Laporan Praktikum
Harvest Walukow
164231104

1. Masuk pada database classicmodels

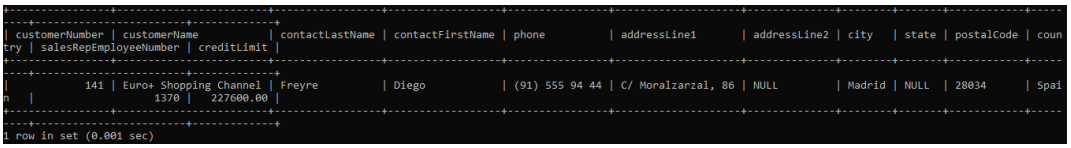
```
USE classicmodels;
```

2. VIEWS
   a. Data lengkap Customer yang paling berkontribusi dalam payment

```
CREATE VIEW customer_amount_terbanyak AS
SELECT * FROM customers
WHERE customers.customerNumber = (
       SELECT customerNumber FROM payments
       GROUP BY customerNumber
       ORDER BY SUM(amount) DESC
       LIMIT 1
);
```

Di buat view dari query untuk mengambil customerNumber yang amount payments-nya paling banyak, lalu ditampilkan semua data customer tersebut dari tabel customers.

Output:



   b. Tampilkan customer.city, dan nilai maksimal pembelian

```
CREATE VIEW city_amount_max AS
SELECT city, MAX(amount) FROM customers JOIN payments
ON customers.customerNumber = payments.customerNumber
GROUP BY city;
```

Dibuat view dari query yang menggabungkan tabel customers dan payments, kemudian ditampilkan nama city-nya dan nilai amount maksimum di tiap kota setelah dikelompokkan kota-nya.

Output:

city	MAX(amount)
Allentown	63357.13
Århus	53745.34
Auckland	75020.13
Barcelona	40473.86
Bergamo	52151.81

...

Tsawassen	37527.58
Vancouver	36527.61
Versailles	53116.99
Wellington	35034.57
White Plains	42339.76

77 rows in set (0.001 sec)

c. Tampilkan jumlah customer yang ditangani setiap employee

```
CREATE VIEW total_customer_employee AS
SELECT firstName, lastName, COUNT(customerNumber) FROM customers
JOIN employees
ON customers.salesRepEmployeeNumber = employees.employeeNumber
GROUP BY employeeNumber;
```

Dibuat view dari query yang menggabungkan tabel customers dan employees, lalu ditampilkan nama employee-nya dan jumlah customer yang ditangani setelah dilakukan pengelompokkan berdasarkan empoyeeNumber.

Output:

firstName	lastName	COUNT(customerNumber)
Leslie	Jennings	6
Leslie	Thompson	6
Julie	Firrelli	6
Steve	Patterson	6
Foon Yue	Tseng	7
George	Vanauf	8
Loui	Bondur	6
Gerard	Hernandez	7
Pamela	Castillo	10
Larry	Bott	8
Barry	Jones	9
Andy	Fixter	5
Peter	Marsh	5
Mami	Nishi	5
Martin	Gerard	6

15 rows in set (0.001 sec)

d. Tampilkan seluruh nama customer yang memiliki jumlah pembelian terkecil

```
CREATE VIEW nama_customer_amount_terkecil AS
SELECT customerName FROM customers WHERE customerNumber = (
       SELECT customerNumber FROM payments WHERE amount = (
              SELECT MIN(amount) FROM payments
       )
);
```

Dibuat view dari query yang menggunakan beberapa nested query, di mana query yang pertama untuk mengambil nilai amount terkecil dari tabel payments, nilai itu lalu digunakan untuk query kedua untuk mendapatkan customerNumber yang melakukan pembelian terkecil tersebut. Lalu, ditampilkan nama customer dengan pembelian terkecil tersebut.

Output:

```
+--------------------------+
| customerName             |
+--------------------------+
| Tokyo Collectables, Ltd |
+--------------------------+
1 row in set (0.001 sec)
```

3. STORED PROCEDURES
   a. Menambah stok produk dengan parameter productcode dan jumlah stok

```
DELIMITER //
CREATE PROCEDURE tambah_stok_produk(
    IN p_productCode VARCHAR(16),
    IN p_quantityInStock INT
)
BEGIN
    UPDATE products
    SET quantityInStock = quantityInStock + p_quantityInStock
    WHERE productCode = p_productCode;
END //

CALL tambah_stok_produk("S10_1678", 7);
```

Setelah query dijalankan terjadi penambahan stok untuk kode produk tersebut:

```
MariaDB [classicmodels]> SELECT productCode, productName, quantityInStock FROM products WHERE productCode = "S10_1678";
+-------------+-----------------------------------+-----------------+
| productCode | productName                       | quantityInStock |
+-------------+-----------------------------------+-----------------+
| S10_1678    | 1969 Harley Davidson Ultimate Chopper |            7940 |
+-------------+-----------------------------------+-----------------+
1 row in set (0.000 sec)
```

   b. Menambah Customer baru dengan parameter yang sesuai

```
DELIMITER //
CREATE PROCEDURE tambah_customer_baru(
    IN p_customerNumber INT,
```

```
    IN p_customerName VARCHAR(50),
    IN p_contactLastName VARCHAR(50),
    IN p_contactFirstName VARCHAR(50),
    IN p_phone VARCHAR(50),
    IN p_addressLine1 VARCHAR(50),
    IN p_addressLine2 VARCHAR(50),
    IN p_city VARCHAR(50),
    IN p_state VARCHAR(50),
    IN p_postalCode VARCHAR(15),
    IN p_country VARCHAR(50),
    IN p_salesRepEmployeeNumber INT,
    IN p_creditLimit DECIMAL(10,2)
)
BEGIN
    INSERT INTO customers (
        customerNumber, customerName, contactLastName,
        contactFirstName, phone, addressLine1, addressLine2, city,
        state, postalCode, country, salesRepEmployeeNumber, creditLimit
    )
    VALUES (
        p_customerNumber, p_customerName, p_contactLastName,
        p_contactFirstName, p_phone, p_addressLine1, p_addressLine2,
        p_city, p_state, p_postalCode, p_country,
        p_salesRepEmployeeNumber, p_creditLimit
    );
END //

CALL tambah_customer_baru(104, 'Tesla Inc.', 'Bro', 'Elon',
'08123456789', 'Jl. Mulyosari', NULL, 'Surabaya', NULL, '60123',
'Indonesia',1165, 50000.00);
```

Setelah query dijalankan:

```
MariaDB [classicmodels]> SELECT * FROM customers WHERE customerNumber = 104;
+----------------+-------------+----------------+-----------------+-------------+--------------+--------------+----------+-------+------------+----------+--------
| customerNumber | customerName | contactLastName | contactFirstName | phone      | addressLine1  | addressLine2 | city     | state | postalCode | country  | salesRep
EmployeeNumber | creditLimit |
+----------------+-------------+----------------+-----------------+-------------+--------------+--------------+----------+-------+------------+----------+--------
---------------+-------------+
|            104 | Tesla Inc.   | Bro            | Elon            | 08123456789 | Jl. Mulyosari | NULL         | Surabaya | NULL  | 60123      | Indonesia |
         1165 |    50000.00 |
+----------------+-------------+----------------+-----------------+-------------+--------------+--------------+----------+-------+------------+----------+--------
---------------+-------------+
1 row in set (0.000 sec)
```

c. Menambah payment baru dengan parameter yang sesuai

```
DELIMITER //
CREATE PROCEDURE tambah_payment_baru(
    IN p_customerNumber INT,
    IN p_checkNumber VARCHAR(50),
    IN p_paymentDate DATE,
    IN p_amount DECIMAL(10,2)
)
BEGIN
    INSERT INTO payments (
        customerNumber, checkNumber, paymentDate, amount
    )
    VALUES (
        p_customerNumber, p_checkNumber, p_paymentDate, p_amount
```

```
    );
END //

CALL tambah_payment_baru(104, 'HRVST29', '2005-29-08', 150000.00);
```

Setelah kueri dijalankan:

```
MariaDB [classicmodels]> SELECT * FROM payments WHERE customerNumber = 104;
+----------------+-------------+-------------+-----------+
| customerNumber | checkNumber | paymentDate | amount    |
+----------------+-------------+-------------+-----------+
|            104 | HRVST29     | 2005-08-29  | 150000.00 |
+----------------+-------------+-------------+-----------+
1 row in set (0.000 sec)
```

4. TRIGGERS
    a. Mengurangi stok produk tertentu setiap ada order masuk (sesuai kuantitas produk)

```
DELIMITER //
CREATE TRIGGER kurangi_stok_produk
AFTER INSERT ON orderdetails
FOR EACH ROW
BEGIN
    UPDATE products
    SET quantityInStock = quantityInStock - NEW.quantityOrdered
    WHERE productCode = NEW.productCode;
END //
```

Trigger di atas akan otomatis mengurangi stok produk pada tabel products setiap kali ada data order baru yang masuk ke tabel orderdetails. Jumlah pengurangan stok sesuai dengan jumlah produk yang dipesan (quantityOrdered).

    b. Membuat record baru dalam tabel baru (BarangHabis (productCode, dateHabis) setiap ada barang yang stoknya habis

```
CREATE TABLE BarangHabis (
    productCode VARCHAR(15) PRIMARY KEY,
    dateHabis DATE
);

DELIMITER //
CREATE TRIGGER insert_barang_habis
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    IF NEW.quantityInStock = 0 AND OLD.quantityInStock > 0 THEN
        INSERT INTO BarangHabis (productCode, dateHabis)
        VALUES (NEW.productCode, CURDATE());
    END IF;
END //
```

Trigger ini akan membuat baris baru pada tabel BarangHabis setiap kali stok suatu produk menjadi habis (dari sebelumnya ada stok menjadi 0). Tanggal habisnya stok dicatat juga.

c. Menghapus record dalam tabel BarangHabis ketika ada stok baru dalam sebuah produk tertentu

```
DELIMITER //
CREATE TRIGGER hapus_barang_habis
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    IF NEW.quantityInStock > 0 AND OLD.quantityInStock = 0 THEN
        DELETE FROM BarangHabis WHERE productCode = NEW.productCode;
    END IF;
END //
```

Trigger ini akan menghapus baris pada tabel BarangHabis jika stok produk yang sebelumnya habis bertambah lagi.