BIG DATA TOOLS FOR MANAGERS

Unit-2 : Data Retrieval using MySQL

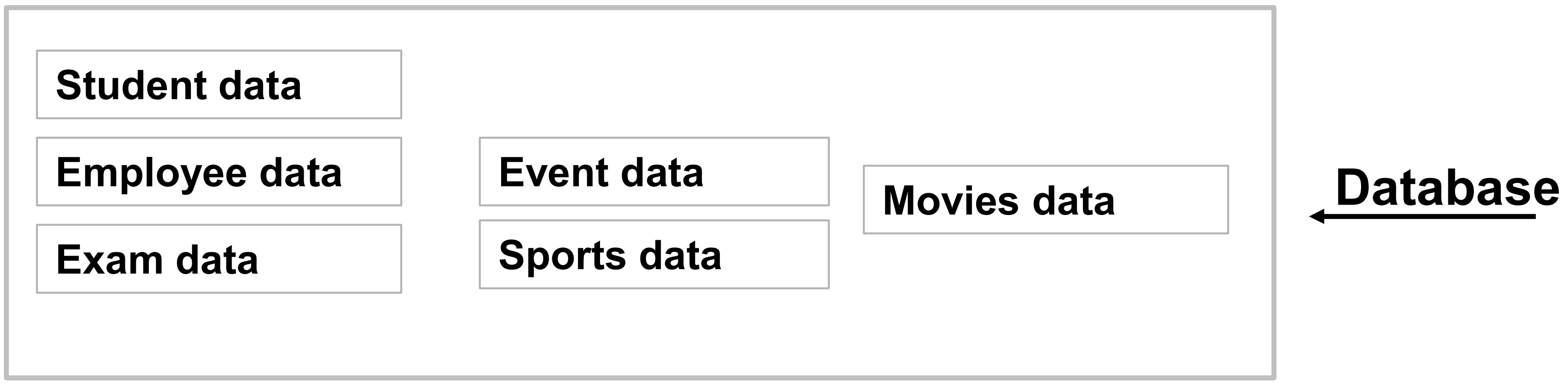
by
Ankit Velani
Adjunct Faculty, Dept. of MBA,
Siddaganga Institute of Technology,Tumkur

Data

- Data is a collection of a small piece of information. It can be used in a variety of forms like text, numbers, media, bytes…etc
- i.e. Student data, Employee data, Events data, Travelling data

Database

- A database is an organized collection of data so that it can be easily managed and accessible.



DBMS

- **D**ata **B**ase **M**anagement **S**ystem is software used to store and retrieve the database.

teradata.
IBM
DB2
Microsoft SQL Server
Amazon RDS
SQL
ORACLE DATABASE
MySQL
SQL
snowflake
PostgreSQL
SAP S/4HANA

RDBMS

- **R**elational **D**ata **B**ase **M**anagement **S**ystem
- All the modern database systems like MySQL, ORACLE…etc are based on RDBMS.
- In RDBMS, data are stored in form of Tables, and it allows to create relationships between tables.

RDBMS

Emp_ID	ENAME	POST	CITY
E1	John	Manager	USA
E2	Nick	Data Analyst	UK
E3	Dom	Product Owner	UK
E4	Paul	Project Owner	UK

RDBMS

Columns/ Attributes/ Fields

Records/ Tuples/ Rows

Emp_ID	ENAME	POST	CITY
E1	John	Manager	USA
E2	Nick	Data Analyst	UK
E3	Dom	Product Owner	UK
E4	Paul	Project Owner	UK

Table Creation

```
CREATE TABLE table_name (
    col-name-1    data-type
    col-name-2    data-type,
    col-name-3    data-type,
    col-name-4    data-type
);
```

Table Creation

```
CREATE TABLE table_name (
    col-name-1   data-type
    col-name-2   data-type,
    col-name-3   data-type,
    col-name-4   data-type
);
```

```
CREATE TABLE employee (
        emp_id   text,
        ename    text,
        post     text,
        city     text
);
```

Insert Records/Rows

```
INSERT INTO table_name VALUES (‘value-1’, ‘value-2’,……);


INSERT INTO employee VALUES ( ‘E1’,
                               ‘John’,
                               ‘Manager’,
                               ‘USA’
                             );
```

Retrieve Data using SELECT

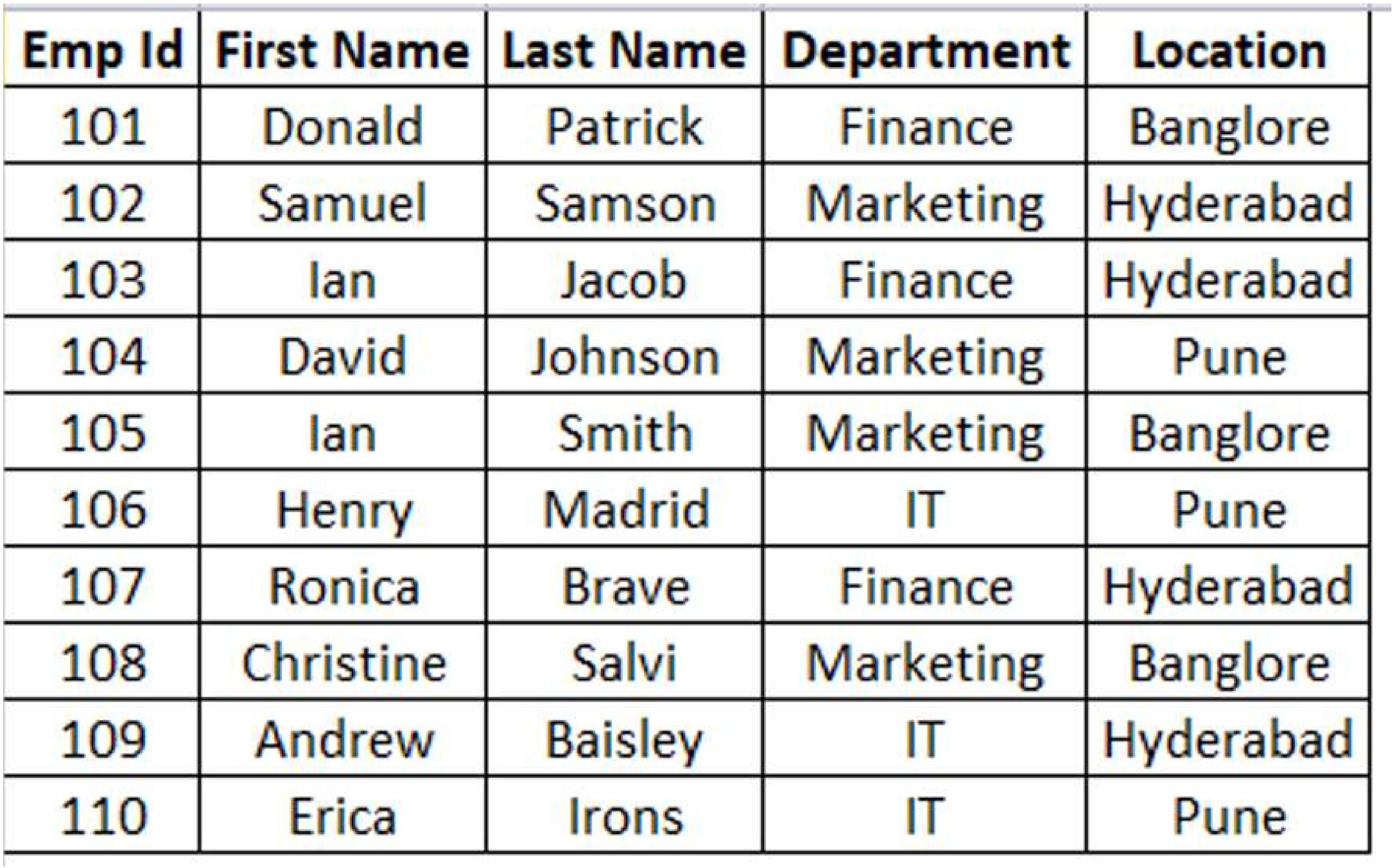
SELECT * FROM table_name;

(asterisk) represent all the columns from table

Retrieve Data using SELECT

```
SELECT * FROM table_name;

SELECT * FROM employee;
SELECT ename FROM employee;
SELECT ename, city FROM employee;
```

Create Employee_2 table in SQL



Emp Id	First Name	Last Name	Department	Location
101	Donald	Patrick	Finance	Banglore
102	Samuel	Samson	Marketing	Hyderabad
103	Ian	Jacob	Finance	Hyderabad
104	David	Johnson	Marketing	Pune
105	Ian	Smith	Marketing	Banglore
106	Henry	Madrid	IT	Pune
107	Ronica	Brave	Finance	Hyderabad
108	Christine	Salvi	Marketing	Banglore
109	Andrew	Baisley	IT	Hyderabad
110	Erica	Irons	IT	Pune

MySQL Data Types

- The data type of a column defines what value the column can hold: integer, character, money, date and time, binary, and so on.
- Each column in a database table is required to have a name and a data type.
- In MySQL there are three main data types: **string**, **numeric**, and **datetime**.

String Types

CHAR(size)

- A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored

Example:

```
CREATE TABLE student(
        usn CHAR(10)
);
```

String Types

VARCHAR(size)

- A variable-length string between 1 and 255 characters in length.

Example:

```
CREATE TABLE student(
        usn VARCHAR(10)
);
```

String Types

TINYTEXT

- TEXT column with a maximum length of 255 characters

MEDIUMTEXT

- TEXT column with a maximum length of 16777215 characters

LONGTEXT

- TEXT column with a maximum length of 4294967295 or 4 GB of characters

String Types

Example:

```
CREATE TABLE student (
                usn      CHAR(10),
                name     VARCHAR(50),
                address  TINYTEXT,
                city     VARCHAR(20),
                state    VARCHAR(20),
                pincode  CHAR(5)
);
```

Numeric Types

Data Type	Signed range(Number with Sign)	Unsigned range
TINYINT	-128 to 127	0 to 255
SMALLINT	-32768 to 32767	0 to 65535
MEDIUMINT	-8388608 to 8388607	0 to 16777215
INT	-2147483648 to 2147483647	0 to 4294967295
BIGINT	-9223372036854775808 to 9223372036854775807	0 to 18446744073709551615
FLOAT	Decimal precision can go to 24 places for a float type	
DOUBLE	Decimal precision can go to 53 places for a double	

DateTime Types

Data Type Syntax	Maximum Size
DATE	Values range from '1000-01-01' to '9999-12-31'.
TIME	Values range from '-838:59:59' to '838:59:59'.
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
TIMESTAMP	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07’ UTC.

Recap…

- String Data Type:
  - CHAR
  - VARCHAR
  - TINYTEXT
  - MEDIUMTEXT
  - LONGTEXT
- Date Data Type:
  - DATE
  - TIME
  - DATETIME
  - TIMESTAMP
- Numeric Data Type:
  - TINYINT
  - SMALLINT
  - MEDIUMINT
  - INT
  - BIGINT
  - FLOAT
  - DOUBLE

Example : String, Int, and DateTime

Example:

```
CREATE TABLE students (
            usn         CHAR(10),
            name        VARCHAR(50),
            dob         DATE,
            age         TINYINT,
            degree_per    FLOAT,
            address   MEDIUMTEXT
);
```

String, Int, and DateTime

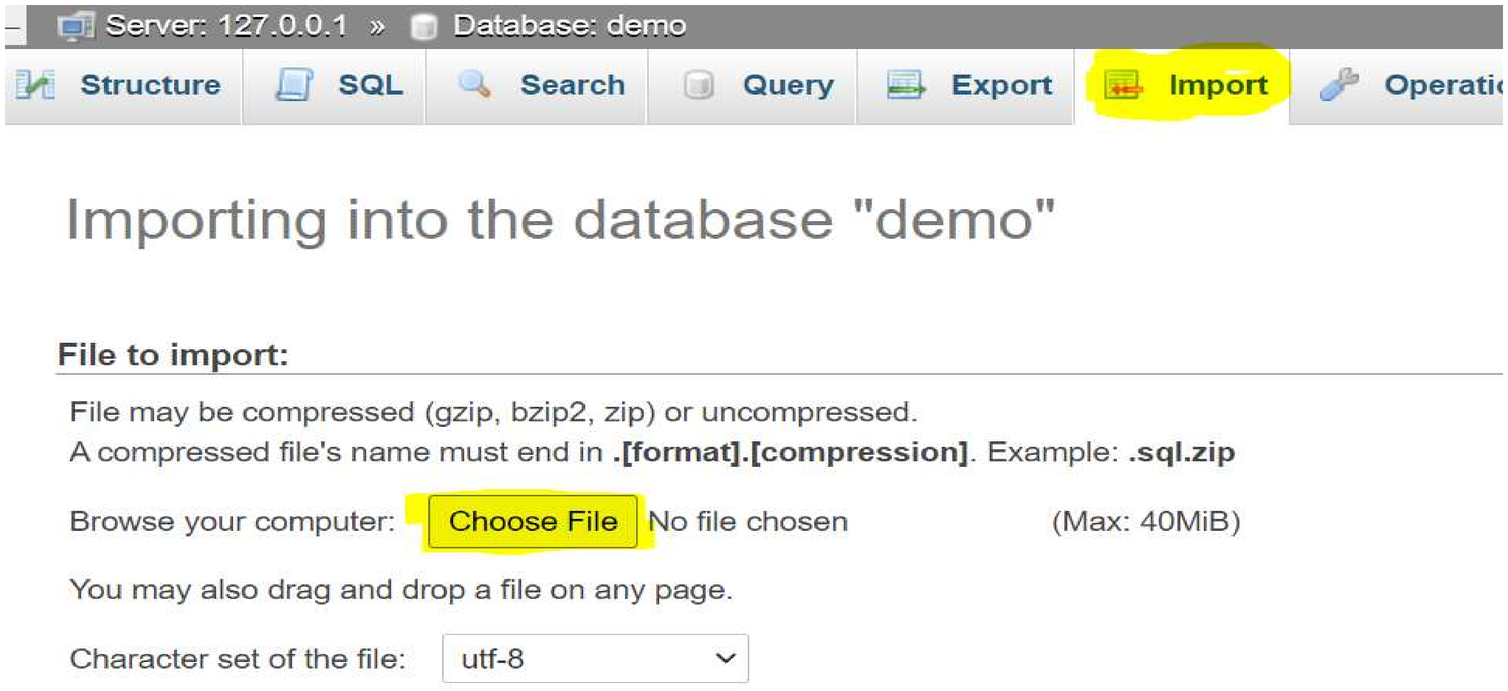
*** Numeric data should be without single or double quotes.*

Example:

```
INSERT INTO students
VALUES (
        'MBA01',
        'Paul S',
        '1991-09-08',
        30,
        70.80,
        'B.H. Road, Tumkur 572103, Karnataka’
);
```

Import data in MySQL

Download customer.sql file :
https://raw.githubusercontent.com/sitmbadept/sitmbadept.github.io/main/BDTM/SQL/customers.sql



Importing into the database "demo"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: No file chosen (Max: 40MiB)

You may also drag and drop a file on any page.

Character set of the file: utf-8

Describe Table

DESCRIBE statement used to view the structure of a table in MySQL.

Syntax:

```
DESCRIBE table_name;
```

Example:

```
DESCRIBE customers;
```

SELECT Statement

- The SELECT statement is used to select data from a database.
- The data returned is stored in a result table, called the result-set.

<u>Syntax</u>

```
SELECT
        column1, column2, column3,...
FROM table_name;
```

SELECT Statement

<u>Example</u>

- Display all the records & columns from Table:
  - SELECT * FROM *customers*;

- Display all the records and selected column from Table:

  -
    ```
    SELECT
            customerNumber,
            customerName
    FROM customers;
    ```

SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax

```
SELECT   DISTINCT
                    column1, column2, column3,...
FROM table_name;
```

SELECT DISTINCT Statement

<u>Example:</u>

- Display all the countries from the country column in customers table.

  ```
  SELECT country  FROM customers;
  ```

- Display unique/distinct countries from the country column in customers table

  ```
  SELECT DISTINCT country  FROM customers;
  ```

SELECT DISTINCT Statement

<u>Example:</u>

- Display values for city and country columns from customers table.

```
SELECT city, country  FROM customers;
```

- Display unique/distinct values for city and country columns from customers table.

```
SELECT DISTINCT city, country  FROM customers;
```

SELECT DISTINCT Statement

Example:

```
SELECT DISTINCT
        city, state, postalCode, country
FROM customers;
```

Working with large text data

Download post.sql file from below link and import into phpMyAdmin

https://raw.githubusercontent.com/sitmbadept/sitmbadept.github.io/main/BDTM/SQL/post.sql

Execute SQL queries for following:

- View Table Structure (`DESCRIBE posts;`)
- View the table data

SQL WHERE Clause

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.

WHERE Syntax:

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

SQL WHERE Clause

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfill a specified condition.

<u>WHERE Syntax:</u>

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

SQL WHERE Clause

The following operators can be used in the WHERE clause:

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

SQL WHERE Clause

Condition can be apply in multiple columns using AND OR operators.

AND operator displays a record if all the conditions separated by AND are TRUE.

OR operator displays a record if any of the conditions separated by OR is TRUE.

SQL WHERE Clause

- Example:

```
SELECT * FROM `customers` WHERE country = 'USA’;


SELECT * FROM customers
WHERE creditLimit BETWEEN 10000 AND 50000;
```

Recap…

- String, Number and Date data types
- Create Table
- Insert data (String, Number, Date)
- Import Data
- Select DISTINCT
- WHERE Clause

UPDATE statement

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax:

UPDATE table_name

SET column1 = value1, $\underbrace{\text{column2 = value2, ...}}_{\text{Column with new values}}$

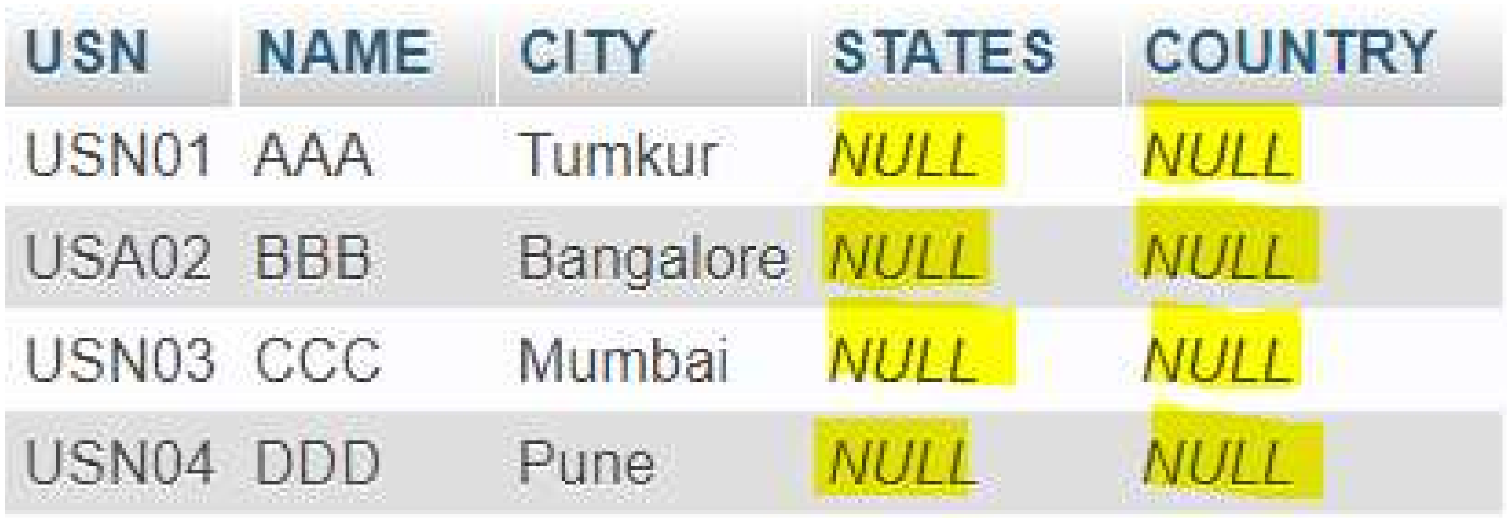
WHERE condition;

***without WHERE condition all the records gets updated in the table.

UPDATE statement

Download Data: https://bit.ly/3TERmqQ

SELECT * FROM students;



USN	NAME	CITY	STATES	COUNTRY
USN01	AAA	Tumkur	NULL	NULL
USA02	BBB	Bangalore	NULL	NULL
USN03	CCC	Mumbai	NULL	NULL
USN04	DDD	Pune	NULL	NULL

UPDATE statement

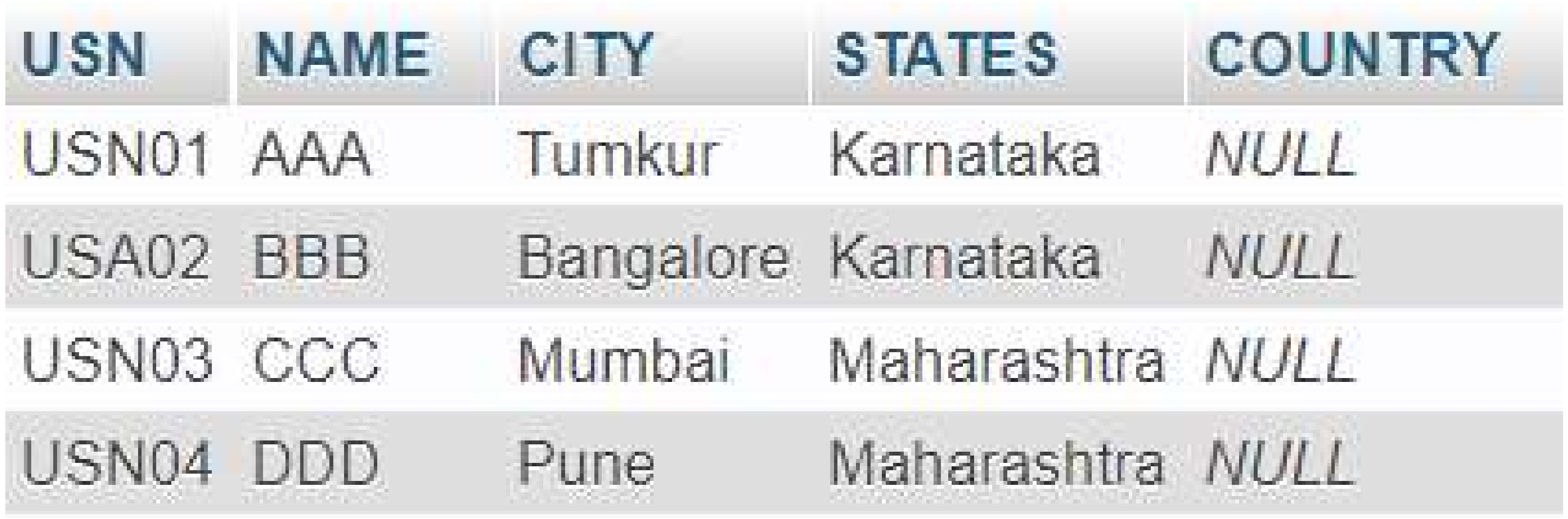
Update state name as per city

```
UPDATE students  SET STATES='Maharashtra'  WHERE CITY= 'Pune';

UPDATE students  SET STATES='Maharashtra'  WHERE CITY= 'Mumbai';


UPDATE students SET STATES='Karnataka'
WHERE CITY IN ('Tumkur','Bangalore');
```

UPDATE statement

View Table Data:

SELECT * FROM students;



USN	NAME	CITY	STATES	COUNTRY
USN01	AAA	Tumkur	Karnataka	*NULL*
USA02	BBB	Bangalore	Karnataka	*NULL*
USN03	CCC	Mumbai	Maharashtra	*NULL*
USN04	DDD	Pune	Maharashtra	*NULL*

UPDATE statement

Update country as India for all the students

Example:

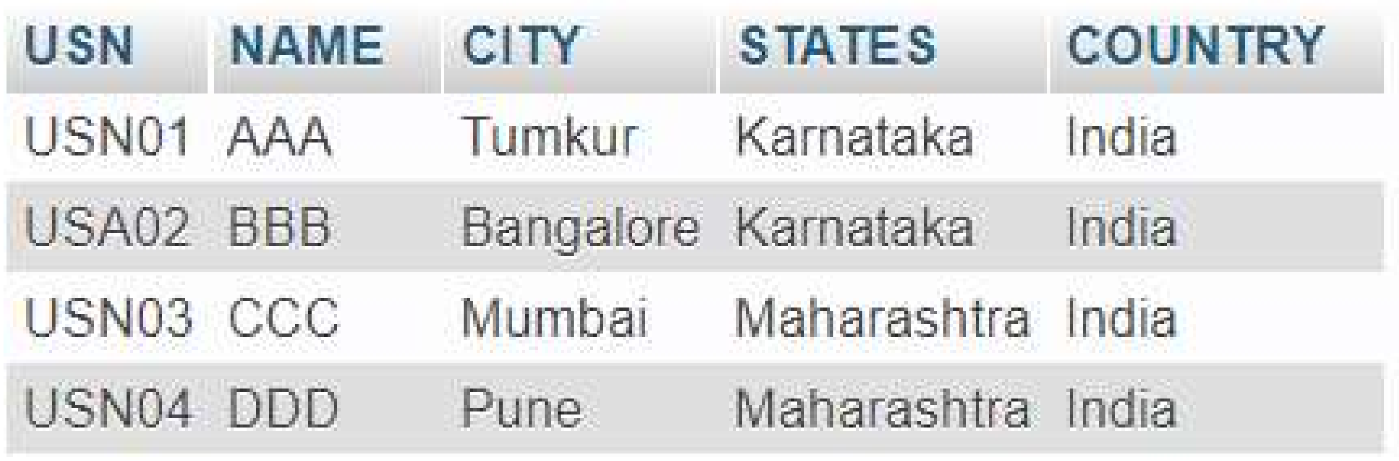
UPDATE students SET COUNTRY = 'India';

***without WHERE condition all the records gets updated in the table.

UPDATE statement

View Table Data:

SELECT * FROM students;



USN	NAME	CITY	STATES	COUNTRY
USN01	AAA	Tumkur	Karnataka	India
USA02	BBB	Bangalore	Karnataka	India
USN03	CCC	Mumbai	Maharashtra	India
USN04	DDD	Pune	Maharashtra	India

DELETE statement

The UPDATE statement is used to modify the existing records in a table.

Syntax:

```
DELETE FROM table_name
WHERE condition;
```

***without WHERE condition all the records gets deleted from the table.

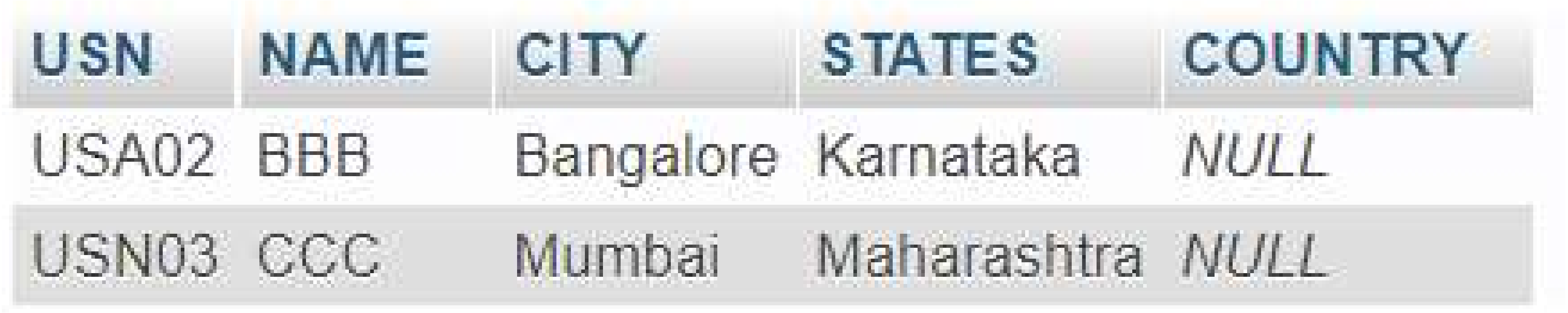
DELETE statement

```
DELETE FROM students WHERE USN='USN01';

DELETE FROM students WHERE City='Pune';
```

DELETE statement

View Table data:

SELECT * FROM students;



USN	NAME	CITY	STATES	COUNTRY
USA02	BBB	Bangalore	Karnataka	*NULL*
USN03	CCC	Mumbai	Maharashtra	*NULL*

TRUNCATE statement

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax:

```
TRUNCATE TABLE table_name;
```

TRUNCATE statement

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Example:

```
TRUNCATE TABLE students;
```

View Table data:

```
SELECT * FROM students;
```

DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

Syntax:

```
DROP TABLE table_name;
```

Example:

```
DROP TABLE students;
```

MySQL Joins

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Orders Table

OrderID	CustomerID	OrderDate
10308	2	2022-08-15
10309	1	2022-08-26
10310	2	2022-09-01

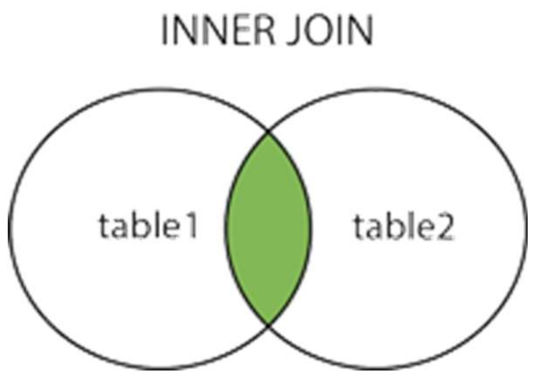
Download Data : https://bit.ly/3TERmqQ

Customers Table

CustomerID	CustomerName	Country
1	John Todd	Germany
2	Dominic Dom	Mexico
3	Paul S	Mexico

INNER JOIN

- The INNER JOIN keyword selects records that have matching values in both tables.



INNER JOIN Syntax

```
SELECT col-1, col-2….columns(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

INNER JOIN

Example: Selecting all the columns from both the table

```
SELECT customers.* ,
       orders.*
FROM customers
INNER JOIN orders
ON orders.CustomerID = customers.CustomerID;
```

INNER JOIN

Example: Selecting specified columns from tables.

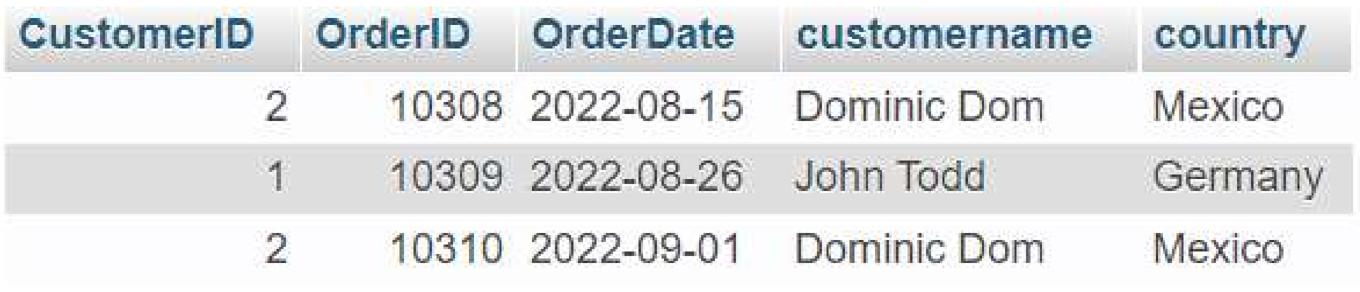
```
SELECT orders.CustomerID,
       orders.OrderID,
       orders.OrderDate,
       customers.CustomerName,
       customers.Country
FROM customers
INNER JOIN orders
ON orders.CustomerID = customers.CustomerID;
```

INNER JOIN

Example: Selecting specified columns from tables

```
SELECT   orders.CustomerID,
         orders.OrderID,
         orders.OrderDate,
         customers.CustomerName,
         customers.Country
FROM customers
INNER JOIN orders
ON orders.CustomerID = customers.CustomerID;
```

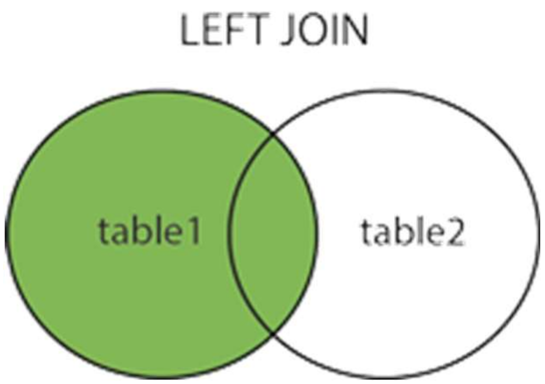
Output:



CustomerID	OrderID	OrderDate	customername	country
2	10308	2022-08-15	Dominic Dom	Mexico
1	10309	2022-08-26	John Todd	Germany
2	10310	2022-09-01	Dominic Dom	Mexico

LEFT JOIN

- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records (if any) from the right table



LEFT JOIN Syntax

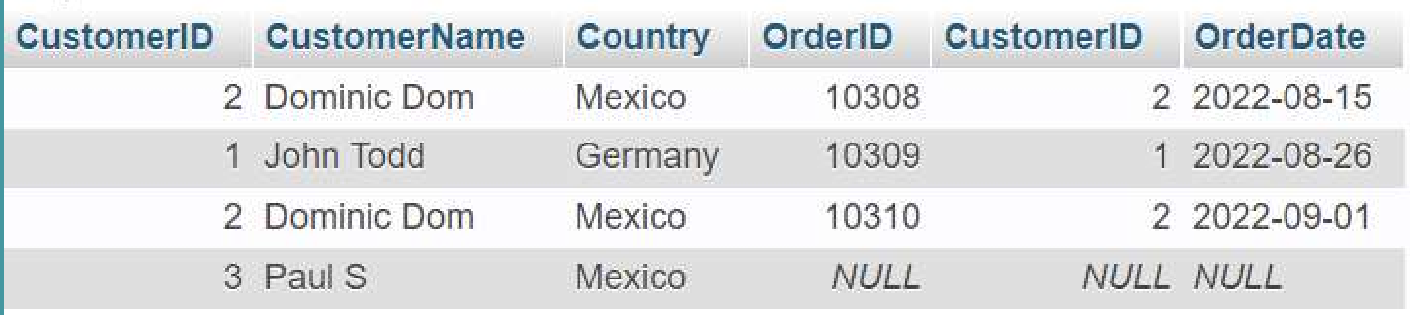
```
SELECT col-1, col-2….columns(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

LEFT JOIN

Example:

```
SELECT   customers.* ,
         orders.*
FROM customers
LEFT JOIN orders
ON orders.CustomerID = customers.CustomerID;
```

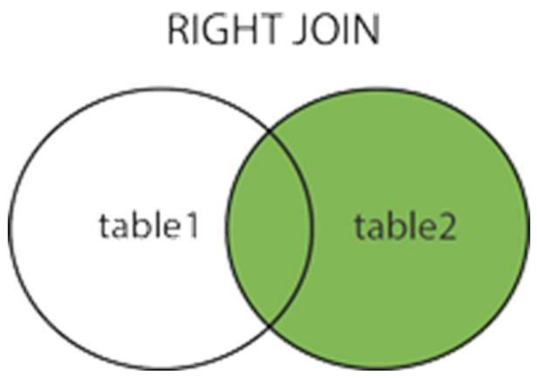
Output:



CustomerID	CustomerName	Country	OrderID	CustomerID	OrderDate
2	Dominic Dom	Mexico	10308	2	2022-08-15
1	John Todd	Germany	10309	1	2022-08-26
2	Dominic Dom	Mexico	10310	2	2022-09-01
3	Paul S	Mexico	*NULL*	*NULL*	*NULL*

RIGHT JOIN

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1)



RIGHT JOIN Syntax

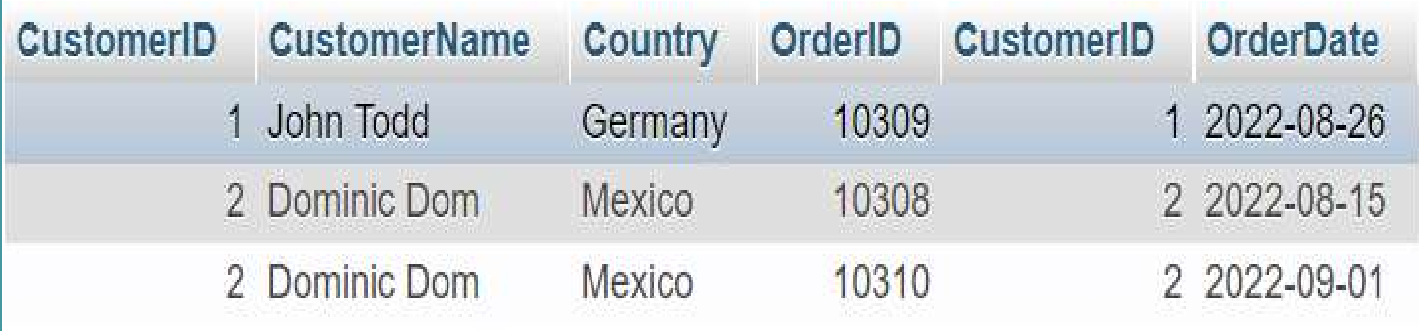
```
SELECT col-1, col-2….columns(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

RIGHT JOIN

Example:

```
SELECT   customers.* ,
         orders.*
FROM customers
RIGHT JOIN orders
ON orders.CustomerID = customers.CustomerID;
```

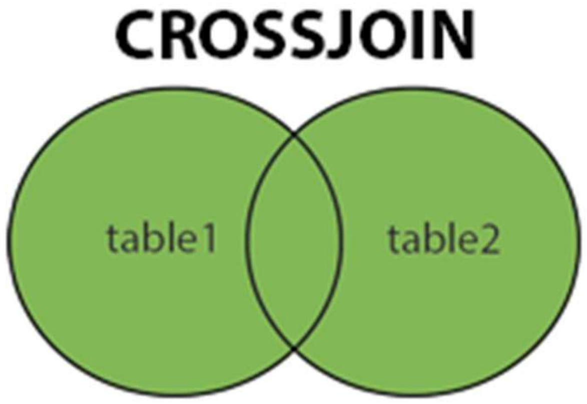
Output:



CustomerID	CustomerName	Country	OrderID	CustomerID	OrderDate
1	John Todd	Germany	10309	1	2022-08-26
2	Dominic Dom	Mexico	10308	2	2022-08-15
2	Dominic Dom	Mexico	10310	2	2022-09-01

CROSS JOIN

- The CROSS JOIN keyword returns all records from both tables (table1 and table2).



CROSS JOIN Syntax

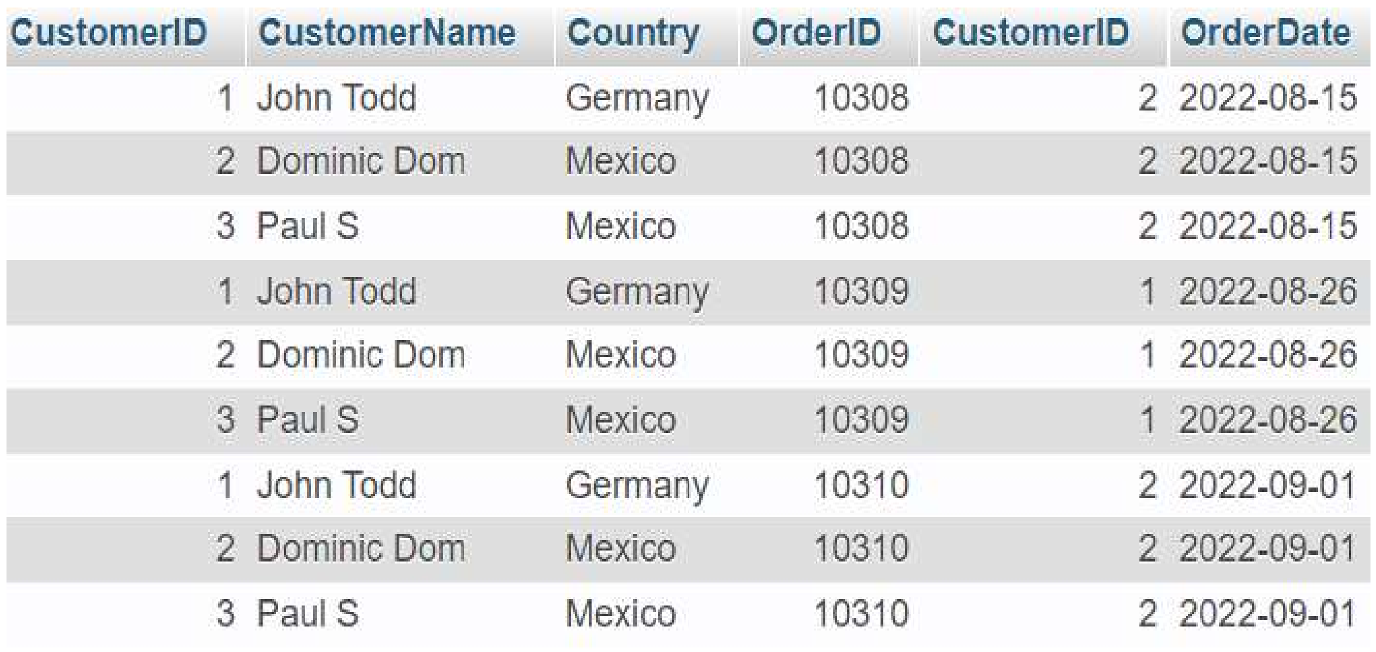
```
SELECT col-1, col-2….columns(s)
FROM table1
CROSS JOIN table2;
```

CROSS JOIN

Example:

```
SELECT   customers.* ,
         orders.*
FROM customers
CROSS JOIN orders;
```

Output:



CustomerID	CustomerName	Country	OrderID	CustomerID	OrderDate
1	John Todd	Germany	10308	2	2022-08-15
2	Dominic Dom	Mexico	10308	2	2022-08-15
3	Paul S	Mexico	10308	2	2022-08-15
1	John Todd	Germany	10309	1	2022-08-26
2	Dominic Dom	Mexico	10309	1	2022-08-26
3	Paul S	Mexico	10309	1	2022-08-26
1	John Todd	Germany	10310	2	2022-09-01
2	Dominic Dom	Mexico	10310	2	2022-09-01
3	Paul S	Mexico	10310	2	2022-09-01

UNION Operator

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Important points for UNION operator.
  - Every SELECT statement within UNION must have the same number of columns.
  - The columns must also have similar data types
  - The columns in every SELECT statement must also be in the same order

UNION Operator

UNION Syntax

```
SELECT col1, col2,..etc FROM table1
UNION
SELECT col1,col2,..etc  FROM table2;
```

UNION ALL Syntax

```
SELECT col1, col2,..etc FROM table1
UNION ALL
SELECT col1,col2,..etc  FROM table2;
```

*****The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL*

UNION

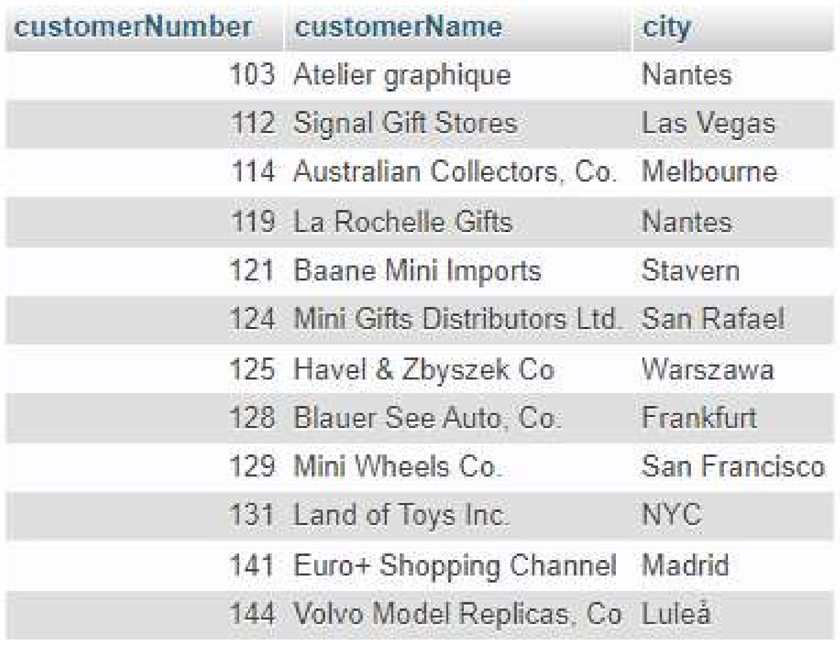
Example:

SELECT customerNumber, customerName, city FROM customers_1

UNION

SELECT customerNumber, customerName, city FROM customers_2;

Output:



customerNumber	customerName	city
103	Atelier graphique	Nantes
112	Signal Gift Stores	Las Vegas
114	Australian Collectors, Co.	Melbourne
119	La Rochelle Gifts	Nantes
121	Baane Mini Imports	Stavern
124	Mini Gifts Distributors Ltd.	San Rafael
125	Havel & Zbyszek Co	Warszawa
128	Blauer See Auto, Co.	Frankfurt
129	Mini Wheels Co.	San Francisco
131	Land of Toys Inc.	NYC
141	Euro+ Shopping Channel	Madrid
144	Volvo Model Replicas, Co	Luleå

UNION ALL

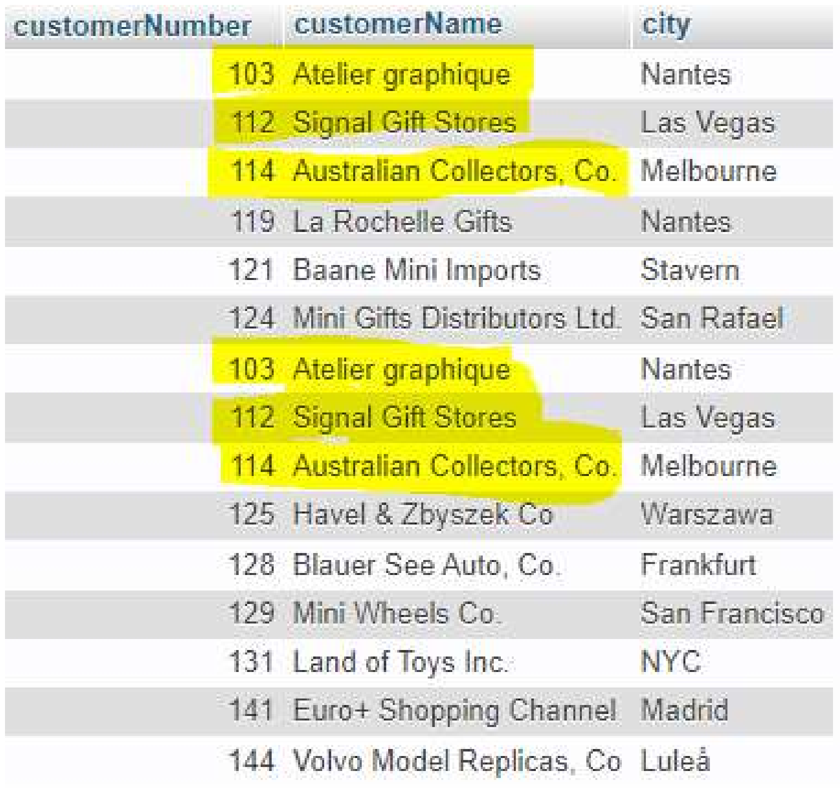
Example:

SELECT customerNumber, customerName, city FROM customers_1

UNION ALL

SELECT customerNumber, customerName, city FROM customers_2;

Output:



customerNumber	customerName	city
103	Atelier graphique	Nantes
112	Signal Gift Stores	Las Vegas
114	Australian Collectors, Co.	Melbourne
119	La Rochelle Gifts	Nantes
121	Baane Mini Imports	Stavern
124	Mini Gifts Distributors Ltd.	San Rafael
103	Atelier graphique	Nantes
112	Signal Gift Stores	Las Vegas
114	Australian Collectors, Co.	Melbourne
125	Havel & Zbyszek Co	Warszawa
128	Blauer See Auto, Co.	Frankfurt
129	Mini Wheels Co.	San Francisco
131	Land of Toys Inc.	NYC
141	Euro+ Shopping Channel	Madrid
144	Volvo Model Replicas, Co	Luleå

GROUP BY Statement

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country.

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

MIN, MAX, COUNT, AVG, SUM Functions

- MIN() function returns the smallest value of the selected column.
- MAX() function returns the largest value of the selected column.
- The COUNT() function returns the number of rows that matches a specified criterion.
- The AVG() function returns the average value of a numeric column.
- The SUM() function returns the total sum of a numeric column.

GROUP BY Statement

Syntax:

```
SELECT col-1, col-2, col-3
FROM table_name
WHERE condition
GROUP BY  col-1, col-2, col-3
ORDER BY  col-1, col-2, col-3;
```

*** ORDER BY for sorting the column values*

GROUP BY Example

Download Data: https://bit.ly/3AJdcRh

Example : Calculate Product count for each order ID

```
SELECT
        OrderID,
        count(ProductID)
FROM order_details
GROUP BY OrderID;
```

GROUP BY Example

Example : Calculate order qty for each order ID

```
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID;
```

GROUP BY Example

```
SELECT
        OrderID,
        min(Quantity)
FROM order_details
GROUP BY OrderID;
```

GROUP BY Example

```
SELECT
        OrderID,
        min(Quantity)
FROM order_details
GROUP BY OrderID;
```

```
SELECT
        OrderID,
        max(Quantity)
FROM order_details
GROUP BY OrderID;
```

GROUP BY Example

```
SELECT
        OrderID,
        min(Quantity)
FROM order_details
GROUP BY OrderID;
```

```
SELECT
        OrderID,
        max(Quantity)
FROM order_details
GROUP BY OrderID;
```

```
SELECT
        OrderID,
        avg(Quantity)
FROM order_details
GROUP BY OrderID;
```

GROUP BY with HAVING

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING clause is equivalent to WHERE clause but HAVING used with only GROUP BY clause.

GROUP BY with HAVING

Syntax:

```
SELECT col-1, col-2, col-3
FROM table_name
WHERE condition
GROUP BY  col-1, col-2, col-3
HAVING condition
ORDER BY  col-1, col-2, col-3;
```

*** ORDER BY for sorting the column values*

GROUP BY with HAVING

Example: Find out Order ID which has Order qty greater than 180

```
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID
HAVING sum(Quantity) > 180;
```

GROUP BY with HAVING

Example: Find out Order ID which has Order qty greater than 180

```
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID
HAVING sum(Quantity) > 180;
```

GROUP BY with HAVING

Example: Find out Order ID which has Order qty less than 10

```
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID
HAVING sum(Quantity) < 10;
```

Views

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

- Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Views

Example:

Create a View which gives Order ID with greater than 180 Order qty

```
CREATE VIEW qty_gt_180 AS
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID
HAVING sum(Quantity) > 180;
```

Views

Example:

Create a View which gives Order ID with greater than 180 Order qty

```
CREATE VIEW qty_gt_180 AS
SELECT
        OrderID,
        sum(Quantity)
FROM order_details
GROUP BY OrderID
HAVING sum(Quantity) > 180;
```

Now Query the View to get the result

```
SELECT * FROM qty_gt_180;
```

Dropping a View

- A view is deleted with the DROP VIEW statement.

- Syntax

  DROP VIEW *view_name*;

Example:

DROP VIEW qty_gt_180;