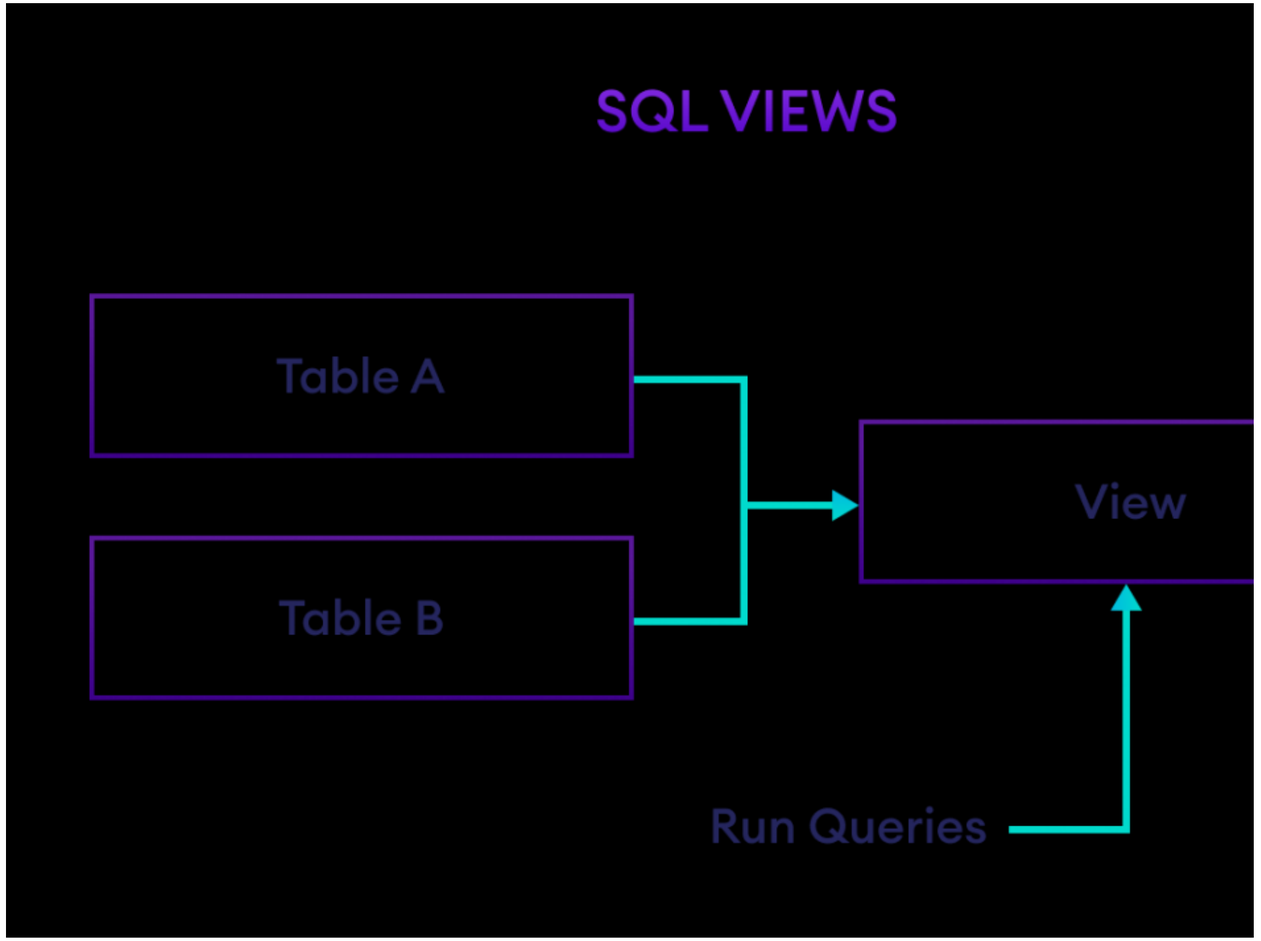
Views

- Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database.

- We can create a view by selecting fields from one or more tables present in the database.

- A View can either have all the rows of a table or specific rows based on certain condition.



Views

- Creating Views from single table
- Creating views from multiple table
- Deleting Views
- Updating Views

StudentDetails

S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

StudentMarks

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

CREATING VIEWS

A View can be created from a single table or multiple tables

Syntax:

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

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows

Examples:

Creating View from a single table:

- In this example we will create a View named DetailsView from the table StudentDetails

```
CREATE VIEW DetailsView AS
SELECT NAME, ADDRESS
FROM StudentDetails
WHERE S_ID < 5;
```

To see the data in the View, we can query the view in the same manner as we query a table.

```
SELECT * FROM DetailsView;
```

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

Creating View from multiple tables:

In this example we will create a View named MarksView from two tables StudentDetails and StudentMark

```
CREATE VIEW MarksView AS
SELECT StudentDetails.NAME,
StudentDetails.ADDRESS, StudentMarks.MARKS
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

To display data of View MarksView:

```
SELECT * FROM MarksView;
```

Output:

NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

DELETING VIEWS

We can delete or drop a View using the DROP statement

DROP VIEW view_name;

view_name: Name of the View which we want to delete.

For example, if we want to delete the View MarksView, we can do this as:

DROP VIEW MarksView;

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is not met, then we will not be allowed to update the view.

- SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause
- The SELECT statement should not have the DISTINCT keyword
- The View should have all NOT NULL values
- The view should not be created using nested queries or complex queries
- The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view

We can use the CREATE OR REPLACE VIEW statement to add or remove fields from a view.

Syntax:

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

For example, if we want to update the view MarksView and add the field AGE to this View from StudentMarks Table

```
CREATE OR REPLACE VIEW MarksView AS
SELECT StudentDetails.NAME, StudentDetails.ADDRESS,
StudentMarks.MARKS, StudentMarks.AGE
FROM StudentDetails, StudentMarks
WHERE StudentDetails.NAME = StudentMarks.NAME;
```

SELECT * FROM MarksView;

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

Inserting a row in a view: We can insert a row in a View in a same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View

INSERT INTO view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);

view_name: Name of the View

Example: In the below example we will insert a new row in the View DetailsView which we have created above in the example of “creating views from a single table”.

```
INSERT INTO DetailsView(NAME, ADDRESS)
VALUES("Suresh","Gurgaon");
```

```
SELECT * FROM DetailsView;
```

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

Deleting a row from a View: Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view

DELETE FROM view_name
WHERE condition;

view_name:Name of view from where we want to delete rows
condition: Condition to select rows

DELETE FROM DetailsView
WHERE NAME="Suresh";

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

Uses of a View

- **Restricting data access** – Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table

- **Hiding Data complexity** – A view can hide the complexity that exists in a multiple table join.

- **Simplify commands for the user** – Views allows the user to select information from multiple tables without requiring the users to actually know how to perform a join

- **Store complex queries** – Views can be used to store complex queries

Trigger

- A trigger is a set of actions that are run automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table.

- A Trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs.

- For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated

BEFORE and AFTER of Trigger:

1. **BEFORE triggers** run the trigger action before the triggering statement is run.

2. **AFTER triggers** run the trigger action after the triggering statement is run.

```
create trigger [trigger_name]
[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]
```

SQL Trigger to problem statement.

Given Student Report Database, in which student marks assessment is recorded.

In such schema, create a trigger so that the total and average of specified marks is automatically inserted whenever a record is insert.

Here, as trigger will invoke after record is inserted so, AFTER Tag can be used.

```
CREATE TRIGGER after_insert_records AFTER INSERT
ON student
BEGIN
  UPDATE student SET total = subject1 + subject2 + subject3;
END
```

Above SQL statement will create a trigger in the student database in which whenever subjects marks are entered, before inserting this data into the database, trigger will compute those two values and insert with the entered values. i.e.,

```
INSERT INTO student VALUES
(11, 'Snehal', 10, 20, 30, 0);

select * from Student;
+-----+-------+-------+-------+-------+-------+-------+
| tid | name  | subj1 | subj2 | subj3 | total |
+-----+-------+-------+-------+-------+-------+-------+
| 11| Snehal|   10|   20 |   30|   0|
+-----+-------+-------+-------+-------+-------+-------+
```

SQL Trigger | BEFORE INSERT

```
CREATE TRIGGER after_insert_records BEFORE INSERT
ON student
BEGIN
UPDATE student SET total = subject1 + subject2 + subject3;
END
```


Thank
you!