Experiment No. 3

Group: - 1 from T4 Batch

0077 – Aakash Joshi

2039 – Akanksha Lokhande

Aim: -

SQL Queries – all types of Join, Sub-Query and View:

Write at least 10 SQL queries for suitable database application using SQL DML statements.

Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join ,Sub-Query and View

Date: - 22/8/2023

Theory: -

Joins in SQL are essential for retrieving data from multiple tables based on relationships between them. They enable you to combine and correlate data from various sources, creating a comprehensive dataset for analysis. SQL offers several types of joins:

1. INNER JOIN: The most common type, INNER JOIN, returns only rows that have matching values in both tables, eliminating non-matching rows. It's used when you want to retrieve data that exists in both tables.

2. LEFT JOIN (or LEFT OUTER JOIN): This returns all rows from the left (or first) table and the matched rows from the right (or second) table. If there are no matches in the right table, NULL values are returned.

3. RIGHT JOIN (or RIGHT OUTER JOIN): Similar to LEFT JOIN, but it returns all rows from the right table and matched rows from the left table.

4. FULL JOIN (or FULL OUTER JOIN): Combines data from both tables, including unmatched rows from both sides. Rows without matches in either table are filled with NULL values.

5. SELF JOIN: A self join occurs when a table is joined with itself, creating a virtual relationship. This is often used for hierarchical data or when comparing records within the same table.

Joins play a pivotal role in creating complex queries to analyze relationships in your data. They are crucial in business intelligence, reporting, and data analysis, allowing users to extract meaningful insights by connecting and aggregating data from various parts of a database. Proper understanding and use of joins are essential skills for anyone working with relational databases.

SQL subqueries, also known as inner queries or nested queries, are a powerful feature that allows you to nest one query within another. These subqueries are encapsulated within the main query and are used to retrieve data necessary for the main query's execution. They play a crucial role in enhancing the flexibility and expressiveness of SQL.

Subqueries can be used in various scenarios:

1. Filtering: Subqueries can filter the results of the outer query. For example, you can use a subquery to find all customers who have placed orders in the last month.

2. Column expressions: Subqueries can be used in the SELECT clause to retrieve a single value, which can then be used as a part of the result set. An example is retrieving the highest or lowest value in a column.

3. Table expressions: Subqueries can be used to retrieve an entire result set, which can be treated as a temporary table within the main query. This is often used in JOIN operations.

4. Correlated Subqueries: These subqueries are executed for each row processed by the outer query, allowing for complex conditions based on values in both the outer and inner queries.

Subqueries can improve code readability and reduce the need for complex joins, especially in scenarios where you need to access data from multiple tables with hierarchical or dependent relationships. However, it's essential to optimize subqueries for performance as poorly constructed subqueries can slow down query execution.

In summary, SQL subqueries are a versatile tool for building complex and efficient queries, enabling developers to manipulate and retrieve data more effectively from relational databases. They are an integral part of SQL, enhancing its capability to handle a wide range of data retrieval and manipulation tasks.

Views in SQL are virtual tables created by a query. They are incredibly useful and versatile tools for simplifying database management, enhancing security, and improving query efficiency. A view is a saved SQL query that can be treated as a table when retrieving data. Here are the key aspects of views in SQL:

1. Data Abstraction: Views abstract the underlying complexity of a database by presenting a subset of data or a different perspective. This simplifies data access for users, making it easier to work with.

2. Security: Views can restrict access to sensitive data. By defining a view that shows only necessary columns and rows, you can control what specific users or roles can see, ensuring data privacy and security.

3. Data Consistency: Views promote data consistency by centralizing complex joins, calculations, and business logic. Users can access this information consistently, reducing the risk of errors.

4. Performance Optimization: Views can improve query performance by pre-computing and storing complex queries. This minimizes redundant calculations and speeds up query execution.

5. Data Partitioning: Views can be used to partition a large table into smaller, manageable pieces, making it easier to maintain and query.

6. Simplified Querying: Instead of writing the same complex query multiple times, views allow you to encapsulate it once and reuse it as if it were a regular table.

7. Schema Evolution: Views can insulate application code from underlying schema changes. Even if the structure of tables changes, views can remain consistent, reducing the impact on application code.

In summary, views in SQL are a powerful feature that enhances data management, security, and query optimization. They provide a layer of abstraction that simplifies data access and maintains data integrity while improving query performance and ensuring that sensitive data remains protected.

Conclusion: -

The task of creating and executing SQL queries encompassing various types of joins, sub-queries, and views provides valuable insights into the capabilities of SQL and its practical application in database management. This exercise allows learners to apply their understanding of complex data retrieval and manipulation scenarios. Here are the key takeaways from this exercise:

1. Understanding of Join Types: Writing SQL queries involving various join types, such as INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN, illustrates the importance of connecting data from multiple tables. This skill is crucial for retrieving relevant information from normalized databases.

2. Data Integration: By composing queries with joins, learners discover how to combine data from different tables and derive meaningful insights. The ability to correlate data from various sources is essential for reporting and analytics.

3. Sub-Queries for Data Filtering: The use of sub-queries demonstrates how to filter and refine data by nesting queries within other queries. This technique is particularly useful for fetching specific subsets of data or for advanced filtering conditions.

4. Aggregation and Calculation: Queries may involve sub-queries for calculating aggregated values, such as sum, average, or count. This showcases the power of SQL for performing complex data calculations.

5. Data Abstraction with Views: Writing SQL queries that create and utilize views enhances the understanding of data abstraction. Views allow users to simplify complex queries and access data in a structured, user-friendly manner.

6. Query Optimization: During the process of writing these queries, learners may encounter situations where query performance can be optimized. This encourages the development of skills in query tuning and optimization.

7. Real-World Application: The queries created reflect real-world scenarios, such as sales reporting, inventory management, or customer analysis. This highlights the practical utility of SQL in various domains.

8. Problem-Solving Skills: Crafting SQL queries is akin to problem-solving, where learners apply their knowledge to extract meaningful information from large datasets. It fosters analytical thinking and logical reasoning.

9. Data Security: Understanding how to use views for data abstraction and access control is vital for ensuring data security and privacy in database systems.

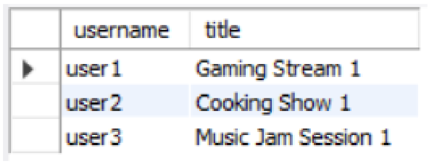
In conclusion, the exercise involving SQL queries with joins, sub-queries, and views is an integral part of mastering database management and SQL. It empowers learners to manipulate data effectively, generate insights, and optimize data retrieval, all of which are indispensable skills in the world of data-driven decision-making and information management.

Code: -

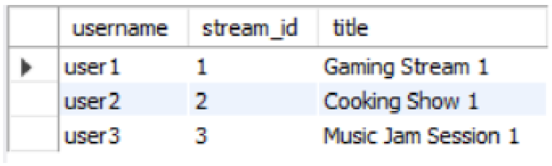
```
SELECT User.username, Stream.title

FROM User

JOIN Stream ON User.streamer_id = Stream.stream_id;


-- Select all users and their associated streams (if any)

SELECT User.username, Stream.stream_id,Stream.title
```

```
FROM User

Right JOIN Stream ON User.streamer_id = Stream.stream_id;


-- Find users with the most popular video

SELECT User.username

FROM User

WHERE User.uid = (

   SELECT Video.video_id

   FROM Video

   ORDER BY Video.viewer_count DESC

   LIMIT 1

);


-- Create a view to display stream information along with categories

CREATE VIEW StreamCategories AS

SELECT Stream.stream_id, Stream.title, Stream.stream_category

FROM Stream;


-- Query data from the "StreamCategories" view

SELECT *

FROM StreamCategories;
```

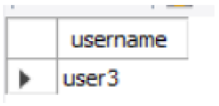
Output: -



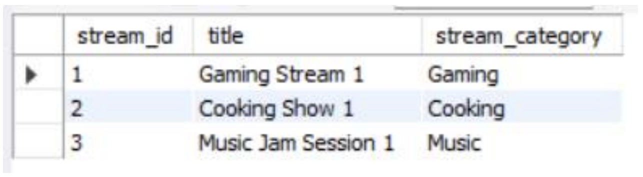
username	title
user1	Gaming Stream 1
user2	Cooking Show 1
user3	Music Jam Session 1



username	stream_id	title
user1	1	Gaming Stream 1
user2	2	Cooking Show 1
user3	3	Music Jam Session 1



username
user3



stream_id	title	stream_category
1	Gaming Stream 1	Gaming
2	Cooking Show 1	Cooking
3	Music Jam Session 1	Music