**C**ase study **11**

# Kings of France - Part 1

**Objective**. This study describes the French royal dynasty since Hughes Capet in 941. Its underlying goal is to study some properties and algorithms of widespread tree data structures. This first document of a series of two analyzes the dynasty of Kings of France, stores it in a database and extracts some simple information from it. The next study will be devoted to the derivation of more complex information.

**Keywords**. genealogy, tree, cyclic data structure, interval, ordering relation, temporal query, de Morgan law.

## 11.1 Introduction

Kings and queens generally have complicated lives, full of wars, plots, crimes, betrayal, family affairs, political alliances, unhappy marriages, mistresses, lovers, bastard and hidden children. Just to begin with.

The genealogy of Kings of France is one of the most turbulent, and therefore interesting lineages. Fortunately, the *Salic law*[1], edicted by King Clovis I in the 6th century, to which the Frankish government system was submitted, introduced important constraints in the royal inheritance principles that will substantially simplify the representation of French royal lineages in a database. One of the titles

1. http://en.wikipedia.org/wiki/Salic_law

(*rules*) of this law, the *agnatic succession*, organizes the succession of monarchs. In particular, it strictly excludes women and favours *father-son* or *brother-brother* succession, whenever possible.[2]

This means that, during nearly a millenary, the Kings of France are organized as a simple **tree**, a structure particularly straightforward to translate into data structures. It is clear that, in this organization, women would have made things much more complicated. Algorithmically speaking, I mean.

Figure 11.1 shows the members of the dynasty of French monarchs originated from **Hughes Capet** (941-996), the founder of the Capetian house. His youngest descendant, **Louis-Philippe I**, reigned until 1848.

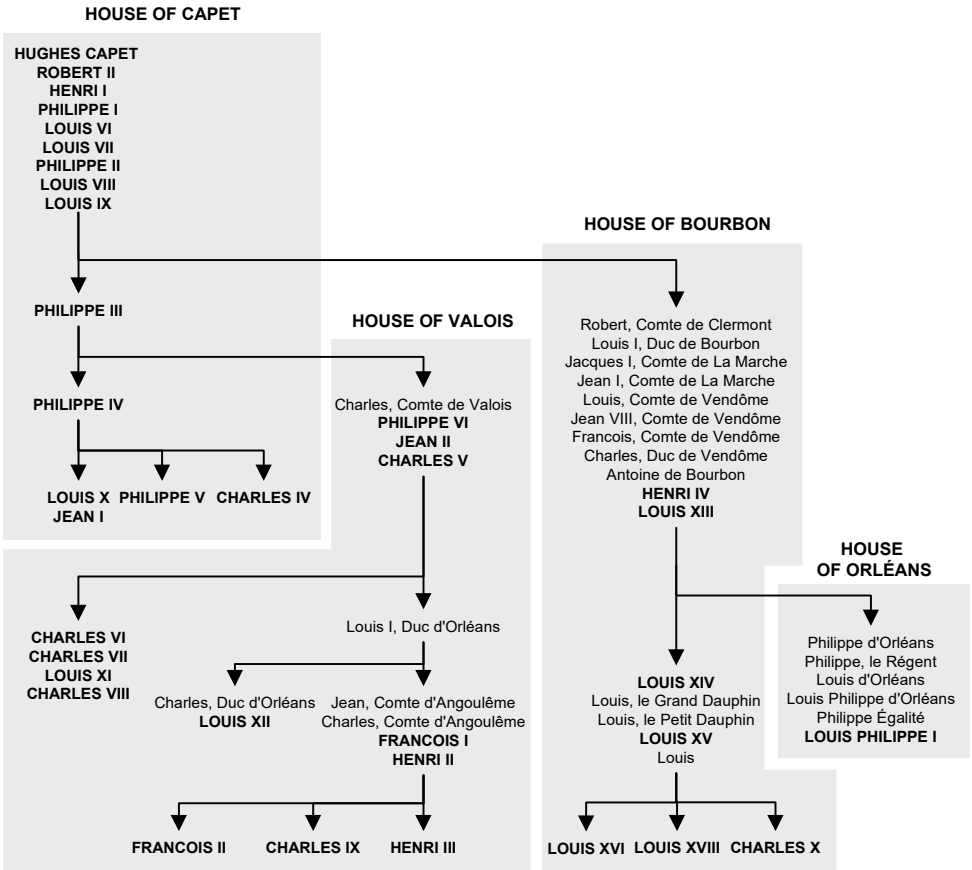


**Figure 11.1 -** Genealogy of Kings of France from 987 to 1848. King names are in bold upper case (*translated from http://en.wikipedia.org/wiki/French_monarchs_family_tree*)

2. The Salic agnatic succession rule still applies in several countries, notably in Belgium, where it was partly repealed in 1991 to allow female descendants of Albert II to ascend the throne (http://fr.wikipedia.org/wiki/Roi_des_Belges).

This main part of French monarchy spanned 9 centuries, thus providing a nice set of data to play with.

Note that this tree represents *kings descent*. It includes kings of course (in boldface and in upper case) but also descendants of kings who, though not having been kings themselves, have kings in their descent. OK, this seems to be a bit complex, so let us examine an example. **Charles**, Comte de Valois, was not king. However, he was the son of **Philippe III**, King of France and his son, **Philippe VI**, also was King of France. So, as a node in the link between two kings, **Charles** appears in the royal tree to ensure the continuity of this tree. This explains why kings of France seem to have so few sons![3]

## 11.2 Data structures

These data are structured as shown in Script 11.1. The data are stored in table **MEMBER** of database **KINGS.db**.

```
createOrReplaceDB KINGS.db;
create table MEMBER(
   PiD            char(6) not null,
   Name           char(20) not null,
   House          char(12) not null,
   Father         char(6),
   Title          char(20) not null,
   DateBirth      integer not null,
   DateDeath      integer not null,
   DateReignBegin integer,
   DateReignEnd   integer,
   primary key (PiD),
   foreign key (Father) references MEMBER);
commitDB;
closeDB;
```

**Script 11.1 -** Schema of the KINGS database

For each member, we record the following information:

- **PiD**, his unique code,
- **Name**, his usual name,
- **House**, the name of the house (family) he belongs to,

3. The site http://www.capedia.fr/ records the descendants of Hughes Capet, the number of which is estimated to 550,000. Most of them were not kings. Fortunately.

- **Father**, the code of his father (optional),
- **Title**, his main title; special value `'Roi de France'` (*King of France*) indicates that he reigned during a part of his life,
- **DateBirth**, (estimated) year of birth,
- **DateDeath**, (estimated) year of death,
- **DateReignBegin**, first year of his reign (optional),
- **DateReignEnd**, last year of his reign (optional).

This table is self-referencing: it includes a foreign key (Father) referencing itself, making it a *cyclic structure*. Since the father of Hughes Capet, the root of the royal tree, is not recorded, the column of this foreign key is nullable.

The loading of sample data is shown in Script 11.2. Excerpts from the database (column names have been shortened to save space) are shown below. Document Table-MEMBER.pdf, in the script directory shows the full contents of this table.

| PiD | Name | House | Fath | Title | Birth | Death | RBegin | REnd |
|---|---|---|---|---|---|---|---|---|
| C00 | HUGHES CAPET | Capet | -- | Roi de France | 939 | 996 | 987 | 996 |
| C01 | ROBERT II | Capet | C00 | Roi de France | 972 | 1031 | 996 | 1031 |
| C02 | HENRI I | Capet | C01 | Roi de France | 1008 | 1060 | 1031 | 1060 |
| C03 | PHILIPPE I | Capet | C02 | Roi de France | 1052 | 1108 | 1060 | 1108 |
| C04 | LOUIS VI | Capet | C03 | Roi de France | 1077 | 1137 | 1108 | 1137 |
| C05 | LOUIS VII | Capet | C04 | Roi de France | 1120 | 1180 | 1137 | 1180 |
| C06 | PHILIPPE II | Capet | C05 | Roi de France | 1165 | 1223 | 1180 | 1223 |
| C07 | LOUIS VIII | Capet | C06 | Roi de France | 1187 | 1226 | 1223 | 1226 |
| C08 | LOUIS IX | Capet | C07 | Roi de France | 1215 | 1270 | 1226 | 1270 |
| C09 | PHILIPPE III | Capet | C08 | Roi de France | 1245 | 1285 | 1270 | 1285 |
| C10 | PHILIPPE IV | Capet | C09 | Roi de France | 1268 | 1314 | 1285 | 1314 |
| C11 | LOUIS X | Capet | C10 | Roi de France | 1289 | 1316 | 1314 | 1316 |
| C12 | PHILIPPE V | Capet | C10 | Roi de France | 1291 | 1322 | 1316 | 1322 |
| C13 | CHARLES IV | Capet | C10 | Roi de France | 1295 | 1328 | 1322 | 1328 |
| C14 | JEAN I | Capet | C11 | Roi de France | 1316 | 1316 | 1316 | 1316 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| V03 | CHARLES V | Valois | V02 | Roi de France | 1337 | 1380 | 1364 | 1380 |
| V04 | CHARLES VI | Valois | V03 | Roi de France | 1368 | 1422 | 1380 | 1422 |
| V05 | CHARLES VII | Valois | V04 | Roi de France | 1403 | 1461 | 1422 | 1461 |
| V06 | LOUIS XI | Valois | V05 | Roi de France | 1423 | 1483 | 1461 | 1483 |
| V07 | CHARLES VIII | Valois | V06 | Roi de France | 1470 | 1498 | 1483 | 1498 |
| V08 | Louis I | Valois | V03 | Duc d'Orléans | 1372 | 1407 | -- | -- |
| V09 | Charles | Valois | V08 | Duc d'Orléans | 1394 | 1465 | -- | -- |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| V13 | FRANCOIS I | Valois | V12 | Roi de France | 1494 | 1547 | 1515 | 1547 |
| V14 | HENRI II | Valois | V13 | Roi de France | 1519 | 1559 | 1547 | 1559 |
| V15 | FRANCOIS II | Valois | V14 | Roi de France | 1544 | 1560 | 1559 | 1560 |
| V16 | CHARLES IX | Valois | V14 | Roi de France | 1550 | 1574 | 1560 | 1574 |
| V17 | HENRI III | Valois | V14 | Roi de France | 1551 | 1589 | 1574 | 1589 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

**Figure 11.2 -** Some rows of the MEMBER table

Since we will do a lot of work on kings, it can be useful to denote them through view KING (Script 11.3).

The data have been collected from various sources, including,

- http://en.wikipedia.org/wiki/French_monarchs_family_tree
- http://en.wikipedia.org/wiki/Members_of_the_French_Royal_Families
- http://en.wikipedia.org/wiki/List_of_French_monarchs

```
createOrReplaceDB KINGS.db;
insert into MEMBER values('C00','HUGHES CAPET','Capet',
       null,'Roi de France',939,996,987,996);
insert into MEMBER values('C01','ROBERT II','Capet',
       'C00','Roi de France',972,1031,996,1031);
insert into MEMBER values('C02','HENRI I','Capet',
       'C01','Roi de France',1008,1060,1031,1060);
...
insert into MEMBER values('O04','Philippe Égalité','Orléans',
       'O03','Duc d''Orléans',1747,1793,null,null);
insert into MEMBER values('O05','LOUIS PHILIPPE I','Orléans',
       'O04','Roi de France',1773,1850,1830,1848);
commitDB;
closeDB;
```

**Script 11.2 -** Loading data about Kings of France

```
create view KING(PiD,Name,House,Father,
       DateBirth,DateDeath,DateReignBegin,DateReignEnd) as
select PiD,Name,House,Father,
       DateBirth,DateDeath,DateReignBegin,DateReignEnd
from   MEMBER
where  Title = 'Roi de France';
```

**Script 11.3 -** A view showing kings only

## 11.3 Warming up: some easy queries

Let us begin with a query *counting the kings provided by each royal house* (Script 11.4). The result set (Figure 11.3) tells us that the *Capetian house* gave the largest number of kings and that the *house of Orléans* provided only one.

| House | Kings |
|---|---|
| Capet | 15 |
| Valois | 13 |
| Bourbon | 7 |
| Orléans | 1 |

**Figure 11.3 -** Number of kings in each royal house

```
select House,count(*) as Kings from KING
group by House
order by Kings desc;
```

**Script 11.4 -** How many kings in each house?

The next query returns the average age of kings when they ascend the throne (Script 11.5). Note the use of the **round** function to avoid meaningless decimals.

```
select round(avg(DateReignBegin - DateBirth),1)
       as  'Beginning reign age'
from   KING;
```

**Script 11.5 -** How old (on the average) were kings when ascending the throne?

```
+----------------------+
| Beginning reign age |
+----------------------+
| 24.9                 |
+----------------------+
```

**Figure 11.4 -** Average age of kings when ascending the throne

The next query extracts the minimum, average and maximum age of death of kings as well as their minimum, average and maximum reign length (Script 11.6). The life span of a member is given by expression `DateDeath - DateBirth` and his reign length by `DateReignEnd - DateReignBegin`. Since these rather long expressions are used several times in the query, we suggest to replace them by shorter names `life` and `reign`, defined by the first two **set** statements.

```
set life = DateDeath - DateBirth;
set reign = DateReignEnd - DateReignBegin;
select min($life$)           as MinLife,
       round(avg($life$),1)  as AvgLife,
       max($life$)           as MaxLife,
       min($reign$)          as MinReign,
       round(avg($reign$),1) as AvgReign,
       max($reign$)          as MaxReign
from   KING
where  DateBirth < DateDeath;
```

**Script 11.6 -** How long do kings live and reign?

Condition `DateBirth < DateDeath` may appear surprising. Actually, the royal lineage includes a singularity, King *Jean I*. This misfortunate king was born on November 16th, 1316, and died ... five days later. During his short life, he was king of France, so that `DateDeath = DateBirth = DateReignEnd = DateReignBegin`. We decide to discard him from the statistics. We get the results of Figure 11.5, showing that king is not always a safe job.

| MinLife | AvgLife | MaxLife | MinReign | AvgReign | MaxReign |
|---|---|---|---|---|---|
| 16 | 49.9 | 79 | 1 | 24.0 | 72 |

**Figure 11.5 -** Statistics on the life of kings

## 11.4 Rebuilding royal families

Through this series of queries, we will collect information on the local family structures of the members: just fathers and sons. The first query associates with each father, the list of his sons (Script 11.7). Expression `group_concat(PiD,',')` creates the comma-separated list of **PiD** values for each group.

```
select Father,group_concat(PiD,',') as Sons
from   MEMBER
group by Father;
```

**Script 11.7 -** Associating his sons with each father

As the root of the royal tree, Hughes Capet (C00) has no father but nevertheless appears as a son (of nobody) in the result (Figure 11.6).

| Father | Sons |
|---|---|
| -- | C00 |
| B00 | B01 |
| ... | ... |
| B10 | B11,O00 |
| B11 | B12 |
| ... | ... |
| B15 | B16,B17,B18 |
| ... | ... |
| C08 | C09,B00 |
| C09 | C10,V00 |
| C10 | C11,C12,C13 |
| C11 | C14 |
| O00 | O01 |

```
| ...      | ...          |
| V00      | V01          |
| ...      | ...          |
| V03      | V04,V08      |
| ...      | ...          |
| V08      | V09,V11      |
| ...      | ...          |
| V14      | V15,V16,V17  |
+--------+-------------+
```

**Figure 11.6 -** Fathers and sons

A variant of this query is shown in Script 11.8. It examines each member (not only fathers) and associates with each of them the list if his sons, which may be empty. This query does not make use of row grouping (`group by`) in the main query but builds the list of son's PiD through a subquery. Not surprisingly, Hughes Capet does not appear as a son in the result set.

```
select PiD, (select group_concat(PiD,',')
             from   MEMBER
             where  Father = M.PiD) as Sons
from   MEMBER M;
```

**Script 11.8 -** Associating his sons with each member

Showing member id's only, these results are not particularly expressive. It would be better to use names instead. However several members appear to have the same name, as shown by the following checking query:

```
select Name, count(*) as N
from   MEMBER group by Name having N > 1;
```

To solve the ambiguity, we write the *name* and *birth date* of both the father and their sons. For instance, we display `'HUGH CAPET (939)'` instead of `'C00'`. So, we replace

```
PiD
```

by[4]

```
Name||' ('||cast(DateBirth as char)||')'
```

Since this fairly complicated expression appears twice in the query, we define variable **`birth`** with value `' ('||cast(DateBirth as char)||')'`. This makes the query simpler to write and to read (Script 11.9, as an extension of Script 11.7).

4. Note that here, `Father` is the alias of the first item in the select list and not the name of the eponym source column. Not all DBMS allow this pattern. If needed, the column alias can be quoted ("Father").

```
set birth = ' ('||cast(DateBirth as char)||')';
select (select Name||$birth$ from MEMBER
        where PiD = M.Father) as Father,
        group_concat(Name||$birth$,',') as Sons
from   MEMBER M
group by Father;
```

**Script 11.9 -** Showing the *father-son* family composition

The result looks much better (Figure 11.7).

| Father | Sons |
|---|---|
| -- | HUGHES CAPET (939) |
| Antoine (1518) | HENRI IV (1553) |
| CHARLES V (1337) | CHARLES VI (1368),Louis I (1372) |
| CHARLES VI (1368) | CHARLES VII (1403) |
| CHARLES VII (1403) | LOUIS XI (1423) |
| Charles (1270) | PHILIPPE VI (1293) |
| Charles (1394) | LOUIS XII (1462) |
| Charles (1459) | FRANCOIS I (1494) |
| Charles (1489) | Antoine (1518) |
| FRANCOIS I (1494) | HENRI II (1519) |
| François (1470) | Charles (1489) |
| HENRI I (1008) | PHILIPPE I (1052) |
| HENRI II (1519) | FRANCOIS II (1544),CHARLES IX (1550),HENRI III (1551) |
| ... | ... |

**Figure 11.7 -** Fathers and sons - A more readable presentation

If we are interested by brotherhood only, that is, by groups of at least two brothers, we could try the query of Script 11.10, which returns the list of Figure 11.8.

| Brothers |
|---|
| LOUIS XIV (1638),Philippe I (1640) |
| LOUIS XVI (1754),LOUIS XVIII (1755),CHARLES X (1757) |
| PHILIPPE III (1245),Robert (1256) |
| PHILIPPE IV (1268),Charles (1270) |
| LOUIS X (1289),PHILIPPE V (1291),CHARLES IV (1295) |
| CHARLES VI (1368),Louis I (1372) |
| Charles (1394),Jean (1399) |
| FRANCOIS II (1544),CHARLES IX (1550),HENRI III (1551) |

**Figure 11.8 -** Brotherhood

```
select group_concat(Name||$birth$,',') as Brothers
from   MEMBER
group by Father
having count(*) > 1;
```

**Script 11.10 -** Brotherhood

The last query of this series is a bit more *sensitive*: it asks the age of members when they had their first son[5] (Script 11.11). The query performs an auto-join that associates two members, the second one (alias S) being the son of the former (alias F).

```
select F.Name||' ('||cast(F.DateBirth as char)||')' as Father,
       count(*) as Sons,
       min(S.DateBirth - F.DateBirth) as AgeFirstSon
from   MEMBER F, MEMBER S
where  F.PiD = S.Father
group by F.PiD,F.Name
order by AgeFirstSon;
```

**Script 11.11 -** How old were kings when they had their first son?

The result is quite interesting (Figure 11.9)!

| **Father** | **Sons** | **AgeFirstSon** |
|---|---|---|
| JEAN II (1319) | 1 | 18 |
| François (1470) | 1 | 19 |
| LOUIS XV (1710) | 1 | 19 |
| CHARLES VII (1403) | 1 | 20 |
| Louis (1661) | 1 | 21 |
| PHILIPPE IV (1268) | 3 | 21 |
| ... | .. | .. |
| LOUIS XI (1423) | 1 | 47 |
| HENRI IV (1553) | 1 | 48 |
| Louis (1376) | 1 | 50 |
| Jean (1399) | 1 | 60 |
| Charles (1394) | 1 | 68 |

**Figure 11.9 -** Age of members when they had their first son

5. As known by the database, which records the members of the royal tree only.

## 11.5 King succession

The database records the *father-son* tree of French members, but does not inform on the succession of kings as they ascend the throne of France. Or does it?

We know for sure that no two kings may reign at the same time. In addition, any gap between two successive reign periods must be avoided, as far as possible. More precisely,

1. Reign periods (DateReignBegin,DateReignEnd) may not overlap.
2. For each reign period, but the last one, there should be another reign period that begins when the former finishes. However, the French revolution forced a break in the succession, so that the rule must be adapted: *for each reign period, but the last one, there should be another reign period that does not begin before the former finishes.*

There are several ways to extract the king succession from table MEMBER. The simplest consists in displaying the kings in ascending order of DateReignBegin:

```
select * from KING
order by DateReignBegin;
```

This works fine except for King *Jean I*, who has the same reign beginning date as his successor, King *Philippe V.* Who succeeds whom is undefined when we only know the year of these dates. Hence the final version of Script 11.12 which discards abnormal periods.

```
select * from KING
where  DateReignBegin < DateReignEnd
order by DateReignBegin;
```

**Script 11.12 -** The king succession

## 11.6 Seeking missing kings

King history may include **breaks**, that is, periods during which no king has been reigning, for some reason (a revolution is a good one). Surprisingly, identifying these breaks is not an easy task. Indeed, we must extract from the database *something that does not exist*!

Following Figure 11.10, a royal period is missing (we will call it **K2**) if there are two periods **K1** and **K3** such that:

- **K1** and **K3** are royal reigns (we use view KING),

- **K1** finishes before **K3** starts (`K1.DateReignEnd < K3.DateReignBegin`), so, there is a gap between them,
- there is no period **K2** between **K1** and **K3** (this gap is empty). Should **K2** exist, it would be a royal period, it would begin at the earliest when **K1** terminates and it would finish at the latest when **K3** begins. We must describe **K2** in this this way, then tell that such period do not exist.


**Figure 11.10 -** Identifying the absence of a period

Script 11.13 translates this definition. It displays king names in **K1** and **K3** surrounding the missing period, as shown in Figure 11.11.

```
+-----------+------+------+-------------+
| Last king | From | To   | Next king   |
+-----------+------+------+-------------+
| LOUIS XVI | 1792 | 1814 | LOUIS XVIII |
+-----------+------+------+-------------+
```

**Figure 11.11 -** A period without king has been detected

There is only one missing period, from **1792**, when **Louis XVI** was suspended before being executed, to **1814**, when **Louis XVIII**, called by the senate, accesses the throne for a period called *Restauration*.[6]

```
select K1.Name           as 'Last king',
       K1.DateReignEnd   as 'From',
       K3.DateReignBegin as 'To',
       K3.Name           as 'Next king'
from   KING K1, KING K3
where  K1.DateReignEnd < K3.DateReignBegin
and not exists(select 1 from KING K2
               where  K2.DateReignEnd <= K3.DateReignBegin
               and    K1.DateReignEnd <= K2.DateReignBegin);
```

**Script 11.13 -** Finding missing kings

6. This period exhibits an interesting property. Indeed, it has been interrupted for 100 days, during Napoleon's return to France in 1815. Though this cannot be represented in our database (where dates are measured in years), the fact that several non contiguous periods can be covered by the same king means that kings and royal periods no longer are synonyms. This may make some queries and algorithms of this chapter invalid. Good exercise for the reader!

## 11.7 Royal conflicts

When studying king succession, we have admitted that *reign periods may not overlap*. However, kings do not always comply with such a wise rule. It is not uncommon that, in troubled times, two kings claim their legitimacy at the same time on the same country. Let us check whether the King of France lineage is devoid of such conflicts.

Checking whether two periods overlap is not as intuitive as we could think. For this, we first examine a simpler problem, that is, determining whether *two periods are disjoint*. Two periods [b1,e1] and [b2,e2] are disjoint if one of them terminates before (or when) the other starts, that is, when either

$$e1 \leq b2$$

or

$$e2 \leq b1$$

So, on the contrary, *two periods overlap* when this condition is not satisfied, that is, when[7]

$$e1 > b2 \text{ and } e2 > b1$$

We translate this condition into Script 11.14, which seeks royal conflicts (overlapping royal periods).

The result is quite comforting. It shows that French kings can been wise:

```
+------------+-------------+
| First king | Second king |
+------------+-------------+
+------------+-------------+
```

```
select K1.Name as 'First king',K2.Name as 'Second king'
from   KING K1, KING K2
where  K1.PiD <> K2.PiD
and    K1.DateReignEnd > K2.DateReignBegin
and    K2.DateReignEnd > K1.DateReignBegin
order by K1.Name,K2.Name;
```

**Script 11.14 -** Identifying overlapping periods

By artificially changing some values of DateReignBegin, we can create overlapping periods. Just to check the validity of the query!

7. Simple application of a *de Morgan* law: **not** (C1 **or** C2) = (**not** C1) **and** (**not** C2).

The scripts of this study are available in directory **SQLfast/Scripts/Case-Studies/Case_Kings_of_France**. They can be run from main script **Kings-MAIN.sql** (see Part 2).