



Tutorial: Aggregate and Nested Queries

This tutorial uses the schema and data of the database created in Tutorial 1.

1. Aggregate Queries.

- (a) How many loans involve an owner and a borrower from the same department?

Solution:

```
1 SELECT COUNT(*)
2 FROM loan l, student s1, student s2
3 WHERE l.owner = s1.email
4       AND l.borrower = s2.email
5       AND s1.department = s2.department;
```

- (b) For each faculty, print the number of loans that involve an owner and a borrower from this faculty?

Solution:

```
1 SELECT d1.faculty, COUNT(*)
2 FROM loan l, student s1, student s2, department d1, department d2
3 WHERE l.owner = s1.email
4       AND l.borrower = s2.email
5       AND s1.department = d1.department
6       AND s2.department = d2.department
7       AND d1.faculty = d2.faculty
8 GROUP by d1.faculty;
```

- (c) What are the average and the standard deviation of the duration of a loan?

Solution:

```
1 SELECT AVG((CASE
2   WHEN l.returned ISNULL
3   THEN '2010-12-31'
4   ELSE l.returned
5   END) - l.borrowed + 1),
6   STDDEV_POP ((CASE
7   WHEN l.returned ISNULL
8   THEN '2010-12-31'
9   ELSE l.returned
10  END) - l.borrowed + 1),
11 FROM loan l;
```

or

```
1 SELECT AVG(temp.duration),  
2 STDDEV_POP (temp.duration)  
3 FROM (SELECT ((CASE  
4     WHEN l.returned ISNULL  
5     THEN '2010-12-31'  
6     ELSE l.returned  
7     END) - l.borrowed + 1) AS duration FROM loan l) AS temp;
```

2. Nested Queries

- (a) Print the titles of the different books that have never been borrowed. Use a nested query.

Solution:

```
1 SELECT b.title  
2 FROM book b  
3 WHERE b.ISBN13 NOT IN (  
4     SELECT l.book  
5     FROM loan l);
```

or, equivalently,

```
1 SELECT b.title  
2 FROM book b  
3 WHERE b.ISBN13 <> ALL (  
4     SELECT l.book  
5     FROM loan l);
```

Always use one of the quantifiers ALL or ANY in front of subqueries wherever possible even though some systems may be lenient with this requirement.

Note that there could be several times the same title (since there could be different books with the same title) but not the same book. There is no need to use DISTINCT since the query asks for the different books but not for the different titles.

- (b) Print the name of the different students who own a copy of a book that they have never lent to anybody.

Solution:

```
1 SELECT s.name  
2 FROM student s  
3 WHERE s.email IN (  
4     SELECT c.owner  
5     FROM copy c  
6     WHERE NOT EXISTS (  
7         SELECT *  
8         FROM loan l  
9         WHERE l.owner = c.owner  
10            AND l.book = c.book  
11            AND l.copy = c.copy));
```

or, equivalently,

```
1 SELECT s.name  
2 FROM student s  
3 WHERE s.email = ANY (  
4     SELECT c.owner  
5     FROM copy c  
6     WHERE NOT EXISTS (SELECT *  
7         FROM loan l  
8         WHERE l.owner = c.owner  
9         AND l.book = c.book  
10        AND l.copy = c.copy));
```

The query can also be written as follows but the highlighted tuple construction does not always work on other systems than PostgreSQL.

```
1 SELECT s.name
2 FROM student s
3 WHERE s.email IN (
4     SELECT c.owner
5     FROM copy c
6     WHERE (c.owner, c.book, c.copy) NOT IN
7         (SELECT l.owner, l.book, l.copy
8          FROM loan l));
```

The following query would print several time the name of those students who own several copies that have never been borrowed. We would not be able to differentiate the repeated names of students who own several copies that have never been borrowed from the repeated names of different students with the same name.

```
1 SELECT s.name
2 FROM student s, copy c
3 WHERE s.email = c.owner
4 AND NOT EXISTS (SELECT *
5                  FROM loan l
6                   WHERE l.owner = c.owner
7                        AND l.book = c.book
8                        AND l.copy = c.copy);
```

- (c) For each department, print the names of the students who lent the most.

Solution:

```
1 SELECT s.department, s.name, count(*)
2 FROM student s, loan l
3 WHERE l.owner = s.email
4 GROUP BY s.department, s.email, s.name
5 HAVING count(*) >= ALL
6     (SELECT count(*)
7      FROM student s1, loan l1
8       WHERE l1.owner = s1.email
9             AND s.department = s1.department
10      GROUP BY s1.email);
```

Notice that there are three such students in the department of geography (that is why one should almost never use TOP N queries).

If we create a new department called Undecidable Computations with some students who never borrowed any book, what would happen? If there were students in the department of Undecidable Computations, should we print all of them or none of them? They would all have borrowed zero books, which would be the maximum in the department... We should print them all (using OUTER JOIN, CASE and ISNULL to consider the cases of 0 loan). Are there students who never borrowed a book?

Note that we need to group by department in order to print the department although there is no ambiguity. Some systems, like PostgreSQL, relax this rule. It is recommended not to use this relaxation for the sake of portability.

- (d) Print the emails and the names of the different students who borrowed all the books authored by Adam Smith.

Solution:

```
1 SELECT s.email, s.name
2 FROM student s
3 WHERE NOT EXISTS (
4     SELECT *
5     FROM book b
6     WHERE authors = 'Adam Smith'
```

```
7      AND NOT EXISTS (  
8          SELECT *  
9          FROM loan l  
10         WHERE l.book = b.isbn13  
11               AND l.borrower = s.email));
```

References

- [1] W3schools online web tutorials. www.w3schools.com. Visited on 26 July 2021.
- [2] S. Bressan and B. Catania. *Introduction to Database Systems*. McGraw-Hill Education, 2006.
- [3] H. Garcia-Molina, J.D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Pearson international edition. Pearson Prentice Hall, 2009.
- [4] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 2002.