Demystifying SQL: Advanced Techniques for Data Integration and Analysis

SQL i.e. Structured Query Language, serves as the foundation of relational databases, providing a robust framework for managing and analyzing data. While fundamental SQL commands are foundational, exploring advanced techniques unlocks a world of opportunities for seamless data integration and insightful analysis. In this article, we explore into the intricacies of advanced SQL concepts, from the nuances of joins to the power of aggregation, all within the context of real-world scenarios.

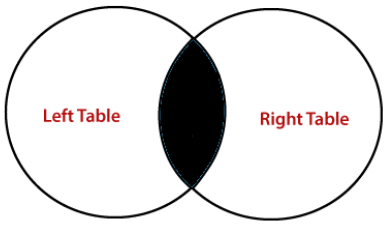
Demystifying SQL Joins: Inner, Left, Right, Full and Cross

In the world of databases, SQL joins play a pivotal role in amalgamating data from multiple tables based on a common column. Thus joins are of various types that are inner, outer, left, and right. Each join serves a distinct purpose in crafting intricate queries that unravel insights hidden within interconnected datasets.

Inner Join: Inner join or explicitly inner join returns rows that have matching values in both tables.

Syntax :-

```
SELECT Column_Name1, Column_Name2,..., Column_NameN
FROM Table_Name1
INNER JOIN Table_Name2
ON
Table_Name1.Column_Name=Table_Name2.Column_Name;
```



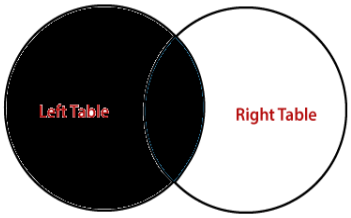
INNER JOIN

Left Join: Left join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain null. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:-

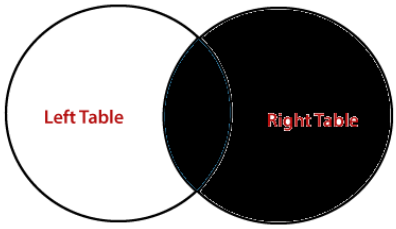
```
SELECT TableName1.columnName1,TableName2.columnName2
```

FROM TableName1
LEFT JOIN TableName2
ON TableName1.ColumnName=TableName2.ColumnName;



Left Join

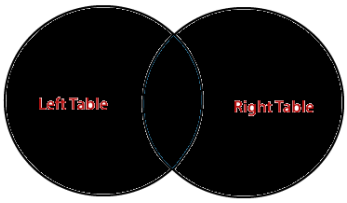
Right Join: RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.
Syntax:-
SELECT TableName1.columnName1, TableName2.columnName2
FROM TableName1
RIGHT JOIN TableName2
ON
TableName1.ColumnName=TableName2.ColumnName;



Right Join

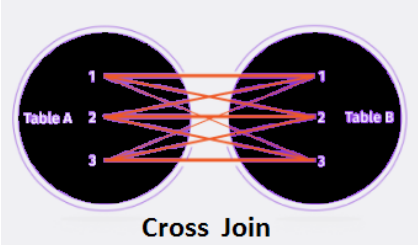
Full Join: FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.
Syntax:-
SELECT * FROM table1
FULL OUTER JOIN table2

ON
table1.column_name=table2.column_name;



Full Join

Cross Join: Cross join returns the Cartesian product of the sets of rows from the tables that are joined.
Syntax:-
SELECT column_name
FROM TableA
CROSS JOIN TableB;



Cross Join

Advanced SQL Techniques: Data Manipulation

Advanced SQL techniques for data manipulation go beyond basic commands, allowing users to perform intricate operations on datasets with precision. These techniques encompass a range of statements, each serving a distinct purpose in modifying or managing data within a database.
Insert: Insert is used to insert a single or a multiple records in a table.
Syntax:-
INSERT INTO table_name
VALUES (value1, value2, value3....);
For Example:-
INSERT INTO STUDENTS
VALUES (1, ABHIRAM, 22, ALLAHABAD),
(2, ALKA, 20, GHAZIABAD),
(3, DISHA, 21, VARANASI);

ROLL_NO	NAME	AGE	CITY
1	ABHIRAM	22	ALLAHABAD
2	ALKA	20	GHAZIABAD
3	DISHA	21	VARANASI

Update: Update is used to change the data of the records held by tables. Which rows is to be update, it is decided by a condition. To specify condition, we use WHERE clause.
Syntax:-
UPDATE table_name
SET column_name = expression
WHERE conditions ;
For Example:-

Student_Id	FirstName	LastName	User_Name
1	Ada	Sharma	sharmili
2	Rahul	Maurya	sofamous
3	James	Walker	jonny

UPDATE students
SET User_Name = 'beinghuman'
WHERE Student_Id = '3' ;

Student_Id	FirstName	LastName	User_Name
1	Ada	Sharma	sharmili
2	Rahul	Maurya	sofamous
3	James	Walker	**beinghuman**

Delete: Delete is used to delete rows from a table. Generally DELETE statement removes one or more records from a table.
Syntax:-
DELETE FROM table_name
WHERE condition;
For Example:-

ID	EMP_NAME	CITY
101	Adarsh Singh	Obra
102	Sanjay Singh	Meerut
103	Priyanka Sharma	Raipur

DELETE FROM EMPLOYEE WHERE ID=101;

ID	EMP_NAME	CITY
102	Sanjay Singh	Meerut
103	Priyanka Sharma	Raipur

Merge: The MERGE statement performs insert, update, or delete operations on a target table based on the results of a join with a source table.

By mastering these advanced SQL techniques for data manipulation, users can effectively manage and transform datasets to meet their analytical and operational needs. Whether updating existing records, removing obsolete data, adding new entries, or synchronizing information across tables, these commands empower users to maintain accurate and efficient databases.

SQL Aggregation: Group By, Having, and Aggregate Functions

SQL offers powerful aggregation capabilities, allowing users to perform calculations across multiple rows, often grouped by specific criteria. Understanding the key components of SQL aggregation is essential for mastering this aspect of database querying:

1) Group by: The Group by is used for organizing similar data into groups.
The SELECT statement is used with the GROUP BY clause.
WHERE clause is placed before the GROUP BY clause.
ORDER BY clause is placed after the GROUP BY clause.
Syntax:-

```
SELECT column1, function_name(column2)
FROM table_name
WHERE condition
GROUP BY column1, column2;
```

For Example:-

S.no	Name	AGE	Salary
1	John	24	25000
2	Nick	22	22000
3	Amara	25	15000
4	Nick	22	22000
5	John	24	25000

SELECT NAME, SUM (SALARY)
FROM Employee
GROUP BY NAME;

NAME	SALARY
John	50000
Nick	44000
Amara	15000

2) Having: Having is used to place conditions on the columns to determine the part of the last result-set of the group. Here, we are not required to use the combined functions like COUNT (), SUM (), etc. with the WHERE clause. After that, we need to use a HAVING clause.
Syntax:-
SELECT column1, function_name(column2)
FROM table_name
WHERE condition
GROUP BY column1, column2
HAVING condition;
For Example:-
SELECT NAME, SUM(SALARY)
FROM Employee
GROUP BY NAME
HAVING SUM(SALARY)>23000;

Name	SUM(SALARY)
John	50000

3) Aggregate Functions: SQL provides a variety of aggregate functions which includes COUNT(), SUM(), AVG(), MIN(), and MAX(), which perform calculations on a set of values within a group.

- **COUNT():-** The COUNT is an aggregate function in SQL which returns the total number of rows from the table.
  Syntax:-
  SELECT COUNT(column_Name) AS Alias_Name FROM Table_Name;
  For Example:-

Product_ID	Product_Name	Product_Quantity
104	P1	10.250
202	P4	15.500
103	P2	18.250
111	P7	25.250
210	P6	15.500
212	P8	19.750
112	P10	10.250

  SELECT COUNT(Product_Quantity) AS Total_Number_ofproduct
  FROM Product_Details;

Total_Number_ofproduct
7

- **SUM():-**The SUM is an aggregate function in SQL which returns the sum of integer column of the table.
  Syntax:-
  SELECT SUM(column_Name) AS Alias_Name FROM Table_Name;
  For Example:-
  SELECT SUM(Product_Quantity) AS Total_sum_ofproductquantity
  FROM Product_Details;

Total_sum_ofproductquantity
114.75

- **AVG():-**The AVG is an aggregate function in SQL which returns the average of values of the integer column of the table.
  Syntax:-
  SELECT AVG(column_Name) AS Alias_Name FROM Table_Name;
  For Example:-
  SELECT AVG(Product_Quantity) AS Average_ofproductquantity
  FROM Product_Details;

Average_ofproductquantity
16.39

- **MIN():-**The MIN is an aggregate function in SQL which returns the minimum or smallest value from the specified column of the table.
  Syntax:-
  SELECT MIN(column_Name) AS Alias_Name FROM Table_Name;
  For Example:-
  SELECT MIN(Product_Quantity) AS Minimum_in_productquantity
  FROM Product_Details;

Minimum_in_productquantity
10.250

- **MAX():-**The MAX is an aggregate function in SQL which returns the maximum or largest value from the specified column of the table.
  Syntax:-
  SELECT MAX(column_Name) AS Alias_Name FROM Table_Name;
  For Example:-
  SELECT MAX(Product_Quantity) AS Maximum_in_productquantity
  FROM Product_Details;

Maximum_in_productquantity
25.250

In conclusion, acquiring mastery over advanced SQL techniques equips users with the ability to adeptly manipulate, integrate, and analyze data sourced from various channels. Through comprehensive comprehension of SQL joins, data manipulation commands, and aggregation techniques, professionals can harness SQL's full potential to tackle intricate data obstacles. By exploring into practical examples and engaging in hands-on experience, individuals can refine their SQL proficiency and effectively utilize data to steer informed decision-making processes.