

Contents

- 1. RDBMS Concepts*
- 2. SQL DDL Statements*
- 3. SQL Syntax*

1. RDBMS Concepts

RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access. A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.

The data in RDBMS is stored in database objects called tables. The table is a collection of related data entries and it consists of columns and rows. Remember, a table is the most common and simplest form of data storage in a relational database. Following is the example of a Student Information table:

	Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
▶	1	Ali	Mohammed	Kamil	1995-10-20	M
	2	Alvaa	Mohammed	Salim	1990-10-10	F
	3	Ameer	Samir	Hamid	1994-08-30	M
	4	Afnan	Hamid	Noman	1996-10-10	F
	5	Ahmed	Sami	Haleem	1996-01-02	M
	6	Avmen	Mohammed	Mustafa	1996-01-01	M
	7	Abbas	Fadhel	Hatem	1995-01-20	M
	8	Havat	Amer	Essa	1996-10-09	F

Every table is broken up into smaller entities called fields. The fields in the Students Information (stud_info table) consist of Stud_ID, F_Name, S_Name, T_Name, BirthDate and Gender. A field is a column in a table that is designed to maintain specific information about every record in the table. A record, also called a row of data, is each individual entry that exists in a table. For example, there are 8 records in the above Student Information table. Following is a single row of data or record in the stud_info table:

▶	1	Ali	Mohammed	Kamil	1995-10-20	M
---	---	-----	----------	-------	------------	---

A record is a horizontal entity in a table. A column is a vertical entity in a table that contains all information associated with a specific field in a table. For example, a column in the stud_info table is BirthDate, which represents location description and would consist of the following:

BirthDate
1995-10-20
1990-10-10
1994-08-30
1996-10-10
1996-01-02
1996-01-01
1995-01-20
1996-10-09

2. SQL DDL Statements

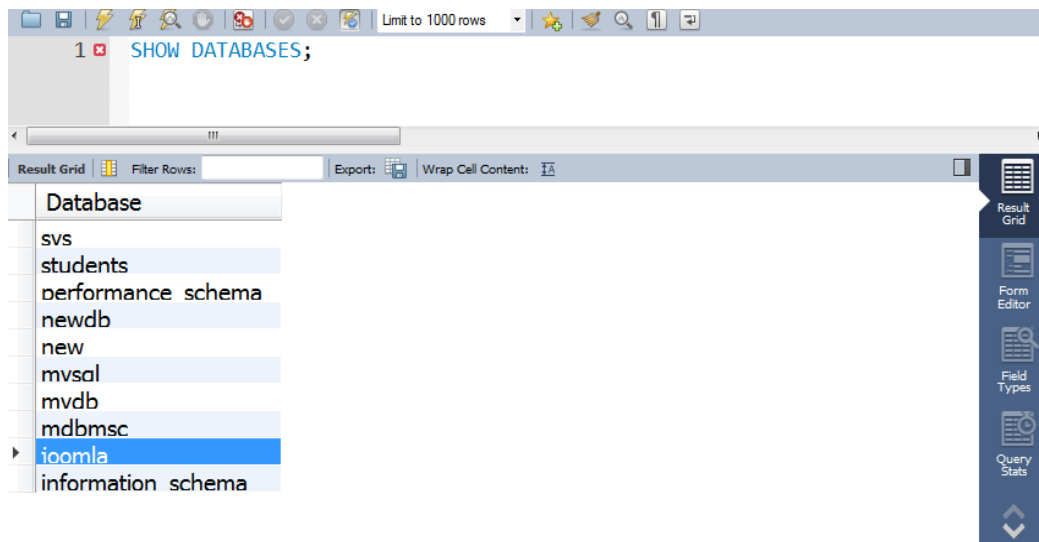
2.1 SQL Create Statement

2.1.1 Create Database

The CREATE DATABASE statement is used to create a database. Always database name should be unique within the RDBMS.

CREATE DATABASE dbname;

Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases as follows:



2.1.2 Create Table

The CREATE TABLE statement is used to create a table in a database. Tables are organized into rows and columns; and each table must have a name.

```
CREATE TABLE table_name(column1 datatype,  
column2 datatype,  
column3 datatype,  
.....  
columnN datatype,  
PRIMARY KEY( one or more columns )  
);
```

The column_name parameters specify the names of the columns of the table. The data_type parameter specifies what type of data the column can hold (e.g. varchar, integer, decimal, date, etc.). The size parameter specifies the maximum length of the column of the table. A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

The new table has the same column definitions. All columns or specific columns can be selected. When you create a new table using existing table, new table would be populated using existing values in the old table.

```
CREATE TABLE NEW_TABLE_NAME AS
SELECT [ column1, column2...columnN ]
FROM EXISTING_TABLE_NAME
[ WHERE ]
```

Example: create table stud_id which contains Stud_ID and Gender from stud_info:

The screenshot shows a SQL IDE interface. The top pane contains the following SQL script:

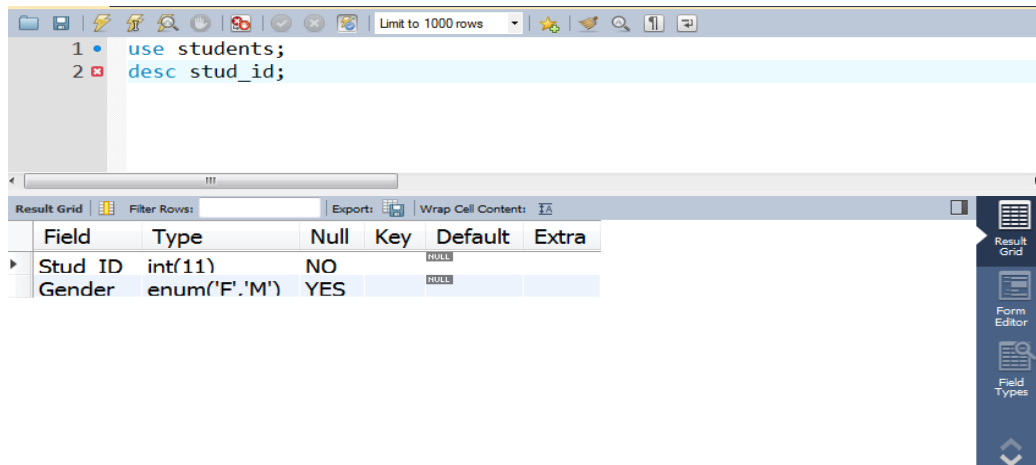
```
1 • use students;
2 ✖ create table stud_id as
3 ✖ select Stud_ID, Gender from stud_info;
4 ✖ select * from stud_id;
```

The bottom pane displays the 'Result Grid' with the following data:

	Stud_ID	Gender
▶	1	M
	2	F
	3	M
	4	F
	5	M
	6	M
	7	M
	8	F

The interface also includes a toolbar at the top with icons for file operations, a 'Limit to 1000 rows' dropdown, and a right sidebar with 'Result Grid', 'Form Editor', and 'Field Types' options.

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use **DESC** command as follows:



2.1.3 Create Index

The CREATE INDEX statement is used to create indexes in tables. Indexes allow the database application to find data fast; without reading the whole table. An index can be created in a table to find data more quickly and efficiently. The users cannot see the indexes, they are just used to speed up searches/queries. Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So you should only create indexes on columns (and tables) that will be frequently searched against.

```
CREATE UNIQUE INDEX index_name  
ON table_name ( column1, column2,...columnN);
```

2.2 Alter Statement

2.2.1 Alter Table

The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table. You would also use ALTER TABLE command to add and drop various constraints on an existing table.

The basic syntax of **ALTER TABLE** to add a new column in an existing table is as follows:

```
ALTER TABLE table_name ADD column_name datatype;
```

The basic syntax of ALTER TABLE to **DROP COLUMN** in an existing table is as follows:

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

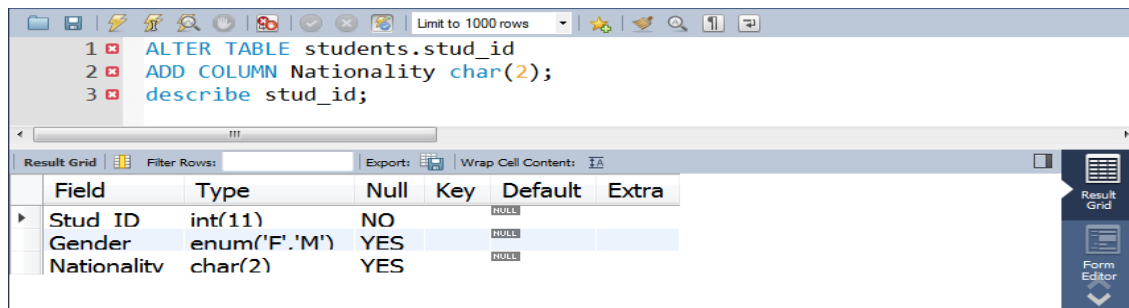
The basic syntax of ALTER TABLE to change the **DATA TYPE** of a column in a table is as follows:

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

The basic syntax of ALTER TABLE to add a **NOT NULL** constraint to a column in a table is as follows:

```
ALTER TABLE table_name MODIFY column_name datatype NOT NULL;
```

Example: add column nationality with two character to table stud_id:



The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows:

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);
```

The basic syntax of ALTER TABLE to **ADD CHECK CONSTRAINT** to a table is as follows:

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

The basic syntax of ALTER TABLE to **ADD PRIMARY KEY** constraint to a table is as follows:

```
ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

The basic syntax of ALTER TABLE to **DROP CONSTRAINT** from a table is as follows:


```
ALTER TABLE table_name  
DROP CONSTRAINT MyUniqueConstraint;
```

If you're using MySQL, the code is as follows:

```
ALTER TABLE table_name  
DROP INDEX MyUniqueConstraint;
```

The basic syntax of ALTER TABLE to **DROP PRIMARY KEY** constraint from a table is as follows:

```
ALTER TABLE table_name  
DROP CONSTRAINT MyPrimaryKey;
```

If you're using MySQL, the code is as follows:

```
ALTER TABLE table_name  
DROP PRIMARY KEY;
```

2.3 SQL Drop Statement

2.3.1 Drop Database

The DROP DATABASE statement is used to delete a database.

DROP DATABASE database_name;

2.3.2 Drop Table

The DROP TABLE statement is used to delete a table.

DROP TABLE table_name;

2.3.3 Drop Index

The DROP INDEX statement is used to delete an index in a table.

```
ALTER TABLE table_name DROP INDEX index_name;
```

3. SQL Syntax

SQL is followed by unique set of rules and guidelines called Syntax. This section gives you a quick start with SQL by listing all the basic SQL Syntax:

All the SQL statements start with any of the keywords like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW and all the statements end with a semicolon (;).

Important point to be noted is that SQL is case insensitive, which means SELECT and select have same meaning in SQL statements, but MySQL makes difference in table names. So if you are working with MySQL, then you need to give table names as they exist in the database.

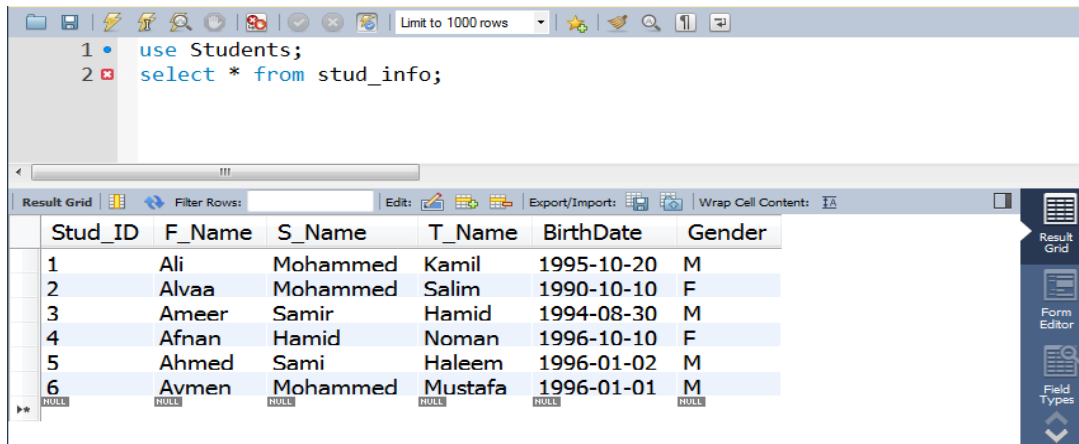
3.1 SQL SELECT Statement

```
SELECT column1, column2....columnN  
FROM table_name;
```

The SELECT statement (DQL Statement) is used to select data from a database. The result is stored in a result table, called the result-set. You can select specific columns from table or select all columns. To select all columns use command:

```
SELECT * FROM table_name;
```

Example: select all rows from table 'stud_info':



The screenshot shows a database query tool interface. The SQL editor contains two lines of code: `1 • use Students;` and `2 ✖ select * from stud_info;`. Below the editor, the 'Result Grid' displays a table with 6 rows and 6 columns: **Stud_ID**, **F_Name**, **S_Name**, **T_Name**, **BirthDate**, and **Gender**. The data rows are as follows:

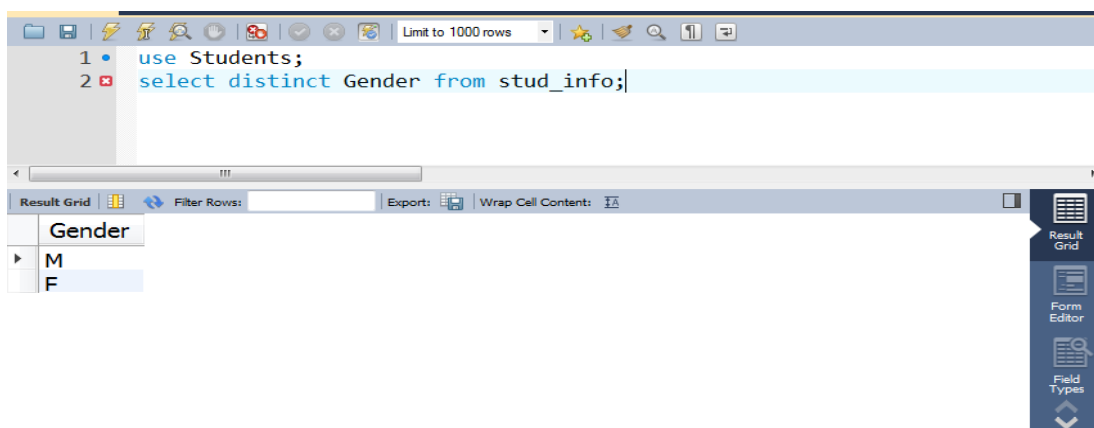
Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
1	Ali	Mohammed	Kamil	1995-10-20	M
2	Alvaa	Mohammed	Salim	1990-10-10	F
3	Ameer	Samir	Hamid	1994-08-30	M
4	Afnan	Hamid	Noman	1996-10-10	F
5	Ahmed	Sami	Haleem	1996-01-02	M
6	Avmen	Mohammed	Mustafa	1996-01-01	M

3.2 Distinct Clause

The `SELECT DISTINCT` statement is used to return only distinct (different) values. In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values. The `DISTINCT` keyword can be used to return only distinct (different) values.

```
SELECT DISTINCT column1, column2,...columnN
FROM table_name;
```

Example: show only the distinct values in column (Gender):



The screenshot shows the same database query tool interface. The SQL editor now contains: `1 • use Students;` and `2 ✖ select distinct Gender from stud_info;`. The 'Result Grid' displays a table with a single column: **Gender**. The data rows are:

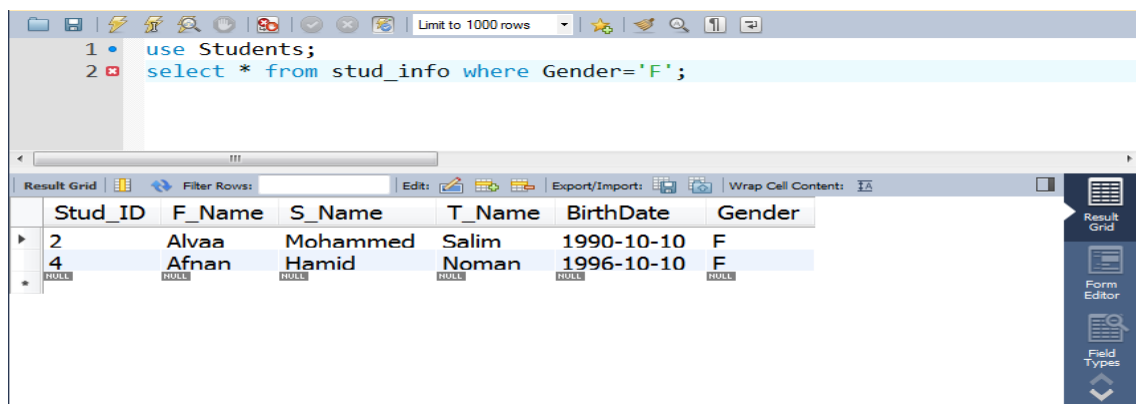
Gender
M
F

3.3 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.

```
SELECT column1, column2,...columnN
FROM   table_name
WHERE  CONDITION;
```

Example: show only the female students:



SQL requires single quotes around text values (most database systems will also allow double quotes). However, numeric fields should not be enclosed in quote. There are many operators can be user with WHERE clause:

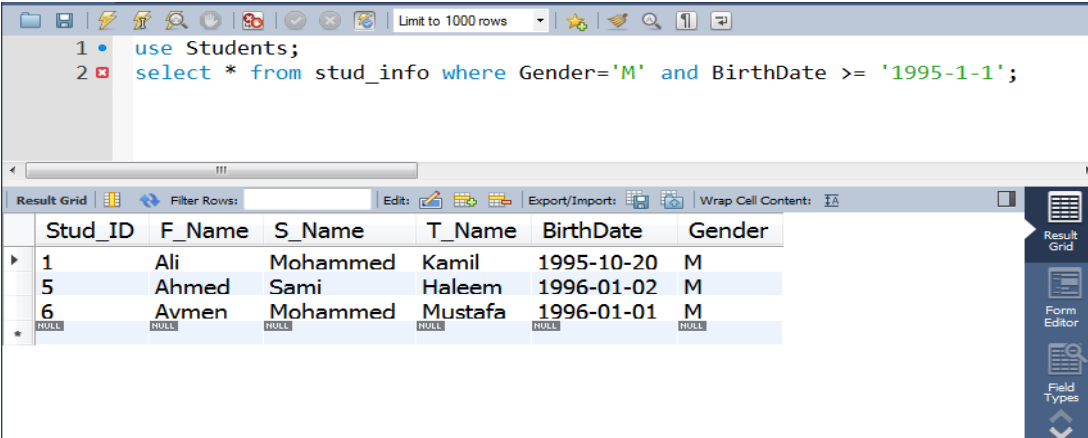
Operator	Description
=	Equal
<>	Not equal. Note: In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

3.3.1 AND/OR Clause

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 are true. The OR operator displays a record if either the first condition OR the second condition is true.

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION-1 {AND|OR} CONDITION-2;
```

Example: show information of male students where birthdate greater than or equal '1995-1-1':



The screenshot shows a database query tool interface. The SQL query editor contains the following code:

```
1 • use Students;
2 select * from stud_info where Gender='M' and BirthDate >= '1995-1-1';
```

The results are displayed in a table with the following columns: Stud_ID, F_Name, S_Name, T_Name, BirthDate, and Gender. The table contains three rows of data:

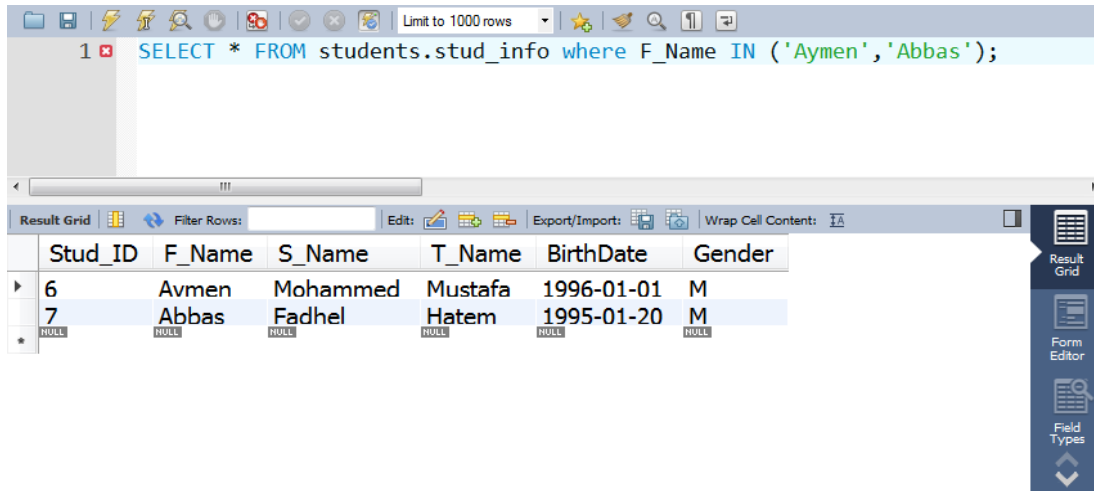
Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
1	Ali	Mohammed	Kamil	1995-10-20	M
5	Ahmed	Sami	Haleem	1996-01-02	M
6	Avmen	Mohammed	Mustafa	1996-01-01	M

3.3.2 IN Clause

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

```
SELECT column1, column2....columnN
FROM   table_name
WHERE  column_name IN (val-1, val-2,...val-N);
```

Example: show only students information whom names are Aymen and Abbas:



The screenshot shows a database query tool interface. At the top, a SQL query is entered: `SELECT * FROM students.stud_info where F_Name IN ('Aymen','Abbas');`. Below the query, the results are displayed in a table with the following columns: Stud_ID, F_Name, S_Name, T_Name, BirthDate, and Gender. The results show two rows: one for Aymen (Stud_ID 6) and one for Abbas (Stud_ID 7). The table is displayed in a 'Result Grid' view.

Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
6	Aymen	Mohammed	Mustafa	1996-01-01	M
7	Abbas	Fadhel	Hatem	1995-01-20	M

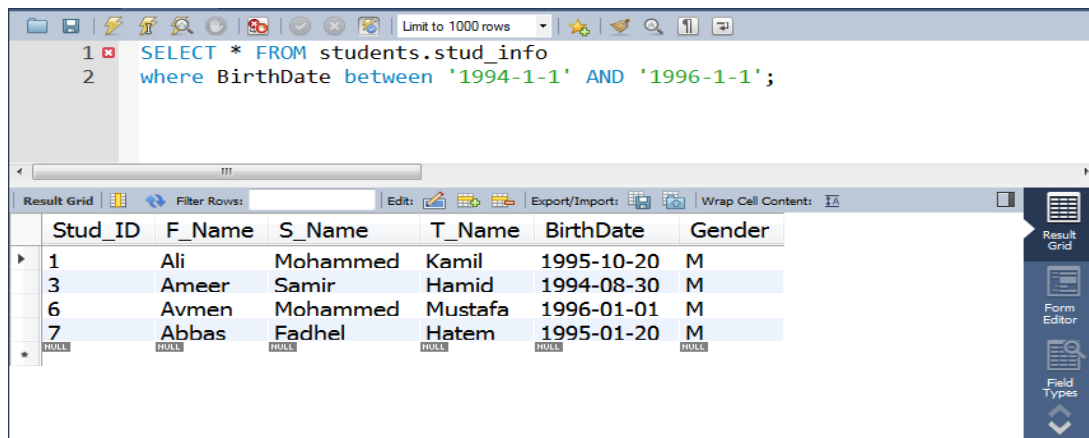
3.3.3 BETWEEN Clause

The BETWEEN operator is used to select values within a range. The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

```
SELECT column1, column2...columnN
FROM table_name
WHERE column_name BETWEEN val-1 AND val-2;
```

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

Example: show students information of students whom birthdate between '1994-1-' and '1996-1-1':



The screenshot shows a database query tool interface. The SQL query entered is:

```
1 SELECT * FROM students.stud_info
2 where BirthDate between '1994-1-1' AND '1996-1-1';
```

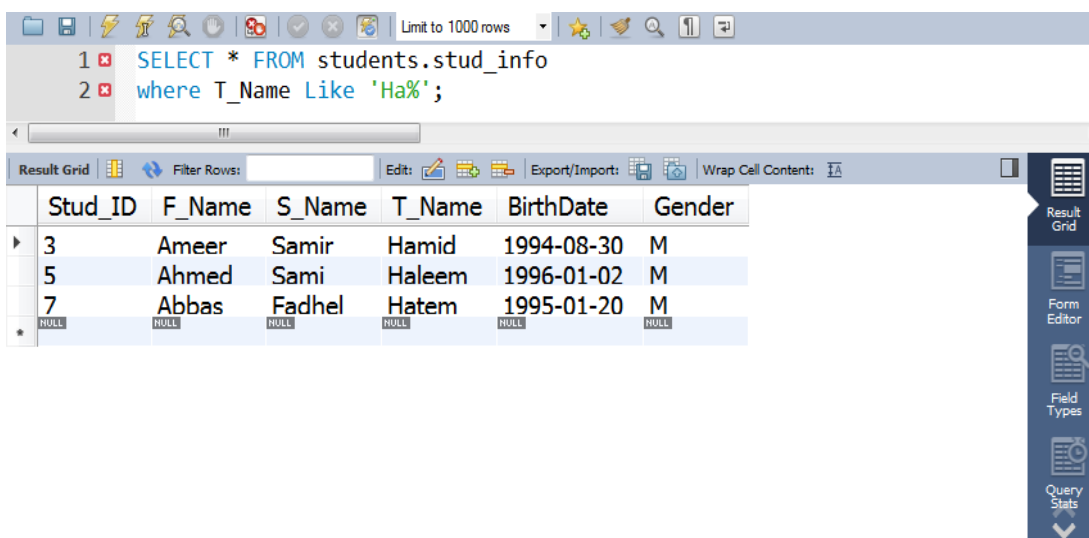
The results are displayed in a table with the following columns: Stud_ID, F_Name, S_Name, T_Name, BirthDate, and Gender. The data rows are:

Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
1	Ali	Mohammed	Kamil	1995-10-20	M
3	Ameer	Samir	Hamid	1994-08-30	M
6	Avmen	Mohammed	Mustafa	1996-01-01	M
7	Abbas	Fadhel	Hatem	1995-01-20	M
NULL	NULL	NULL	NULL	NULL	NULL

3.3.4 LIKE Clause

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.

Example: show only students information whom their third names start with 'Ha':



The screenshot shows a database query tool interface. The SQL query entered is:

```
1 SELECT * FROM students.stud_info
2 where T_Name Like 'Ha%';
```

The results are displayed in a table with the following columns: Stud_ID, F_Name, S_Name, T_Name, BirthDate, and Gender. The data rows are:

Stud_ID	F_Name	S_Name	T_Name	BirthDate	Gender
3	Ameer	Samir	Hamid	1994-08-30	M
5	Ahmed	Sami	Haleem	1996-01-02	M
7	Abbas	Fadhel	Hatem	1995-01-20	M
NULL	NULL	NULL	NULL	NULL	NULL

The "%" sign is used to define wildcards (missing letters) both before and after the pattern. Using the NOT keyword allows you to select records that do NOT match

the pattern. A wildcard character can be used to substitute for any other character(s) in a string. In SQL, wildcard characters are used with the SQL LIKE operator. SQL wildcards are used to search for data within a table.

With SQL, the wildcards are:

Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for a single character
[<i>charlist</i>]	Sets and ranges of characters to match
[<i>^charlist</i>] or [<i>!charlist</i>]	Matches only a character NOT specified within the brackets