Database Designs

Section 2: Database Tables

1.1 Introducing the SELECT Statement

Using the below Students Table we will look at how we can query the table to retrieve information from the database using SQL SELECT Statement.

STUDENTS				
studentID	firstName	lastName	gender	email
1	John	Doe	M	j.d@email.com
2	Jenny	Smith	F	j.s@email.com
3	Navdeep	Singh	F	n.s@email.com
4	Alberto	Diaz	M	a.d@email.com
5	Martha	Blyth	F	m.b@email.com

The SQL SELECT Statement is used to fetch data out of a database table(s). The SELECT statement is used to fetch specific rows and columns out of a database. Below is a very basic example of fetching data from a database using the SELECT Statement SQL Syntax.

We need to always answer three simple questions:
1. Which rows do we want to fetch?
2. Which columns in those rows are we interested in?
3. From which tables should we fetch this information from?

These are the three questions you need to think about in order to structure your thoughts and structure the SELECT statement.

If we want to get all emails of female students we can use the three questions to structure our thoughts and our SELECT statement to retrieve the data. First we are interested only in the rows which are female students. We are interested only in the column that holds the emails. Finally, we are interested in the Students Table which holds the information we want to extract. Having answered these questions we can now write the SQL query:

```
SELECT email
FROM STUDENTS
WHERE gender = 'F';
```

The keywords are highlighted in blue. These keywords are uniformly used to form the syntax of a basic SELECT statement. We would use these three keywords in a variety of ways to extract the most complicated of information from a database.

The WHERE keyword is used to answer the first question noted above i.e. which rows do we want to fetch. The WHERE clause is followed by a list of conditions which is used to figure out which of the rows we are interested in. Whichever rows satisfies the condition is selected as part of the select statement.
If we wanted all student emails from the database we can simply omit the WHERE clause from the SELECT statement.

The SELECT keyword is used to answer the second question noted above i.e. which columns are we interested in fetching. We select all the columns by the column names existing in the table separated by a comma. These are the values whose values will be retrieved for the records that satisfies the WHERE condition above.
The asterisk (*) is a special character to select all columns from a table rather than listing each column individually.

Finally, the FROM keyword is used to answer the last question noted above i.e. which table should we fetch the rows and columns data from. This should be a table stored in the database being queried.

The semicolon (;) at the end of the statement is extremely important because it terminates/ends the statement. This indicates to the DBMS system that we have completed our Query Statement.

Below is an example of a SELECT statement where we want to fetch all columns of all female students with the first name of Jenny.

```
SELECT *
FROM STUDENTS
WHERE gender = 'F' AND firstName = 'Jenny';
```

Below is an example of a SELECT statement where we want to fetch the student id and email of all female students or students with the first name of Alberto.

```
SELECT studentID, email
FROM STUDENTS
WHERE gender = 'F' OR firstName = 'Alberto';
```

The logical AND/OR operator is used to add multiple WHERE queries. We can chain the logical operators to produce more complex and interesting queries.

1.2 Columns Data Type

Database Columns have data types. There are different data types such as Strings, Numbers, Boolean, Null, etc. The data types for columns are specified when the tables are created. The data types of columns govern how a column is treated in SQL queries. Below is a table of the various data types that can be created for databased tables:

Data Type	Description
Char	Holds fixed length strings.
Varchar	Holds variable length strings.
Int	Holds integer values i.e. full numbers.
Decimal	Holds floating point values i.e. decimal numbers.
DateTime	Holds the date and time stamp.
Date	Holds only the date stamp.
Time	Holds only the time stamp.
Blob	Holds binary larg objects. This holds data types that are not easily represented by the other data types.

1.3 Single Quotes, Escapes and NULLS

Data types of Char, Vachar, DateTime, Date and Time values all need to be enclosed in Single Quotes. What is the string itself contains a single quote? The escape character allows us to escape the single quote with a backslash (\ ).

```
SELECT buildingID
FROM property_address
WHERE buildingname = 'Akbar\'s House';
```

The escape characters tells the database that the character after the backslash should be taken literally and any special character recognition that character has is no longer true (i.e. accept the character literally as it).

Different database systems can accept different characters as the special escape characters. For example some databases escape a single quote with another preceding single quote. This will depend on the parsing rules of the database.

```
SELECT buildingID
FROM property_address
WHERE buildingname = 'Akbar''s House';
```

The NULL data value implies that a value does not exist. Null is not the same as a blank string or the number zero. A Blank or a zero number is a value that exists. Any columns can contain a value of NULL and this can be defaulted to a column if no values are specified for the column for the data record.

To specify whether a column can have a value of NULL is done at the creation of the table. We have to define the table in a way that allows NULL values in that column.

The NULL values is neither True or False and is just a NULL/Non-existent value. To query a database for NULL values the syntax is slightly different.

```
SELECT studentID
FROM STUDENTS
WHERE email IS NULL;
```

```
SELECT studentID
FROM STUDENTS
WHERE email IS NOT NULL;
```

The equal logical operator (=) checks whether a value exists in a table. As mentioned above NULL is the absence of value. Therefore, we cannot use the equal logical operator to find a NULL value. Instead we would is IS NULL or IS NOT NULL to query whether a NULL value does or does not exist in the table.

1.4 Using the LIKE Operator

SQL allows us to use the **LIKE** keyword within the **WHERE** clause to retrieve data that contains a specific string within a string whether at the beginning, end or in-between the string. The below example demonstrates how to use the **LIKE** keyword to find 'gmail' within the string of the email column.

SELECT	Which Columns?	**email**
FROM	Which Tables?	**STUDENTS**
WHERE	Which Rows?	**email contains the string 'gmail'**

```
SELECT email
FROM STUDENTS
WHERE email LIKE '%gmail%';
```

The **LIKE** is a special keyword similar to **IS NULL** and **IS NOT NULL** special keywords but it allows us to use something called Wildcards. Wildcards are similar to wildcards in Regular Expressions and the symbols have special meanings. A wildcards can match portion of strings and the percentage symbol (**%**) is a wildcard.

The **%** wildcard means anything of any length i.e. this can be made up of characters, numbers, special characters, etc. In the above example the first **%** allows for anything before the 'gmail' string while the second **%** allows for anything after the 'gmail' string, even if that anything is nothing. Therefore, the **LIKE** keyword will match any string contain the string 'gmail' even if the string itself is 'gmail'.

The _ wildcard means anything of a length that is exactly one character i.e. whatever fills that underscore in the string position should be exactly one random character.

Wildcards are extremely useful; however, the make queries execute very slowly due to all the processing the database has to do in order to check and match against the wildcards.