upGrad

#LifeKoKaroLift

TechM Full Stack Software Development

upGrad

Course: Foundation of Databases
Lecture On: Subqueries and Joins
Instructor: Vishwa Mohan

upGrad

Today's Agenda

1. Subqueries
    1. NESTED
    2. CORRELATED
2. Alias and Synonyms
3. Joins
    - Natural Join
    - Inner Join
    - Left Join
    - Right Join
    - Outer Join
    - Self Join
4. Views

Schema - Movie Booking application

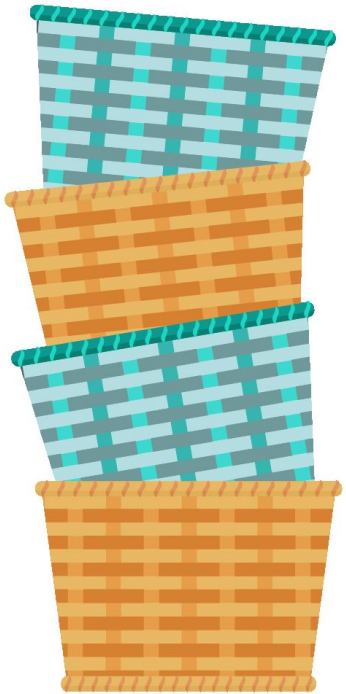
upGrad

- In the last class, we learnt about the normalization process and how it helps in resolving data anomalies.
- We have discussed the movie booking application throughout the course so far.
- Let us see what the final schema of application looks like.

Let’s take a look at Project Schema.

Subqueries

upGrad

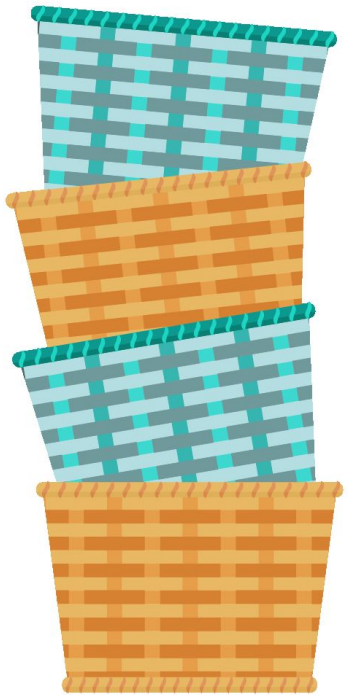


A subquery is query within another query.
The outer query is called the main query and the inner query is called as the subquery.

A subquery can be used with statements such as SELECT, UPDATE, INSERT and DELETE, and operators such as IN, NOT IN and BETWEEN.

Subqueries

upGrad



Rules for writing subqueries are as follows:

1. A subquery should be enclosed within parentheses.

2. A subquery should generally have only one column in the *SELECT* clause.

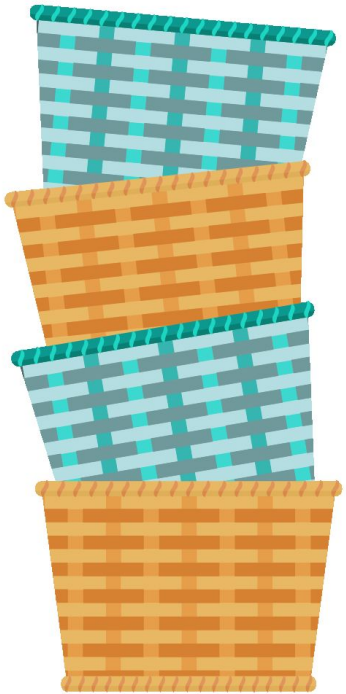
3. *The ORDER BY* clause cannot be used within a subquery; instead, the *GROUP BY* clause is used to perform the same function.

4. If a subquery returns multiple records, then it should be handled in the *WHERE* clause with operators such as *IN.*

5. *The BETWEEN* operator cannot be used with a subquery; however, it can be used within a subquery.

Subqueries

upGrad



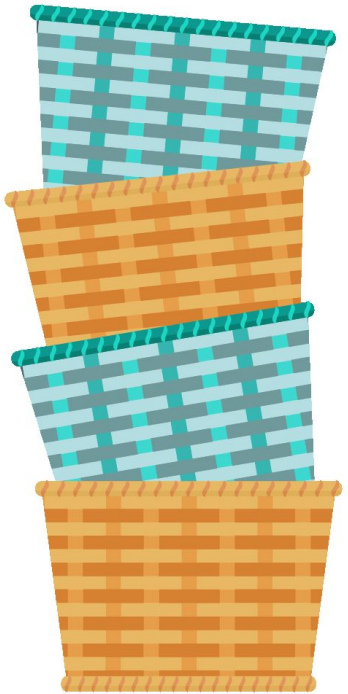
Subqueries are of the following two types:

1. **Nested Subqueries**

   Nested subquery is one where the innermost queries are executed and their result is used to execute the outer query.

2. **Correlated Subqueries**

   Correlated subqueries adopt the approach opposite to that of nested subqueries. Each subquery is executed once for every row of the outer query.



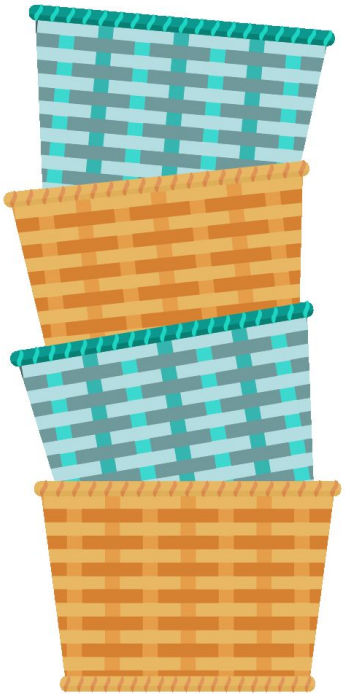
Nested Subqueries

Nested subquery is one where the innermost queries are executed and their result is used to execute the outer query.

The syntax for nested subqueries is as follows:

```
SELECT column1, column2
FROM table1
WHERE column3 operator
(SELECT column3
FROM table2 where <some condition>);
```

Let’s take a look at Example 1.



Correlated Subqueries

Correlated queries adopt the approach opposite to that of normal subqueries. They are executed after the execution of the outer query.

A correlated subqueries should be used when the result depends on each row of the parent statement.

The syntax for correlated subqueries is as follows:

```
SELECT column1, column2
FROM table1 outer
WHERE column1 operator
(SELECT func(column1)
FROM table2
WHERE expr1 = outer.expr2);
```

Let’s take a look at Example 2.

Subquery: Hands-On (10 min)

1. Find out the details of all the theatres where Hindi movies are screened.
2. Find out all upcoming movies with duration greater than average duration of all upcoming movies

Poll 1 (45 sec)

Which of the following type of subquery should be used when the result depends on each row of the parent statement?

1. Nested Subquery

2. Correlated Subquery

Poll 1 (Answer)

Which of the following type of subquery should be used when the result depends on each row of the parent statement?

1. Nested Subquery

2. **Correlated Subquery**

Poll 2 (45 sec)

State whether the following statement is true or false:

In a correlated subquery, the subquery uses the correlation name of the outer query.

1. True

2. False

Poll 2 (Answer)

State whether the following statement is true or false:

In a correlated subquery, the subquery uses the correlation name of the outer query.

1. True

2. False

Poll 3 (45 sec)

State whether the following statement is true or false:

The ORDER BY clause can be used within a subquery.

1. True

2. False

Poll 3 (Answer)

State whether the following statement is true or false:

The ORDER BY clause can be used within a subquery.

1. True

2. False

Synonyms

upGrad

A synonym is an alias given to database objects such as tables, views and stored procedures.

The syntax for creating a synonym is as follows:

CREATE SYNONYM *synonym_name* FOR *table_name*;

It is beneficial to use synonyms as they can:

1. Simplify complex object names and
2. Enable name changes to objects

Let’s take a look at Example 3.

Alias vs Synonym

upGrad

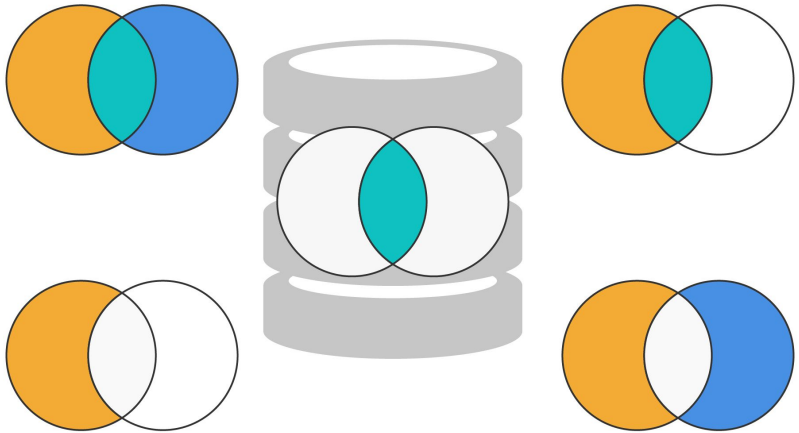
Differences between Aliases and Synonyms

Alias	Synonym
It is temporary for each query.	It is permanent for each query and can be used for multiple queries.
It is not stored anywhere.	It is stored in a local or a remote server.

Continuing Example 3.

SQL JOIN

upGrad



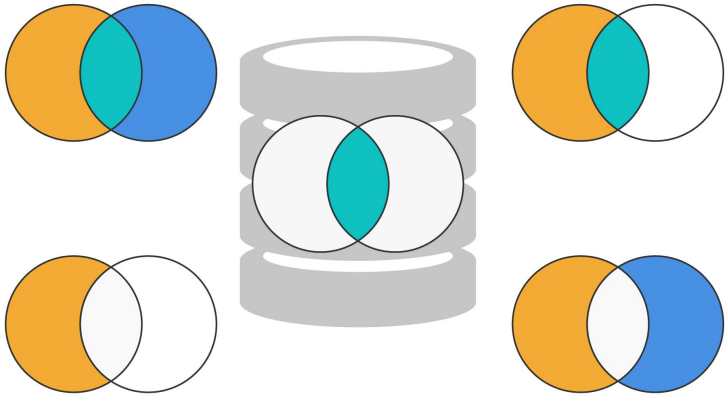
The JOIN clause is used to combine rows from two or more tables.

JOINs in SQL are of the following types:

1. Natural Join
2. Inner Join
3. Left Join
4. Right Join
5. Full Outer Join
6. Self Join

SQL JOIN

upGrad

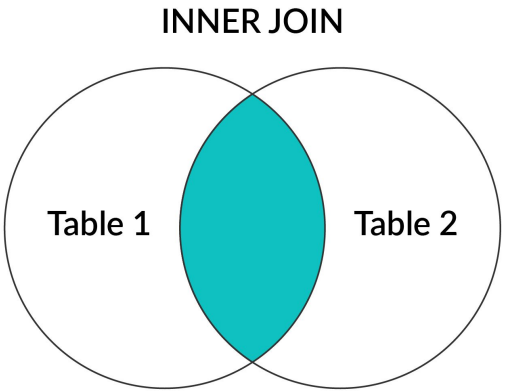


NATURAL JOIN is a JOIN operation that joins two tables based on the common columns that have the same name and data types.

The syntax for natural JOIN is as follows:

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

Let's take a look at Example 4.



INNER JOIN

An INNER JOIN returns all the records that have matching values in both tables.

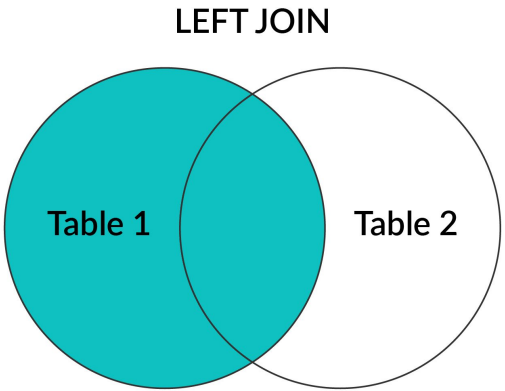
The syntax for INNER JOIN is as follows:

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

Let’s take a look at Example 5.

Differences between a NATURAL JOIN and an INNER JOIN

NATURAL JOIN	INNER JOIN
It is possible only when the two tables have columns having the same name.	It can be achieved even if the names of the two columns are not the same.
It creates an implicit join.	In this, we explicitly mention the columns with the ON clause which are used to join two tables.
It is mostly used in extremely simple joins.	It can be used in complex join queries.
The common column appear only once in the output.	The common columns appear twice in the output.



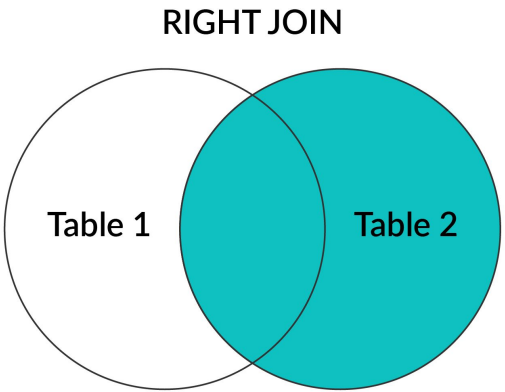
LEFT JOIN

A LEFT JOINS returns all the records from the left table and the matched records from the right table.In other words, all the columns of the left table are returned with matching records from the right table in case found.

The syntax for LEFT JOIN is as follows:

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

Continuing Example 5



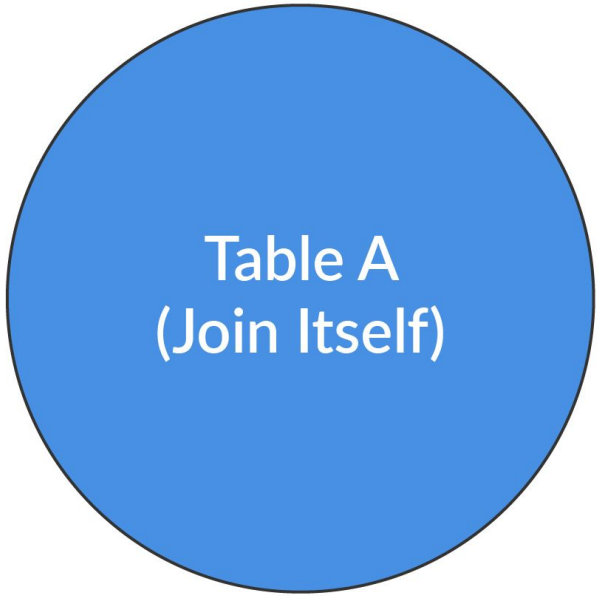
RIGHT JOIN

A right join returns all the records from the right table and the matched records from the left table. In other words, all the columns of the right table are returned with matching records from the left table in case found.

The syntax for RIGHT JOIN is as follows:

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

Continuing Example 5



Self Join

FULL OUTER JOIN

A full outer join returns all the records from both the tables irrespective of whether there is a match in the records in the left or the right table. In other words, all the columns both the tables are returned with matching records wherever found.

The syntax for OUTER JOIN is as follows:

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

Continuing Example 5

Poll 4 (45 sec)

State whether the following statement is true or false:

A self join is used to join a table to itself.

1. True

2. False

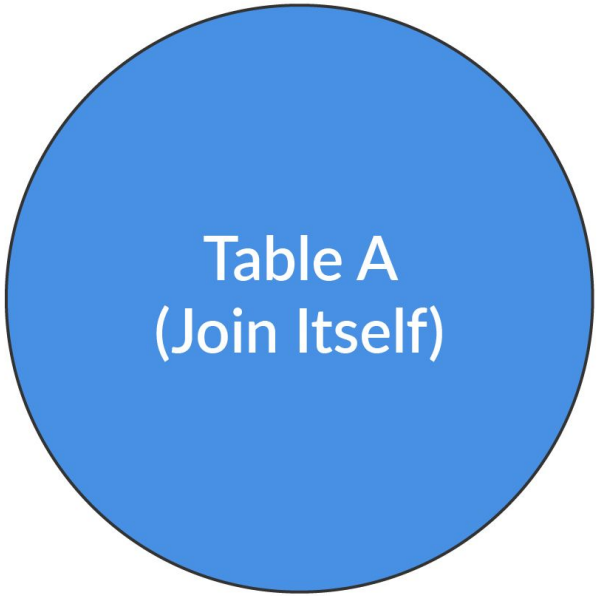
Poll 4 (Answer)

State whether the following statement is true or false:

A self join is used to join a table to itself.

1. True

2. False



Self Join

SELF JOIN

It is a join that joins a table to itself. It useful in comparing columns in the same table or querying hierarchical data.

The syntax for SELF JOIN is as follows:

```
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;
```

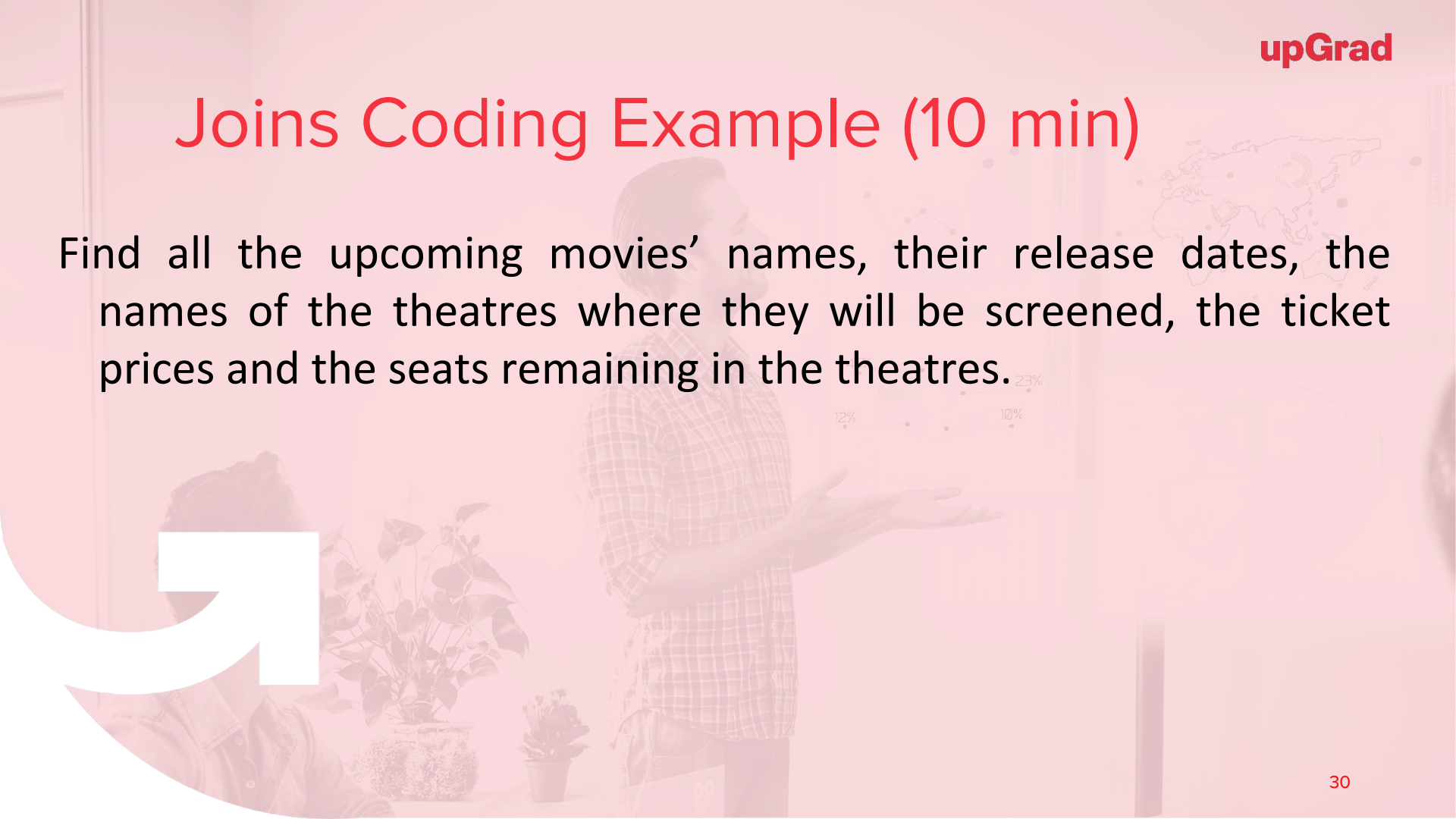
Continuing Example 5

Joins: Hands-On Excercise (10 min)

1. Find the names of all the customers who have booked movie tickets.
2. Find the list of all the customers including those who have not booked movie tickets.
3. Find the list of all the people who have booked movie tickets, even if their names are not present in the ‘customers’ table.
4. Find the details of all the customers and ticket bookings.

Joins Coding Example (10 min)

Find all the upcoming movies' names, their release dates, the names of the theatres where they will be screened, the ticket prices and the seats remaining in the theatres.



Poll 5 (45 sec)

Which of the following is the default Natural JOIN?

1. Natural Left Join

2. Natural Inner Join

3. Natural Right Join

4. Natural Full Join

Poll 5 (Answer)

Which of the following is the default Natural JOIN?

1. Natural Left Join

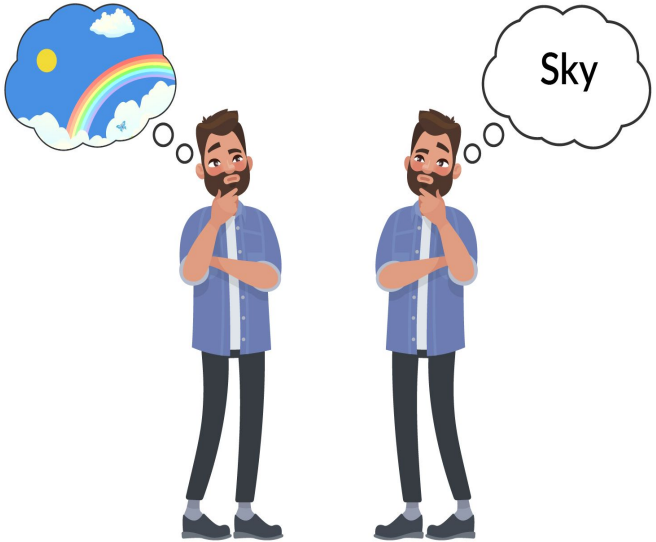
2. Natural Inner Join

3. Natural Right Join

4. Natural Full Join

View

upGrad



What is a **VIEW**?

A **VIEW** is a virtual table based on the results of an SQL query. We can run complex join queries and present the result in the form of this table.

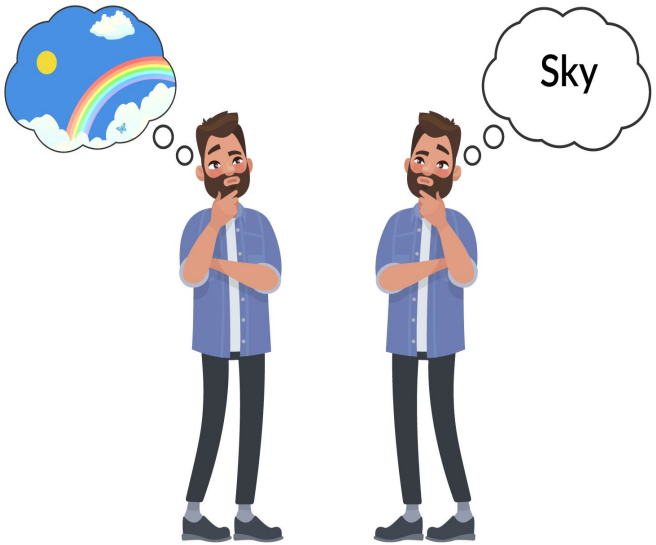
The syntax used to create a VIEW is as follows:

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

Let’s take a look at Example 6.

Materialized VIEW

upGrad



A materialized VIEW, which is similar to a normal **VIEW**, is based on the results of a SQL query. However, it is physically stored on the disk.

The syntax used to create a materialized VIEW is as follows:

```
CREATE MATERIALIZED VIEW view_name
BUILD [IMMEDIATE/DEFERRED]
REFRESH [FAST/COMPLETE/FORCE]
ON [COMMIT/DEMAND] AS
      <query expression>
```

Note:

The BUILD clause decides when to populate data into view.

The REFRESH clause decides how to update data.

The TRIGGER clause decides when to update data.

Let’s take a look at Example 7.

upGrad

Differences between a Normal View and a Materialised View

Normal View	Materialised View
It is a virtual table.	It is a type of actual table that stores data.
It always fetches the latest data.	It fetches stale data unless it is refreshed.
It results in a slow query.	It results in a fast query, as you can apply different ways of query optimisation.

Views: Hands-On (10 min)

1. Create a view that returns data about theatres and the cities in which they lie.
2. Create a materialised view to store the results of all customer details, such as their first names, usernames and the number of seats that they have booked, if any.

Poll 6 (45 sec)

Which of the following are benefits of using views?

1. Better security

2. Hiding unnecessary data

3. Both 1 and 2

4. None of the above

Poll 6 (Answer)

Which of the following are benefits of using views?

1. Better security

2. Hiding unnecessary data

3. Both 1 and 2

4. None of the above

Poll 7 (45 sec)

State yes or no.

Is a view a virtual table?

1. Yes
2. No

Poll 7 (Answer)

State yes or no.

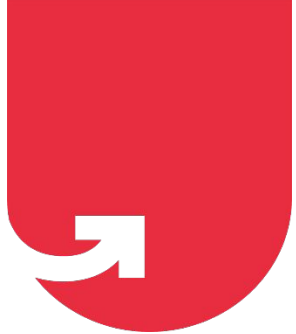
Is a view a virtual table?

1. **Yes**
2. No


upGrad
DISCUSSION TIME

upGrad

#LifeKoKaroLift



Thank You!