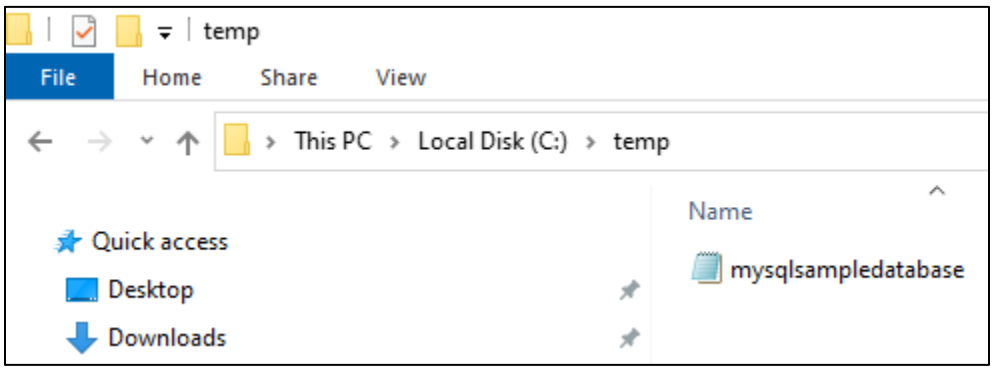
PRAKTEK DBMS LANJUT
MINGGU 3-MySQL VIEW LANJUTAN

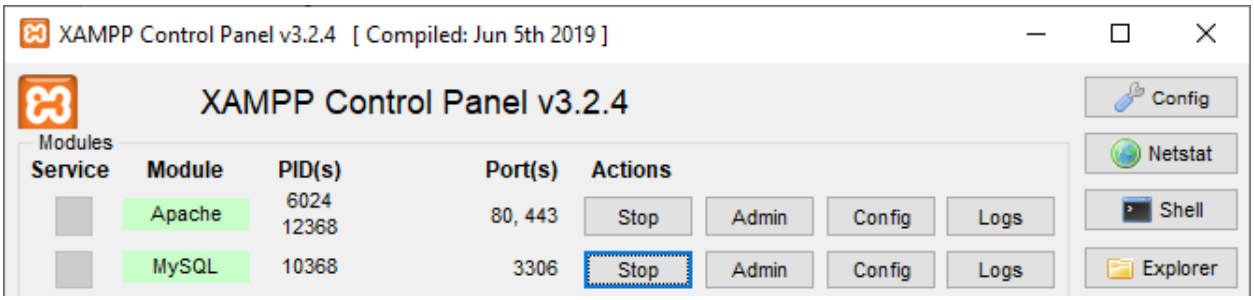
Tahap 1-Import Database

Download contoh database `classicmodels` dari link di bawah ini:
https://www.mysqltutorial.org/mysql-sample-database.aspx

Buat folder dengan nama temp di drive C, unzip file **mysqlsampledatabase.rar** ke folder temp.

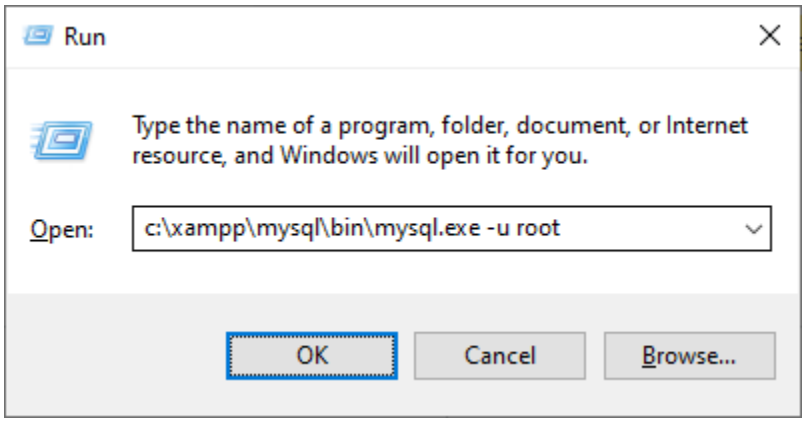


Jalankan XAMPP Control Panel

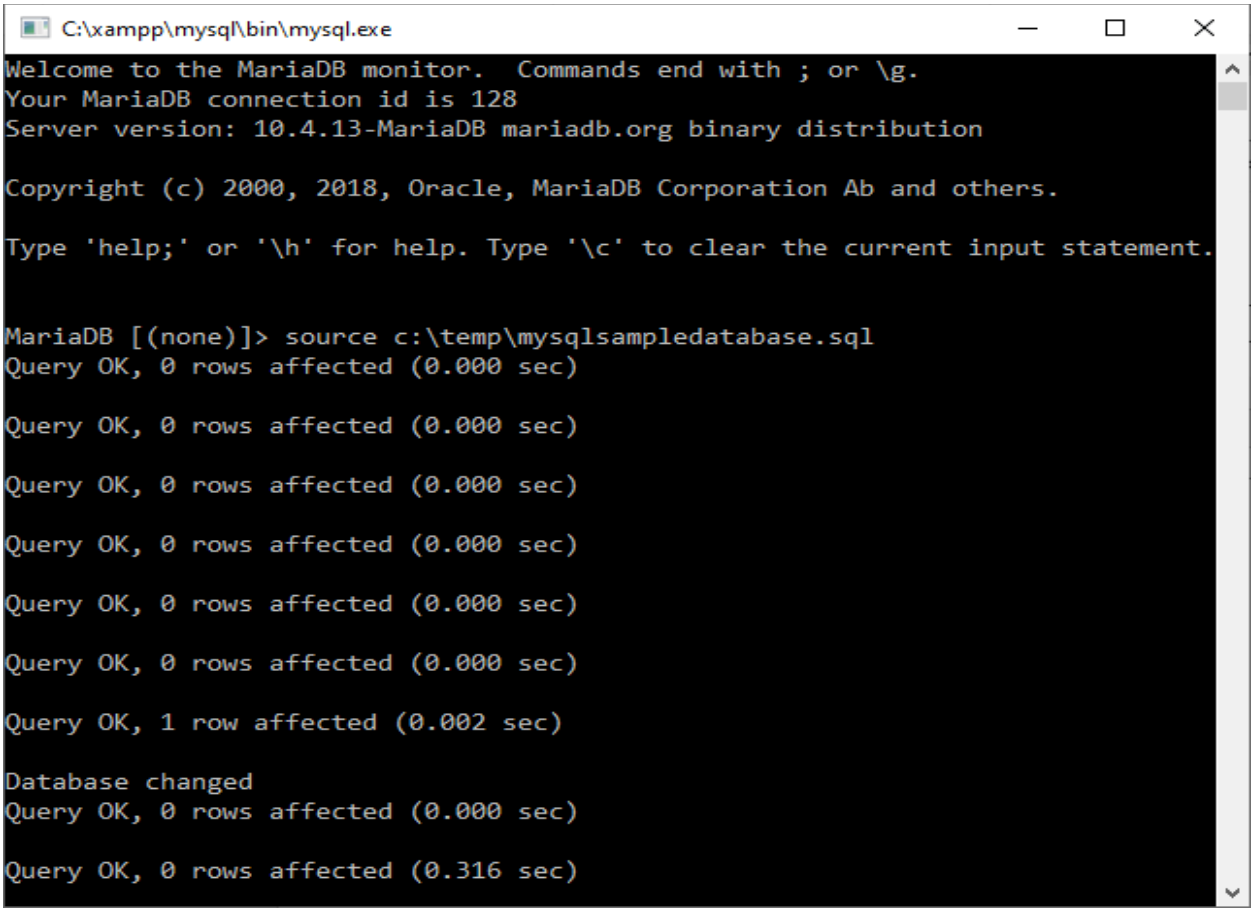


Gunakan phpmyadmin untuk meng-import database **mysqlsampledatabase.sql**

Atau gunakan MySQL Clients
Tekan tombol Windows+R, ketik perintah berikut:



Klik OK.
Ketik perintah untuk import database
`source c:\temp\mysqlsampledatabase.sql`



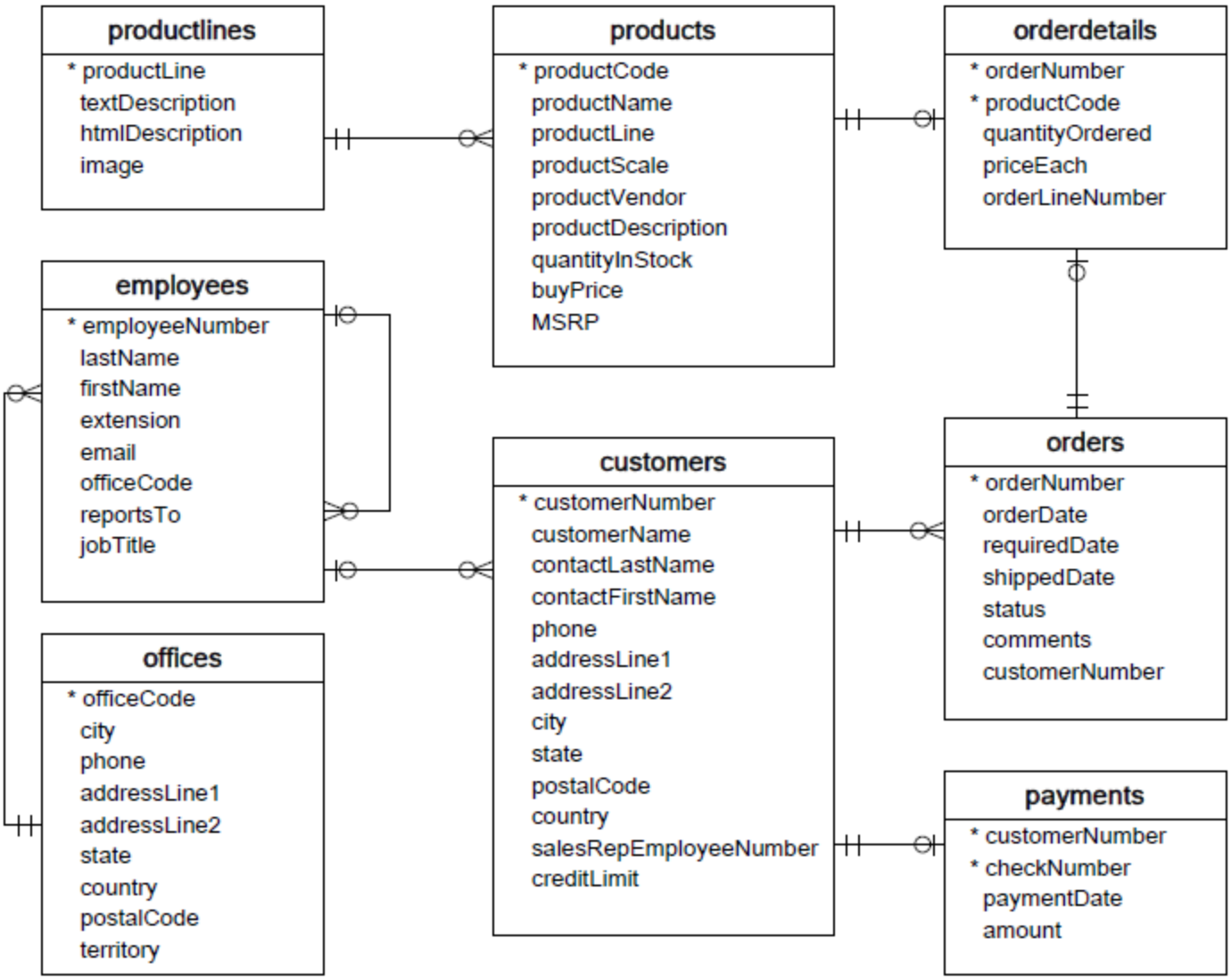
Ketikkan perintah show tables untuk menampilkan struktur tables yang ada

di database.

```
MariaDB [classicmodels]> show tables;
+-------------------------+
| Tables_in_classicmodels |
+-------------------------+
| customers               |
| employees               |
| offices                 |
| orderdetails            |
| orders                  |
| payments                |
| productlines            |
| products                |
+-------------------------+
8 rows in set (0.001 sec)
```

The MySQL sample database schema consists of the following tables:

- **Customers**: stores customer's data.
- **Products**: stores a list of scale model cars.
- **ProductLines**: stores a list of product line categories.
- **Orders**: stores sales orders placed by customers.
- **OrderDetails**: stores sales order line items for each sales order.
- **Payments**: stores payments made by customers based on their accounts.
- **Employees**: stores all employee information as well as the organization structure such as who reports to whom.
- **Offices**: stores sales office data.



Tahap 2- MySQL CREATE VIEW

Introduction to MySQL CREATE VIEW statement

The `CREATE VIEW` statement creates a new view in the database. Here is the basic syntax of the `CREATE VIEW` statement:

```
CREATE [OR REPLACE] VIEW [db_name.]view_name [(column_list)]
AS
  select-statement;
```

In this syntax:

First, specify the name of the view that you want to create after the `CREATE VIEW` keywords. The name of the view is unique in a database. Because views

and tables in the same database share the same namespace, the name a view cannot the same as the name of an existing table.

Second, use the `OR REPLACE` option if you want to replace an existing view if the view already exists. If the view does not exist, the `OR REPLACE` has no effect.

Third, specify a list of columns for the view. By default, the columns of the view are derived from the select list of the `SELECT` statement. However, you can explicitly specify the column list for the view by listing them in parentheses following the view name.

Finally, specify a `SELECT` statement that defines the view. The `SELECT` statement can query data from tables or views. MySQL allows you to use the `ORDER BY` clause in the `SELECT` statement but ignores it if you select from the view with a query that has its own `ORDER BY` clause.

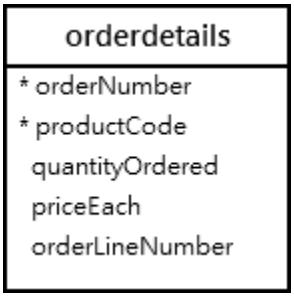
By default, the `CREATE VIEW` statement creates a view in the current database. If you want to explicitly create a view in a given database, you can qualify the view name with the database name.

MySQL CREATE VIEW examples

Let's take some example of using the `CREATE VIEW` statement to create new views.

1) Creating a simple view example

Let's take a look at the `orderDetails` table from the sample database:



This statement uses the `CREATE VIEW` statement to create a view that represents total sales per order.

```
CREATE VIEW salePerOrder AS
    SELECT
        orderNumber,
        SUM(quantityOrdered * priceEach) total
    FROM
        orderDetails
    GROUP by orderNumber
    ORDER BY total DESC;
```

If you use the `SHOW TABLE` command to view all tables in the `classicmodels` database, you will see the view `salesPerOrder` is showing up in the list.

```
SHOW TABLES;
```

	Tables_in_classicmodels
▶	customers
	employees
	offices
	orderdetails
	orders
	payments
	productlines
	products
	saleperorder

This is because the views and tables share the same namespace as mentioned earlier.

To know which object is a view or table, you use the `SHOW FULL TABLES` command as follows:

```
SHOW FULL TABLES;
```

	Tables_in_classicmodels	Table_type
▶	customers	BASE TABLE
	employees	BASE TABLE
	offices	BASE TABLE
	orderdetails	BASE TABLE
	orders	BASE TABLE
	payments	BASE TABLE
	productlines	BASE TABLE
	products	BASE TABLE
	saleperorder	VIEW

The `table_type` column in the result set specifies the type of the object: view or table (base table).

If you want to query total sales for each sales order, you just need to execute a simple `SELECT` statement against the `SalePerOrder` view as follows:

```
SELECT * FROM salePerOrder;
```

orderNumber	total
10165	67392.85
10287	61402.00
10310	61234.67
10212	59830.55
10207	59265.14
10127	58841.35
10204	58793.53
10126	57131.92
10222	56822.65
10142	56052.56
10390	55907.50

2) Creating a view based on another view example

MySQL allows you to create a view based on another view.

For example, you can create a view called `bigSalesOrder` based on the `salesPerOrder` view to show every sales order whose total is greater than `60,000` as follows:

```
CREATE VIEW bigSalesOrder AS
    SELECT
        orderNumber,
        ROUND(total,2) as total
    FROM
        salePerOrder
    WHERE
        total > 60000;
```

Now, you can query the data from the `bigSalesOrder` view as follows:

```
SELECT
    orderNumber,
    total
FROM
    bigSalesOrder;
```

orderNumber	total
10165	67392.85
10287	61402.00
10310	61234.67

3) Creating a view with join example

The following example uses the `CREATE VIEW` statement to create a view based on multiple tables. It uses the `INNER JOIN` clauses to join tables.

```
CREATE OR REPLACE VIEW customerOrders AS
SELECT
```

```
    orderNumber,
    customerName,
    SUM(quantityOrdered * priceEach) total
FROM
    orderDetails
INNER JOIN orders o USING (orderNumber)
INNER JOIN customers USING (customerNumber)
GROUP BY orderNumber;
```

This statement selects data from the `customerOrders` view:

```
SELECT * FROM customerOrders
ORDER BY total DESC;
```

This picture shows the partial output:

orderNumber	customerName	total
10165	Dragon Souveniers, Ltd.	67392.85
10287	Vida Sport, Ltd	61402.00
10310	Toms Spezialitäten, Ltd	61234.67
10212	Euro+ Shopping Channel	59830.55
10207	Diecast Collectables	59265.14
10127	Muscle Machine Inc	58841.35
10204	Muscle Machine Inc	58793.53
10126	Corrida Auto Replicas, Ltd	57131.92
10222	Collectable Mini Designs Co.	56822.65
10142	Mini Gifts Distributors Ltd.	56052.56
10390	Mini Gifts Distributors Ltd.	55902.50

4) Creating a view with a subquery example

The following example uses the `CREATE VIEW` statement to create a view whose `SELECT` statement uses a subquery. The view contains products whose buy prices are higher than the average price of all products.

```
CREATE VIEW aboveAvgProducts AS
    SELECT
        productCode,
        productName,
        buyPrice
    FROM
        products
    WHERE
        buyPrice > (
            SELECT
                AVG(buyPrice)
            FROM
                products)
    ORDER BY buyPrice DESC;
```

This query data from the `aboveAvgProducts` is simple as follows:

```
SELECT * FROM aboveAvgProducts;
```

5) Creating a view with explicit view columns example

This statement uses the `CREATE VIEW` statement to create a new view based on the customers and orders tables with explicit view columns:

```
CREATE VIEW customerOrderStats (
   customerName ,
   orderCount
)
AS
    SELECT
        customerName,
        COUNT(orderNumber)
    FROM
        customers
            INNER JOIN
        orders USING (customerNumber)
    GROUP BY customerName;
```

This query returns data from the `customerOrderStats` view:

```
SELECT
    customerName,
    orderCount
FROM
    customerOrderStats
ORDER BY
      orderCount,
    customerName;
```

	customerName	orderCount
▶	Bavarian Collectables Imports, Co.	1
	Amica Models & Co.	2
	Auto Associés & Cie.	2
	Boards & Toys Co.	2
	CAF Imports	2
	Cambridge Collectables Co.	2
	Canadian Gift Exchange Network	2
	Classic Gift Ideas, Inc	2
	Clover Collections, Co.	2
	Collectable Mini Designs Co.	2
	Daedalus Designs Imports	2

TUGAS MINGGU 3:

Screenshot setiap tahapan dan hasilnya untuk mempraktekkan materi di atas, upload file dokumentasi ke elearning, nama file: **DBMSL-20211-REG-KELAS-M3-NOABSEN-NAMA.docx**