Functional Testing :(Frond End (UI) +Backend Testing (Database Testing))

- Behavior coverage
- Input domain coverage
- Error handling coverage
- Service level coverage
- Calculation based Coverage
- **Backend Coverage(Database coverage/Database Testing)**

Backend coverage testing / database testing

- Validating all **frond end operations** are **stored in database**
- Front end operation or **data entered in the front end is stored in the database** or backend
- As a tester, we check or validate, whenever data has been entered in the front end that data has been stored in the database or not **(validating front end with backend)**
- We check or validate data can be **fetched or not**
- So, we simply fire the **SQL queries** & we get response in the form of table (rows & column), where we check whether **data is stored or not**

Or

- SQL query is used to **fetch data or to check stored data**

Web Based application:

Ex.1: Sign Up Form: (First Name, Last Name, Email, Mob, & Address): If we have to check on UI == to check if signup is success. We can check for success message.

Customers_Sign_up table:

- Validate Existence of the values in the database table
- Validate Correctness of the values in the database table
- Validate Completeness of the values in a database table

Database Systems:

Database: Storage Location: Collection of related data.

- **Structured Data**- IRCTC, Facebook, Flipkart, Amazon, MSEB Board, MPSC
- **Unstructured Data**- Random Data—Social Media Post- Data can not be stored in db in the form of Database.

Database Management Systems (DBMS):

- To perform any operations on the data we need the database management systems
- SQL Server,MySQL,Oracle,DB2,MongoDB

When we deal with structured data we use **RDBMS**.

SQL

1. **SQL** (*Structured Query Language*) is **used** to **perform operations** on the **records stored** in the **database** such as **updating** records, **deleting** records, **creating** and **modifying tables**, views, etc.
2. SQL is just a **query language**; it is **not a database**. To **perform** SQL queries, you **need** to **install** any **database**, for example, Oracle, MySQL, MongoDB, PostGre SQL, SQL Server, DB2, etc.

What is SQL?

1. SQL stands for **Structured Query Language**.
2. It is **designed** for **managing data** in a relational database management system (**RDBMS**).
3. It is pronounced as S-Q-L or sometime **See-Qwell**.
4. SQL is a database language, it is **used** for database creation, deletion, fetching rows, and modifying rows, etc.

- All **DBMS** like MySQL, Oracle, MS Access, Sybase, Informix, PostgreSQL, and SQL Server use SQL as **standard** database language.

Why SQL is required?

SQL is required:

1. To **create** new databases, tables and views
2. To **insert** records in a database
3. To **update** records in a database
4. To **delete** records from a database
5. To **retrieve** data from a database

What SQL does?

- With SQL, we can query our database in several ways, using English-like statements.
- With SQL, a user can access data from a relational database management system.
- It allows the user to describe the data.
- It allows the user to define the data in the database and manipulate it when needed.
- It allows the user to create and drop database and table.
- It allows the user to create a view, stored procedure, function in a database.
- It allows the user to set permission on tables, procedures, and views.

SQL Syntax

SQL **follows** some **unique** set of **rules** and **guidelines** called **syntax**. Here, we are providing all the basic SQL syntax.

- **SQL** is **not** case sensitive. Generally SQL **keywords** are **written** in **uppercase.**
- SQL statements are **dependent** on text lines. We can place a single SQL statement on one or multiple text lines.
- You can perform most of the **action** in a database **with** SQL statements.

SQL statement

SQL statements are **started** with any of the SQL **commands/keywords** like SELECT, INSERT, UPDATE, DELETE, ALTER, DROP etc. and the **statement ends with a semicolon (;).**

Example of SQL statement:

1. **SELECT** *"column_name"* **FROM** *"table_name";*

Why semicolon is used after SQL statements:

Semicolon is **used** to **separate** SQL statements. It is a **standard way** to **separate** SQL statements in a **database system** in which more than one SQL statements are used in the same call.

SQL Commands

These are the some important SQL command:

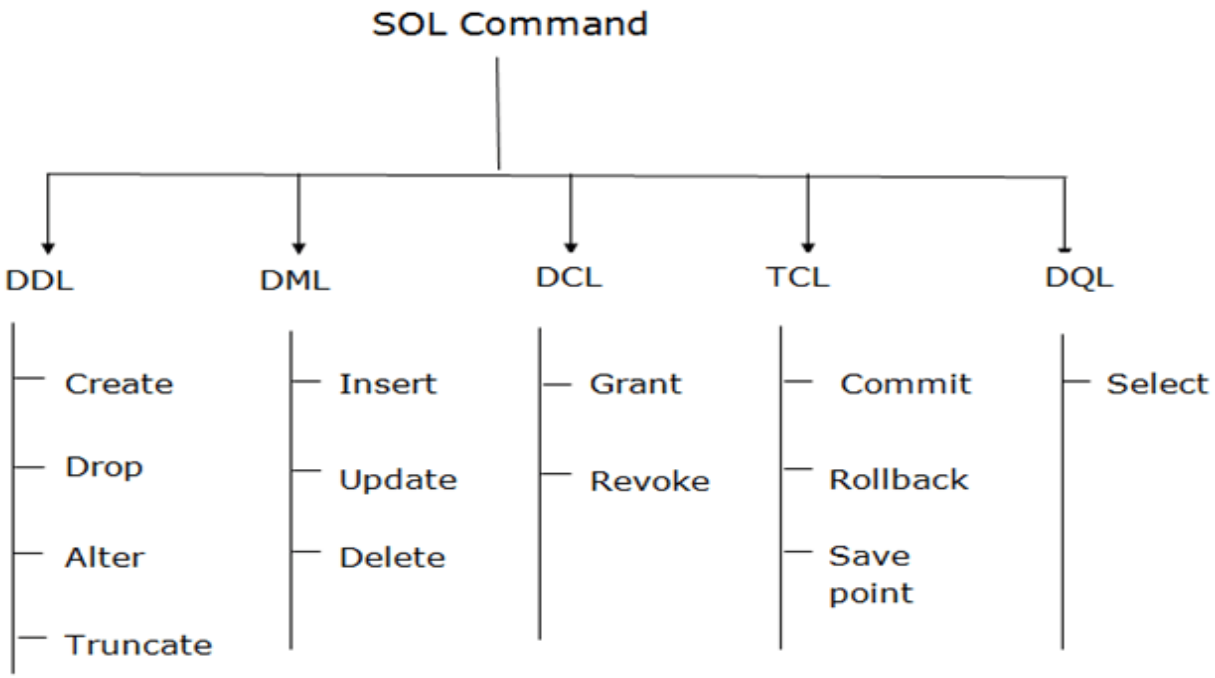
- **SELECT:** it extracts data from a database.
- **UPDATE:** it updates data in database.
- **DELETE:** it deletes data from database.
- **CREATE TABLE:** it creates a new table.
- **ALTER TABLE:** it is used to modify the table.
- **DROP TABLE:** it deletes a table.
- **CREATE DATABASE:** it creates a new database.
- **ALTER DATABASE:** It is used to modify a database.
- **INSERT INTO:** it inserts new data into a database.
- **CREATE INDEX:** it is used to create an index (search key).
- **DROP INDEX:** it deletes an index.

SQL Commands

- SQL commands are **instructions**. It is **used** to communicate **with** the database. It is also **used** to **perform** specific tasks, functions, and queries of **data**.
- SQL can **perform** various **tasks** like create a table, add data to tables, drop the table, modify the table, and set **permission for users**.

Types of SQL Commands

There are **five** types of SQL commands: **DDL, DML, DCL, TCL, and DQL.**



1. Data Definition Language (DDL)

- DDL **changes** the structure of the table **like** creating a table, deleting a table, altering a table, etc.
- **All** the **command** of DDL are **auto-committed** that **means** it permanently **save** all the **changes** in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

2. Data Manipulation Language

- DML commands are **used** to **modify** the database. It is responsible for all form of changes in the database.
- The command of DML is **not auto-committed** that means it **can't permanently save** all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

3. Data Query Language

DQL is **used** to **fetch** the data from the database.

It uses only one command:

- SELECT

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Sign Up" or "Login Credentials" or “Orders” or “Employ Info”). Tables contain records (rows) with data.

Table Name- **Sign Up**

ID	**FN**	**LN**	**MobNo**	**EmailId**	**City**
1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	Anna	Hajare	3333	3@gmail.com	Pune
4	Sharad	Pawar	4444	4@gmail.com	Baramati
5	Soniya	Gandhi	5555	5@gmail.com	Pune

The table above contains five records (one for each user) and 6 columns (ID, FN, LN, MobNo, EmailId, & City).

1. **SQL SELECT Statement**

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

The following SQL statement selects or fetch all the records in the "SignUp" table:

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

SELECT Syntax

```
SELECT * FROM table_name;
```

Example

```
SELECT * FROM SignUp;
```

Output / Result

ID	FN	LN	MobNo	EmailId	City
1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	Anna	Hajare	3333	3@gmail.com	Pune
4	Sharad	Pawar	4444	4@gmail.com	Baramati
5	Soniya	Gandhi	5555	5@gmail.com	Pune

The following SQL statement selects or fetch the particular column data / records in the "Sign Up" table:

The following SQL statement selects the "FN" and "LN" columns from the "SignUp" table:

Syntax

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

Example

```
SELECT FN, LN
FROM SignUp;
```

Output / Result

	FN	**LN**
1	Rahul	Gandhi
2	Arvind	Kejriwal
3	Anna	Hajare
4	Sharad	Pawar
5	Soniya	Gandhi

2. **DISTINCT**

The SELECT DISTINCT statement is **used** to **return** / show only distinct (different) values or unique values in the particular column.

Inside a table, a column often **contains many** duplicate values; and sometimes you only want to list the different (distinct) values.

The following SQL statement selects only the DISTINCT values from the "LN" column in the "SignUp" table:

Syntax

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

Example

```
SELECT DISTINCT LN
FROM SignUp;
```

Output / Result

	LN
1	Gandhi
2	Kejriwal
3	Hajare
4	Pawar

SELECT Example Without DISTINCT

The following SQL statement selects all (including the duplicates) values from the "LN" column in the "SignUp" table:

Syntax

```
SELECT Column1
FROM table_name;
```

Example

```
SELECT LN
FROM SignUp;
```

Output / Result

	LN
1	Gandhi
2	Kejriwal
3	Hajare
4	Pawar
5	Gandhi

The following SQL statement lists the number of different (distinct) LN:

Syntax

```
SELECT COUNT (DISTINCT column1, column2,...)
FROM table_name;
```

Example

```
SELECT COUNT (DISTINCT LN)
FROM SignUp;
```

Output / Result

Number of Records-4

3. TOP

It is used to specify the number of records to return.

It is a SQL statement it used to show / display the top records in a particular column.

It is a SQL statement use to select top values in a particular table

The following SQL statement selects the first three records from the "Signup" table (for SQL Server/MS Access):

Syntax

```
SELECT TOP value / number * FROM table_name;
```

Example

```
SELECT TOP 3 * FROM SignUp;
```

Output

	ID	FN	LN	MobNo	EmailId	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	3	Anna	Hajare	3333	3@gmail.com	Pune

The following SQL statement selects the first 50% of the records from the "Sign Up" table (for SQL Server/MS Access):

Syntax

```
SELECT TOP 50 PERCENT * FROM table_name;
```

Example

```
SELECT TOP 50 PERCENT * FROM SignUp;
```

Output

	ID	FN	LN	MobNo	EmailId	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	3	Anna	Hajare	3333	3@gmail.com	Pune

4. ORDER BY

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.

It is a SQL statement it used to display the records by ascending & descending order from selected column.

The following SQL statement selects all SignUp from the "SignUp" table, sorted by the "FN" column:

Syntax

```
SELECT * FROM table_name
ORDER BY column1;
```

Example

```
SELECT * FROM SignUp
ORDER BY FN;
```

Output

	ID	FN	LN	MobNo	EmailId	City
1	3	Anna	Hajare	3333	3@gmail.com	Pune
2	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	1	Rahul	Gandhi	1111	1@gmail.com	Pune
4	4	Sharad	Pawar	4444	4@gmail.com	Baramati
5	5	Soniya	Gandhi	5555	5@gmail.com	Pune

ORDER BY DESC Example

The following SQL statement selects all SignUp from the "SignUp" table, sorted DESCENDING by the "FN" column:

Syntax

```
SELECT * FROM table_name
ORDER BY column1 DESC;
```

Example

```
SELECT * FROM SignUP
ORDER BY FN DESC;
```

Output

	ID	FN	LN	MobNo	EmailId	City
2	5	Soniya	Gandhi	5555	5@gmail.com	Pune
1	4	Sharad	Pawar	4444	4@gmail.com	Baramati
3	1	Rahul	Gandhi	1111	1@gmail.com	Pune
4	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
5	3	Anna	Hajare	3333	3@gmail.com	Pune

5. **Where Clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

Note: The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.!

The following SQL statement selects all the SignUp from the City "Pune", in the "SignUp" table:

Syntax

```
SELECT * FROM table_name
WHERE Condition or Column Name=Value;
```

Example

```
SELECT * FROM SignUp
WHERE City='Pune';
```

Output

	ID	FN	LN	MobNo	EmailId	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	3	Anna	Hajare	3333	3@gmail.com	Pune
3	5	Soniya	Gandhi	5555	5@gmail.com	Pune

Syntax

```
SELECT * FROM table_name
WHERE Condition;
```

Example

```
SELECT * FROM SignUp
WHERE ID=1;
```

Output

	ID	FN	LN	MobNo	EmailId	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune

Syntax

```
SELECT Column1, Column2
FROM table_name
WHERE Condition;
```

Example

```
SELECT EmailId
FROM SignUp
WHERE City='Pune';
```

Output

	Email Id
1	1@gmail.com
2	3@gmail.com
3	5@gmail.com

Types of Where Clause

1. OR
2. AND

The WHERE clause can be combined with AND & OR 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.

1. **OR 1 1=1 1 0 =1 0 1=1 0 0=0**

It is a SQL statement use to select records when both condition are true or one condition is true.

It is a SQL statement use to select records either one condition must be true then output will be true.

The following SQL statement selects all fields from "SignUp" where FN is "Rahul" OR LN is "Gandhi":

Syntax

```
SELECT * FROM table_name
WHERE Condition1 OR Condition 2;
```

Example

```
SELECT * FROM SignUp
WHERE FN='Rahul' OR LN='Gandhi';
```

1 1 = 1 0 1=1

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	5	Soniya	Gandhi	5555	5@gmail.com	Pune

Syntax

```
SELECT Column1, Column2
FROM table_name
WHERE Condition1 OR Condition 2;
```

Example

```
SELECT Email Id
FROM SignUp
WHERE FN='Rahul' OR LN='Gandhi';
```

Output

	Email Id
1	1@gmail.com
2	5@gmail.com

2. **AND**

It is a SQL statement in that select record when both the condition must be true then output will be true.

The following SQL statement selects all fields from "SignUp" where FN is "Rahul" AND LN is "Gandhi":

Syntax

```
SELECT * FROM table_name
WHERE condition1 AND condition2
```

Example

```
SELECT * FROM SignUp
WHERE FN='Rahul' AND LN='Gandhi';      1  1 =1
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune

Syntax

```
SELECT Column1, Column2
FROM table_name
WHERE Condition1 AND Condition 2;
```

Example

```
SELECT Mob. No
FROM SignUp
WHERE FN='Rahul' AND LN='Gandhi';
```

Output

	Mob. No
1	1111

6. LIKE Operator

The 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

The following SQL statement selects all SignUp with a FN starting with "A":

- **For Starting with Specific alphabets**

Syntax

```
SELECT * FROM table_name
WHERE column LIKE pattern;
```

Example

```
SELECT * FROM SignUp
WHERE FN LIKE "A%";
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
2	3	Anna	Hajare	3333	3@gmail.com	Pune

Syntax

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

Example

```
SELECT LN
FROM SignUp
WHERE FN LIKE "A%";
```

Output

	LN
1	Kejriwal
2	Hajare

- **For Ending with Specific alphabets**

The following SQL statement selects all SignUp with a FN ending with "d":

Syntax

```
SELECT * FROM table_name
WHERE column LIKE pattern;
```

Example

```
SELECT * FROM SignUp
WHERE FN LIKE "%d";
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
2	4	Sharad	Pawar	4444	4@gmail.com	Baramati

Syntax

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

Example

```
SELECT LN
FROM SignUp
WHERE FN LIKE "%d";
```

Output

	LN
1	Kejriwal
2	Pawar

Select all records where the value of the FN column does NOT start with the letter "A".

Syntax

```
SELECT * FROM table_name
WHERE column NOT LIKE Pattern;
```

Example

```
SELECT * FROM SignUp
WHERE FN  NOT LIKE "A%";
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	4	Sharad	Pawar	4444	4@gmail.com	Baramati
3	5	Soniya	Gandhi	5555	5@gmail.com	Pune

Select all records where the value of the column starts with the letter "A".

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "A%";
```

Select all records where the value of the column *ends* with the letter "a".

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "%a";
```

Select all records where the value of the column starts with letter "A" and ends with the letter "b".

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "A%b";
```

Select all records where the value of the column contains the letter "a".

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "%a%";
```

Select all records where the value of the column does NOT start with the letter "A".

Syntax

```
SELECT * FROM table_name
WHERE column NOT LIKE "A%";
```

For Starting with alphabets ABC

The following SQL statement selects all Sign Up with a FN starting with "A", "B", or "C":

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "[ABC]%";
```

For Ending with alphabets ABC

The following SQL statement selects all Sign Up with a FN ending with "A", "B", or "C":

Syntax

```
SELECT * FROM table_name
WHERE column LIKE "%[ABC]";
```

LIKE Operator	Description
WHERE Column LIKE 'a%'	Finds any values that start with "a"
WHERE Column LIKE '%a'	Finds any values that end with "a"
WHERE Column LIKE '%or%'	Finds any values that have "or" in any position
WHERE Column LIKE '_r%'	Finds any values that have "r" in the second position
WHERE Column LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE Column LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE Column LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

7. WILDCARD

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the LIKE operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

The following SQL statement selects all SignUp with a FN starting with any character, followed by "ahul":

Syntax

```
SELECT * FROM table_name
WHERE Column LIKE Specified pattern;
```

Example

```
SELECT * FROM SignUp
WHERE FN LIKE '_ahul';
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune

8. IN Operator

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

It is used to fetch one or more data from the table.

It is use to select those records which specify in query.

The following SQL statement selects all SignUp that are Present in "Rahul", "Soniya":

Syntax

```
SELECT * FROM table_name
WHERE Column IN ('Value1', 'Value2', 'Value3'…)
```

Example

```
SELECT * FROM SignUp
WHERE FN IN ('Rahul', 'Soniya');
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	5	Soniya	Gandhi	5555	5@gmail.com	Pune

The following SQL statement selects all Sign Up that are NOT present in "Rahul", "Soniya":

Syntax

```
SELECT * FROM table_name
WHERE Column NOT IN ('Value1', 'Value2', 'Value3'...);
```

Example

```
SELECT * FROM SignUp
WHERE FN NOT IN ('Rahul', 'Soniya');
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
2	3	Anna	Hajare	3333	3@gmail.com	Pune
3	4	Sharad	Pawar	4444	4@gmail.com	Baramati

9. 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.

It is a SQL statement use to select value between ranges which specified.

Syntax

```
SELECT * FROM table_name
WHERE Column_name  BETWEEN Value 1 AND Value 2;
```

Example

```
SELECT * FROM SignUp
WHERE ID BETWEEN 2 AND 4;
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
2	3	Anna	Hajare	3333	3@gmail.com	Pune
3	4	Sharad	Pawar	4444	4@gmail.com	Baramati

Example

```
SELECT * FROM SignUp
WHERE Email Id  BETWEEN '1@gmail.com' AND '3@gmail.com';
```

Output

ID	FN	LN	Mob. No	Email Id	City
1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
3	Anna	Hajare	3333	3@gmail.com	Pune

Example

```
SELECT * FROM Sign Up
WHERE FN BETWEEN Arvind AND Soniya;
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	2	Arvind	Kejriwal	2222	2@gmail.com	Mumbai
2	3	Anna	Hajare	3333	3@gmail.com	Pune
3	4	Sharad	Pawar	4444	4@gmail.com	Baramati
4	5	Soniya	Gandhi	5555	5@gmail.com	Pune

NOT BETWEEN Example

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

It is a SQL statement use to select value between range which specified.

Syntax

```
SELECT * FROM table_name
WHERE Column_name  NOT BETWEEN Value 1 AND Value 2;
```

Example

```
SELECT * FROM SignUp
WHERE ID NOT BETWEEN 2 AND 4;
```

Output

	ID	FN	LN	Mob. No	Email Id	City
1	1	Rahul	Gandhi	1111	1@gmail.com	Pune
2	5	Soniya	Gandhi	5555	5@gmail.com	Pune