===================== MySQL by Testing world ======================

\*\* what do we test in database testing…

When we enter data from UI then it goes to server and the server store the data into database, and I want to check/test all datas are successfully stored or not !

At database testing we do test the data in 3 ways…

1. validate existance of values in database table- datas are stored into database or not !

2. validate correctness of values in database table- correct datas are stored ot not !

3. validate completeness of values in database table- complete datas are stored ot not ! (like st. instead of street)

: in real time we only use Select query, in 90% company testers don’t have access to delete or update data.

What to test...

#1) Database Schemas

A Database Schema is nothing more than a formal definition of how the data is going to be organized inside a DB. To test it:

before that first we need to know the requirement to validate- what is table name, columns name, column size, data type etc.

these are available in the reference design document, so first we need to have it before conducting the test.

#2) Transactions

When testing Transactions it is important to make sure that they satisfy the ACID properties.

#3) Stored Procedures

Stored Procedures are more or less similar to user-defined functions.

These can be invoked by Call Procedure/Execute Procedure statements and the output is usually in the form of result sets.

#4) Field Constraints

The Default value, Unique value, and Foreign key:

🡺 Approach : first we need to do….

> MySQL server : MySQL workbench(client software)

: download and set up MySQL server

: create user name and password, you can create multiple user.

: it is a server where database will be stored, can be installed in my machine.

: In organization (mysql) database server will be placed in separate machine and every person who want to connect with the database and execute Query has to download the client software MySQL workbench. Then need to make connection with the database server.

: now connect MySQL workbench(client software to MySQL (database) server…

: start MySQL workbench

: click MySql connection(+)

: do a connection name   
: enter host name(IP Address)

: port(3306)

: enter (user name + password)

: click test connection - ok.

> set up MySQL workbench(client software) : it is used to connect to the server, run query, fetch data

> create database and tables.

> go to google > type MySQL database > download mysql installer(community)

> click on downloaded msi file.

> click msi file > select MySQL server > setup user name & pass word > next > next > execute > finish

> click msi file > add > select MySQL workbench > [=>] > next > execute > finish.

> test Connection : open workbench > MySQL connections(+) > connection name(do any name) > enter uid, password > click test connection > ok > ok.

> click on the database (that you created) > can start writing Query.

> for practice : download sample database <https://drive.google.com/file/d/1pYGglElnaAbsms_lmLoPgv9HVbna6zXJ/view?usp=sharing>

> unzip file > copy file path

> go to workbench > administrator > data import > imprort from a file

> select/paste file path > start import(let finished)

> schema > refresh (now you can see the database ‘classic model’)

> right click on database > set as default schema.

> now check database : write query > select \* from customer

> setup Environment : you can practice on w3school environment…

**W3School : Sql statements :**

Database testing:

SQL - - Structured Query Language

DDL (Data Definition Language) : create, alter, drop

DML (Data Definition Language) : (functional tester’s job)

select, insert, update, delete

TCL (Data Definition Language) : commit, rollback

DCL (Data Definition Language) : grant, revoke

SELECT \* FROM Customers; [\* means everything]

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

//to see all meta data/columns info from schema.

Describe information\_schema.columns ;

//to connect databaseName/to start work on a database.

USE classicModels;

//to see the tables in database;

SHOW TABLES;

// to see the total number of columns.

SELECT count(\*) AS numberOFcolumns FROM information\_schema.columns WHERE table\_name = ‘Customers’;

// to see the names of the columns.

SELECT column\_name FROM information\_schema.columns WHERE table\_name = ‘Customers’;

//to see column names + dataType of the columns.

SELECT column\_name, data\_type FROM information\_schema.columns WHERE table\_name = ‘Customers’;

//to see column names + dataSize of the columns.

SELECT column\_name, column\_type FROM information\_schema.columns WHERE table\_name = ‘Customers’;

//to see column names null or NOT of the columns.

SELECT column\_name, is\_nullable FROM information\_schema.columns WHERE table\_name = ‘Customers’;

SELECT column1, column2, ...  
FROM table\_name;

SELECT \* FROM table\_name;

SELECT CustomerName, City, Country FROM Customers;

SELECT DISTINCT column1, column2, ...  
FROM table\_name;

SELECT Country

FROM Customers;

SELECT DISTINCT Country

FROM Customers;

SELECT COUNT(DISTINCT Country) FROM Customers;[COUNT returns total row no]

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

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

SELECT \* FROM Customers  
WHERE CustomerID = 1;

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 AND condition2 AND condition3 ...;

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition1 OR condition2 OR condition3 ...;

SELECT column1, column2, ...  
FROM table\_name  
WHERE NOT condition;

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC; (column’s alphabatical order)

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

UPDATE table\_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

DELETE FROM table\_name WHERE condition;

DELETE FROM table\_name;

SELECT column\_name(s)  
FROM table\_nameWHERE condition  
LIMIT number;

SELECT MIN(column\_name) [MIN() function returns smallest value of the column]  
FROM table\_name  
WHERE condition;

SELECT MAX(column\_name) [MAX() function returns highest value of the column]  
FROM table\_name  
WHERE condition;

SELECT COUNT(column\_name) [COUNT() function returns the number of rows]  
FROM table\_name  
WHERE condition;

SELECT AVG(column\_name) [AVG() function returns Average value of column]  
FROM table\_name  
WHERE condition;

SELECT SUM(column\_name) [SUM() function returns total of a numeric column]  
FROM table\_name  
WHERE condition;

SELECT column1, column2, ...  
FROM table\_name  
WHERE columnN LIKE pattern; [%=anyValue, \_=1digit length value]

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

[return value if it is IN any one of them]

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name NOT IN (value1, value2, ...);

[return value if it is NOT IN any one of them]

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (*SELECT* STATEMENT);

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

[BETWEEN selects values within a given range]

SELECT column\_name AS alias\_name  
FROM table\_name;

SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;

SELECT column\_name(s)  
FROM table\_name AS alias\_name;

SELECT column\_name(s)  
FROM table1 INNER JOIN table2

ON table1.column\_name = table2.column\_name;

[INNER JOIN selects records that have matching values in both tables]

SELECT column\_name(s)  
FROM table1 LEFT JOIN table2ON table1.column\_name = table2.column\_name;

[LEFT JOIN returns all records from left table matching records from right table]

SELECT column\_name(s)  
FROM table1 RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

[RIGHT JOIN returns all records from right table matching records left table]

SELECT column\_name(s)  
FROM table1 CROSS JOIN table2;

[CROSS JOIN returns all records from both tables. table1 and table2]

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

[self join is a regular join, but the table is joined with itself]

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID (<> means != (not equal to))  
AND A.City = B.City  
ORDER BY A.City;

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

[UNION is used to combine result-set of two or more SELECT statements

And removes duplicate rows.]

SELECT column\_name(s) FROM table1  
UNION ALL  
SELECT column\_name(s) FROM table2;

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

[GROUP BY statement groups the rows that have same values]

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

SELECT column\_name(s)  
FROM table\_name  
WHERE EXISTS  
(SELECT column\_name FROM table\_name WHERE condition);

[EXISTS operator returns TRUE if subquery returns one or more records]

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ANY  
  (SELECT column\_name  FROM table\_name  WHERE condition);

[ANY returns TRUE if ANY of the subquery values meet the condition]

SELECT ALL column\_name(s)  
FROM table\_name  
WHERE condition;

[ALL returns TRUE if ALL of the subquery values meet the condition]

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ALL  
  (SELECT column\_name  FROM table\_name  WHERE condition);

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

INSERT INTO table2 (column1, column2, column3, ...)  
SELECT column1, column2, column3, ...  
FROM table1  
WHERE condition;

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

[CASE returns the value from only first true conditon].

---------------------------------------------------------

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

SELECT \* FROM Customers  
WHERE City = 'Berlin' OR City = 'Stuttgart';

SELECT \* FROM Customers  
WHERE Country = 'Germany' OR Country = 'Spain';

SELECT \* FROM Customers  
WHERE NOT Country = 'Germany';

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

SELECT \* FROM Customers  
WHERE NOT Country = 'Germany' AND NOT Country = 'USA';

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

SELECT \* FROM Customers  
ORDER BY Country;

SELECT \* FROM Customers  
ORDER BY Country DESC;

SELECT \* FROM Customers  
ORDER BY Country, CustomerName;

SELECT \* FROM Customers  
ORDER BY Country ASC, CustomerName DESC;

INSERT INTO table\_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);

INSERT INTO table\_name  
VALUES (value1, value2, value3, ...);

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

INSERT INTO Customers (CustomerName, City, Country)  
VALUES ('Cardinal', 'Stavanger', 'Norway');

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NULL;

SELECT CustomerName, ContactName, Address  
FROM Customers  
WHERE Address IS NOT NULL;

UPDATE Customers  
SET ContactName = 'Alfred Schmidt', City = 'Frankfurt'  
WHERE CustomerID = 1;

UPDATE Customers  
SET PostalCode = 00000  
WHERE Country = 'Mexico';

UPDATE Customers  
SET PostalCode = 00000;

DELETE FROM table\_name WHERE condition;

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

DELETE FROM table\_name;

DELETE FROM Customers;

SELECT \* FROM Customers  
LIMIT 3;

SELECT \* FROM Customers  
WHERE Country='Germany'  
LIMIT 3;

SELECT MIN(Price) AS SmallestPrice  
FROM Products;

SELECT MAX(Price) AS LargestPrice  
FROM Products;

SELECT COUNT(ProductID)  
FROM Products;

SELECT AVG(Price)  
FROM Products;

SELECT SUM(Quantity)  
FROM OrderDetails;

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a%';

SELECT \* FROM Customers  
WHERE CustomerName LIKE '%a';

SELECT \* FROM Customers  
WHERE CustomerName LIKE '%or%';

SELECT \* FROM Customers  
WHERE CustomerName LIKE '\_r%';

SELECT \* FROM Customers  
WHERE CustomerName LIKE 'a\_\_%';

SELECT \* FROM Customers  
WHERE ContactName LIKE 'a%o';

SELECT \* FROM Customers  
WHERE CustomerName NOT LIKE 'a%';

SELECT \* FROM Customers  
WHERE City LIKE 'ber%';

SELECT \* FROM Customers  
WHERE City LIKE '%es%';

SELECT \* FROM Customers  
WHERE City LIKE '\_ondon';

SELECT \* FROM Customers  
WHERE City LIKE 'L\_n\_on';

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

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

SELECT \* FROM Customers  
WHERE Country IN (SELECT Country FROM Suppliers);

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

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

SELECT \* FROM Products  
WHERE Price BETWEEN 10 AND 20  
AND CategoryID NOT IN (1,2,3);

SELECT \* FROM Products  
WHERE ProductName BETWEEN 'Carna Tigers' AND 'Mozzarella di Giovanni'  
ORDER BY ProductName;

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

SELECT \* FROM Orders  
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;

SELECT CONCAT(Address,',', PostalCode, City, Country) AS Address

FROM Customers;

select Orders.OrderID, Orders.OrderDate, Customers.CustomerName from Orders, Customers

where Orders.CustomerID = Customers.CustomerID

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

SELECT Orders.OrderID, Customers.CustomerName  
FROM Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName  
FROM ((Orders  
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID);

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID  
ORDER BY Customers.CustomerName;

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
ORDER BY Orders.OrderID;

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
CROSS JOIN Orders;

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
CROSS JOIN Orders  
WHERE Customers.CustomerID=Orders.CustomerID;

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City  
FROM Customers A, Customers B  
WHERE A.CustomerID <> B.CustomerID  
AND A.City = B.City  
ORDER BY A.City;

SELECT City FROM Customers  
UNION : selects only distinct values  
SELECT City FROM Suppliers  
ORDER BY City;

SELECT City FROM Customers  
UNION ALL : selects all values  
SELECT City FROM Suppliers  
ORDER BY City;

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

SELECT City, Country FROM Customers  
WHERE Country='Germany'  
UNION ALL  
SELECT City, Country FROM Suppliers  
WHERE Country='Germany'  
ORDER BY City;

SELECT 'Customer' AS Type, ContactName, City, Country  
FROM Customers  
UNION  
SELECT 'Supplier', ContactName, City, Country  
FROM Suppliers;

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

SELECT COUNT(CustomerID), Country **[COUNT means total count of Raws]**  
FROM Customers  
GROUP BY Country;

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

SELECT Shippers.ShipperName, COUNT(Orders.OrderID) AS NumberOfOrders FROM Orders  
LEFT JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID  
GROUP BY ShipperName;

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)HAVING conditionORDER BY column\_name(s);

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

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

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM (Orders  
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID)  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 10;

SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders  
FROM Orders  
INNER JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID  
WHERE LastName = 'Davolio' OR LastName = 'Fuller'  
GROUP BY LastName  
HAVING COUNT(Orders.OrderID) > 25;

SELECT column\_name(s)  
FROM table\_name  
WHERE EXISTS  
(SELECT column\_name FROM table\_name WHERE condition);

SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price < 20);

SELECT SupplierName  
FROM Suppliers  
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.SupplierID = Suppliers.supplierID AND Price = 22);

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ANY  
  (SELECT column\_name  FROM table\_name  WHERE condition);

SELECT ALL column\_name(s)  
FROM table\_name  
WHERE condition;

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name operator ALL  
  (SELECT column\_name  FROM table\_name  WHERE condition);

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity = 10);

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity > 99);

SELECT ProductName  
FROM Products  
WHERE ProductID = ANY  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity > 1000);

SELECT ALL ProductName  
FROM Products  
WHERE TRUE;

SELECT ProductName  
FROM Products  
WHERE ProductID = ALL  
  (SELECT ProductID  
  FROM OrderDetails  
  WHERE Quantity = 10);

INSERT INTO table2  
SELECT \* FROM table1WHERE condition;

INSERT INTO table2 (column1, column2, column3, ...)  
SELECT column1, column2, column3, ...  
FROM table1  
WHERE condition;

INSERT INTO Customers (CustomerName, City, Country)  
SELECT SupplierName, City, Country FROM Suppliers;

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
SELECT SupplierName, ContactName, Address, City, PostalCode, Country FROM Suppliers;

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)  
SELECT SupplierName, ContactName, Address, City, PostalCode, Country FROM Suppliers;

CASE  
    WHEN condition1 THEN result1  
    WHEN condition2 THEN result2  
    WHEN conditionN THEN resultN  
    ELSE result  
END;

SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'  
    WHEN Quantity = 30 THEN 'The quantity is 30'  
    ELSE 'The quantity is under 30'  
END AS QuantityText  
FROM OrderDetails;

SELECT CustomerName, City, Country  
FROM Customers  
ORDER BY  
(CASE  
    WHEN City IS NULL THEN Country  
    ELSE City  
END);

SELECT ProductName, UnitPrice \* (UnitsInStock + UnitsOnOrder)  
FROM Products;

SELECT ProductName, UnitPrice \* (UnitsInStock + IFNULL(UnitsOnOrder, 0))  
FROM Products;

SELECT ProductName, UnitPrice \* (UnitsInStock + COALESCE(UnitsOnOrder, 0))  
FROM Products;

-- Select all: for coment  
SELECT \* FROM Customers;

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

MySQL Arithmetic Operators

+ Add

- Subtract

\* Multiply

/ Divide

% Modulo

MySQL Bitwise Operators

& Bitwise AND

| Bitwise OR

^ Bitwise exclusive OR

MySQL Comparison Operators

= Equal to

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to

<> Not equal to

+= Add equals

-= Subtract equals

\*= Multiply equals

/= Divide equals

%= Modulo equals

&= Bitwise AND equals

^-= Bitwise exclusive equals

|\*= Bitwise OR equals

MySQL Logical Operators

ALL TRUE if all of the subquery values meet the condition

AND TRUE if all the conditions separated by AND is TRUE

ANY TRUE if any of the subquery values meet the condition

BETWEEN TRUE if the operand is within the range of comparisons

EXISTS TRUE if the subquery returns one or more records

IN TRUE if the operand is equal to one of a list of expressions

LIKE TRUE if the operand matches a pattern

NOT Displays a record if the condition(s) is NOT TRUE

OR TRUE if any of the conditions separated by OR is TRUE

SOME TRUE if any of the subquery values meet the condition