## **Step 1: Create a Database**

## CREATE DATABASE my\_database;

## **Step 2: Use the Database**

## Before creating tables, you need to select the database.

## USE my\_database;

## **Step 3: Create a Table**

## Now, you can create a table within the database. Below is an example of creating a users table with some common columns.

## CREATE TABLE users (

## id INT AUTO\_INCREMENT PRIMARY KEY,

## username VARCHAR(50) NOT NULL,

## City VARCHAR(50) NOT NULL,

## password VARCHAR(255) NOT NULL,

## created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

## );

## **Explanation:**

## **id**: An integer that automatically increments with each new record.

## **username**: A string with a maximum of 50 characters, cannot be null.

## **email**: A string with a maximum of 100 characters, cannot be null.

## **password**: A string with a maximum of 255 characters, cannot be null.

## **created\_at**: A timestamp that defaults to the current date and time when the record is created.

## INSERT INTO Example

The INSERT INTO statement is used to insert new records in a table.

The following SQL statement inserts a new record in the "Customers" table:

### **Example**

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

insert into customers (customer\_id ,customer\_fname , customer\_lname )values (1000521 , 'mahmoud' ,'khorshed' );

SELECT \* FROM customers WHERE Customer\_ID=1000521;

## UPDATE Table

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.

### **Example**

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'  
WHERE CustomerID = 1;

## The SQL DELETE Statement

The DELETE statement is used to delete existing records in a table.

### **DELETE Syntax**

DELETE FROM table\_name WHERE condition;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

Delete All Records

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM Customers;

## Delete a Table

To delete the table completely, use the DROP TABLE statement:

### **Example**

Remove the Customers table:

DROP TABLE Customers;

## The SQL MIN() and MAX() Functions

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

### **MIN Example**

Find the lowest price:

SELECT MIN(Price)  
FROM Products;

### **MAX Example**

Find the highest price:

SELECT MAX(Price)  
FROM Products;

## Set Column Name (Alias)

When you use MIN() or MAX(), the returned column will be named MIN(field) or MAX(field) by default. To give the column a new name, use the AS keyword:

### **Example**

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

Give the counted column a name by using the AS keyword.

### **Example**

Name the column "number of records":

SELECT COUNT(\*) AS [number of records]  
FROM Products;

### **Example**

Find the number of products where Price is higher than 20:

SELECT COUNT(ProductID)  
FROM Products  
WHERE Price > 20;

### **Example**

Insert a NULL value to better understand the example above:

UPDATE Products  
SET ProductName = NULL  
WHERE ProductName = 'Chang';

**Sum Example**

Return the number of orders made for the product with ProductID 11:

SELECT SUM(Quantity) as total  
FROM OrderDetails  
WHERE ProdictId = 11;

## SUM() With an Expression

The parameter inside the SUM() function can also be an expression.

If we assume that each product in the OrderDetails column costs 10 dollars, we can find the total earnings in dollars by multiply each quantity with 10:

### **Example**

Use an expression inside the SUM() parenthesis:

SELECT SUM(Quantity \* 10)  
FROM OrderDetails;

select order\_item\_product\_id , sum(order\_item\_product\_price ) from order\_items group by order\_item\_product\_id order by sum(order\_item\_product\_price) desc;

## The SQL AVG() Function

The AVG() function returns the average value of a numeric column.

### **Example**

Find the average price of all products:

SELECT AVG(Price) as average\_price  
FROM Products  
WHERE CategoryID = 1;

## Higher Than Average

To list all records with a higher price than average, we can use the AVG() function in a sub query:

### **Example**

Return all products with a higher price than the average price:

SELECT \* FROM Products  
WHERE price > (SELECT AVG(price) FROM Products);

select order\_item\_product\_id , order\_item\_product\_price from order\_items where order\_item\_product\_price > (select AVG(order\_item\_product\_price) from order\_items)

The SQL LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern;

|  |  |
| --- | --- |
| **LIKE Operator** | **Description** |
| WHERE CustomerName LIKE 'a%' | Finds any values that start with "a" |
| WHERE CustomerName LIKE '%a' | Finds any values that end with "a" |
| WHERE CustomerName LIKE '%or%' | Finds any values that have "or" in any position |
| WHERE CustomerName LIKE '\_r%' | Finds any values that have "r" in the second position |
| WHERE CustomerName LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| WHERE CustomerName LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| WHERE ContactName LIKE 'a%o' | Finds any values that start with "a" and ends with "o" |

The following SQL statement selects all customers with a CustomerName starting with "a":

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

## Using the [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

### **Example**

SELECT \* FROM Customers  
WHERE City LIKE '[bsp]%';

## The SQL IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

SELECT \* FROM Customers  
WHERE Country IN ('Germany', 'France', 'UK');

The following SQL statement selects all customers that are from the same countries as the suppliers:

### **Example**

SELECT \* FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);

## BETWEEN Example

### **Example**

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20;

## BETWEEN Dates Example

The following SQL statement selects all orders with an OrderDate between '01-July-1996' and '31-July-1996':

### **Example**

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN #07/01/1996# AND #07/31/1996#;

select order\_date,dayname(order\_date),count(order\_date) from orders where order\_date between '2014-07-07' and '2014-07-13' group by order\_date ;

select order\_date,count(order\_date) AS COUNT\_Orders ,dayname(order\_date) from orders where order\_date between '2014-07-07' and '2014-07-13' group by order\_date order by COUNT\_Orders DESC;

## SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the "Orders" table:

|  |  |  |
| --- | --- | --- |
| **OrderID** | **CustomerID** | **OrderDate** |
| 10308 | 2­­ | 1996-09-18 |
| 10309 | 37 | 1996-09-19 |
| 10310 | 77 | 1996-09-20 |

Then, look at a selection from the "Customers" table:

|  |  |  |  |
| --- | --- | --- | --- |
| **CustomerID** | **CustomerName** | **ContactName** | **Country** |
| 1 | Alfreds Futterkiste | Maria Anders | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mexico |

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

Trials:

Logical error

### **Example**

SELECT orders.order\_id, customers.customer\_fname, orders.order\_date  
FROM orders  
INNER JOIN customers

­ON orders.order\_customer\_id = customers.customer\_id;

### **Example**

Join OrderDetails with Products, and use SUM() to find the total amount:

SELECT SUM(Price \* Quantity)  
FROM OrderDetails  
LEFT JOIN Products

ON OrderDetails.ProductID = Products.ProductID;

SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Customers AS c, Orders AS o  
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;

select sum(o.order\_item\_quantity \* p.product\_price),p.product\_name from order\_items as o

left join products as p on o.order\_item\_product\_id = p.product\_id

where p.product\_price > 100

group by p.product\_name limit 20;