Basic SQL-1

By Mithilesh Singh

SQL

It stands for Structural Query Language for storing, manipulating and retrieving data in databases.

Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access

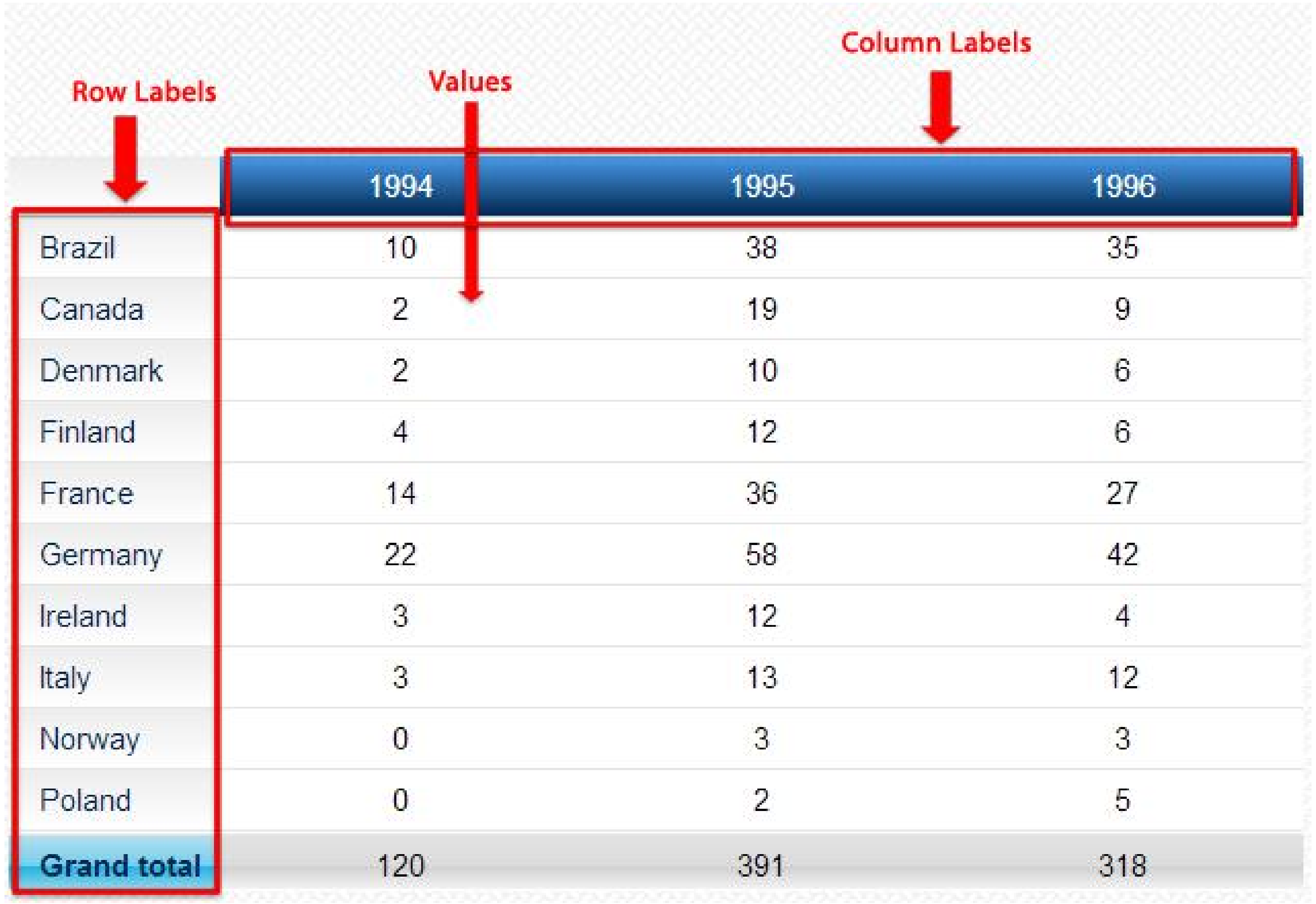
What Can SQL do??

Execute	SQL can execute queries against a database
Retrieve	SQL can retrieve data from a database
Insert	SQL can insert records in a database
Update	SQL can update records in a database
Delete	SQL can delete records from a database
Create	SQL can create new databases
Create	SQL can create new tables in a database
Create	SQL can create stored procedures in a database
Create	SQL can create views in a database
Set	SQL can set permissions on tables, procedures, and views

Table Structure:

- 1. A record, also called a row, is each individual entry that exists in a table

2. A column is a vertical entity in a table that contains all information associated with a specific field in a table.


Row Labels
Values
Column Labels

	1994	1995	1996
Brazil	10	38	35
Canada	2	19	9
Denmark	2	10	6
Finland	4	12	6
France	14	36	27
Germany	22	58	42
Ireland	3	12	4
Italy	3	13	12
Norway	0	3	3
Poland	0	2	5
Grand total	120	391	318

Some of The Most Important SQL Commands

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

SELECT COMMAND:

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

  Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

- `SELECT * FROM table_name;`

DISTINCT

Whenever there are duplicate entries in Database and we want to get only unique values we use DISTINCT in our query.

```
SELECT DISTINCT <col name>FROM <table name>;
```

If user wants to get count instead of name then use this:

```
SELECT COUNT(DISTINCT <col name>) FROM <table name>;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Where Clause

- Where clause is use to fetch the data from the database based on condition.

Syntax:

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

Operators in The 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

AND, OR and NOT Operators

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

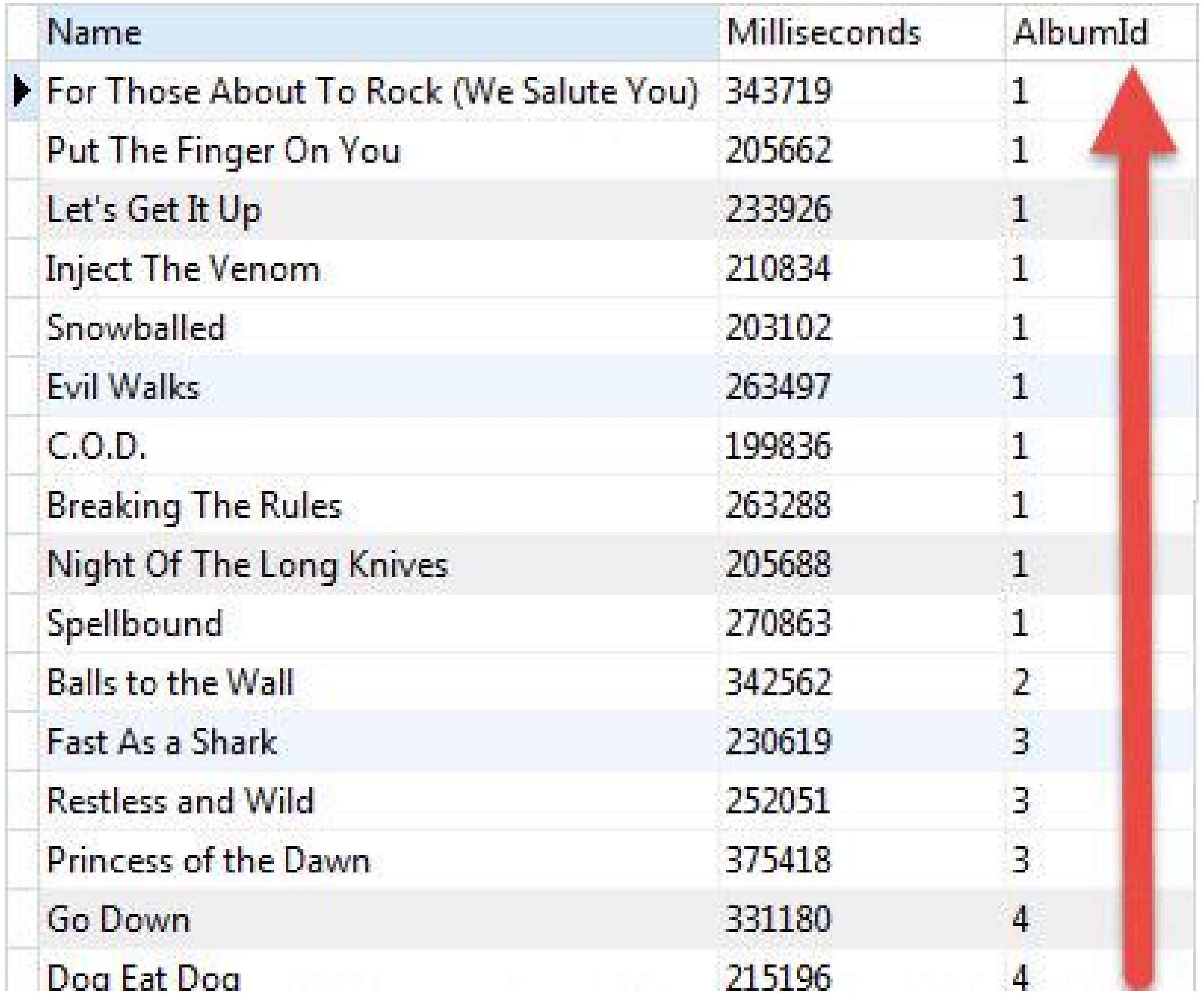
2. The **OR** operator displays a record if any of the conditions separated by OR is TRUE.

3. The **NOT** operator displays a record if the condition(s) is NOT TRUE.

AND Syntax: SELECT *column1*, *column2, ...*
FROM table_name
WHERE *condition1* AND *condition2* AND *condition3 ...;*

OR Syntax:
SELECT *column1, column2, ...*
FROM table_name
WHERE *condition1* OR *condition2* OR *co ndition3 ...;*

NOT Syntax: SELECT *column1, column2, ...*
FROM table_name
WHERE NOT *condition;*



Name	Milliseconds	AlbumId
For Those About To Rock (We Salute You)	343719	1
Put The Finger On You	205662	1
Let's Get It Up	233926	1
Inject The Venom	210834	1
Snowballed	203102	1
Evil Walks	263497	1
C.O.D.	199836	1
Breaking The Rules	263288	1
Night Of The Long Knives	205688	1
Spellbound	270863	1
Balls to the Wall	342562	2
Fast As a Shark	230619	3
Restless and Wild	252051	3
Princess of the Dawn	375418	3
Go Down	331180	4
Dog Eat Dog	215196	4

ORDER BY Keyword

- The ORDER BY keyword is used to sort the result-set in ascending or descending order.
SELECT column1, column 2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;

SQL INSERT INTO Statement

- There is basically two possible ways to write the INSERT INTO statement,

1.INSERT INTO *table_name* (*column1*, *column2*, *c olumn3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);
2. If you are adding values for all the columns of the table.
INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

SQL UPDATE Statement

UPDATE Syntax

- UPDATE *table_name*
  SET column1 = value1, column 2 = value2, ...
  WHERE *condition;*

NOTE:

- **Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

SQL DELETE Statement

- DELETE FROM *table_name* WHERE *condition*;
  Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

- Delete All the Records

  Syntax:
  DELETE FROM *table_name*;

SQL TOP Clause

- Top clause use to get limited no of data from huge amount of data.

  Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

- SQL Server:
  Syntax:
  SELECT TOP *number|percent column_name(s)*
  FROM table_name
  WHERE condition;
  Example:
- SELECT TOP 10 * FROM Customers;
- SELECT TOP 50 PERCENT * FROM C ustomers;
- SELECT TOP 3 * FROM Customers WHERE Country=India;

MIN(), MAX() Functions

The **MIN()** function returns the smallest value of the selected column.

The **MAX()** function returns the largest value of the selected column.

- Syntax:

SELECT MAX (<column name>) FROM "table_name";

example:
SELECT MAX(Sales) FROM Store_Information;

O/p:

MAX(Sales)
1500

LIKE Operator

- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column;
- % - The percent sign represents zero, one, or multiple characters;
- _ - The underscore represents a single character;
- Syntax:
  SELECT column1, column2, ...
  FROM table_name
  WHERE columnN LIKE patte

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

IN Operator

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

- Syntax:
  SELECT *column_name(s)*
  FROM table_name
  WHERE column_name IN (*value1, value2, ...*);
  ex:
  SELECT * FROM Customers
  WHERE Country IN ('Germany', 'France', 'UK
  ');
- SELECT *column_name(s)*
  FROM table_name
  WHERE column_name IN (*SELECT STATEME
  NT*);
  ex:
  SELECT * FROM Customers
  WHERE Country IN (SELECT Country FROM
  Suppliers);

BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Example:

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

BETWEEN with IN Operator

The following SQL statement selects all products with a price BETWEEN 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3:

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

SQL Aliases

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.

Note:

An alias only exists for the duration of the query.

Column Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

Table Syntax:

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Aliases

- **Note:** It requires double quotation marks or square brackets if the alias name contains spaces:
  Ex:
  SELECT CustomerName AS Customer, ContactName AS [Contact Person]
  FROM Customers;
  Aliases can be useful when:
- There are more than one table involved in a query
- Functions are used in the query
- Column names are big or not very readable
- Two or more columns are combined together

SQL JOIN

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

Syntax:

```
SELECT Orders.OrderID,
Customers.CustomerName,
Orders.OrderDate
FROM Orders
INNER JOIN Customers ON Order
s.CustomerID=Customers.Customer
ID;
```

OrderID	CustomerID	OrderDate
10308	1	1996-09-18

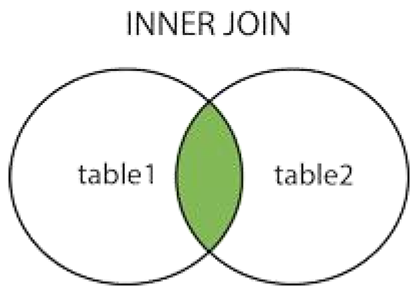
CustomerID	CustomerName	ContactName	Country
1	Alfreds	Maria	Germany

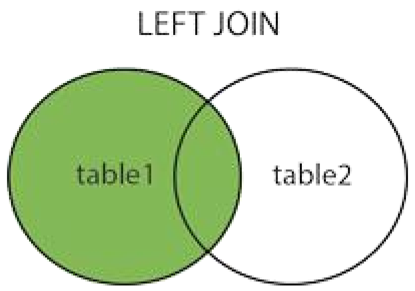
OrderID	CustomerName	OrderDate
10308	Alfreds	9/18/1996

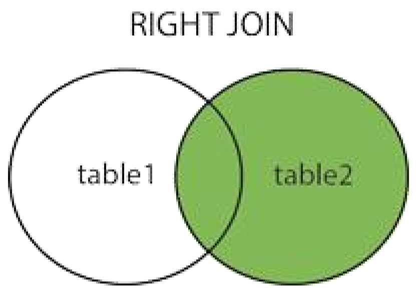
Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.
Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

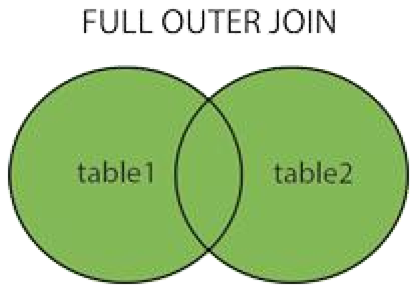
Types of Join's

Here are the different types of the JOINs in SQL:









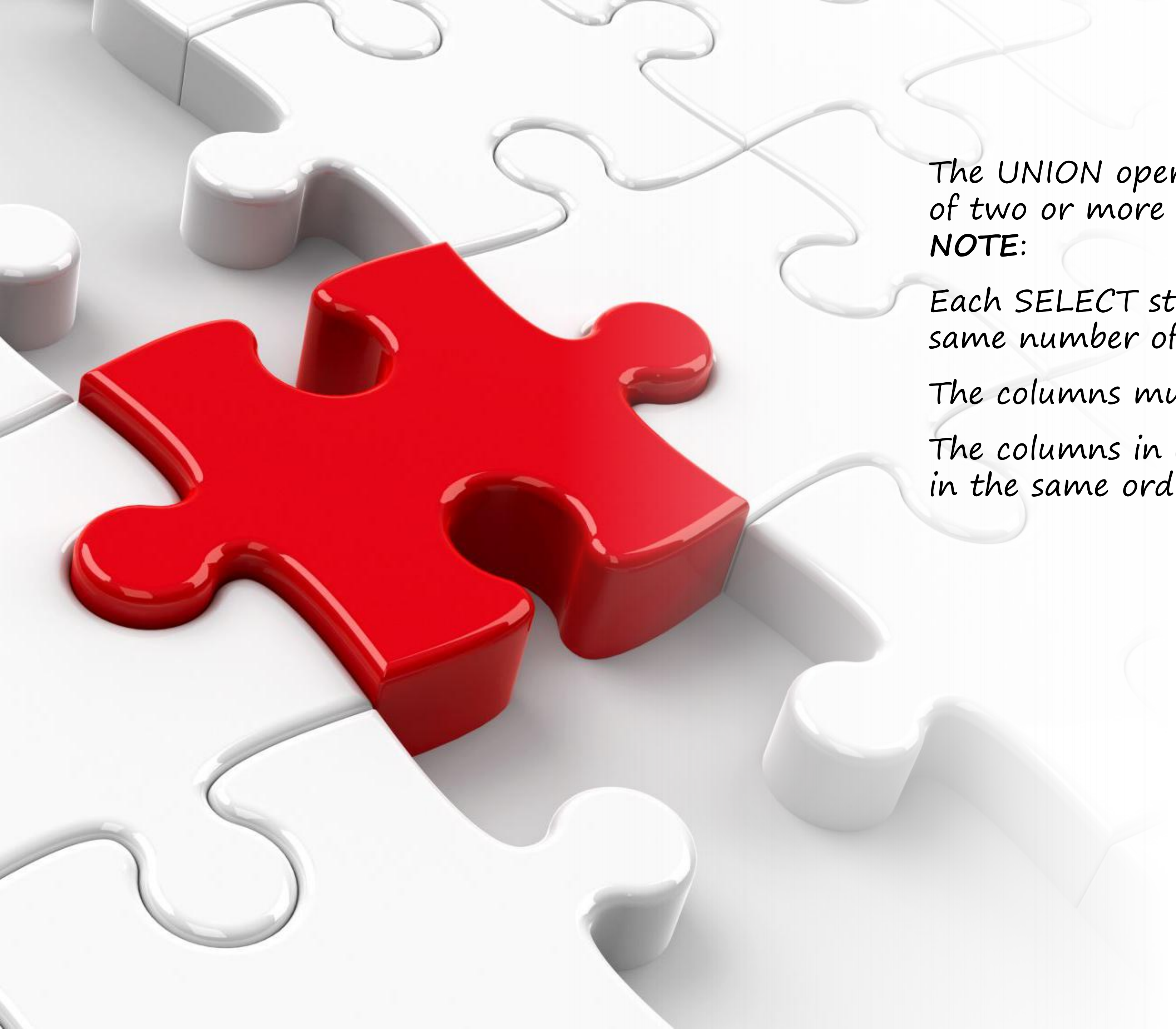
- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

SQL INNER JOIN

INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name
= table2.column_name;
```

- **Note:** The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns



The UNION operator is used to combine the result-set of two or more SELECT statements:
NOTE:

Each SELECT statement within UNION must have the same number of columns

The columns must also have similar data types

The columns in each SELECT statement must also be in the same order

UNION Operato r

Union/Union All Syntax:

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

Union Syntax:

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

```
SLECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

GROUP BY Statement

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

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

GROUP BY Examples

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
```

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

HAVING Clause

WHERE keyword could not be used with aggregate functions that is why HAVING clause was added to SQL.

HAVING Syntax

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Having Advance level Example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Ajay	Maria	Obere Str. 57	Berlin	12209	Germany
2	Bimal	Ana	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	champion	Antonio	Mataderos 2312	México D.F.	05023	Mexico

SQL EXISTS Operator

The EXISTS operator is used to test for the existence of any record in a subquery.
Syntax:
SELECT *column_name(s)*
FROM table_name
WHERE EXISTS
(SELECT *column_name* F
ROM *table_name* WHERE
condition);

ProductID	ProductName	SupplierID	CategoryID	Price
4	Test	2	2	22

SupplierID	SupplierName	City	PostalCode	Country
2	Tester	Pune	70117	India

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

This will return true or false as an output.
If we want value as output, Replace: 'Exists'
operator with 'Any' and 'All' operator.
```

CASE Statement

- The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.
-
```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

SQL NULL Functions

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	NULL

Suppose that the "UnitsOnOrder" column is optional, and may contain NULL values.

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

In the example above, if any of the "UnitsOnOrder" values are NULL, the result will be NULL.

The MySQL IFNULL() function lets you return an alternative value if an expression is NULL:

Solution:

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

SQL Stored Procedures for SQL Server

What is a Stored Procedure?

if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

Stored Procedure Syntax

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Execute a Stored Procedure

```
EXEC procedure_name;
```

Note: This is bit tricky when we use where condition in our SQL Syntax:

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30)
AS
SELECT * FROM Customers WHERE City = @City
GO;
```

Execute a Stored Procedure

```
EXEC SelectAllCustomers @City = 'London';
```

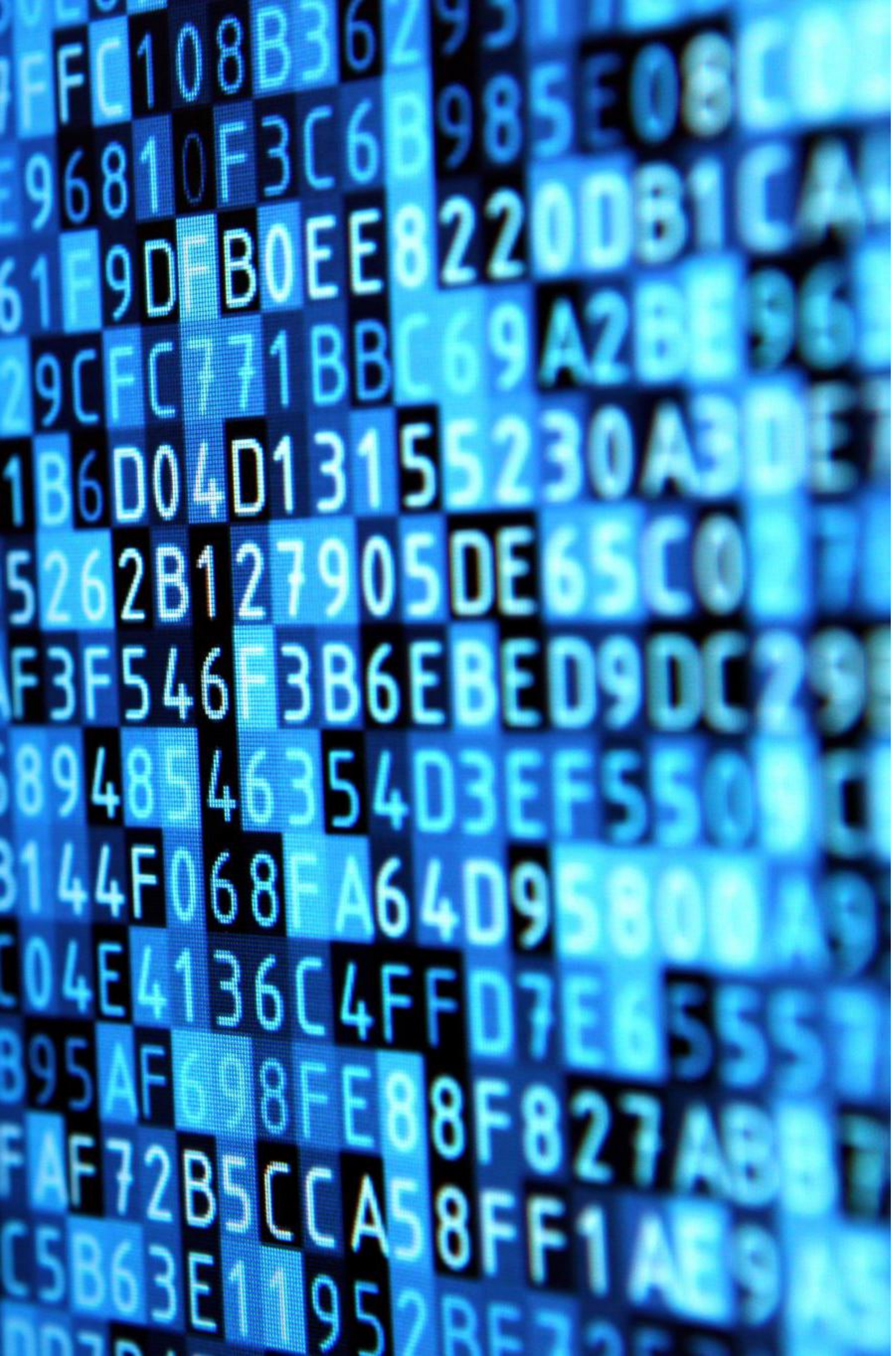
Basic SQL-- 02

Mithilesh Singh

SQL CREATE DATABASE

The CREATE DATABASE statement is used to create a new SQL database.
Syntax: CREATE DATABASE <database_name>.

NOTE:
Tip: Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases with the following SQL command
Syntax: SHOW DATABASES;



DROP DATABASE

- The DROP DATABASE statement is used to drop an existing SQL database.
  Syntax: DROP DATABASE <database_name>
- **Note:** Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

SQL BACKUP DATABASE

- The BACKUP DATABASE statement is used in SQL Server to create a full back up of an existing SQL database.
  Syntax:
  BACKUP DATABASE <datanase_name>
  TO DISK = 'filename';
  NOTE:

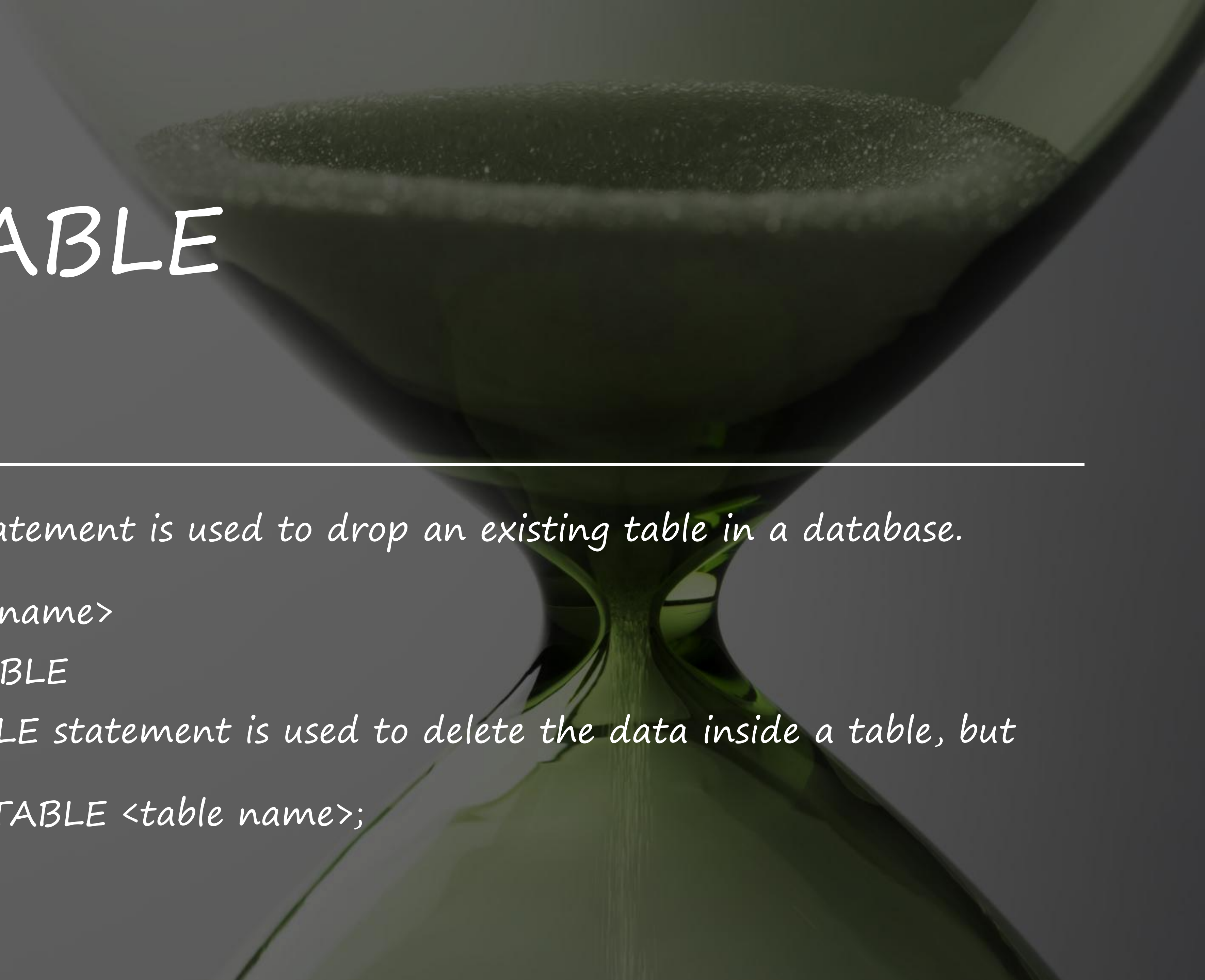
  A differential back up only backs up the parts of the database that have changed since
  the last full database backup.
  NOTE:
  BACKUP DATABASE testDB
  TO DISK = 'D:\backups\testDB.bak'
  WITH DIFFERENTIAL;

SQL CREATE TABLE

PersonID	LastName	FirstName	Address	City

- CREATE TABLE statement is used to create a new table in a database.
- Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```



DROP TABLE

- The DROP TABLE statement is used to drop an existing table in a database.
  Syntax:
  DROP TABLE <table name>
- Note: TRUNCATE TABLE
- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.
  Syntax: TRUNCATE TABLE <table name>;

ALTER TABLE

- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.
- **ALTER TABLE - ADD Column**
  Syntax: ALTER TABLE <table name>
  ADD column_name datatype;

  ALTER TABLE - DROP COLUMN
  Syntax: ALTER TABLE *table_name*
  DROP COLUMN *column_name;*
  Modify the column:
  Syntax: ALTER TABLE *table_name*
  ALTER COLUMN *column_name datatype;*

SQL Constraints

- SQL constraints are used to specify rules for the data in a table.
- Constraints are used to limit the type of data that can go into a table.
  Note:
  Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
- **The following constraints are commonly used in SQL:**

  NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Uniquely identifies a row/record in another table
- CHECK - Ensures that all values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column when no value is specified
- INDEX - Used to create and retrieve data from the database very quickly

UNIQUE Constraint
The UNIQUE constraint ensures that all values in a column are different.
Example:
```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int);
```
Drop UNIQUE Constraint:
```
    ALTER TABLE Persons
    DROP CONSTRAINT UC_Person;
```

PRIMARY KEY Constraint
The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values.
Example:
```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```
SQL PRIMARY KEY on ALTER TABLE
```
ALTER TABLE Persons
ADD PRIMARY KEY (ID);
```
Note: If you use the ALTER TABLE statement to add a primary key, the primary key column(s) must already have been declared to not contain NULL values (when the table was first created).

FOREIGN KEY Constraint

A FOREIGN KEY is a key used to link two tables together.
A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Example:

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID
int FOREIGN KEY REFERENCES Persons
(PersonID)
);
```

DROP a FOREIGN KEY Constraint

```
ALTER TABLE Orders
DROP CONSTRAINT FK_PersonOrder;
```

SQL CHECK Constraint

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

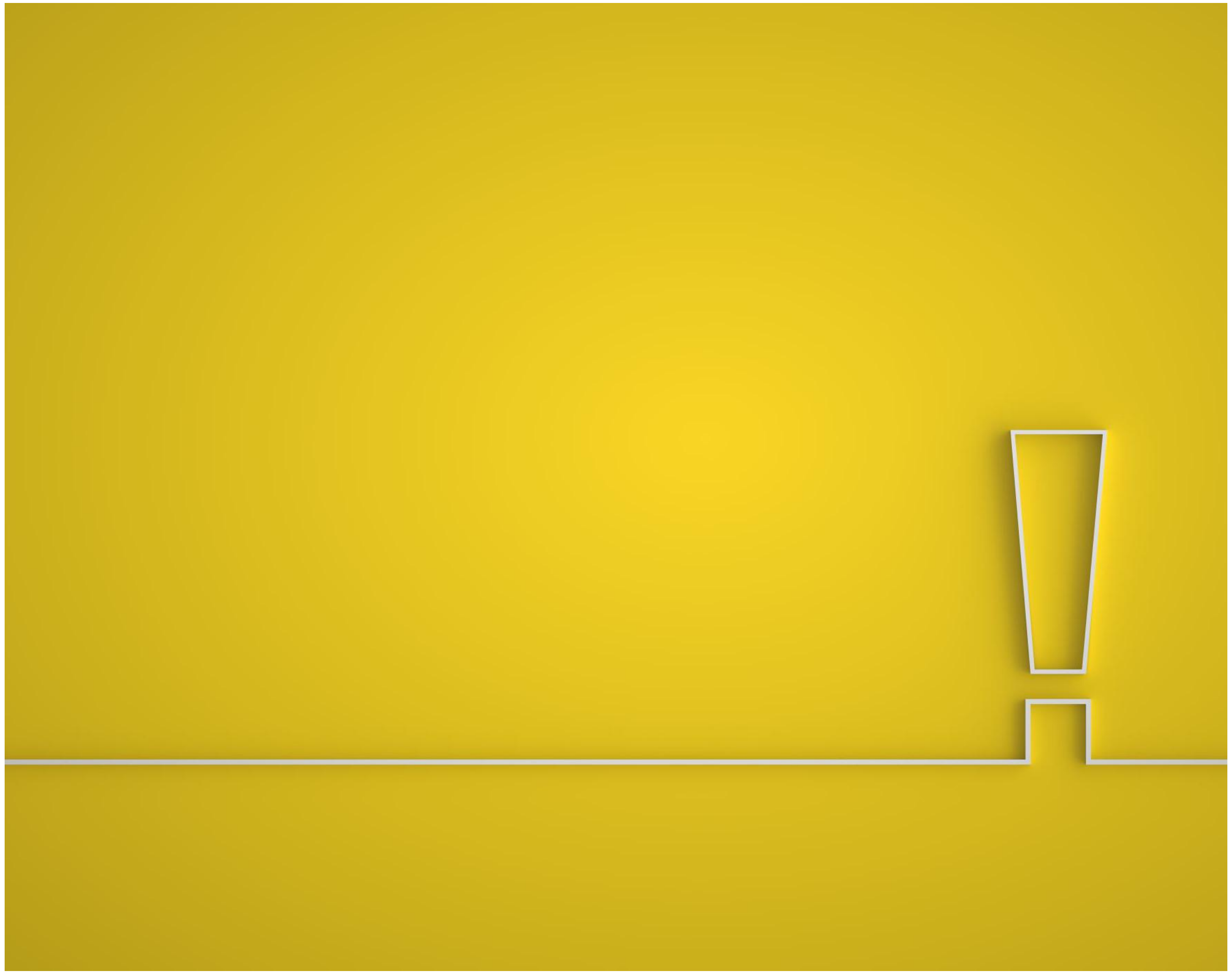
If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

Example:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

DROP a CHECK Constraint

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```



SQL DEFAULT Constraint

The DEFAULT constraint is used to provide a default value for a column.
The default value will be added to all new records IF no other value is specified.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName
varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City
varchar(255) DEFAULT 'Sandnes'
);
```

SQL CREATE INDEX

The CREATE INDEX statement is used to create indexes in tables.
Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.
Note: Updating a table with indexes takes more time than updating a table without (because the indexes also need an update).
CREATE INDEX Syntax:
CREATE INDEX *index_name*
ON table_name (column1, column2, ...);
SQL Server:
DROP INDEX *table_name.index_name;*

AUTO INCREMENT

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

Syntax for SQL Server:

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

SQL Keywords

https://onedrive.live.com/edit.aspx?resid=D7FE449AE7B1E08F%21130&nd=1

Keyword	Description
ADD	Adds a column in an existing table
ADD CONSTRAINT	Adds a constraint after a table is already created
ALTER	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table
ALTER COLUMN	Changes the data type of a column in a table
ALTER TABLE	Adds, deletes, or modifies columns in a table
ALL	Returns true if all of the subquery values meet the
AND	Only includes rows where both conditions is true
ANY	Returns true if any of the subquery values meet the condition
AS	Renames a column or table with an alias
ASC	Sorts the result set in ascending order
BACKUP DATABASE	Creates a back up of an existing database
BETWEEN	Selects values within a given range
CASE	Creates different outputs based on conditions

Some basic MCQ questiones with correct answer: https://onedrive.live.com/?cid=D7FE449AE7B1E08F&id=D7FE449AE7B1E08F%21133&parId=root&o=OneUp

Mithilesh Singh