# JOINS & SUBQUERY IN SQL

Thursday 25
(145 - 220) Wk 21

## JOINS - INTRODUCTION

Ques. I want to display customer name & name of the policy taken by each customer.

```
SELECT cname, pname FROM customer c
    JOIN policy p ON c.cid = p.cid;
```

1. A join is a query that combines records from two or more tables.
2. A join will be performed whenever multiple tables appear in the FROM clause of the query.
3. The select list of the query can select any columns from any of these tables
4. If join condition is omitted or invalid then a cartesian product is formed. (all the rows in the first table are joined to all rows in second table).
5. If any 2 of these tables have a column name in common, then must qualify these columns throughout the query with table alias name to avoid ambiguity.
6. Most join queries contain at least one join condition either in the FROM clause or in the WHERE clause.


26 Friday
(146 - 219) Wk 21

## CARTESIAN JOIN

↳ Also known as cross-join.
↳ There is a join for each row of one table to every row of another table.

- Display the details of the customers who have taken the policy.

CUSTOMER

| CID | CNAME |
|---|---|
| 1 | Tom |
| 2 | John |
| 3 | Ram |

POLICYENROLLEMENT

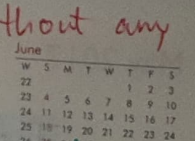
| EnrollmentID | CID | PID |
|---|---|---|
| 101 | 3 | MBP |
| 102 | 1 | PP |
| 103 | 2 | PP |

```
SELECT c.cid, cname, enrollmentID, p.cid,
pid FROM customer c, policyenrollment p
Order by c.cid, enrollmentid;
```

OUTPUT

| CID | CNAME | ENROLLMENTID | CID | PID |
|---|---|---|---|---|
| 1 | Tom | 101 | 3 | MBP |
| 1 | Tom | 102 | 1 | PP |
| 1 | Tom | 103 | 2 | PP |
| 2 | John | 101 | 3 | MBP |
| 2 | John | 102 | 1 | PP |
| 2 | John | 103 | 2 | PP |
| 3 | Ram | 101 | 3 | MBP |
| 3 | Ram | 102 | 1 | PP |
| 3 | Ram | 103 | 2 | PP |


Begining of the Holy Month of Ramadan (Malaysia / Maldives)




27 Saturday
(147 - 218) Wk 21

## CARTESIAN PRODUCT

When you join a table without any condition, it will lead to Cartesian Product.

All rows in the first table joined to all rows in the second table

To avoid a Cartesian product, always include a valid join condition.

```
SELECT c.cid, cname, enrollmentID, p.cid,
pid FROM customer c, policyenrollment p
WHERE c.cid = p.cid Order by c.cid;
```

OUTPUT

| CID | CNAME | EnrollmentID | CID | PID |
|---|---|---|---|---|
| 1 | Tom | 102 | 1 | PP |
| 2 | John | 103 | 2 | PP |
| 3 | Ram | 102 | 3 | MBP |

## JOIN SYNTAX

```
SELECT table1.column, table2.column
FROM table1, table2
WHERE table1.column1 = table2.column2;
```

```
SELECT table1.column, table2.column
FROM table1 JOIN table2 ON table1.column1 =
                           table2.column2;
```

↳ ANSI style join

JOIN & ON keywords are used in the FROM clause of the statement as per ANSI standard.

May 2017

# Type of Joins

28 Sunday

(148 - 217) Wk 22

- Equijoin
- Non Equijoin
- Outer join
- Self join

## EQUIJOIN

Equi joins are also called as simple joins or inner joins. When two tables are joined using = operator in the join condition it is called as an equi join.

Ques. What is the join query to display the details of the customers who have taken the Policy?

```
SELECT c.cid, cname, pid FROM customer c,
policyenrollment p WHERE c.cid = p.cid;

SELECT c.cid, cname, pid FROM customer
c JOIN policyenrollment p ON c.mid = p.mid;
```

Join condition

c is the alias name for the table customer & p for policyenrollment

- A non equijoin is a join cond" containing something other than an equality operator.

May 2017

## OUTER JOIN

Monday 29

(149 - 216) Wk 22

- Equi join or inner join returns rows only when there is at least one row from both the tables that matches the join cond".

- Outer join, return all the rows from one of the tables mentioned in the FROM clause even if the cond" is not matched.

Types of outer join:
- Left outer join
- Right outer join
- Full outer join

## LEFT OUTER JOIN

- The LEFT OUTER JOIN returns all the rows from the left table, with the matching rows of the right table.

- The result is NULL on the right side when there is no match.

Syntax:

```
SELECT table1.column, table2.
column FROM table1 LEFT
OUTER JOIN table2 ON
table1.column1 = table2.column2;
```

30 Tuesday
(150 - 215) Wk 22

Example: Display the member details along with the book they haven taken & the members who have not taken any book,

```
SELECT c.cid, cname, pid FROM customer c
LEFT OUTER JOIN policyenrollment p ON
c.cid = p.cid;
```

| CID | CNAME | PID |
|---|---|---|
| 1 | Tom | PP |
| 2 | John | PP |
| 3 | Ram | MBP |
| 4 | Tiny | |

## RIGHT OUTER JOIN

- The RIGHT OUTER JOIN returns all the rows from the right table, with the matching rows of the left table.
- The result is NULL in the left side when there is no match.

Syntax:-
```
SELECT table1.column, table2.column
FROM table1 RIGHT OUTER JOIN
table2 ON table1.column1 =
table2.column2;
```

Example: Display the policies that are taken & not taken by customers

```
SELECT p.pid, pname FROM policyenrollment
pp RIGHT OUTER JOIN policy
p ON pp.pid = p.pid
```

| PID | PNAME |
|---|---|
| MBP | Money Back Plan |
| PP | Personal Protect |
| MBP | Money Back Plan |

## FULL OUTER JOIN

- The full OUTER JOIN returns all rows from both the tables.
- The result is NULL on the left or right side when there is no match!

Syntax -
```
SELECT table1.column, table2.column
FROM table1 FULL OUTER JOIN table2.ON
table1.column1 = table2.column2;
```

Example: Display the policies that are taken & not taken. Customers who have taken policies & not taken any policies.

```
SELECT c.cid, cname, p.pid, pname FROM
policy p FULL OUTER JOIN policyenrollment
pp ON p.pid = pp.pid FULL OUTER JOIN
customer c ON pp.cid = c.cid;
```

Wednesday 31
(151 - 214) Wk 22

June 2017

16 Friday
(167 - 198) Wk 24

joins the table to itself. It is done by creating alias names for the same table

## SELF JOIN

- A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY.

- To join a table itself means that each row of the table is combined with itself & with every other row of the table.

- The self join can be viewed as a join of two copies of the same table.

Ex: Query to display the books which are of the same price.

```
SELECT b1.bid, b2.bname, b2.bprice
FROM book b1, book b2 WHERE b1.bid!=
b2.bid AND b1.bprice = b2.bprice;
```

## NATURAL JOIN

- NATURAL JOIN is a type of EQUI JOIN that compares the common columns of both the tables with each other.

- The associated tables can have one or more pairs of identically named columns.

- The columns must have the same data type.

Saturday 17
(168 - 197) Wk 24

```
SELECT cid, cname, pid FROM customer
NATURAL JOIN policyenrollment;
```

* Check whether common columns exist in both tables before doing a natural join.

## JOINS WITH THE USING CLAUSE

The USING clause specifies which columns to test for equality when two tables are joined. It can be used instead of an ON or where clause.

```
SELECT cid, cname, pid FROM customer
JOIN policyenrollment USING (cid);
```

## JOINING MULTIPLE TABLES

To join n number of tables, you need a minimum of n-1 join conditions.

for example, to join three tables, two join conditions are required.

- 
```
SELECT table1.column, table2.column, table3.
column FROM table1, table2, table3
WHERE table1.column1 = table2.column2 and
table1.column1 = table3.column1;
```

18 Sunday
(169 - 196) Wk 25

- SELECT table1.column, table2.column, table3.column FROM table1 JOIN table2 ON table1.column1 = table2.column2 JOIN table3 ON table1.column1 = table3.column1;

Example - Query to display policies taken by each customer. Display the policy name, customer name and due date.

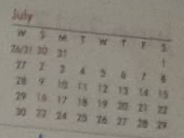
- You need to join policy, customer and policyenrollment table. The Syntax to join multiple table is:

```
SELECT cname, pname, duedate FROM customer
c JOIN policyenrollment p ON c.cid=p.cid
JOIN policy pp ON pp.pid=p.pid;
```

## SUBQUERY

Ques. How to display the customers who are elder to 'Tiny'?

First get what is Tiny's DOB using subquery. Then we can get the result set.



19 Monday
(170 - 195) Wk 25

- A subquery is a SELECT statement that is embedded in a clause of another SQL statement.

- They can be very useful to select rows from a table with a condition that depends on the data in the same or another table.

- The subquery can be placed in the following SQL clauses:

The WHERE clause

The HAVING clause

The FROM clause

SYNTAX for using SUBQUERY in WHERE clause

```
SELECT Column_List FROM table WHERE
    columnname operator
        (SELECT columnlist FROM table);
```

Example :-

A subquery is also called as inner query or inner select. While the statement containing the subquery is also called an outer query or outer select.

20 Tuesday
[171 - 194] Wk 25

Query to display the customer who are elder to 'Tiny'

Main query

```
SELECT CID, CNAME, DOB from Customer
WHERE dob > (SELECT dob FROM
customer WHERE cname = 'tiny');
```

Sub query. Enclose sub queries in parentheses.

| CID | CNAME | DOB |
|---|---|---|
| 2 | John | 27-JUL-85 |
| 3 | Ram | 31-DEC-84 |

Subquery output is 28-May-86.
Only output of the main query will be displayed.

## SUBQUERIES - TYPES

SUBQUERY

- SINGLE ROW
- MULTIROW

```
SELECT Column_List
FROM table
WHERE columnname operator
(SELECT columnlist
FROM table);
```

Returns only one result

```
SELECT Column_List
FROM table
WHERE columnname operator
(SELECT columnlist
FROM table);
```

Returns more than one result

Wednesday 21
[172 - 193] Wk 25

Sub Queries may return more than one column from the inner SELECT statement.

## OPERATORS IN SUBQUERY

- **Single Row Subquery**
  - ↳ Returns only one row
  - ↳ Operators used are:
    =, <, <=, >, >=, <>
- **Multiple Row Subquery**
  - ↳ Returns more than one row
  - ↳ Operators used are:

| Operator | Meaning |
|---|---|
| - in | - Equal to any member in the list |
| - all | - Compare to each value returned by the subquery |
| - any | - Compare to any value returned by the subquery |

Relational operators like: ≥, ≤, >, < can't be used when subquery returns more than one result.

June 2018

MULTIROW SUBQUERY - EXAMPLE

22 Thursday (173 - 192) Wk 25

Display the customers who paid the highest penalty

SELECT cid, cname, address from customer where cid = (select cid from policyenrollment where penalty = (Select max(penalty) from policyenrollment));

1, 2

200

The sub query returns more than one value, but = operator checks only one value.

Use 'in' operator, instead of =

Select cid, cname, address from customer where cid IN (select cid from policyenrollment where penalty = (Select max(penalty) from policyenrollment));

MULTI ROW OPERATORS - IN, ANY AND ALL

Combinations of ANY

- <ANY means - less than any one of the available value.
- >ANY means - more than any one of the available values.
- =ANY is equivalent to IN.

Combination of ALL

Friday 23 (174 - 191) Wk 25

- <ALL means less than all available values.
- >ALL means more than all available values.

The NOT operator can be used with IN, ANY, and ALL operators.

SAMPLE RECORDS

| Enrollment ID | Cid | Pid | Due Date | Paid Date | Amount | Penalty |
|---|---|---|---|---|---|---|
| 101 | 3 | MBP | 12-Dec-2017 | 11-Dec-2017 | 2000 | 0 |
| 102 | 1 | PP | 15-Mar-2018 | 23-Mar-2018 | 3000 | 200 |
| 103 | 2 | MBP | 22-Apr-2018 | 27-Apr-2018 | 4000 | 120 |
| 104 | 2 | PP | 15-Feb-2018 | 22-Feb-2018 | 4000 | 200 |
| 105 | 4 | MBP | 23-Apr-2018 | 25-Apr-2018 | 2500 | 70 |
| 106 | 3 | PP | 22-Mar-2018 | 28-Mar-2018 | 2400 | 150 |

| Pid | PName | PPeriodInYears | MinAmountPerMonth |
|---|---|---|---|
| MBP | Money Back Plan | 20 | 1000 |
| PP | Personal Protect | 15 | 1500 |

24 Saturday

(175 - 190) Wk 25

## ANY OPERATOR - EXAMPLE

- Display the customers who have a greater penalty than the penalty paid by customer 4 or 3 (cid).

```
SELECT cid, pid, penalty FROM policyenrollment
WHERE penalty > ANY (SELECT penalty
from policyenrollment WHERE mid IN (4,3));
```

(70, 150)

| CID | PID | PENALTY |
|---|---|---|
| 1 | PP | 200 |
| 2 | MGP | 120 |
| 2 | PP | 200 |

## ALL OPERATOR - EXAMPLE

- Display the customers who have a greater penalty than the penalty paid by both customers whose ID is 4 and 3.

```
SELECT cid, pid, penalty FROM policyenrollment
WHERE penalty >= ALL (SELECT penalty
FROM policyenrollment WHERE mid IN (4,3));
```

(70, 150)

| CID | PID | PENALTY |
|---|---|---|
| 1 | PP | 200 |
| 2 | PP | 200 |

Count(*) will count all the rows.

Hari Raya Puasa / Eid Ul-Fitr (End of Ramadan) (Malaysia / Maldives / Singapore)

25 Sunday

(176 - 189) Wk 26

## SUBQUERY IN THE SELECT LIST

- Query to display the cid, cname and the number of policies they have taken.

```
SELECT cid, cname, (SELECT count(*)
FROM policyenrollment p WHERE p.cid=
c.cid) as cnt FROM customer c;
```

- Subquery used SELECT clause must to provided with an alias name.

## SUBQUERY AS A TABLE NAME

- When you insert a select statement into a FROM clause, it becomes a subquery.
- The subquery returns a temporary table in the database server's memory and then it is used by the outer query for further processing.
- Display the policies that have been taken the maximum number of times.

```
SELECT pid, count(*) maxcount FROM
policyenrollment GROUP BY pid HAVING COUNT(*)
= (SELECT MAX(cnt) maxcnt FROM (SELECT
pid, count(*) cnt FROM policyenrollment
GROUP BY pid));
```

June 2017

Ramazan / Id- ul-Fitr (End of Ramadan) (India / Kenya / Singapore / Sri Lanka)

26 Monday

(177 - 188) Wk 26

## Correlated Subquery

- Correlated subquery is a subquery that uses values from the outer query.

- The subquery is evaluated onces for each row processed by the outer query.

Query to display the policies of the same minimum amount to be paid per month.

```
SELECT p.pid, p.pname, p.MinAmount FROM
policy p WHERE p.Minamount = (SELECT
p1.minamount FROM policy p1 WHERE
p1.minamount = p.minamount AND
p1.pid != p.pid);
```

## SUBQUERY - INSERT

All the records in the policyenrollment need to be copied into the policyenrollment_history table.

```
INSERT INTO policyenrollment_history SELECT*
FROM policyenrollment;
```

↳ Ensure you have created the policyenrollment_history table before inserting

June

July

## SUBQUERY - CREATE

Tuesday 27

(178 - 187) Wk 26

Is there any way to copy the records without creating the policyenrollment_history table?

```
CREATE TABLE policyenrollment_history AS
SELECT * FROM policyenrollment;
```

## SUBQUERY - DELETE

What is the query to delete all the policyenrollment made by 'John'?

```
DELETE FROM policyenrollment WHERE cid =
(SELECT cid FROM customer WHERE
cname = 'John')
```