Salesforce Developer Soql Joins

salesforce-developer-soql

```
// Salesforce - Developer - SOQL - Semi-joins and Anti-joins:

A semi-join is a subquery on another object in an IN clause.  An anti-join is a
subquery on another object in a NOT IN clause.

SELECT Id, Name FROM Account
WHERE Id IN (SELECT AccountId FROM Opportunity WHERE StageName = 'Closed Lost')

Notice that the left operand, Id, of the IN clause is an ID field.  The subquery
returns a single field of the same type as the field to which it is compared.

SELECT Id FROM Tasks
WHERE WhoId IN (SELECT Id FROM Contact WHERE MailingCity = 'Twin Falls');

SELECT Id FROM Account
WHERE Id NOT IN (SELECT AccountId FROM Opportunity WHERE IsClosed = false)

SELECT Id FROM Opportunity
WHERE AccountId NOT IN (SELECT AccountId FROM Contact WHERE LeadSource = 'Web')

SELECT Id, Name FROM Account
WHERE Id IN (SELECT AccountId FROM Contact WHERE LastName LIKE '%Apple%')
  AND Id IN (SELECT AccountId FROM Opportunity WHERE isClosed = false);

We can use at most two subqueries in a single semi-join or anti-join query.
Multiple semi-joins and anti-joins queries are also subjected to existing
limits on subqueries per query.

We can create a semi-join or anti-join that evaluates a relationship query in
a SELECT clause:

SELECT Id, (SELECT Id FROM OpportunityLineItems)
FROM Opportunity
WHERE Id IN (SELECT OpportunityId FROM OpportunityLineItem
  WHERE totalPrice > 10000)

Because a great deal of processing work is required for semi-join and anti-join
queries, Salesforce impose the following restrictions:

1. We cannot have more than two IN clause or NOT IN clause per WHERE clause.

2. We cannot use the NOT operator as a conjunction with semi-joins and
   anti-joins.  Using it converts a semi-join to an anti-join, and the reverse.
   Instead of using the NOT operator, write the query in the appropriate
   semi-join or anti-join form.

3. The left operand must query a single ID (primary key) or reference (foreign
   key) field.  The selected field in a subquery can be a reference field.  For
   example:

   SELECT Id FROM Idea WHERE (Id IN (
     SELECT ParentId FROM Vote WHERE CreateDate > LAST_WEEK AND
       Parent.Type='Idea'
     )
   );

4. The left operand cannot use relationships.  For example, the following
   semi-join query is not valid due to the Account.Id relationship field:

   SELECT Id FROM Contact WHERE Account.Id IN (SELECT ... )

5. A subquery must query a field referencing the same object types as the main
   query.

6. There is no limit on the number of records matched in a subquery.  Standard
   SOQL query limits apply to the main query.

7. The selected column in a subquery must be a foreign key field, and cannot
   traverse relationships.  This means that we cannot use dot notation in a
   selected field of a subquery.

8. We cannot query on the same object in a subquery as in the main query.  We
   can write such self semi-join queries without using semi-joins or anti-joins.
   For example, the following self semi-join query is not valid:

   SELECT Id, Name FROM Account WHERE ID IN (
     SELECT ParentId FROM Account WHERE Name = 'myaccount'
   )

   However, it is simple to rewrite the query as:

   SELECT Id, Name FROM Account WHERE Parent.Name = 'myaccount'

9. We cannot nest a semi-join or anti-join statement inside another semi-join
   or anti-join statement.

10. We can use semi-joins and anti-joins in the main WHERE statement, but not
    in a subquery WHERE clause.  For example, the following query is valid:

    SELECT Id FROM Idea
    WHERE (Idea.Title LIKE 'Vacation%') AND
      (Idea.LastCommentDate > YESTERDAY) AND
      (Id IN (SELECT ParentId FROM Vote WHERE CreatedById = '...'
        AND Parent.Type='Idea'))

    But the following query is not valid because the nested query is an extra
    level deep:

    SELECT Id FROM Idea WHERE
      ((Idea.Title LIKE 'Vacation%') AND
      (CreatedDate > YESTERDAY) AND
      (Id IN (SELECT ParentId FROM Vote WHERE CreatedById = '...'
        AND Parent.Type='Idea')
      ) OR (Idea.Title LIKE 'ExcellentIdea%'))

11. We cannot use subqueries with OR

12. COUNT, FOR UPDATE, ORDER BY, and LIMIT are not supported in subqueries.

13. The following objects are currently not supported in subqueries:
    ActivityHistory, Attachments, Event, EventAttendee, Note, OpenActivity,
    Tags (AccountTag, ContactTag, and all other tags objects), Tasks
```