Database fundamentals
Part 2

Agenda

- What is SQL?
- What Can SQL do?
- SQL Syntax.
- DDL Commands
- DML Commands

What is SQL?

- SQL stands for **S**tructured **Q**uery **L**anguage.
- SQL lets you access and manipulate databases.
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.

What Can SQL do?

- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can insert records in a database.
- SQL can update records in a database.
- SQL can delete records from a database.
- SQL can create new databases.
- SQL can create new tables in a database.
- SQL can create stored procedures in a database.
- SQL can create views in a database.
- SQL can set permissions on tables, procedures, and views.

Is SQL Standard ?

- SQL is a Standard - BUT....
- Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.
- However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

Using SQL in Your Web Site

- To build a web site that shows data from a database, you will need:
- An RDBMS database program (i.e., MS Access, SQL Server, MySQL):
  - To use a server-side scripting language, like PHP or ASP.
  - To use SQL to get the data you want.
  - To use HTML / CSS to style the page.
- What is RDBMS?
  - RDBMS stands for Relational Database Management System.
  - RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
  - The data in RDBMS is stored in database objects called tables.
  - A table is a collection of related data entries, and it consists of columns and rows.

SQL Syntax

Database Tables

- A database most often contains one or more tables.
- Each table is identified by a name (e.g., "Customers" or "Orders").
- Tables contain records (rows) with data.

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

- The table above contains five records (one for each customer) and seven columns (CustomerID, CustomerName, ContactName, Address, City, PostalCode, and Country).

SQL Statements

- Most of the actions you need to perform on a database are done with SQL statements.
- The following SQL statement selects all the records in the "Customers" table:
- `SELECT * FROM Customers;`
- Now ; we will teach you all about the different SQL statements.
- Keep in Mind That...
  - SQL keywords are NOT case sensitive: **select** is the same as **SELECT**
- Semicolon after SQL Statements?
  - Some database systems require a semicolon at the end of each SQL statement.
  - Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

The Most Important SQL Commands

- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database

SQL Create Database Statement

- To create a new database in SQL, you typically use the **CREATE DATABASE** statement. Here's the basic syntax:

```
CREATE DATABASE database_name;
```

- Where database_name is the name you want to give to your new database.

- Example:-

```
CREATE DATABASE my_database;
```

SQL Drop Database Statement

- To drop a new database in SQL, you typically use the **DROP DATABASE** statement. Here's the basic syntax:

```
DROP DATABASE database_name;
```

- Where database_name is the name of the database you want to delete.

- Example:-

```
DROP DATABASE my_database;
```

SQL Create Table Command

- The **CREATE TABLE** command in SQL is used to create a new table in a relational database.

  Here's the basic syntax:

```
CREATE TABLE table_name (
    column1 datatype [constraints],
    column2 datatype [constraints],
    ...
    columnN datatype [constraints]
);
```

- table_name is the name of the table you want to create.
- column1, column2, ..., columnN are the names of the columns in the table.
- datatype specifies the data type for each column.
- [constraints] are optional constraints that can be applied to each column, such as NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, etc.

SQL Create Table Command Example

```sql
CREATE TABLE users (
  id INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT,
  email VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  phone_number VARCHAR(15),
  created TIMESTAMP NOT NULL DEFAULT NOW()
);

```

SQL Drop Table Command

- The **DROP TABLE** statement in SQL is used to delete an existing table and all its data from the database. Here's the basic syntax:

```
DROP TABLE table_name;
```

- Where table_name is the name of the table you want to delete.
- It's important to exercise caution when using DROP TABLE because it permanently deletes the table and all its data, and this action cannot be undone.

- Example:-

```
DROP TABLE employees;
```

SQL Alter Table Command

- The **ALTER TABLE** statement in SQL is used to modify an existing table structure. It can be used to add, modify, or drop columns, as well as add or drop constraints.
- Here's the basic syntax:

```
ALTER TABLE table_name
    action;
```

- Where table_name is the name of the table you want to modify, and action specifies the alteration you want to perform. Some common actions include:

SQL Alter Table Command

Some common actions include:

1. **Adding a column:**

```sql
ALTER TABLE table_name
ADD column_name datatype [constraints];
```

2. **Dropping a column:**

```sql
ALTER TABLE table_name
DROP COLUMN column_name;
```

3. **Modifying a column** (for example, changing its data type):

```sql
ALTER TABLE table_name
MODIFY COLUMN column_name new_datatype;
```

4. **Adding a constraint:**

```sql
ALTER TABLE table_name
ADD CONSTRAINT constraint_name constraint_definition;
```

5. **Dropping a constraint:**

```sql
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

SQL Alter Table Command Example

- Adding a new column `age` of type `INT` to the table `employees`:

```sql
ALTER TABLE employees
ADD age INT;
```

- Dropping the column `age` from the table `employees`:

```sql
ALTER TABLE employees
DROP COLUMN age;
```

- Modifying the data type of the column `email` from `VARCHAR(100)` to `VARCHAR(255)` in the table `employees`:

```sql
ALTER TABLE employees
MODIFY COLUMN email VARCHAR(255);
```

- Adding a new constraint named `fk_department_id` as a foreign key constraint in the table `employees`, referencing the `department_id` column in the `departments` table:

```sql
ALTER TABLE employees
ADD CONSTRAINT fk_department_id
FOREIGN KEY (department_id)
REFERENCES departments (department_id);
```

- Dropping the constraint named `fk_department_id` from the table `employees`:

```sql
ALTER TABLE employees
DROP CONSTRAINT fk_department_id;
```

SQL Rename Table Command

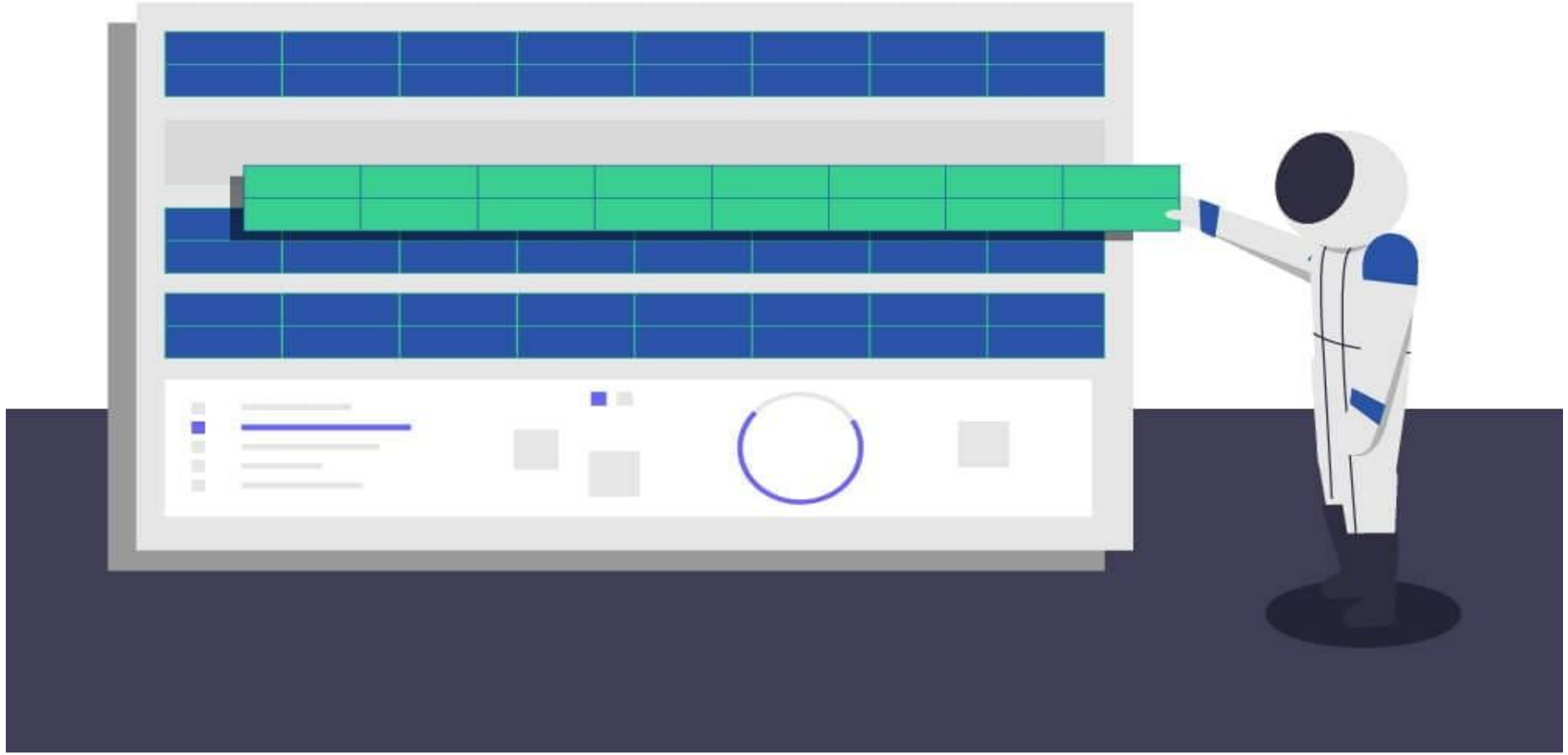
- To rename a table in SQL, you typically use the **ALTER TABLE** statement with the **RENAME TO** clause.
- Here's the basic syntax:

```
ALTER TABLE old_table_name
RENAME [redacted] new_table_name;
```

- OR

```
RENAME TABLE old_table_name To new_table_name;
```

INSERT INTO Statement



SQL INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- INSERT INTO Syntax:
- It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SQL INSERT INTO Statement example

- Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

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

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

Insert Data Only in Specified Columns

- It is also possible to only insert data in specific columns.
- The following SQL statement will insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway’);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

SQL NULL Values

- What is a NULL Value?
  - A field with a NULL value is a field with no value.
  - If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
- Note:
  - A NULL value is different from a zero value or a field that contains spaces.
  - A field with a NULL value is one that has been left blank during record creation!
- How to Test for NULL Values?
  - It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
  - We will have to use the IS NULL and IS NOT NULL operators instead.

SQL NULL Values…

- IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

- IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

SQL NULL operators … examples

- The IS NULL Operator
  - The IS NULL operator is used to test for empty values (NULL values).
  - The following SQL lists all customers with a NULL value in the "Address" field:

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

Result:

No result.

Your Database:

Tablename	Records
Customers	91
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

SQL NULL operators … examples

- The IS NOT NULL Operator
- The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).
- The following SQL lists all customers with a value in the "Address" field:

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;
```

Result:

Number of Records: 91

CustomerName	ContactName	Address
Alfreds Futterkiste	Maria Anders	Obere Str. 57
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312

SQL UPDATE Statement

SQL UPDATE Statement

- The UPDATE statement is used to modify the existing records in a table.
- UPDATE Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Note :

- The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

SQL UPDATE Statement
example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

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

UPDATE Multiple Records

Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

- It is the WHERE clause that determines how many records will be updated.
- The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico

SQL DELETE Statement

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

```
DELETE FROM table_name WHERE condition;
```

- Note:
  - Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement.
  - The WHERE clause specifies which record(s) should be deleted.
  - If you omit the WHERE clause, all records in the table will be deleted!

SQL DELETE Statement
example

- The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL DELETE ALL records

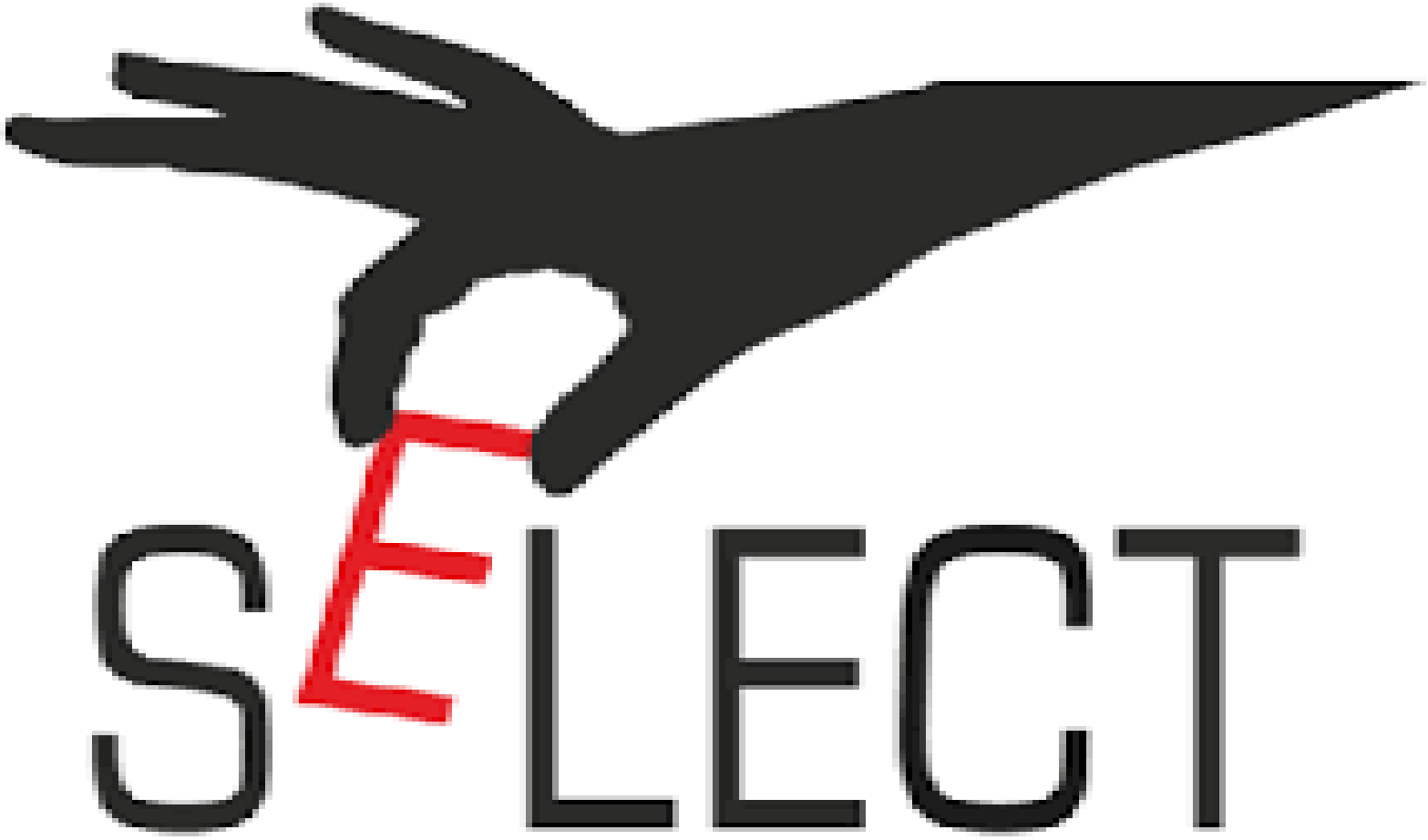
- Delete All Records
- It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:
- `DELETE FROM table_name;`
- Example : `DELETE FROM Customers;`

Result:

You have made changes to the database. Rows affected: 92

Your Database:

Tablename	Records
Customers	0
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29


SELECT

SQL SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.
- **SELECT Syntax** :
  - ➢ `SELECT column1, column2, ...`
  - ➢ `FROM table_name;`

➢ Here, column1, column2, ... are the field names of the table you want to select data from.

➢ If you want to select all the fields available in the table, use the following syntax:

- ➢ `SELECT * FROM table_name;`

SQL SELECT Statement…
example:

- Ex1: write the select statement which select the Customer Name and its City from the "Customers" table
  - ➢`SELECT CustomerName, City FROM Customers;`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

- Ex2: write the select statement which select all data about Customer stored in "Customers" table
  - ➢`SELECT * FROM Customers;`

SQL SELECT DISTINCT Statement…

- The SELECT DISTINCT statement is used to return only distinct (different) values.
- Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
- SELECT DISTINCT Syntax
  - ➢
```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Number of Records: 91

Country
Germany
Mexico
Mexico

Number of Records: 21

Country
Germany
Mexico
UK

- Ex: selects all (including the duplicates) values from the "Country" column in the "Customers" table:
  - ➢
```
SELECT Country FROM Customers;
```
- Ex: selects only the DISTINCT values from the "Country" column in the "Customers" table:
  - ➢
```
SELECT DISTINCT Country FROM Customers;
```

SQL WHERE Clause

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.
- WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Note: The WHERE clause is not only used in SELECT statements, but also in UPDATE, DELETE, etc.!
- Ex: write SQL statement to selects all the customers from the country "Mexico", in the "Customers" table:

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Operators in The WHERE Clause

Operator	Description	
=	Equal	`SELECT * FROM Products WHERE Price = 18;`
>	Greater than	`SELECT * FROM Products WHERE Price > 30;`
<	Less than	`SELECT * FROM Products WHERE Price < 30;`
>=	Greater than or equal	`SELECT * FROM Products WHERE Price >= 30;`
<=	Less than or equal	`SELECT * FROM Products WHERE Price <= 30;`
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=	
BETWEEN	Between a certain range	`SELECT * FROM Products WHERE Price BETWEEN 50 AND 60;`
LIKE	Search for a pattern	`SELECT * FROM Customers WHERE City LIKE 's%';` All countries start with S
IN	To specify multiple possible values for a column	`SELECT * FROM Customers WHERE City IN ('Paris','London');`

Customer table

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et	Frédérique	24, place	Strasbourg	67000	France

SQL AND, OR and NOT Operators

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition:
  - The AND operator displays a record if all the conditions separated by AND are TRUE.
  - The OR operator displays a record if any of the conditions separated by OR is TRUE.
  - The NOT operator displays a record if the condition(s) is NOT TRUE.

SQL AND, OR and NOT Operators…

- AND Syntax

```
➢SELECT column1, column2, ...
 FROM table_name
 WHERE condition1 AND condition2 AND condition3 ...;
```

- Ex: write SQL statement that selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

```
• SELECT * FROM Customers
  WHERE Country='Germany' AND City='Berlin';
```

Number of Records: 1

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

SQL AND, OR and NOT Operators…

- OR Syntax
  - ➢
```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```
- Ex: type SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":
  -
```
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';
```

Number of Records: 2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

SQL AND, OR and NOT Operators…

- **NOT** Syntax
  - ```
    SELECT column1, column2, ...
    FROM table_name
    WHERE NOT condition;
    ```
- Ex: type SQL statement selects all fields from "Customers" where country is NOT "Germany":
  - ```
    SELECT * FROM Customers
    WHERE NOT Country='Germany';
    ```

Number of Records: 80

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Combining AND, OR and NOT

- Ex: The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "München" (use parenthesis to form complex expressions):

```
➢SELECT * FROM  Customers
 WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

Number of Records: 2

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

- Ex: The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

```
➢SELECT * FROM Customers
 WHERE NOT Country='Germany' AND NOT Country='USA';
```

Number of Records: 67

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

SQL ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
- The ORDER BY keyword sorts the records in ascending order by default. (from A to Z or 1 to 10)
- To sort the records in descending order, use the DESC keyword.
- ORDER BY Syntax:

```
➢ SELECT column1, column2, ...
  FROM table_name
  ORDER BY column1, column2, ... ASC|DESC;
```

- Ex: The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

```
➢ SELECT * FROM Customers ORDER BY Country;
```

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium

SQL MIN and MAX Functions

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.
- MIN() Syntax
  - ```
    SELECT MIN(column_name)
    FROM table_name
    WHERE condition;
    ```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

- Ex: The following SQL statement finds the price of the cheapest product:
  - ```
    SELECT MIN(Price) AS SmallestPrice
    FROM Products;
    ```

Result:

Number of Records: 1

SmallestPrice
2.5

SQL MIN() and MAX() Functions

- MAX() Syntax
  - ```
    SELECT MAX(column_name)
    FROM table_name
    WHERE condition;
    ```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

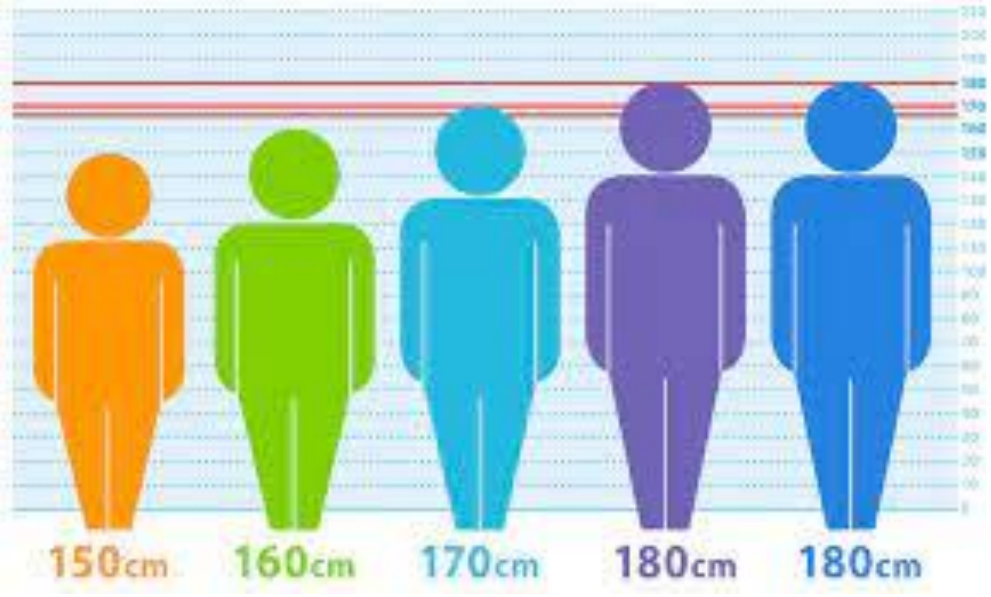
- Ex: The following SQL statement finds the price of the cheapest product:
  - ```
    SELECT MAX(Price) AS LargestPrice
    FROM Products;
    ```

Result:

Number of Records: 1

LargestPrice
263.5

SQL COUNT(), AVG() and SUM() Functions



SQL COUNT(), AVG() and SUM() Functions

- The COUNT() function returns the number of rows that matches a specified criterion.
- COUNT() Syntax
  - ```
    SELECT COUNT(column_name)
    FROM table_name
    WHERE condition;
    ```
- Ex: The following SQL statement finds the number of products:
  - ```
    SELECT COUNT(ProductID)
    FROM Products;
    ```

Your Database:

Tablename	Records
Customers	0
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Result:

Number of Records: 1

COUNT(ProductID)
77

SQL COUNT(), AVG() and SUM() Functions

- The AVG() function returns the average value of a numeric column.
- AVG() Syntax
  - ```
    SELECT AVG(column_name)
    FROM table_name
    WHERE condition;
    ```
- Ex: The following SQL statement finds the average price of all products:
  - ```
    SELECT  AVG(Price)
     FROM Products;
    ```

Your Database:

Tablename	Records
Customers	0
Categories	8
Employees	10
OrderDetails	518
Orders	196
Products	77
Shippers	3
Suppliers	29

Result:

Number of Records: 1

AVG(Price)
28.866363636363637

SQL COUNT(), AVG() and SUM() Functions

- The SUM() function returns the total sum of a numeric column.
- SUM() Syntax
  - ```
    SELECT SUM(column_name)
    FROM table_name
    WHERE condition;
    ```
- Ex: The following SQL statement finds the sum of the "Quantity" fields in the "OrderDetails" table:
  - ```
    SELECT SUM(Quantity)
    FROM OrderDetails;
    ```

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

Result:

Number of Records: 1

SUM(Quantity)
12743

SQL LIKE Operator

SQL LIKE Operator

- LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
- There are two wildcards often used in conjunction with the LIKE operator:
- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character
- LIKE Syntax
  - `SELECT column1, column2, ...`
  - `FROM table_name`
  - `WHERE columnN LIKE pattern;`

- Tip: You can also combine any number of conditions using AND or OR operators.

SQL LIKE Operator…

- Here are some examples showing different LIKE operators with '%' and '_' wildcards:

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"

SQL LIKE Operator…

- Ex: The following SQL statement selects all customers with a CustomerName starting with "a":
  - ```
    SELECT * FROM Customers
    WHERE CustomerName LIKE 'a%';
    ```

Result:

Number of Records: 4

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
    ```

SQL LIKE Operator…

- Ex: The following SQL statement selects all customers with a CustomerName ending with "a":
  - ```
    SELECT * FROM Customers
    WHERE CustomerName LIKE '%a';
    ```

Result:

Number of Records: 7

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
30	Godos Cocina Típica	José Pedro Freyre	C/ Romero, 33	Sevilla	41101	Spain
    ```

SQL LIKE Operator…

- Ex: The following SQL statement selects all customers with a CustomerName that have "or" in any position:"a":
  - ```
    SELECT * FROM Customers
    WHERE CustomerName LIKE '%or%';
    ```

Result:

Number of Records: 11

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA
    ```

SQL LIKE Operator…

- Ex: The following SQL statement selects all customers with a CustomerName that have "r" in the second position:
  - ```
    SELECT * FROM Customers
    WHERE CustomerName LIKE '_r%';
    ```

Result:

Number of Records: 11

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
    ```

SQL LIKE Operator…

- Ex: The following SQL statement selects all customers with a ContactName that starts with "a" and ends with "o":
-
```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%o';
```

Result:

Number of Records: 3

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
69	Romero y tomillo	Alejandra Camino	Gran Vía, 1	Madrid	28001	Spain

SQL IN Operator

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.
- IN Syntax
  - ```
    SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (value1, value2, ...);
    ```
- Or
  - ```
    SELECT column_name(s)
    FROM table_name
    WHERE column_name IN (SELECT STATEMENT);
    ```

SQL IN Operator example

- The IN The following SQL statement selects all customers that are located in "Germany", "France" or "UK":

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

Result:

Number of Records: 29

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
    ```

SQL IN Operator example

- The following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":
  - ```
    SELECT * FROM Customers
    WHERE Country NOT IN ('Germany', 'France', 'UK');
    ```

Result:

Number of Records: 62

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
    ```

SQL BETWEEN Operator

SQL BETWEEN Operator

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: *begin and end values are included*.
- BETWEEN Syntax
  - ```
    SELECT column_name(s)
    FROM table_name
    WHERE column_name BETWEEN value1 AND value2;
    ```
    ```

SQL BETWEEN Operator
example

- The following SQL statement selects all products with a price between 10 and 20:

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

Result:

Number of Records: 29

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
15	Genen Shouyu	6	2	24 - 250 ml bottles	15.5
16	Pavlova	7	3	32 - 500 g boxes	17.45
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
25	NuNuCa Nuß-Nougat-Creme	11	3	20 - 450 g glasses	14

NOT BETWEEN
example

- To display the products outside the range of the previous example, use NOT BETWEEN:
  - ```
    SELECT * FROM Products
    WHERE Price NOT BETWEEN 10 AND 20;
    ```

Number of Records: 48

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97
10	Ikura	4	8	12 - 200 ml jars	31
11	Queso Cabrales	5	4	1 kg pkg.	21
12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38
13	Konbu	6	8	2 kg box	6
    ```

BETWEEN with IN *example*

- The following SQL statement selects all products with a price between 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3:
  - ```
    SELECT * FROM Products
    WHERE Price BETWEEN 10 AND 20;
    AND CategoryID NOT IN (1,2,3);
    ```

Number of Records: 9

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12.5
36	Inlagd Sill	17	8	24 - 250 g jars	19
40	Boston Crab Meat	19	8	24 - 4 oz tins	18.4
42	Singaporean Hokkien Fried Mee	20	5	32 - 1 kg pkgs.	14
46	Spegesild	21	8	4 - 450 g glasses	12
57	Ravioli Angelo	26	5	24 - 250 g pkgs.	19.5
58	Escargots de Bourgogne	27	8	24 pieces	13.25
    ```

BETWEEN Text Values
example

- The following SQL statement selects all products with a ProductName between Carnarvon Tigers and Mozzarella di Giovanni:
  - ```
    SELECT * FROM Products
    WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
    ORDER BY ProductName;
    ```

Result:

Number of Records: 37

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
39	Chartreuse verte	18	1	750 cc per bottle	18
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
48	Chocolade	22	3	10 pkgs.	12.75
    ```

BETWEEN Dates
example

- The following SQL statement selects all orders with an OrderDate between '01-July-1996' and '31-July-1996':
  - ```
    SELECT * FROM Orders
    WHERE OrderDate BETWEEN "07/01/1996"  AND  "07/31/1996";
    ```

Result:

Number of Records: 22

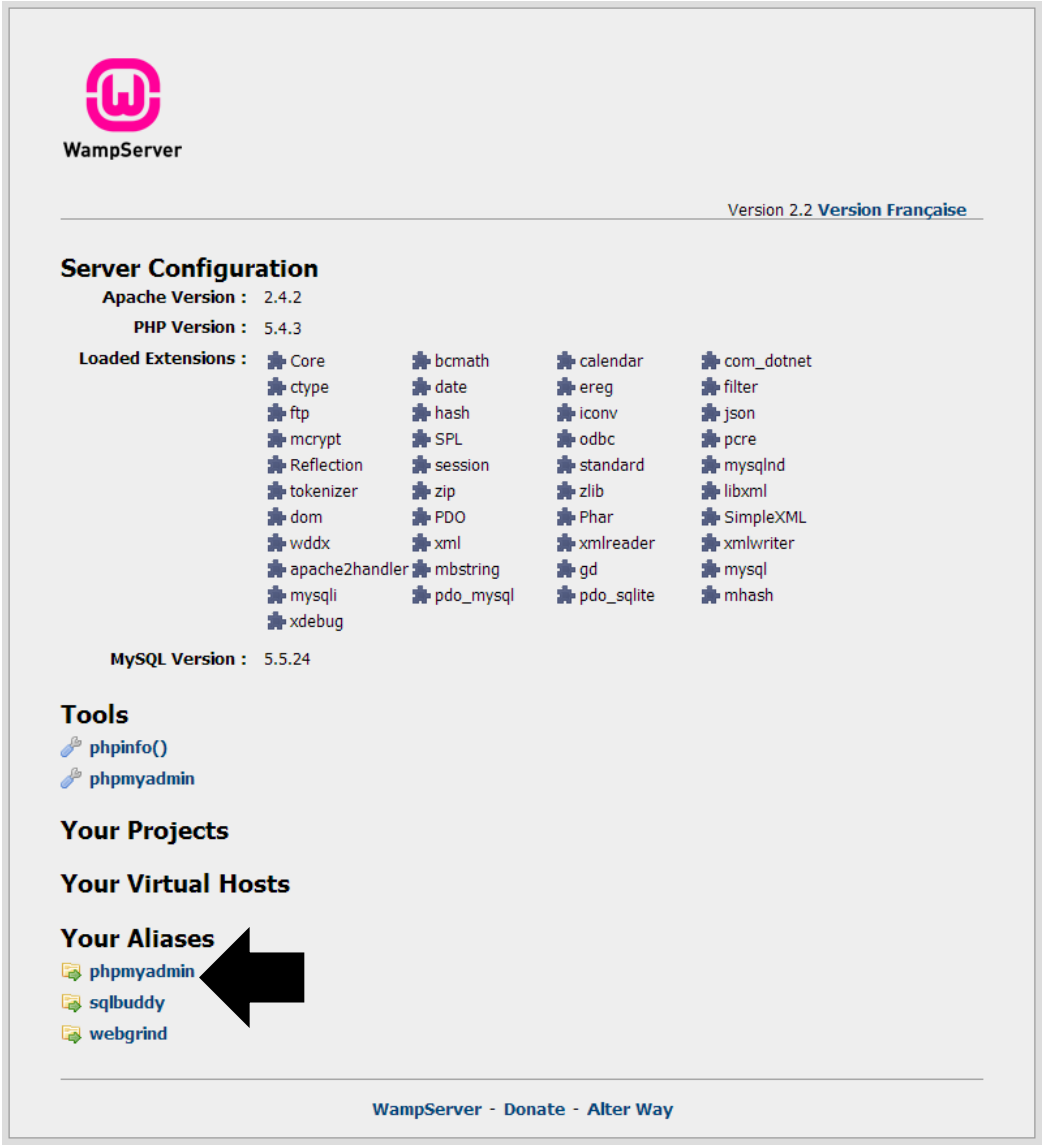
OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/8/1996	1
10252	76	4	7/9/1996	2
10253	34	3	7/10/1996	2
10254	14	5	7/11/1996	2
10255	68	9	7/12/1996	3
    ```

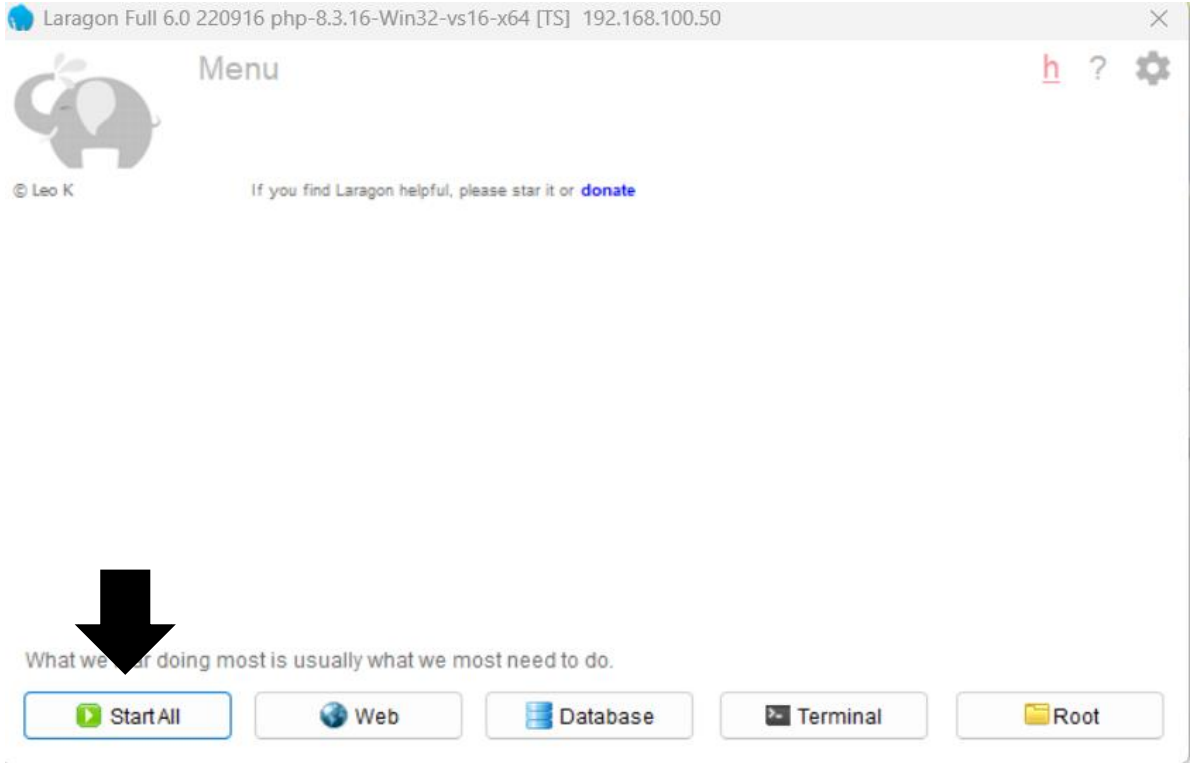
Programs

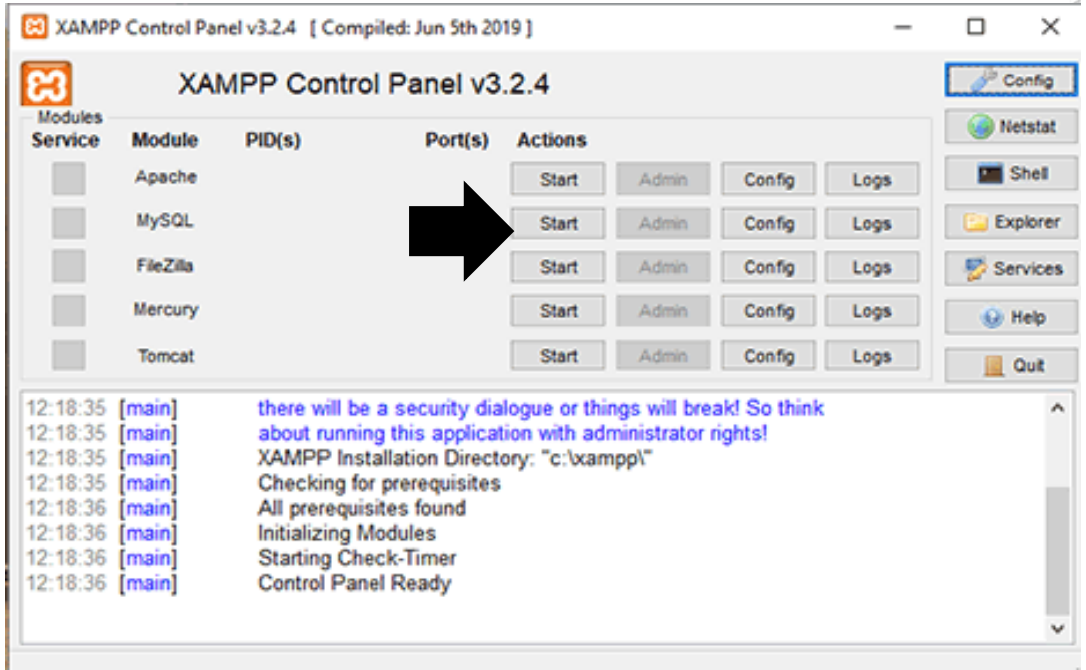
- XAMPP: https://www.apachefriends.org/
- WAMP: https://www.wampserver.com/
- LARAGON: https://laragon.org/download

- To access database: http://localhost/phpmyadmin

Programs


WampServer
Version 2.2 Version Française
Server Configuration
Apache Version : 2.4.2
PHP Version : 5.4.3
Loaded Extensions : Core, bcmath, calendar, com_dotnet, ctype, date, ereg, filter, ftp, hash, iconv, json, mcrypt, SPL, odbc, pcre, Reflection, session, standard, mysqlnd, tokenizer, zip, zlib, libxml, dom, PDO, Phar, SimpleXML, wddx, xml, xmlreader, xmlwriter, apache2handler, mbstring, gd, mysql, mysqli, pdo_mysql, pdo_sqlite, mhash, xdebug
MySQL Version : 5.5.24
Tools
phpinfo()
phpmyadmin
Your Projects
Your Virtual Hosts
Your Aliases
phpmyadmin
sqlbuddy
webgrind
WampServer - Donate - Alter Way


Laragon Full 6.0 220916 php-8.3.16-Win32-vs16-x64 [TS] 192.168.100.50
Menu
© Leo K
If you find Laragon helpful, please star it or donate
What we doing most is usually what we most need to do.
Start All
Web
Database
Terminal
Root


XAMPP Control Panel v3.2.4 [Compiled: Jun 5th 2019]
XAMPP Control Panel v3.2.4
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
MySQL
FileZilla
Mercury
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
12:18:35 [main] there will be a security dialogue or things will break! So think
12:18:35 [main] about running this application with administrator rights!
12:18:35 [main] XAMPP Installation Directory: "c:\xampp\"
12:18:35 [main] Checking for prerequisites
12:18:36 [main] All prerequisites found
12:18:36 [main] Initializing Modules
12:18:36 [main] Starting Check-Timer
12:18:36 [main] Control Panel Ready

Any Questions ?