SQL(Week 3)

DSBA CURRICULUM DESIGN

FOUNDATIONS	CORE COURSES	DOMAIN APPLICATIONS
Python for Data Science	**Advanced Statistics**	**Financial Risk Analytics**
Statistical Methods for Decision Making	**Data Mining**	**Marketing Retail Analytics**
	Predictive Modelling	
	Machine Learning	
	Data Visualization	
	SQL (Week-3/3)	
	Time Series Forecasting	

Table of Contents

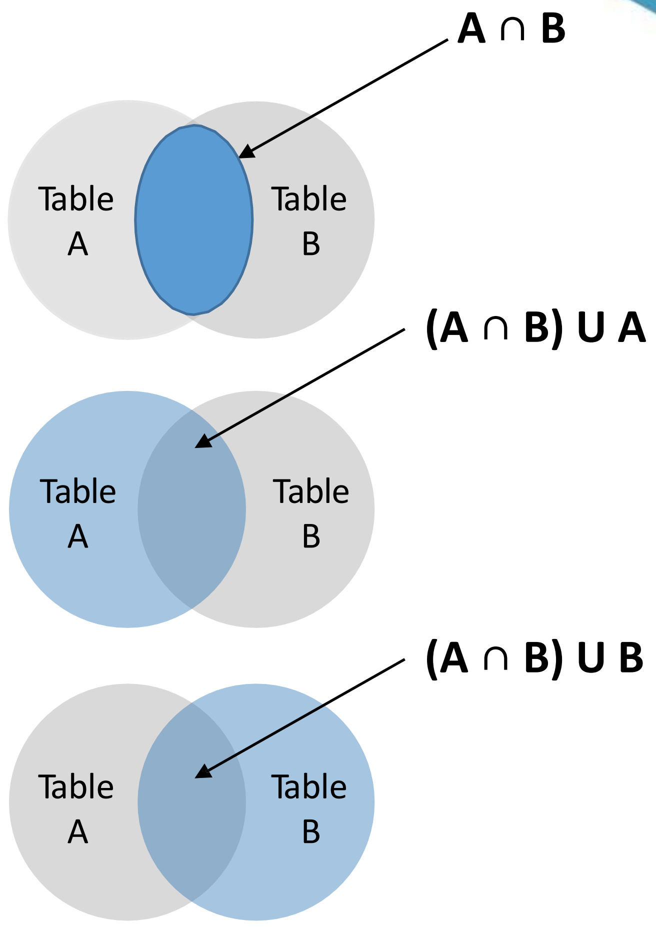
greatlearning
Learning for Life

SQL Joins

SQL JOINS

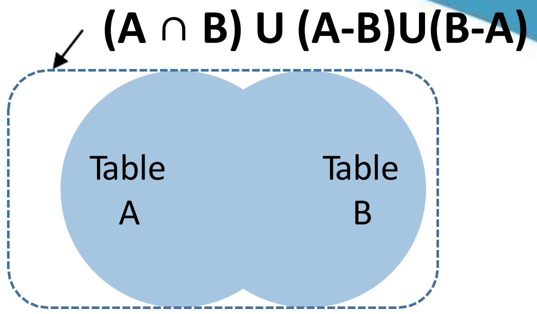
Combine rows from two or more tables, based on a related column between them.

- **INNER JOIN** – Returns rows when there is a match in both tables.
- **LEFT JOIN** – Returns all rows from the left table, even if there are no matches in the right table.
- **RIGHT JOIN** – Returns all rows from the right table, even if there are no matches in the left table.

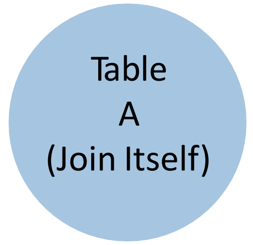


SQL JOINS

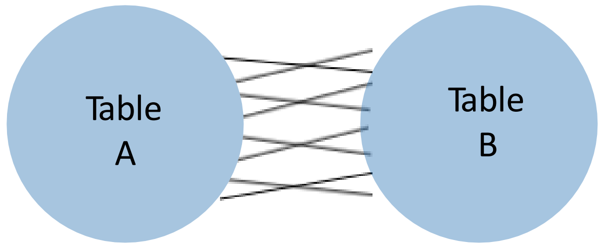
- **FULL OUTER JOIN** – Returns rows when there is a match in one of the tables.



- **SELF JOIN** – Used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.



- **CARTESIAN JOIN (CROSS JOIN)** – Returns the Cartesian product of the sets of records from the two or more joined tables.



SQL INNER JOIN

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate.

SYNTAX :

```
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
INNER JOIN table2
ON table1.commonfield = table2.commonfield;
```

SQL INNER JOIN

- Display details of employee and department
- department_id is a common column between employees & departments tables
- 'e' and 'd' are alias for the table names

SQL INNER JOIN

employee_id	first_name	last_name	salary	department_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

department_id	department_name	manager_id	location_id
10	IT	200	1700
11	Marketing	201	1800
13	Resources	203	2400
14	Shipping	121	1500

```
SELECT e.employee_id,
e.first_name, e.last_name,
d.department_id,
d.department_name
FROM employees e
INNER JOIN departments d
ON
e.department_id=d.department_id
;
```

employee_id	first_name	last_name	department_id	department_name
101	Steven	Cohen	10	IT

SQL LEFT JOIN

The LEFT JOIN returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

SYNTAX :

```
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
LEFT JOIN table2
ON table1.commonfield = table2.commonfield;
```

SQL LEFT JOIN

- List all employees those with and without departments
- Department names will be NULL for those employees in which there are no departments

greatlearning
Learning for Life

SQL LEFT JOIN

employee_id	first_name	last_name	salary	Department_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

department_id	department_name	manager_id	location_id
10	IT	200	1700
11	Marketing	201	1800
13	Resources	203	2400
14	Shipping	121	1500

```
SELECT e.employee_id,
e.first_name, e.last_name,
d.department_id, d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON e.department_id =
d.department_id;
```

employee_id	first_name	last_name	department_id	department_name
101	Steven	Cohen	10	IT
102	Edwin	Thomas	11	Marketing
103	Harry	Potter	Null	Null
100	Erik	John	Null	Null

SQL RIGHT JOIN

- The RIGHT JOIN returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

SYNTAX :

```
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
RIGHT JOIN table2
ON table1.commonfield = table2.commonfield;
```

SQL RIGHT JOIN

- List all departments those with and without managers
- For those departments where there are no managers, their names will be shown as NULL

SQL RIGHT JOIN

employee_id	first_name	last_name	salary	department_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

department_id	department_name	manager_id	location_id
10	IT	200	1700
11	Marketing	201	1800
13	Resources	203	2400
14	Shipping	121	1500

```
SELECT e.employee_id, e.first_name,
e.last_name, d.department_id,
d.department_name
FROM employees e
RIGHT JOIN departments d
ON e.department_id=d.department_id;
```

employee_id	first_name	last_name	department_id	department_name
101	Steven	Cohen	10	IT
102	Edwin	Thomas	11	Marketing
Null	Null	Null	13	Resources
Null	Null	Null	14	Shipping

SQL FULL OUTER JOIN

The FULL OUTER JOIN combines the results of both left and right outer joins. The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.

SYNTAX :

```
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
Left JOIN table2
ON table1.commonfield = table2.commonfield;
Union
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
Right JOIN table2
ON table1.commonfield = table2.commonfield;
```

greatlearning
Learning for Life

SQL FULL OUTER JOIN

employee_id	first_name	last_name	salary	department_id
103	Harry	Potter	20000	12
102	Edwin	Thomas	15000	11
101	Steven	Cohen	10000	10
100	Erik	John	10000	12

10	IT	200	1700
11	Marketing	201	1800
13	Resources	203	2400
14	Shipping	121	1500

```
SELECT e.employee_id, e.first_name, e.last_name,
       d.department_id, d.department_name
FROM employees e
LEFT JOIN departments d
ON e.department_id=d.department_id
UNION
SELECT e.employee_id, e.first_name, e.last_name,
       d.department_id, d.department_name
FROM employees e
RIGHT JOIN departments d
ON e.department_id=d.department_id;
```

employee_id	first_name	last_name	department_id	department_name
101	Steven	Cohen	10	IT
102	Edwin	Thomas	11	Marketing
103	Harry	Potter	Null	Null
100	Erik	John	Null	Null
Null	Null	Null	13	Resources
		Null	14	Shipping

SQL SELF JOIN

- The SELF JOIN joins a table to itself; temporarily renaming at least one table in the SQL statement.

SYNTAX:

```
SELECT a.col1, b.col2,..., a.coln
FROM table1 a, table1 b
WHERE a.commonfield = b.commonfield;
```

SQL SELF JOIN

employee_id	first_name	manager_id
1	Steven	
2	Neena	1
3	Lex	2
4	Alex	3
5	Bruce	3

```
SELECT emp.employee_id,
emp.first_name, emp.manager_id,
mgr.first_name      FROM
employeess emp inner join
employeess mgr      on
emp.manager_id = mgr.employee_id;
```

employee_id	first_name	manager_id	first_name
2	Neena	1	Steven
3	Lex	2	Neena
4	Alex	3	Lex
5	Bruce	3	Lex

SQL CROSS JOIN

- The CROSS JOIN produces a result set with the number of rows in the first table multiplied by the number of rows in the second.

SYNTAX:

```
SELECT table1.col1, table2.col2,…, table1.coln
FROM table1
CROSS JOIN table2;
```

SQL CROSS JOIN

- Assume there are 4 records in table1 and 3 records in table2

```
SELECT * FROM table1
CROSS JOIN table2;
```

Table-1

alpha
A
B
C
D

Table-2

Num
1
2
3

A	1
A	2
A	3
B	1
B	2
B	3
C	1
C	2
C	3
D	1
D	2
D	3

Sub Query

Sub Query

- ❑ **Single Row Subquery:** It either returns zero or a single row
- ❑ **Multiple Row Subquery:** It returns one or multiple rows of a table
- ❑ **Multiple Column Subquery:** It returns one or multiple columns
- ❑ **Correlated Subqueries:** It refers to one or more columns in the outer SQL query. (Only Brief overview to be covered)

1. MySQL subquery on WHERE Clause

❑ General Format

```
SELECT Field 1, Field2  FROM table 1 WHERE field 3 = (SELECT
MAX(field 3) FROM table 1);
```

Important points to note

✓ Any number of fields could be specified in the first select statement

✓ Field 3 specified in the “where” clause has to be same as the output generated from second “select” statement

✓ Second “Select” statement of a sub-query statement can use either the same table specified in the first query (similar to self join) or it could have a different table

2. MySQL Subquery with operators IN and NOT IN

❑ General Format

```
SELECT Field 1, Field2  FROM table 1 WHERE field 3 NOT IN (SELECT
DISTINCT(field 3) FROM table 2);
```

Important points to note

- ✓ Any number of fields could be specified in the first select statement
- ✓ Field 3 specified in the “where” clause has to be same as the output generated from second “select” statement
- ✓ Second “select” statement of a sub-query statement should refer to a table which is different from the table specified in the first “select” statement.

3. MySQL subquery with FROM clause

- ❑ When we use a subquery with FROM clause, the result set returned is considered as a temporary table rows which is then known as a derived or materialized subquery.

```
SELECT I.Field 1, I.Field 2 FROM (select Ave(Field 1) as
averagefield1 from Table 2) as temptable, Table 1 as I WHERE
I.Field 1 > temptable.averagefield1;
```

Let’s understand the query:

- ✓ Alias is used here – Table 1 alias is I

- ✓ Subquery output is stored in the temp table and the output which is average in the above query is used in the where clause

Data Science @ Work

Apply **Data Science at your workplace** to gain some instant benefits:

- Get noticed by your management with your outstanding analysis backed by data science.
- Create an impact in your organization by taking up small projects/initiatives to solve critical issues using data science.
- Network with members from the data science vertical of your organization and seek opportunities to contribute in small projects.
- Share your success stories with us and the world to position yourself as a subject matter expert in data science.

greatlearning
Learning for Life

?

ANY QUESTIONS

greatlearning
Learning for Life

HAPPY LEARNING