**Tugas Nested / Subquery**

Nama : Muhammad Nabil Afrizal Rahmadani

NRP : 5025231014

Kelas : Sistem Basis Data D

1. Section 4: Joining Tables
2. Table & Column Aliases
3. Query ini berfungsi untuk menggabungkan dua kolom berbeda yaitu lastName dan firstName, memberikan alias dengan nama Full name dan mengurutkan nya berdasarkan urutan alphabet

SELECT CONCAT\_WS(', ', lastName, firstname) `Full name`

FROM employees

ORDER BY `Full name`;

A screenshot of a phone

Description automatically generated

1. Query ini menampilkan dimana jumlah total dari order lebih dari 6000 dari tabel orderDetails. Query ini juga menggunakan GROUP BY untuk mengelompokkan berdasarkan “Order no.”

SELECT orderNumber `Order no.`,

    SUM(priceEach \* quantityOrdered) total

FROM orderDetails

GROUP BY `Order no.`

HAVING total > 60000;

A screenshot of a computer

Description automatically generated

1. Query ini memberikan alias kepada tabel employees dengan nama e

SELECT e.firstName,

    e.lastName

FROM employees e

ORDER BY e.firstName;

1. Query ini menampilkan informasi dari 2 tabel customers yang di aliaskan sebagai c dan tabel orders yang di aliaskan sebagai o, query ini melakukan INNER JOIN sehingga menyambungkan tabel orders dan customers dengan customerNumber yang sama. Query ini akan menampilkan nama customer dan total order yang telah dilakukan oleh customer itu dan diurutkan berdasarkan total.

SELECT

    customerName,

    COUNT(o.orderNumber) total

FROM

    customers c

INNER JOIN orders o ON c.customerNumber = o.customerNumber

GROUP BY

    customerName

ORDER BY

    total DESC;



1. INNER JOINS
2. Query ini berfungsi menampilkan data productCode, productName dan textDesc dari 2 tabel berbeda, karena textDesc berada di tabel berbeda, maka dilakukan INNER JOIN berdasarkan foreign key nya yaitu productLine sehingga dapat ditampilkan data nya.

SELECT

    productCode,

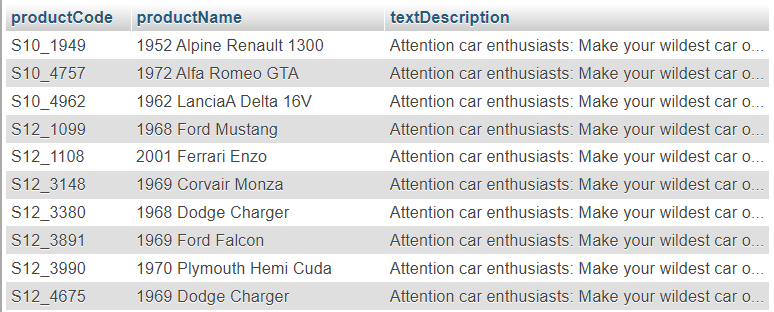
    productName,

    textDescription

FROM

    products

INNER JOIN productlines USING (productline);



1. Query ini berfungsi menampilkan data orderNumber, status, dan total nya. Karena quantityOrdered dan priceEach berada di tabel yang berbeda, dilakukan INNER JOIN berdasarkan orderNumber sehingga dan data di GROUP BY berdasarkan orderNumber nya lalu ditampilkan

SELECT

    orderNumber,

    status,

    SUM(quantityOrdered \* priceEach) total

FROM

    orders

INNER JOIN orderdetails USING (orderNumber)

GROUP BY orderNumber;

A screenshot of a computer

Description automatically generated

1. Query ini menampilkan orderNumber, orderDate, customerName, orderLineNumber, productName, quantityOrdered dan priceEach. Dari tabel order dilakukan INNER JOIN dengan tabel orderdetails menggunakan foreign key nya orderNumber, lalu dilakukan lagi INNER JOIN dengan tabel products menggunakan foreign key nya yaitu productCode, lalu dilakukan INNER JOIN lagi dengan tabel customers menggunakan foreign key nya yaitu customerNumber yang di dapat dari tabel orders

SELECT

    orderNumber,

    orderDate,

    customerName,

    orderLineNumber,

    productName,

    quantityOrdered,

    priceEach

FROM

    orders

INNER JOIN orderdetails

    USING (orderNumber)

INNER JOIN products

    USING (productCode)

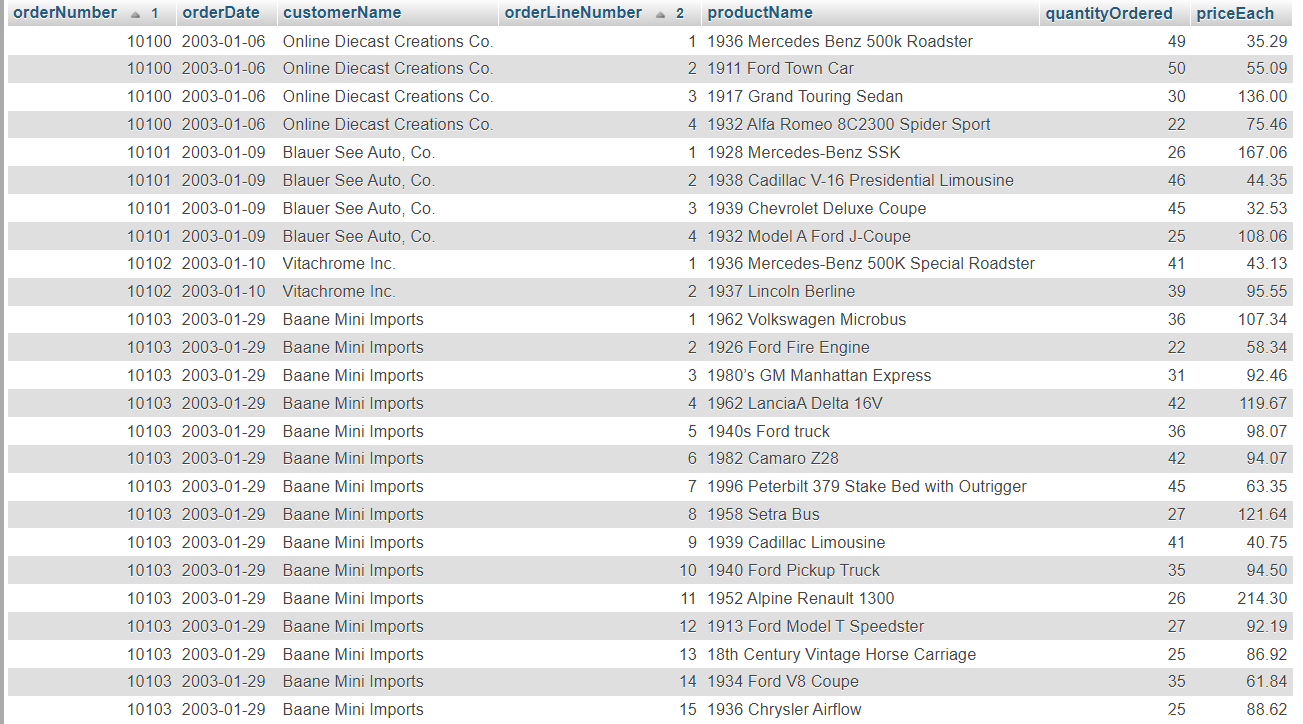
INNER JOIN customers

    USING (customerNumber)

ORDER BY

    orderNumber,

    orderLineNumber;



1. Menampilkan data orderNumber, productName, msrp, priceEach dari productCode “S10\_1678”. Dari tabel products di INNER JOIN dengan tabel orderdetails, tetapi hanya tampilkan dimana msrp > priceEach

SELECT

    orderNumber,

    productName,

    msrp,

    priceEach

FROM

    products p

INNER JOIN orderdetails o

   ON p.productcode = o.productcode

      AND p.msrp > o.priceEach

WHERE

    p.productcode = 'S10\_1678';

1. LEFT JOIN
2. Query ini menampilkan data customerNumber, customerName, orderNumber dan status dari tabel customers lalu di LEFT JOIN dengan tabel orders menggunakan customerNumber. Query ini akan menampilkan semua data customers, meskipun customer tersebut belum pernah melakukan order.

SELECT

    customerNumber,

    customerName,

    orderNumber,

    status

FROM

    customers

LEFT JOIN orders USING (customerNumber);

A screenshot of a website

Description automatically generated

1. Query ini menampilkan data customerNumber, customerName, orderNumber dan status dari tabel customers lalu di LEFT JOIN dengan tabel orders menggunakan customerNumber. Query ini akan menampilkan data dimana customer tersebut tidak pernah melakukan order.

SELECT

    c.customerNumber,

    c.customerName,

    o.orderNumber,

    o.status

FROM

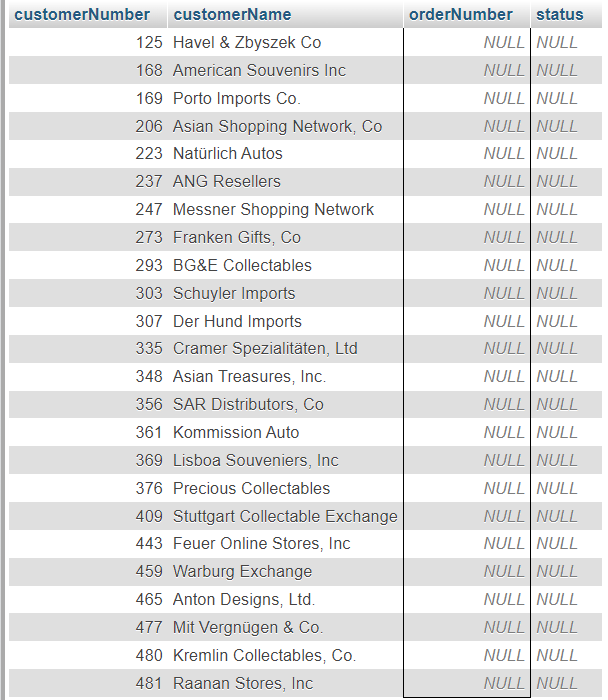
    customers c

LEFT JOIN orders o

    ON c.customerNumber = o.customerNumber

WHERE

    orderNumber IS NULL;



1. WHERE vs ON
   1. Pada query ini data dari orders di LEFT JOIN dengan data dari orderDetails dengan orderNumber 10123.

SELECT

    o.orderNumber,

    customerNumber,

    productCode

FROM

    orders o

LEFT JOIN orderDetails

    USING (orderNumber)

WHERE

    orderNumber = 10123;

A screenshot of a computer

Description automatically generated

* 1. Pada query ini data dari orders di LEFT JOIN dengan data dari orderDetails, tetapi hanya data dengan orderNumber 10123 yang akan menampilkan product code nya.

A screenshot of a number

Description automatically generated

1. RIGHT JOIN
2. Query ini menggabungkan table customer dengan employee menggunakan RIGHT JOIN dengan salesRepEmployeeNumber dari tabel customer = employeeNumber dari tabel employee lalu diurutkan berdasarkan employeeNumber.

SELECT

    employeeNumber,

    customerNumber

FROM

    customers

RIGHT JOIN employees

    ON salesRepEmployeeNumber = employeeNumber

ORDER BY

    employeeNumber;

A screenshot of a number

Description automatically generated

1. Query ini menampilkan data employee yang tidak pernah berinteraksi dengan customer

SELECT

    employeeNumber,

    customerNumber

FROM

    customers

RIGHT JOIN employees ON

    salesRepEmployeeNumber = employeeNumber

WHERE customerNumber is NULL

ORDER BY employeeNumber;

A screenshot of a computer

Description automatically generated

1. SELF JOIN
2. Query ini menampilkan data manager dan bawahan nya (direct report), menggunakan foreign key reportsTo dengan INNER JOIN

SELECT

    CONCAT(m.lastName, ', ', m.firstName) AS Manager,

    CONCAT(e.lastName, ', ', e.firstName) AS 'Direct report'

FROM

    employees e

INNER JOIN employees m ON

    m.employeeNumber = e.reportsTo

ORDER BY

    Manager;

A table with names on it

Description automatically generated

1. Query ini menampilkan data manager dan bawahan nya menggunakan LEFT JOIN jika reportsTo adalah NULL maka manager akan terisi dengan Top Manager.

SELECT

    IFNULL(CONCAT(m.lastname, ', ', m.firstname),

            'Top Manager') AS 'Manager',

    CONCAT(e.lastname, ', ', e.firstname) AS 'Direct report'

FROM

    employees e

LEFT JOIN employees m ON

    m.employeeNumber = e.reportsto

ORDER BY

    manager DESC;

A screenshot of a phone

Description automatically generated

1. Query ini berguna untuk membandingkan row berurutan dari satu tabel yang sama. Dalam konteks ini, jika kota nya sama maka tampilkan lalu di ORDER BY city

SELECT

    c1.city,

    c1.customerName,

    c2.customerName

FROM

    customers c1

INNER JOIN customers c2 ON

    c1.city = c2.city

    AND c1.customername > c2.customerName

ORDER BY

    c1.city;

A screenshot of a computer

Description automatically generated

1. CROSS JOIN
2. Pada tabel ini diberikan sebuah database *playing cards*. Agar semua symbol dapat mendapat semua ranks. Dilakukan CROSS JOIN.

SELECT

  suit\_name,

  rank\_name

FROM

  suits CROSS

  JOIN ranks

ORDER BY

  suit\_name,

  rank\_name;



1. Diberikan sebuah database bernama salesdb, pertama dilakukan INNER JOIN products dengan sales berdasarkan products id nya. Lalu INNER JOIN dilakukan lagi antara stores dengan sales berdasarkan id nya. Lalu di group berdasarkan store\_name dan product name. Lalu untuk menemukan toko yang tidak menjual suatu barang dilakukan CROSS JOIN untuk mendapatkan semua data lalu di filter dengan IFNULL.

SELECT

    b.store\_name,

    a.product\_name,

    IFNULL(c.revenue, 0) AS revenue

FROM

    products AS a

        CROSS JOIN

    stores AS b

        LEFT JOIN

    (SELECT

        stores.id AS store\_id,

        products.id AS product\_id,

        store\_name,

            product\_name,

            ROUND(SUM(quantity \* price), 0) AS revenue

    FROM

        sales

    INNER JOIN products ON products.id = sales.product\_id

    INNER JOIN stores ON stores.id = sales.store\_id

    GROUP BY stores.id, products.id, store\_name , product\_name) AS c ON c.store\_id = b.id

        AND c.product\_id= a.id

ORDER BY b.store\_name;

