, dan Product dengan hubungan melalui SaleID dan ProductID. Query ini hanya menyertakan produk yang memiliki berat lebih dari 180 dan harga lebih dari 1500. Data hasil seleksi ini kemudian dikelompokkan berdasarkan ID penjualan, ID produk, dan tanggal transaksi untuk menghasilkan tampilan yang diinginkan.

10. Create a view named ‘SpentAboveMaximumTotalSalesCustomerIn2023’ to display Customer Name (obtained from uppercase CustomerName), Total Transaction (obtained from total multiplication of ProductPrice and Quantity), and Max Total Transaction (obtained from max of multiplication ProductPrice and Quantity) for each Sales that occurred in 2023 and the Total multiplication of ProductPrice and Quantity is more than maximum of multiplication ProductPrice and Quantity.  
  
CREATE VIEW SpentAboveMaximumTotalSalesCustomerIn2023 AS

SELECT UPPER(CustomerName) AS [Customer Name],

SUM(ProductPrice\*Quantity) AS [Total Transaction],

MAX(ProductPrice\*Quantity) AS [Max Total Transaction]

FROM Sale s

INNER JOIN Customer c ON s.CustomerID = c.CustomerID

INNER JOIN SaleDetail sd ON s.SaleID = sd.SaleID

INNER JOIN Product p ON sd.ProductID = p.ProductID

WHERE YEAR(TransactionDate) = 2023

GROUP BY CustomerName

HAVING SUM(ProductPrice\*Quantity) > MAX(ProductPrice\*Quantity);

Penjelasan : Query ini membuat sebuah view bernama `SpentAboveMaximumTotalSalesCustomerIn2023` yang menampilkan pelanggan yang total transaksi penjualannya di tahun 2023 melebihi transaksi terbesar tunggal mereka dalam tahun yang sama. Dalam view ini, kolom yang ditampilkan meliputi nama pelanggan dalam huruf besar, total transaksi pelanggan, dan transaksi tunggal terbesar pelanggan. Data ini diperoleh dengan menggabungkan tabel Sale, Customer, SaleDetail, dan Product berdasarkan relasi CustomerID dan ProductID. Query ini hanya menyertakan transaksi dari tahun 2023, dan hasilnya dikelompokkan berdasarkan nama pelanggan. Setelah itu, filter `HAVING` memastikan hanya pelanggan yang total transaksinya lebih besar dari transaksi tunggal terbesar mereka yang akan ditampilkan dalam view tersebut.