unionbank

TURBOCHARGED
FOR SUCCESS

Advanced Data Analytics using SQL

November 2021

SQL Fundamentals

Introduction to SQL

SQL is a standard language for storing, manipulating and retrieving data in databases.

The history of SQL begins in an IBM laboratory in San Jose, California, where SQL was developed in the late 1970s for IBM's DB2 product (a relational database management system

The initials stand for Structured Query Language, and the language itself is often referred to as "sequel."

SQL Fundamentals

Introduction to SQL

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

SQL Fundamentals

SQL Activities

Data Definition Language (DDL):

Use DDL commands to specify database schema:

CREATE: This is used to create a new database or objects in a database.

ALTER: This is used to alter a database or objects in a database.

DROP: This is used to delete a database or objects in a database.

TRUNCATE: This is used to remove all data from a table instantaneously.

SQL Fundamentals

SQL Activities

Data Manipulation Language (DML):

Use DML commands to query and modify data:

SELECT: This is used to retrieve data from a database.

INSERT: This is used to insert data into a database.

UPDATE: This is used to update data in a database.

DELETE: This is used to remove data from a database.

SQL Fundamentals

SQL Activities

Data Control Language(DCL):

Use DCL commands to control permissions and translations:

GRANT: This is used to give access to a user.

REVOKE: This is used to take access away from a user.

COMMIT: This is used to save changes in a transaction.

ROLLBACK: This is used to remove the saved changes in a transaction.

SQL Fundamentals

Elements of SQL

The SQL language comprises several elements.

These elements include the following:

Queries that retrieve data based on specific criteria.

Clauses that are components of statements or queries.

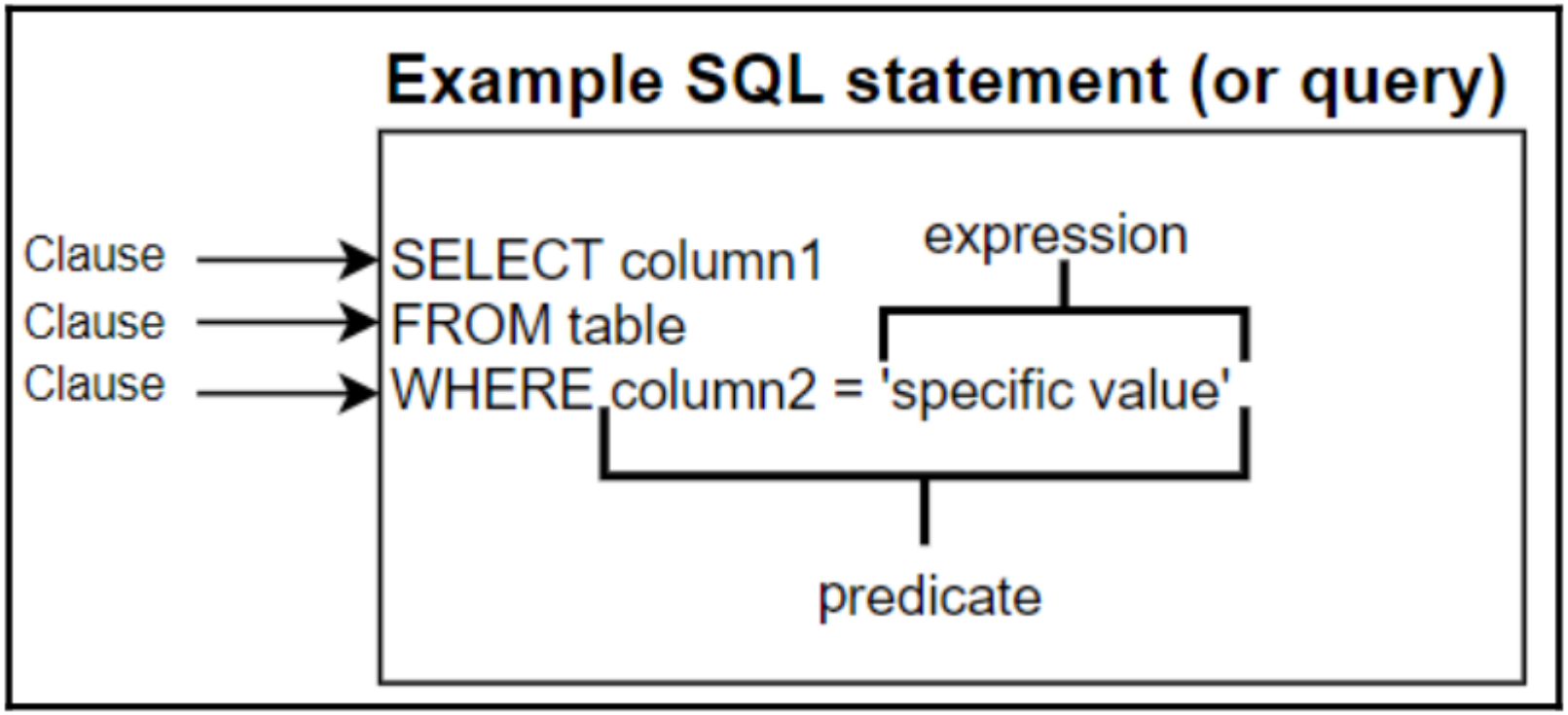
Predicates that are logical conditions that evaluate to true or false. These help you to narrow down the results of your queries.

Expressions that produce either scalar values or tables of columns and rows, expressions are a part of predicates.

SQL Fundamentals

Elements of SQL

The diagram shows you the components of a SQL statement, which is also called a SQL query, you can see the different elements of a SQL statement



SQL Fundamentals

Understanding databases

A database is a collection of data.

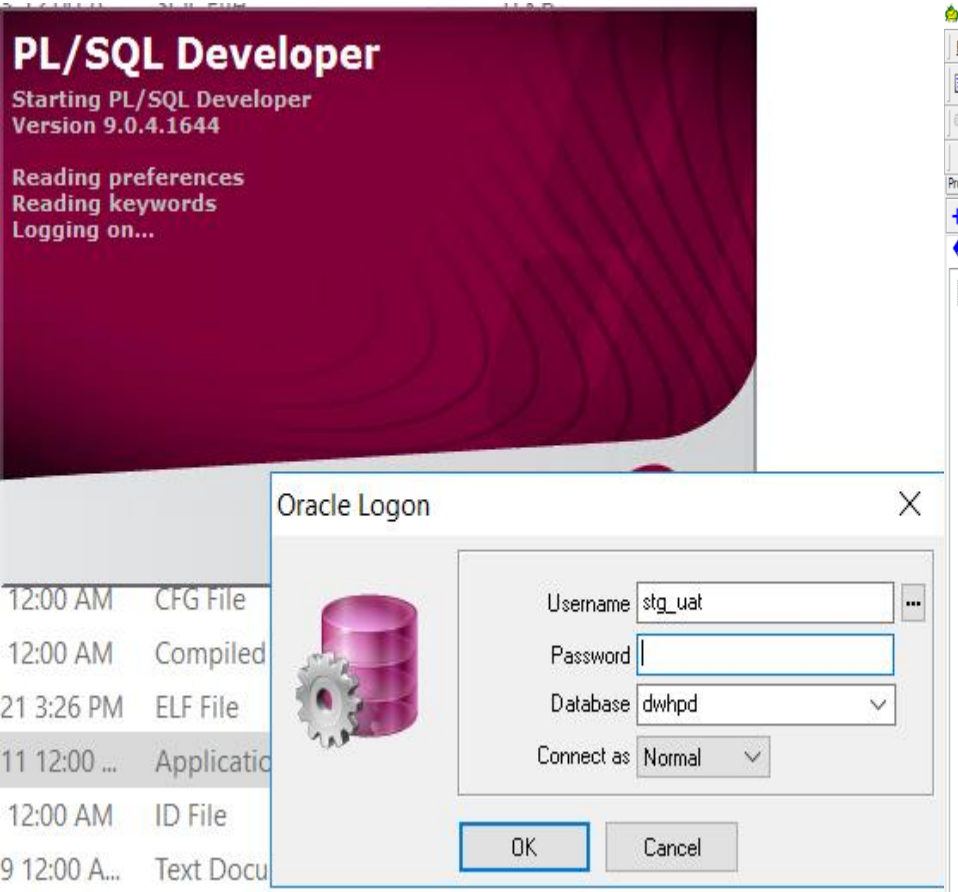
You store databases in a relational database management system (RDMS).

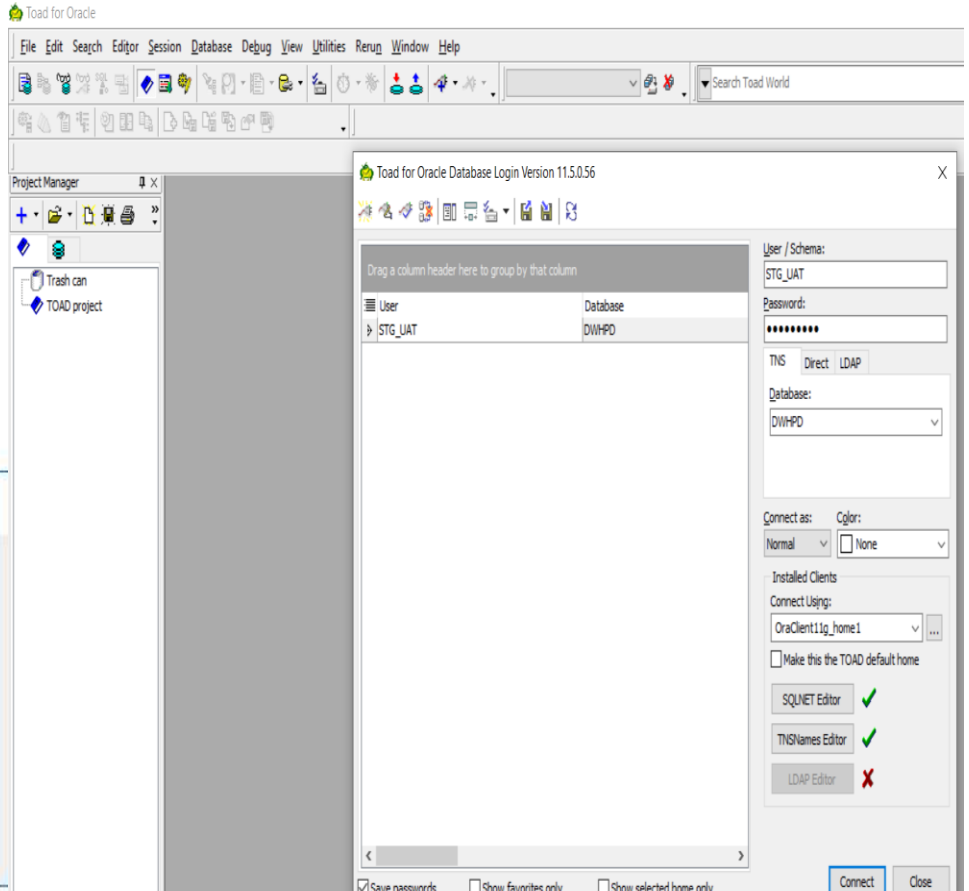
The RDMS is the basis for modern database systems like MySQL, SQL Server, Oracle, PostgreSQL, and others.

SQL Fundamentals

Getting Started with Oracle SQL Tools

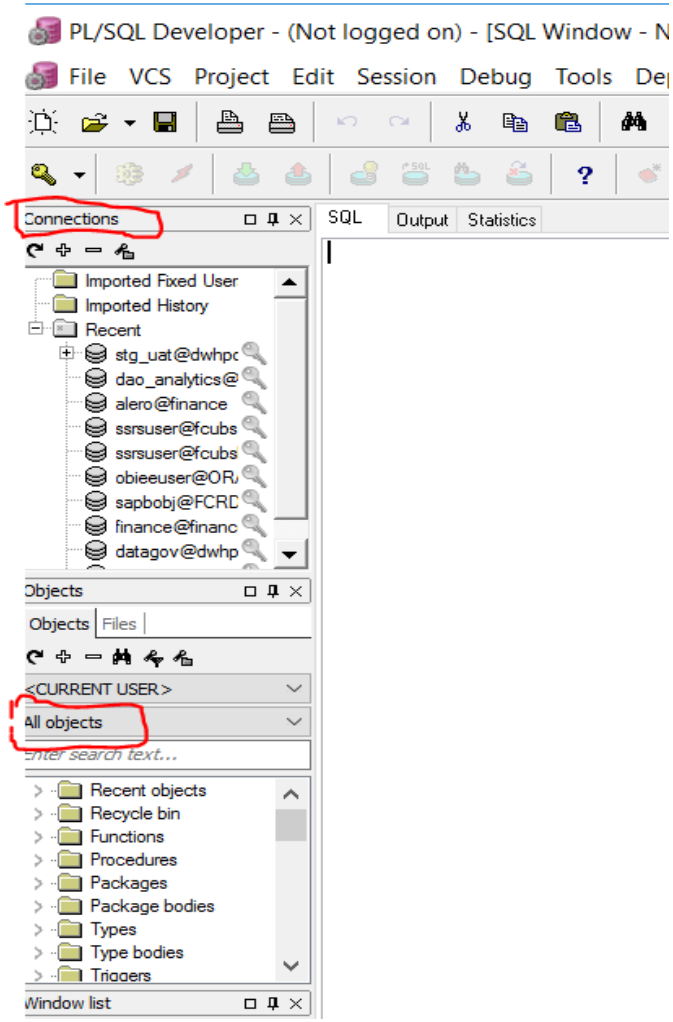
- PL/SQL DEVELOPER
- SQL NAVIGATOR
- TOAD FOR ORACLE





SQL Fundamentals

PL/SQL Layout Overview



SQL Fundamentals

The **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 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:

SQL Fundamentals

The **SELECT** Statement

Demo Database

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

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

SQL Fundamentals

The **SELECT** Statement

SELECT Column Example

The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

```
SELECT CustomerName, City FROM Customers;
```

Result:

Number of Records: 91

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå
Blauer See Delikatessen	Mannheim
Blondel père et fils	Strasbourg
Bólido Comidas preparadas	Madrid

SQL Fundamentals

The **SELECT** Statement

SELECT * Example

The following SQL statement selects all the columns from the "Customers" table:

```
SELECT * FROM Customers;
```

Result:

Number of Records: 91

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 fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain

SQL Fundamentals

The **SELECT** Statement

<u>The SQL SELECT DISTINCT Statement</u>

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;
```

SQL Fundamentals

The **SELECT** Statement

The SQL SELECT DISTINCT Statement

Example

SELECT DISTINCT Country FROM Customers;

Result:

Number of Records: 21

Country
Argentina
Austria
Belgium
Brazil
Canada
Denmark
Finland
France

Questions

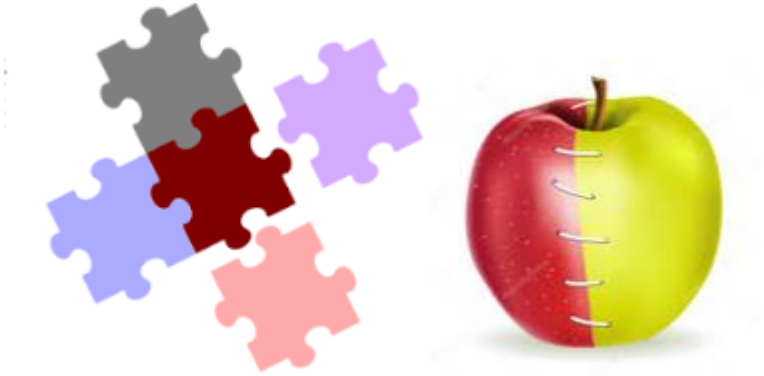
SQL Fundamentals

Expression and Condition

An expression is a combination of one or more values, operators and SQL functions that evaluate to a value.

These SQL EXPRESSIONs are like formulae and they are written in query language.

You can also use them to query the database for a specific set of data.



SQL Fundamentals

Expression and Condition

The SQL WHERE Clause

The WHERE clause is used to filter records.

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

WHERE Syntax

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

SQL Fundamentals

Expression and Condition

The SQL WHERE Clause

WHERE Clause Example

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

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

Result:

Number of Records: 5

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
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico
80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123	México D.F.	05033	Mexico

SQL Fundamentals

Expression and Condition

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes). However, numeric fields should not be enclosed in quotes:

Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

Result:

Number of Records: 1

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

SQL Fundamentals

Expression and Condition

Operators in The WHERE Clause

An operator is a reserved word or a character used primarily in an SQL statement's WHERE clause to perform operation(s), such as comparisons and arithmetic operations.

These Operators are used to specify conditions in an SQL statement and to serve as conjunctions for multiple conditions in a statement.

Arithmetic operators
Comparison operators
Logical operators
Operators used to negate conditions

SQL Fundamentals

Expression and Condition

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

Operator	Description
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL Fundamentals

Expression and Condition

<u>The SQL AND, OR and NOT Operators</u>

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 is 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 Fundamentals

Expression and Condition

The SQL AND, OR and NOT Operators

AND Syntax

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

OR Syntax

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

SQL Fundamentals

Expression and Condition

The SQL AND, OR and NOT Operators

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

SQL Fundamentals

Expression and Condition

The SQL AND, OR and NOT Operators

AND Example

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

Example

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

Result:

Number of Records: 1

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

SQL Fundamentals

Expression and Condition

The SQL AND, OR and NOT Operators

OR Example

The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

Example
SELECT * FROM Customers
WHERE City='Berlin' OR City='München';

Result:

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 Fundamentals

Expression and Condition

The SQL AND, OR and NOT Operators

NOT Example

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

Example

```
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
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

SQL Fundamentals

Expression and Condition

Combining AND, OR and NOT

You can also combine the AND, OR and NOT operators.
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):

Example

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

Result:

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 Fundamentals

Expression and Condition

<u>The SQL LIKE Operator</u>

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

There are two wildcards used in conjunction with the LIKE operator:

% - The percent sign represents zero, one, or multiple characters
_ - The underscore represents a single character

The percent sign and the underscore can also be used in combinations!

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

SQL Fundamentals

Expression and Condition

The 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 starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends 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 starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

SQL LIKE Examples

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

Example

```
SELECT * FROM Customers
WHERE ContactName LIKE 'a%';
```

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

SQL LIKE Examples

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

Example

```
SELECT * FROM Customers
WHERE ContactName LIKE '%a';
```

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

SQL LIKE Examples

The following SQL statement selects all customers with a CustomerName that have "or" in any position:

Example

```
SELECT * FROM Customers
WHERE ContactName LIKE '%or%';
```

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

SQL LIKE Examples

The following SQL statement selects all customers with a CustomerName that have "r" in the second position:

Example

```
SELECT * FROM Customers
WHERE ContactName LIKE '_r%';
```

SQL Fundamentals

Expression and Condition

The SQL LIKE Operator

SQL LIKE Examples

The following SQL statement selects all customers with a CustomerName that starts with "a" and are at least 3 characters in length:

Example

```
SELECT * FROM Customers
WHERE ContactName LIKE 'a_%_%';
```

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

The 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.

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

BETWEEN Example

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

Example

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

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

BETWEEN Example

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
31	Gorgonzola Telino	14	4	12 - 100 g pkgs	12.5

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

NOT BETWEEN Example

To display the products outside the range of the previous example, use NOT BETWEEN:

Example

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

BETWEEN Dates Example

The following SQL statement selects all orders with an OrderDate BETWEEN '04-July-1996' and '09-July-1996':

Example

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN ‘07/04/1996’ AND ‘07/09/1996’;
```

This is similar to :

```
SELECT * FROM Orders
WHERE OrderDate >= '07/04/1996' AND
OrderDate <='07/09/1996'
```

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

BETWEEN Text Values Example

The following SQL statement selects all products with a ProductName BETWEEN 'Carnarvon Tigers' and 'Mozzarella di Giovanni':

Example

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni';
```

SQL Fundamentals

Expression and Condition

SQL BETWEEN Operator

NOT BETWEEN Text Values Example

The following SQL statement selects all products with a ProductName NOT BETWEEN 'Carnarvon Tigers' and 'Mozzarella di Giovanni':

Example

```
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND
'Mozzarella di Giovanni';
```

SQL Fundamentals

Expression and Condition

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 Fundamentals

Expression and Condition

SQL IN Operator

IN Operator Examples

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

Example

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

SQL Fundamentals

Expression and Condition

SQL IN Operator

IN Operator Examples

The following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":

Example

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```

SQL Fundamentals

Expression and Condition

SQL IN Operator

IN Operator Examples

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);
```

Amazing ?
We will discuss more on this in the next slides

SQL Fundamentals

Expression and Condition

SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of the query.

Aliases can be useful when:

There are more than one table involved in a query
Functions are used in the query
Column names are big or not very readable
Two or more columns are combined together

SQL Fundamentals

Expression and Condition

SQL Aliases

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

SQL Fundamentals

Expression and Condition

SQL Aliases

Alias for Columns Examples

The following SQL statement creates two aliases, one for the CustomerID column and one for the CustomerName column:

Example

```
SELECT CustomerID as ID, ContactName AS Customer
FROM Customers;
```

SQL Fundamentals

Expression and Condition

SQL Aliases

Alias for Columns Examples

The following SQL statement creates two aliases, one for the CustomerName column and one for the ContactName column.

Note: It requires double quotation marks or square brackets if the alias name contains spaces:

Example

```
SELECT customer_name AS CustomerName
FROM Customer_master;

SELECT customer_name AS “Customer Name ”
FROM Customer_master;
```

SQL Fundamentals

Expression and Condition

SQL Aliases

Alias for Columns Examples

The following SQL statement creates an alias named "Address" that combine four columns (Address, PostalCode, City and Country):

Example

```
SELECT ContactName, Address + ', ' + PostalCode + ' ' + City + ', ' +
Country AS Address
FROM Customers;
```

SQL Fundamentals

Expression and Condition

SQL TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact on performance.

SQL Fundamentals

Expression and Condition

SQL TOP Clause

The following SQL statement selects the first three records from the "Customers" table:

Example

```
SELECT  * FROM Customer_master
FETCH FIRST 3 ROWS ONLY;
```

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

SQL Fundamentals

Expression and Condition

SQL NULL Values

What is a NULL Value?

A field with a NULL value is a field with no value.

Note: It is very important to understand that 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!

SQL Fundamentals

Expression and Condition

SQL NULL Values

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 Fundamentals

Expression and Condition

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;
```

SQL Fundamentals

Expression and Condition

SQL NULL Values

The IS NULL Operator

The following SQL statement uses the IS NULL operator to list all persons that have no address:

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NULL;
```

The result-set will look like this:

LastName	FirstName	Address
Bloggs	Joe	
Roe	Jane	

SQL Fundamentals

Expression and Condition

SQL NULL Values

The IS NULL Operator

The following SQL statement uses the IS NULL operator to list all persons that have no address:

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NULL;
```

The result-set will look like this:

LastName	FirstName	Address
Bloggs	Joe	
Roe	Jane	

Tip: Always use IS NULL to look for NULL values.

SQL Fundamentals

Expression and Condition

SQL NULL Values

The IS NOT NULL Operator

The following SQL statement uses the IS NOT NULL operator to list all persons that do have an address:

```
SELECT LastName, FirstName, Address FROM Persons
WHERE Address IS NOT NULL;
```

The result-set will look like this:

LastName	FirstName	Address
Doe	John	542 W. 27th Street
Smith	John	110 Bishopsgate

Questions

SQL Fundamentals

Applying Comments

Comments are used to explain sections of SQL statements, or to prevent execution of SQL statements.

Single Line Comments

Single line comments start with --.

Any text between -- and the end of the line will be ignored (will not be executed).

The following example uses a single-line comment as an explanation:

SQL Fundamentals

Applying Comments

The following example uses a single-line comment as an explanation:

Example

```
--Select all:
SELECT * FROM Customers;
```

The following example uses a single-line comment to ignore the end of a line:

Example

```
SELECT * FROM Customers -- WHERE City='Berlin';
```

SQL Fundamentals

Applying Comments

The following example uses a single-line comment to ignore a statement:

Example

```
--SELECT * FROM Customers;
SELECT * FROM Products;
```

The following example uses a single-line comment to ignore the end of a line:

Example

```
SELECT * FROM Customers -- WHERE City='Berlin';
```

SQL Fundamentals

Applying Comments

Multi-line Comments

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored.

The following example uses a multi-line comment as an explanation:

Example

```
/*Select all the columns
of all the records
in the Customers table:*/
SELECT * FROM Customers;
```

SQL Fundamentals

Applying Comments

Multi-line Comments

The following example uses a multi-line comment to ignore many statements:

Example

```
/*SELECT * FROM Customers;
SELECT * FROM Products;
SELECT * FROM Orders;
SELECT * FROM Categories;*/
SELECT * FROM Suppliers;
```

The following example uses a comment to ignore part of a line:

Example

```
SELECT CustomerName, /*City,*/ Country FROM Customers;
```

SQL Fundamentals

Applying Comments

Multi-line Comments

The following example uses a comment to ignore part of a statement:

Example

```
SELECT * FROM Customers WHERE (CustomerName LIKE 'L%'
OR CustomerName LIKE 'R%' /*OR CustomerName LIKE 'S%'
OR CustomerName LIKE 'T%'*/ OR CustomerName LIKE 'W%')
AND Country='USA'
ORDER BY CustomerName;
```

Questions

SQL Fundamentals

Sorting Data

<u>The SQL ORDER BY Keyword</u>

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.** To sort the records in descending order, use the DESC keyword.

ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition //the where clause is optional
ORDER BY column1, column2, ... ASC|DESC;
```

SQL Fundamentals

Sorting Data

The SQL ORDER BY Keyword

ORDER BY Example

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

Example

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

SQL Fundamentals

Sorting Data

The SQL ORDER BY Keyword

ORDER BY DESC Example

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

Example

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

SQL Fundamentals

Sorting Data

The SQL ORDER BY Keyword

ORDER BY Several Columns Example

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

Example

```
SELECT * FROM Customers
ORDER BY Country, CompanyName;
```

SQL Fundamentals

Sorting Data

The SQL ORDER BY Keyword

ORDER BY Several Columns Example 2

The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CompanyName" column:

Example

```
SELECT * FROM Customers
ORDER BY Country ASC, CompanyName DESC;
```

SQL Fundamentals

Basic Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value.

They are allowed in the select list or the HAVING clause of a SELECT statement.

You can use an aggregation in combination with the GROUP BY clause to calculate the aggregation on categories of rows.

SQL Fundamentals

Basic Aggregate Functions

The SQL COUNT(), AVG() , SUM() , MIN() and MAX() Functions

The COUNT() function returns the number of rows that matches a specified criteria.

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

The SUM() function returns the total sum of a numeric column.

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

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

SQL Fundamentals

Basic Aggregate Functions

COUNT() Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

AVG() Syntax

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

SUM() Syntax

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

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

SQL Fundamentals

Basic Aggregate Functions

COUNT() Example

The following SQL statement finds the number of records:
Example

```
SELECT COUNT(*)
FROM Products;
```

The following SQL statement finds the number of products:
Example

```
SELECT COUNT(ProductID)
FROM Products;
```

SQL Fundamentals

Basic Aggregate Functions

AVG() Example

The following SQL statement finds the average price of all products:

Example

```
SELECT AVG(UnitPrice)
FROM Products;
```

SQL Fundamentals

Basic Aggregate Functions

Aggregate functions

SUM() Example

The following SQL statement finds the sum of the "Quantity" fields in the "OrderDetails" table:

Example

```
SELECT SUM(Quantity)
FROM [Order Details];
```

SQL Fundamentals

Basic Aggregate Functions

MIN() Example

The following SQL statement finds the price of the cheapest product:

Example

```
SELECT MIN(UnitPrice) AS SmallestPrice
FROM Products;
```

SQL Fundamentals

Basic Aggregate Functions

MAX() Example

The following SQL statement finds the price of the most expensive product:

Example

```
SELECT MAX(UnitPrice) AS LargestPrice
FROM Products;
```

SQL Fundamentals

Summarizing Data

SQL GROUP BY Statement

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

GROUP BY Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition  // where clause is optional
GROUP BY column_name(s)
ORDER BY column_name(s); // order by clause is optional
```

SQL Fundamentals

Summarizing Data

SQL GROUP BY Statement

SQL GROUP BY Examples

The following SQL statement lists the number of customers in each country:

Example

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country;
```

SQL Fundamentals

Summarizing Data

SQL GROUP BY Statement

SQL GROUP BY Examples

The following SQL statement lists the number of customers in each country, sorted high to low:

Example

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

SQL Fundamentals

Summarizing Data

SQL HAVING Clause

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

SQL Fundamentals

Summarizing Data

SQL HAVING Clause

SQL HAVING Examples

The following SQL statement lists the number of customers in each country. Only include countries with more than 5 customers:

Example

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

SQL Fundamentals

Summarizing Data

SQL HAVING Clause

SQL HAVING Examples

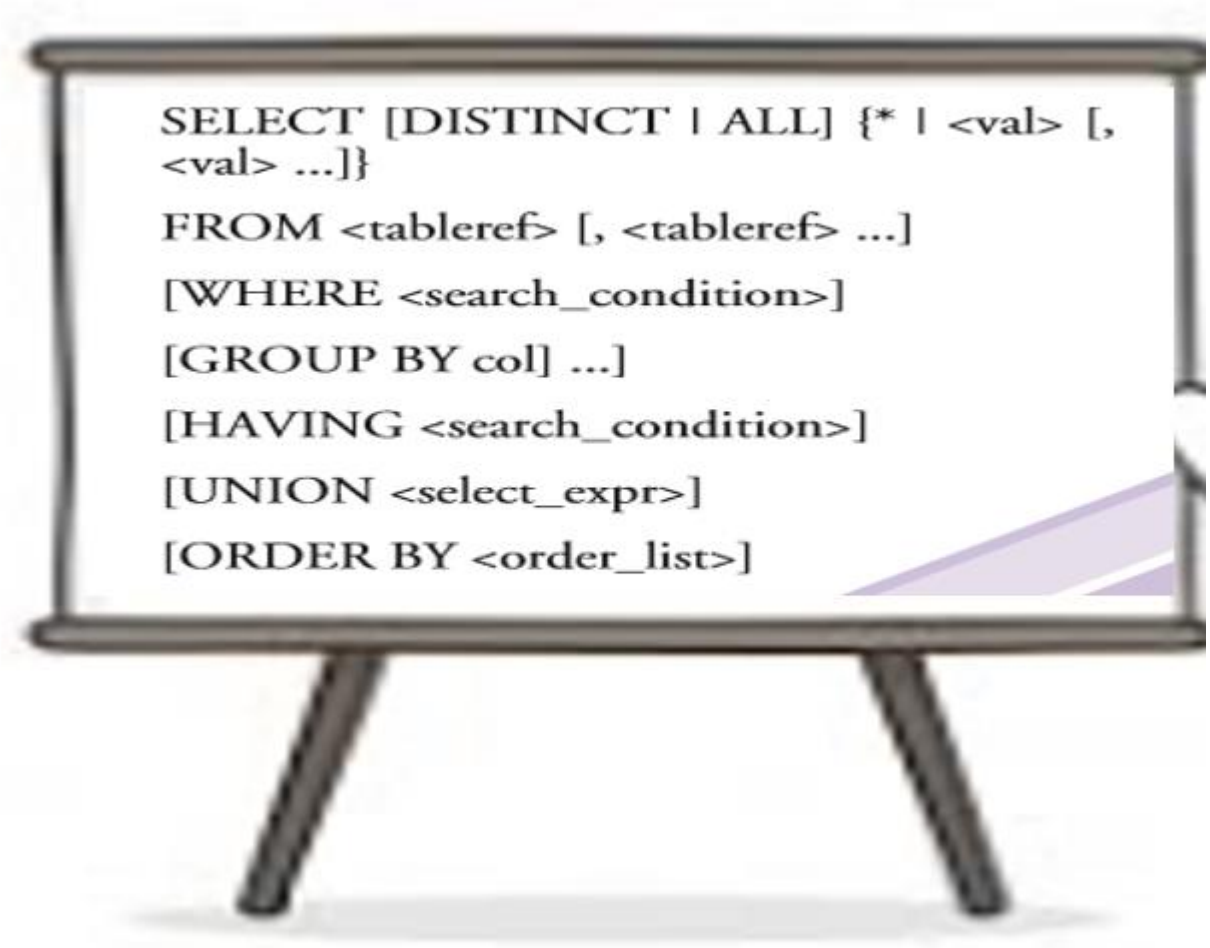
The following SQL statement lists the number of customers in each country, sorted high to low (Only include countries with more than 5 customers):

Example

```
SELECT Country, COUNT(CustomerID)
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

SQL Fundamentals

General SELECT Syntax



Questions

SQL Fundamentals

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*, ...);

OR

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

SQL Fundamentals

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;
```

SQL Fundamentals

SQL DELETE STATEMENT

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

DELETE Syntax

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

SQL Fundamentals

Questions

What does SQL stand for?

- ○ Strong Question Language
- ○ Structured Question Language
- ○ Structured Query Language

SQL Fundamentals

Questions

With SQL, how do you select all the columns from a table named "Persons"?

- ○ SELECT *.Persons
- ○ SELECT * FROM Persons
- ○ SELECT [all] FROM Persons
- ○ SELECT Persons

SQL Fundamentals

Questions

Which SQL statement is used to extract data from a database?

- ○ EXTRACT
- ○ GET
- ○ OPEN
- ○ SELECT

SQL Fundamentals

Questions

With SQL, how do you select all the records from a table named "Persons" where the value of the column "FirstName" is "Peter"?

- SELECT [all] FROM Persons WHERE FirstName='Peter'
- SELECT * FROM Persons WHERE FirstName<>'Peter'
- SELECT [all] FROM Persons WHERE FirstName LIKE 'Peter'
- SELECT * FROM Persons WHERE FirstName='Peter'

SQL Fundamentals

Questions

. With SQL, how do you select all the records from a table named "Persons" where the value of the column "FirstName" starts with an "a"?

- ○ SELECT * FROM Persons WHERE FirstName='%a%'
- ○ SELECT * FROM Persons WHERE FirstName LIKE 'a%'
- ○ SELECT * FROM Persons WHERE FirstName='a'
- ○ SELECT * FROM Persons WHERE FirstName LIKE '%a'

SQL Fundamentals

Questions

The OR operator displays a record if ANY conditions listed are true. The AND operator displays a record if ALL of the conditions listed are true

- False
- True

SQL Fundamentals

Questions

. With SQL, how do you select all the records from a table named "Persons" where the "LastName" is alphabetically between (and including) "Hansen" and "Pettersen"?

- SELECT LastName>'Hansen' AND LastName<'Pettersen' FROM Persons
- SELECT * FROM Persons WHERE LastName BETWEEN 'Hansen' AND 'Pettersen'
- SELECT * FROM Persons WHERE LastName>'Hansen' AND LastName<'Pettersen'

SQL Fundamentals

Questions

Which SQL statement is used to return only different values?

- SELECT DISTINCT
- SELECT UNIQUE
- SELECT DIFFERENT

SQL Fundamentals

Questions

Which SQL keyword is used to sort the result-set?

- SORT
- ORDER BY
- ORDER
- SORT BY

SQL Fundamentals

Questions

. With SQL, how can you return all the records from a table named "Persons" sorted descending by "FirstName"?

- ○ SELECT * FROM Persons ORDER BY FirstName DESC
- ○ SELECT * FROM Persons ORDER FirstName DESC
- ○ SELECT * FROM Persons SORT 'FirstName' DESC
- ○ SELECT * FROM Persons SORT BY 'FirstName' DESC

SQL Fundamentals

Questions

Which operator is used to select values within a range?

- WITHIN
- BETWEEN
- RANGE

THANK YOU