SQL ZERO TO HERO DAY-3

Created by:
Devikrishna R

SQL BASICS_

CREATE DATABASES

Create database SQLSERIES;

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.
Below is an example of a table called "Person":

P_Id	LastName	FirstName	Address	Country
1	R	DEVI	India Nagar	india
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK

The table above contains three records (one for each person) and five columns (P_Id, LastName, FirstName, Address, and City).

SQL Statements

Most of the actions you need to perform on a database are done with SQL statements.
The following SQL statement will select all the records in the "Person" table:

```
SELECT * FROM Person;
```

Note:

- SQL is not case sensitive
- 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.
- We are using MS Access and SQL Server 2000 and we do not have to put a semicolon after each SQL statement, but some database programs force you to use it.

SQL DML and DDL

The query and update commands form the **DML part of SQL**:

- **SELECT -** extracts data
- **UPDATE -** updates data
- **DELETE -** deletes database
- **INSERT INTO -** inserts new data into a database

The **DDL part of SQL** :

- **CREATE DATABASE -** creates a new
- **ALTER DATABASE** – make changes
- **DROP TABLE -** deletes a table
- **CREATE INDEX -** creates an index (search key)
- **DROP INDEX -** deletes an index

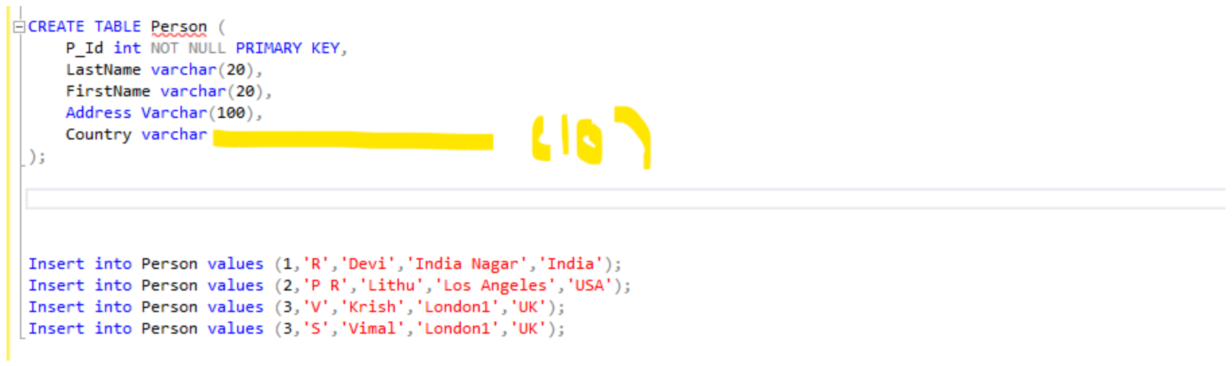
The DATABASE Statement

CREATE DATABASE SQLSERIES;

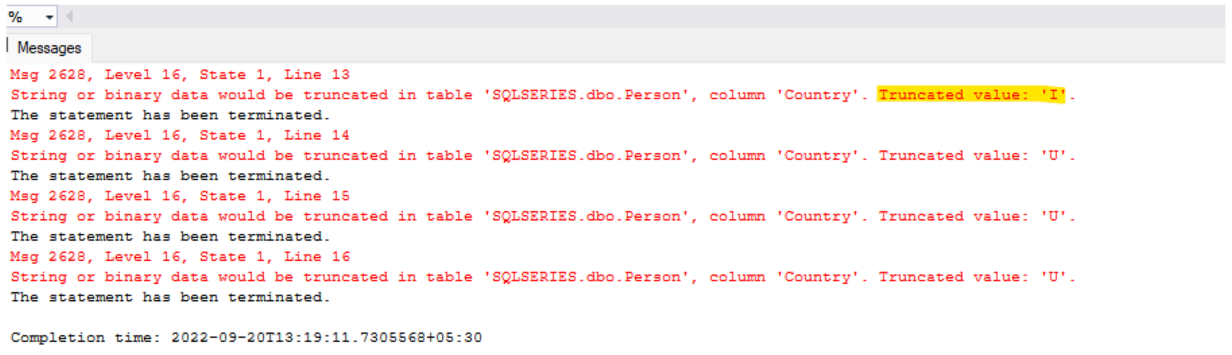
The TABLE Statement

```
CREATE TABLE Person (
    P_Id int NOT NULL PRIMARY KEY,
    LastName varchar(20),
    FirstName varchar(20),
    Address Varchar(100),
    Country varchar
);
```

Error:



```
CREATE TABLE Person (
    P_Id int NOT NULL PRIMARY KEY,
    LastName varchar(20),
    FirstName varchar(20),
    Address Varchar(100),
    Country varchar
);


Insert into Person values (1,'R','Devi','India Nagar','India');
Insert into Person values (2,'P R','Lithu','Los Angeles','USA');
Insert into Person values (3,'V','Krish','London1','UK');
Insert into Person values (3,'S','Vimal','London1','UK');
```



```
Msg 2628, Level 16, State 1, Line 13
String or binary data would be truncated in table 'SQLSERIES.dbo.Person', column 'Country'. Truncated value: 'I'.
The statement has been terminated.
Msg 2628, Level 16, State 1, Line 14
String or binary data would be truncated in table 'SQLSERIES.dbo.Person', column 'Country'. Truncated value: 'U'.
The statement has been terminated.
Msg 2628, Level 16, State 1, Line 15
String or binary data would be truncated in table 'SQLSERIES.dbo.Person', column 'Country'. Truncated value: 'U'.
The statement has been terminated.
Msg 2628, Level 16, State 1, Line 16
String or binary data would be truncated in table 'SQLSERIES.dbo.Person', column 'Country'. Truncated value: 'U'.
The statement has been terminated.

Completion time: 2022-09-20T13:19:11.7305568+05:30
```

DROP Table

Drop table Person;

RECREATE THE TABLE

```
CREATE TABLE Person (
    P_Id int NOT NULL PRIMARY KEY,
    LastName varchar(20),
    FirstName varchar(20),
    Address Varchar(100),
    Country varchar (100)
);
```

INSERT INTO TABLE

```
Insert into Person values (1,'R','Devi','India Nagar','India');
Insert into Person values (2,'P R','Lithu','Los Angeles','USA');
Insert into Person values (3,'V','Krish','London1','UK');
Insert into Person values (4,'S','Vimal','London1','UK');
```

The SQL SELECT Statement

- The SELECT statement is used to select data from a database.

SQL SELECT Syntax

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

OR

```
SELECT * FROM table_name;
```

Note: SQL is not case-sensitive. SELECT is the same as select.

An SQL SELECT Example

The "Person" table:

SELECT * FROM Person;

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	London1	UK

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above.

We use the following SELECT statement:

```
SELECT LastName, FirstName FROM Person;
```

The result-set will look like this:

LastName	FirstName
R	Devi
P R	Lithu
V	Krish
S	Vimal

SELECT * Example

Now we want to select all the columns from the "Person" table.
We use the following SELECT statement:

```
SELECT * From Person;
```

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	London1	UK

The SQL SELECT DISTINCT Statement

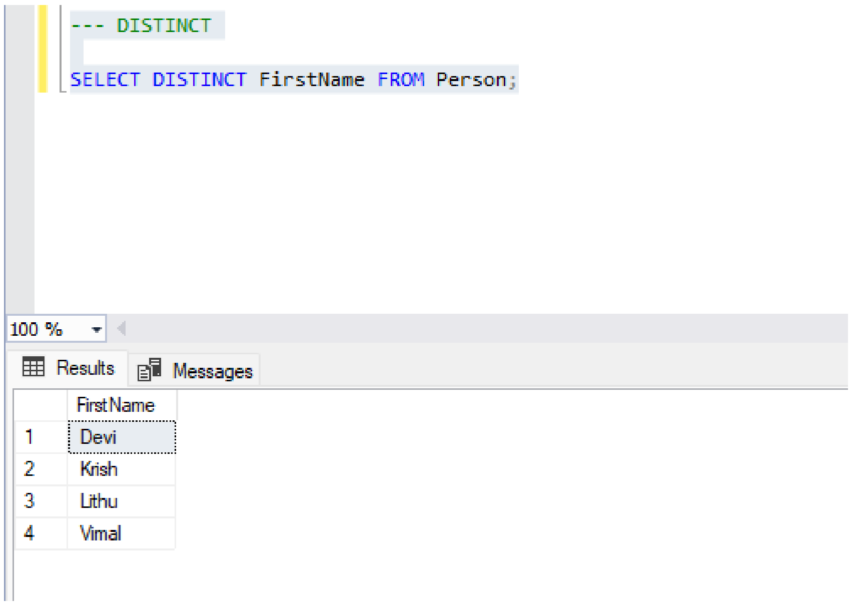
The DISTINCT keyword can be used to return only distinct (different) values.

SQL SELECT DISTINCT Syntax

```
SELECT DISTINCT column_name(s)
FROM table_name
```

SELECT DISTINCT Example

```
SELECT DISTINCT FirstName FROM Person;
```



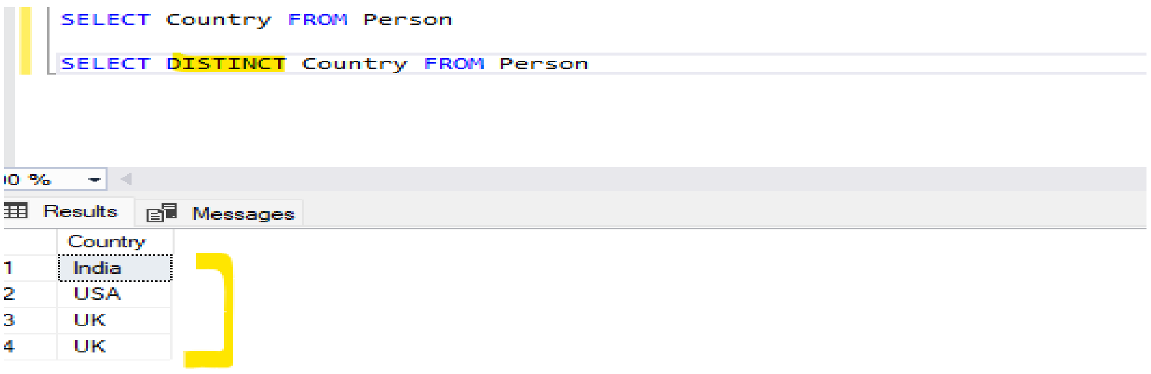
```
--- DISTINCT

SELECT DISTINCT FirstName FROM Person;
```

	FirstName
1	Devi
2	Krish
3	Lithu
4	Vimal

Now we want to select only the distinct values from the column named "Country" from the table above. We use the following SELECT statement:

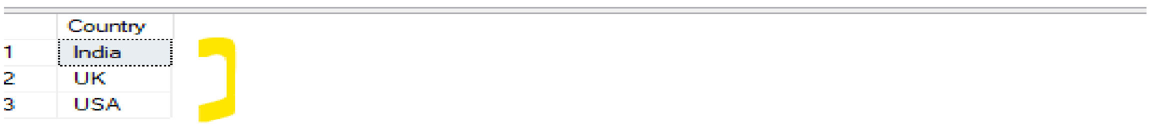
SELECT COUNTRY FROM PERSON;

SELECT DISTINCT Country FROM Person



```
SELECT Country FROM Person
SELECT DISTINCT Country FROM Person
```

	Country
1	India
2	USA
3	UK
4	UK



	Country
1	India
2	UK
3	USA

The WHERE Clause

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

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

WHERE Clause Example

The "Person" table:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	London1	UK

Now we want to select only the Person living in the COUNTRY "" from the table above.
We use the following SELECT statement:

```
SELECT * FROM Person WHERE COUNTRY ='UK'
```

P_Id	LastName	FirstName	Address
3	V	Krish	London1
4	S	Vimal	London1

The result-set will look like this:

Quotes Around Text Fields

SQL uses single quotes around text values (most database systems will also accept double quotes). Although, numeric values should not be enclosed in quotes.

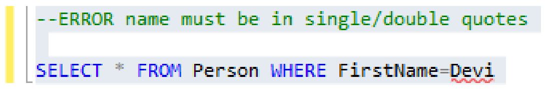
For text values:

This is correct:

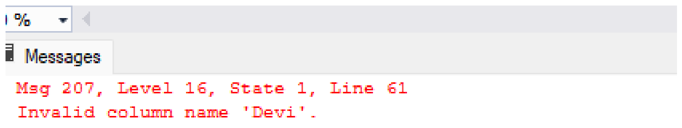
    SELECT * FROM Person WHERE FirstName='Devi'

This is wrong:

    SELECT * FROM Person WHERE FirstName=Devi



```
--ERROR name must be in single/double quotes

SELECT * FROM Person WHERE FirstName=Devi
```



```
Msg 207, Level 16, State 1, Line 61
Invalid column name 'Devi'.
```

For numeric values:

This is correct:

    SELECT * FROM Person WHERE Year=1965

This is wrong:

    SELECT * FROM Person WHERE Year='1965'

Operators Allowed in the WHERE Clause

With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

Note: In some versions of SQL the <> operator may be written as !=

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"

```
Select * from person where firstname like '%dev%'

Select * from person where P_Id between 1 and 3

select * from person where country in ('uk','India');
```

The AND & OR Operators

- The AND & OR operators are used to filter records based on more than one condition.
- The AND operator displays a record if both the first condition and the second condition is true.
- The OR operator displays a record if either the first condition or the second condition is true.

AND Operator Example

The "Person" table:

Now we want to select only the Person with the first name equal to "Devi" AND the last name equal to "R":

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE FirstName='Devi' AND LastName='R'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India

OR Operator Example

Now we want to select only the Person with the first name equal to "Devi" OR the first name equal to "Ola":

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE FirstName='Devi' OR FirstName='Ola'
```

The result-set will look like this:

Combining AND & OR

You can also combine AND and OR (use parenthesis to form complex expressions).

Now we want to select only the Person with the last name equal to "R" AND the first name equal to "Devi" OR to "Ola":

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE LastName='R' AND (FirstName='Devi' OR FirstName='Ola')
```

The result-set will look like this:

The ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set.
- The ORDER BY keyword is used to sort the result-set by a specified column.
- The ORDER BY keyword sort the records in ascending order by default.
- If you want to sort the records in a descending order, you can use the DESC keyword.
- SQL ORDER BY Syntax

```
SELECT column_name(s)
FROM table_name ORDER BY column_name(s) ASC|DESC
```

ORDER BY Example

The "Person" table:

Now we want to select all the Person from the table above, however, we want to sort the Person by their last name.

We use the following SELECT statement:

```
SELECT * FROM Person ORDER BY LastName
```

P_Id	LastName	FirstName	Address	Country
2	P R	Lithu	Los Angeles	USA
1	R	Devi	India Nagar	India
4	S	Vimal	London1	UK
3	V	Krish	London1	UK

ORDER BY DESC Example

Now we want to select all the Person from the table above, however, we want to sort the Person descending by their last name.

We use the following SELECT statement:

```
SELECT * FROM Person ORDER BY LastName DESC
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	Country
3	V	Krish	London1	UK
4	S	Vimal	London1	UK
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA

The INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.
- The first form doesn't specify the column names where the data will be inserted, only their values:

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

- The second form specifies both the column names and the values to be inserted:

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

SQL INSERT INTO Example

We have the following "Person" table:

Now we want to insert a new row in the "Person" table.

We use the following SQL statement:

```
INSERT INTO Person VALUES (5,'Neena', 'N', 'blr 1', 'India')
```

The "Person" table will now look like this:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	London1	UK
5	Neena	N	blr 1	India

Insert Data Only in Specified Columns

It is also possible to only add data in specific columns.

The following SQL statement will add a new row, but only add data in the "P_Id", "LastName" and the "FirstName" columns:

```
INSERT INTO Person (P_Id, LastName, FirstName) VALUES (6, 'Thuli', 'Jb')
```

The "Person" table will now look like this:

The UPDATE Statement

- The UPDATE statement is used to update records in a table.
- SQL UPDATE Syntax:
  - UPDATE table_name SET column1=value, column2=value2,... WHERE some_column=some_value

SQL UPDATE Example

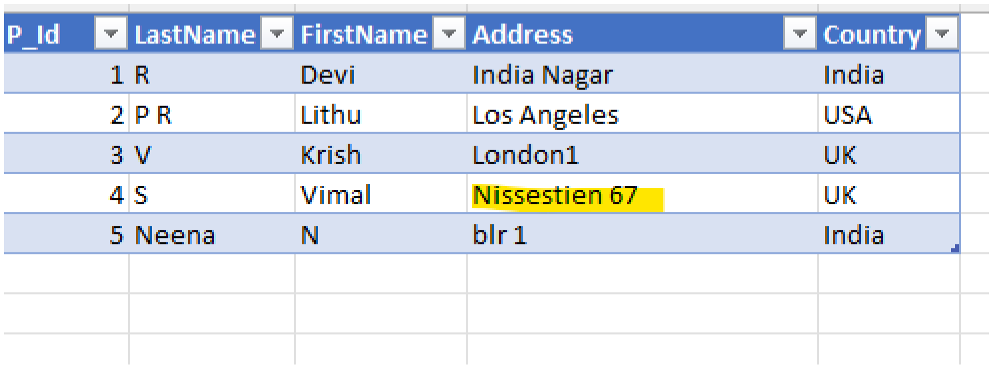
The "Person" table:

Now we want to update the person "vimal, s" in the "Person" table. We use

the following SQL statement:

```
UPDATE Person
SET Address='Nissestien 67'
WHERE LastName='s' AND FirstName='Vimal'
```

The "Person" table will now look like this:



P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	Nissestien 67	UK
5	Neena	N	blr 1	India

SQL UPDATE Warn

Be careful when updating records. If we had omitted the WHERE clause in the example above, like this:

```
UPDATE Person
SET Address='Nissestien 67', Country='UK'
```

The DELETE Statement

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

```
DELETE FROM table_name
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

SQL DELETE Example

The "Person" table:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	Nissestien 67	UK
5	Neena	N	blr 1	India

Now we want to delete the person "Neena, N" in the "Person" table. We use

the following SQL statement:

```
DELETE FROM Person
   WHERE LastName='Neena' AND FirstName='N'
```

The "Person" table will now look like this:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	Nissestien 67	UK

Delete All Rows

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
```

or

```
DELETE * FROM table_name
```

SQL ADVANCE

The TOP Clause

- The TOP clause is used to specify the number of records to return.
- The TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance.

  Note: Not all database systems support the TOP clause.

- SQL Server Syntax:

```
SELECT TOP number|percent column_name(s)
FROM table_name
```

- SQL SELECT TOP Equivalent in MySQL and Oracle:

  - MySQL Syntax:

```
SELECT column_name(s)
FROM table_name
LIMIT number
```

Example:

```
SELECT *
FROM Person
LIMIT 5
```

- Oracle Syntax

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

Example

```
SELECT *
FROM Person
WHERE ROWNUM <=5
```

SQL TOP Example

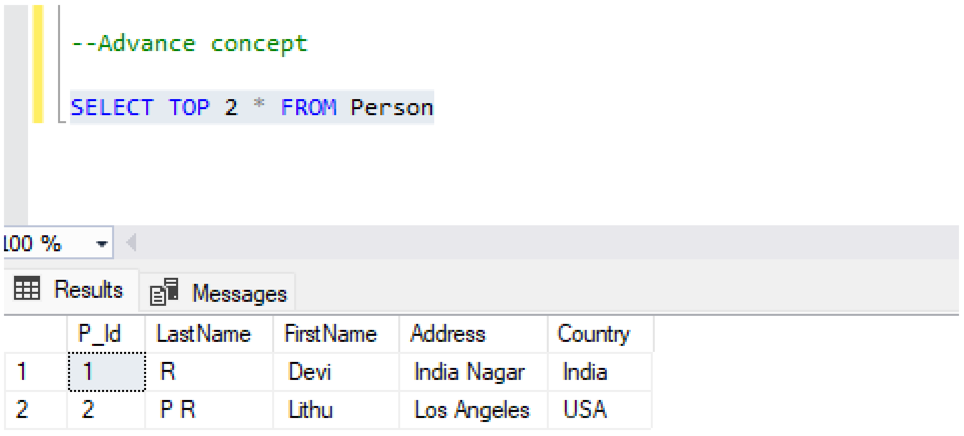
The "Person" table:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	Nissestien 67	UK

Now we want to select only the two first records in the table above.

We use the following SELECT statement:

SELECT TOP 2 * FROM Person



```
--Advance concept

SELECT TOP 2 * FROM Person
```

	P_Id	LastName	FirstName	Address	Country
1	1	R	Devi	India Nagar	India
2	2	P R	Lithu	Los Angeles	USA

SQL TOP PERCENT Example

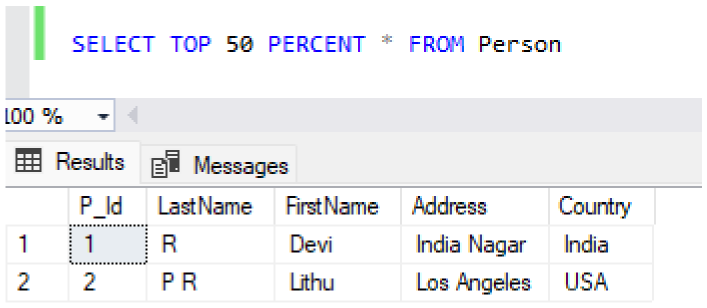
The "Person" table:

Now we want to select only 50% of the records in the table above.

We use the following SELECT statement:

SELECT TOP 50 PERCENT * FROM Person

The result-set will look like this:



```
SELECT TOP 50 PERCENT * FROM Person
```

	P_Id	LastName	FirstName	Address	Country
1	1	R	Devi	India Nagar	India
2	2	P R	Lithu	Los Angeles	USA

SQL Wildcards

- SQL wildcards can be used when searching for data in a database.
- SQL wildcards can substitute for one or more characters when searching for data in a database.

- SQL wildcards must be used with the SQL LIKE operator.
- With SQL, the following wildcards can be used:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for exactly one character

SQL Wildcard Examples

We have the following "Person" table:

P_Id	LastName	FirstName	Address	Country
1	R	Devi	India Nagar	India
2	P R	Lithu	Los Angeles	USA
3	V	Krish	London1	UK
4	S	Vimal	Nissestien 67	UK

Using the % Wildcard

Now we want to select the Person living in a city that starts with "sa" from the "Person" table.

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE Country LIKE 'u%'
```

Next, we want to select the Person living in a city that contains the pattern "es" from the "Person" table.

We use the following SELECT statement:

SELECT * FROM Person WHERE Address LIKE '%es%'

Using the _ Wildcard

Now we want to select the Person with a first name that starts with any character, followed by "la" from the "Person" table.

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE FirstName LIKE
'_ev'
```

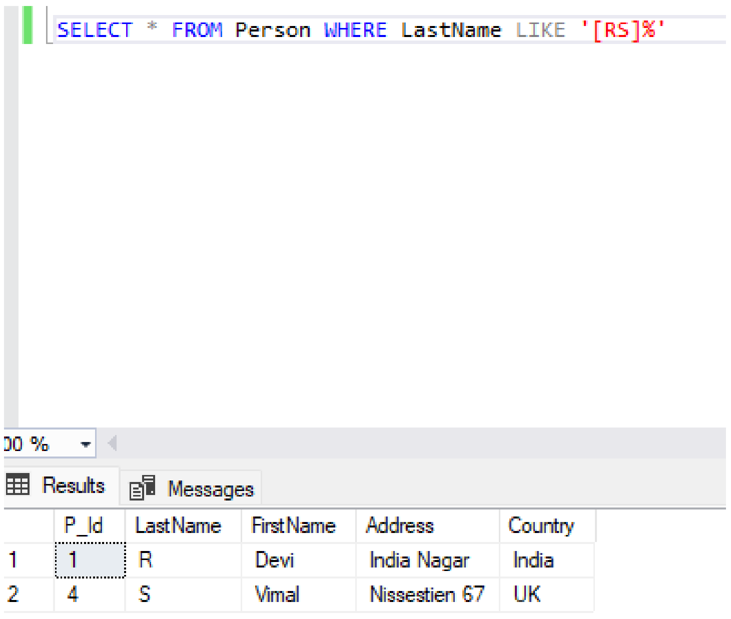
Using the [charlist] Wildcard

Now we want to select the Person with a last name that starts with "b" or "s" or "p" from the "Person" table.

We use the following SELECT statement:

```
SELECT * FROM Person
WHERE LastName LIKE '[RS]%'
```

The result-set will look like this:



```
SELECT * FROM Person WHERE LastName LIKE '[RS]%'
```

	P_Id	LastName	FirstName	Address	Country
1	1	R	Devi	India Nagar	India
2	4	S	Vimal	Nissestien 67	UK

Next, we want to select the Person with a last name that do not start with "R" or "S" from the "Person" table.

We use the following SELECT statement:

```
SELECT * FROM Person WHERE
LastName LIKE '[!RS]%'
```

The LIKE Operator

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

- SQL LIKE Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern
```

The IN Operator

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

- SQL IN Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,..
```

IN Operator Example

The "Person" table:

We use the following SELECT statement:

```
select * from person where country

in('uk','India')
```

The BETWEEN Operator

- The BETWEEN operator is used in a WHERE clause to select a range of data between two values.
- The BETWEEN operator selects a range of data between two values. The values can be numbers, text, or dates.

- SQL BETWEEN Syntax:

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

SQL Alias

- With SQL, an alias name can be given to a table or to a column.
- You can give a table or a column another name by using an alias. This can be a good thing to do if you have very long or complex table names or column names.
- An alias name could be anything, but usually it is short.

- SQL Alias Syntax for Tables:

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

- SQL Alias Syntax for Columns:

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

The SQL SELECT INTO Statement

- The SQL SELECT INTO statement can be used to create backup copies of tables.
- The SELECT INTO statement selects data from one table and inserts it into a different table.
- The SELECT INTO statement is most often used to create backup copies of tables.
- SQL SELECT INTO Syntax

We can select all columns into the new table:

```
SELECT *
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```

Or we can select only the columns we want into the new table:

```
SELECT column_name(s)
INTO new_table_name [IN externaldatabase]
FROM old_tablename
```