



Tutorial: Simple and Algebraic Queries

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

1. Simple Queries.

- (a) Print the different departments.

Solution:

```
1 SELECT d.department
2 FROM department d;
```

Notice that the query does not require `DISTINCT` to eliminate duplicates. Duplicates are guaranteed not to occur because `department` is the `PRIMARY KEY` of the table `department`

- (b) Print the different departments in which students are enrolled.

Solution:

There could be departments in which no student is enrolled. This is the case of the department of `Undecidable Computations`. We need to look into the `student` table.

```
1 SELECT DISTINCT s.department
2 FROM student s;
```

Notice that the query requires `DISTINCT` to eliminate duplicates since it is very likely that there is more than one student in most departments.

- (c) Let us check the integrity of the data. Print the emails of the students who borrowed or lent a copy of a book before they joined the university. There should not be any. Use a simple query.

Solution:

```
1 SELECT DISTINCT s.email
2 FROM loan l, student s
3 WHERE (s.email = l.borrower AND l.borrowed < s.year)
4 OR (s.email = l.owner AND l.borrowed < s.year);
```

- (d) For each copy that has been borrowed and returned, print the duration of the loan. Order the results in ascending order of the ISBN13 and descending order of duration.

Solution:

```
1 SELECT book, returned - borrowed + 1 AS duration
2 FROM loan
3 WHERE NOT (returned ISNULL)
4 ORDER BY book ASC, duration DESC;
```

ASC is the default, but it is strongly recommended to indicate it for clarity.

Notice that the duration can be null if the book has not been returned yet. For a complete answer, you need to calculate the duration until December 31, 2010 to include the books that have not been returned yet.

```
1 SELECT book,
2     ((CASE
3      WHEN returned ISNULL
4      THEN '2010-12-31'
5      ELSE returned
6      END) - borrowed + 1) AS duration
7 FROM loan
8 ORDER BY book ASC, duration ASC;
```

- (e) For each loan of a book published by Wiley that has not been returned, print the title of the book, the name and faculty of the owner and the name and faculty of the borrower. Use CROSS JOIN.

Solution:

We join primary keys and foreign keys to stitch tables together properly.

```
1 SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS borrowerfaculty
6 FROM loan l, book b, copy c,
7     student s1, student s2,
8     department d1, department d2
9 WHERE l.book=b.ISBN13
10 AND c.book = l.book
11 AND c.copy = l.copy
12 AND c.owner = l.owner
13 AND l.owner = s1.email
14 AND l.borrower = s2.email
15 AND s1.department = d1.department
16 AND s2.department = d2.department
17 AND b.publisher = 'Wiley'
18 AND l.returned ISNULL;
```

You can omit the table copy and the copy column since the existence of the corresponding rows and values is guaranteed by design and by the foreign and primary key constraints.

```
1 SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS borrowerfaculty
6 FROM loan l, book b,
7     student s1, student s2,
8     department d1, department d2
9 WHERE l.book=b.ISBN13
10 AND l.owner = s1.email
11 AND l.borrower = s2.email
12 AND s1.department = d1.department
13 AND s2.department = d2.department
14 AND b.publisher = 'Wiley'
15 AND l.returned ISNULL;
```

2. Algebraic Queries.

- (a) For each loan of a book published by Wiley that has not been returned, print the title of the book, the name and faculty of the owner and the name and faculty of the borrower. Use INNER JOIN.

Solution:

```
1 SELECT b.title,
2       s1.name AS ownername,
3       d1.faculty AS ownerFaculty,
4       s2.name AS borrowername,
5       d2.faculty AS borrowerfaculty
6 FROM loan l
7      INNER JOIN book b ON l.book=b.ISBN13
8      INNER JOIN copy c ON c.book = l.book
9      AND c.copy = l.copy
10     AND c.owner = l.owner
11      INNER JOIN student s1 ON l.owner = s1.email
12      INNER JOIN student s2 ON l.borrower = s2.email
13      INNER JOIN department d1 ON s1.department = d1.department
14      INNER JOIN department d2 ON s2.department = d2.department
15 WHERE b.publisher = 'Wiley'
16        AND l.returned ISNULL;
```

You can omit the table `copy` and the `copy` column since the existence of the corresponding rows and values is guaranteed by design and by the foreign and primary key constraints.

```
1 SELECT b.title,
2       s1.name AS ownername,
3       d1.faculty AS ownerFaculty,
4       s2.name AS borrowername,
5       d2.faculty AS borrowerfaculty
6 FROM loan l
7      INNER JOIN book b ON l.book=b.ISBN13
8      INNER JOIN student s1 ON l.owner = s1.email
9      INNER JOIN student s2 ON l.borrower = s2.email
10     INNER JOIN department d1 ON s1.department = d1.department
11     INNER JOIN department d2 ON s2.department = d2.department
12 WHERE b.publisher = 'Wiley'
13        AND l.returned ISNULL;
```

- (b) Print the emails of the different students who borrowed or lent a copy of a book on the day that they joined the university. Use an algebraic query.

Solution:

```
1 SELECT s.email
2 FROM loan l, student s
3 WHERE s.email = l.borrower AND l.borrowed = s.year
4 UNION
5 SELECT s.email
6 FROM loan l, student s
7 WHERE s.email = l.owner AND l.borrowed = s.year;
```

DISTINCT is not needed because UNION eliminates duplicates (so do INTERSECT, EXCEPT and MINUS). The corresponding simple query is generally preferable.

```
1 SELECT DISTINCT s.email
2 FROM loan l, student s
3 WHERE (s.email = l.borrower OR s.email = l.owner)
4 AND l.borrowed = s.year;
```

The simple query requires an explicit DISTINCT.

- (c) Print the emails of the different students who borrowed and lent a copy of a book on the day that they joined the university. Use an algebraic query.

Solution:

```
1 SELECT s.email
2 FROM loan l, student s
3 WHERE s.email = l.borrower AND l.borrowed = s.year
4 INTERSECT
5 SELECT s.email
6 FROM loan l, student s
7 WHERE s.email = l.owner AND l.borrowed = s.year;
```

Note that the corresponding simple query is more complicated. It needs two loan tables.

```
1 SELECT DISTINCT s.email
2 FROM loan l1, loan l2, student s
3 WHERE s.email = l1.borrower AND l1.borrowed = s.year
4 AND s.email = l2.owner AND l2.borrowed = s.year;
```

- (d) Print the emails of the students who borrowed but did not lend a copy of a book on the day that they joined the university. Use an algebraic query.

Solution:

```
1 SELECT s.email
2 FROM loan l, student s
3 WHERE s.email = l.borrower AND l.borrowed = s.year
4 EXCEPT
5 SELECT s.email
6 FROM loan l, student s
7 WHERE s.email = l.owner AND l.borrowed = s.year;
```

There is no corresponding simple query. We would need to use nested or aggregate queries for this type of questions.

- (e) Print the ISBN13 of the books that have never been borrowed. Use an algebraic query.

Solution:

```
1 SELECT b.ISBN13
2 FROM book b
3 EXCEPT
4 SELECT l.book
5 FROM loan l
```

or, using an OUTER JOIN, which introduces NULL values,

```
1 SELECT b.ISBN13
2 FROM book b LEFT OUTER JOIN loan l ON b.isbn13 = l.book
3 WHERE l.book ISNULL;
```

There is no corresponding simple query. We would need to use nested or aggregate queries for this type of questions.