SQL Case Study

"I have a main data table that lists events. Each event has a series of fields including duration, reason code and status.

The database has a table of statuses and there is a common field between the main table and this allows me to show the status name rather than the code.

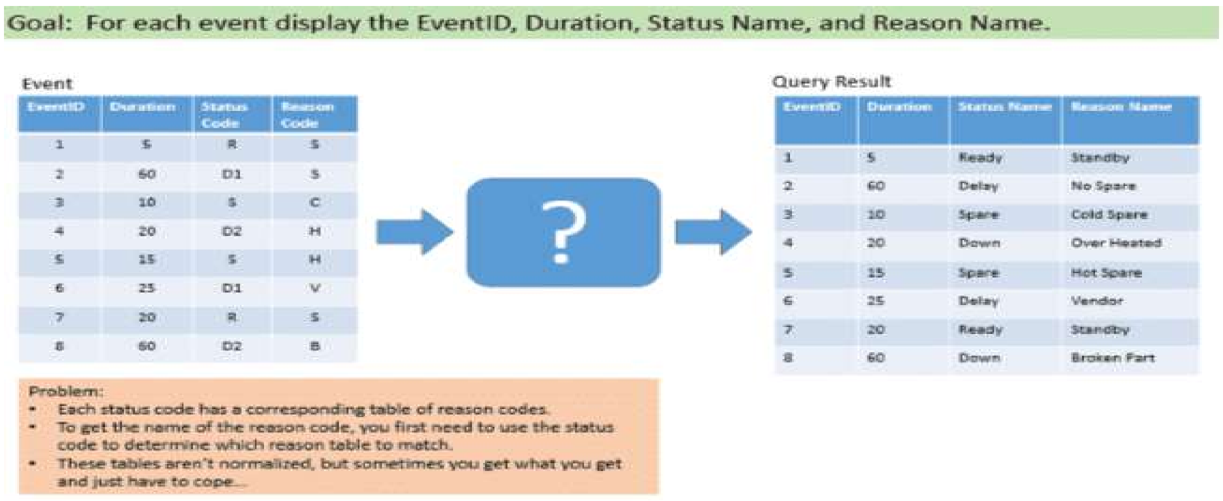
There are four statuses (ready, delay, spare and down) and there are a range of reasons associated with each status. It is possible for the same code to exist in two statuses (e.g. delay maintenance and down maintenance).

For some reason the source database has four tables of reasons (one for each status). Each of these has a field which links to the main reason code. What is missing is a status field in those tables.

What I'm looking for is something along the lines of ''when the main table record status is 'delay' then retrieve the name in the 'delay' table which corresponds to the main table delay code. When the status is 'down', do the same thing but look up the name in the 'down' table

So the goal is to create a result that uses descriptive values rather than codes.

Case Study Goal:



Goal: For each event display the EventID, Duration, Status Name, and Reason Name.

Event

EventID	Duration	Status Code	Reason Code
1	5	R	S
2	60	D1	S
3	10	S	C
4	20	D2	H
5	15	S	H
6	25	D1	V
7	20	R	S
8	60	D2	B

Query Result

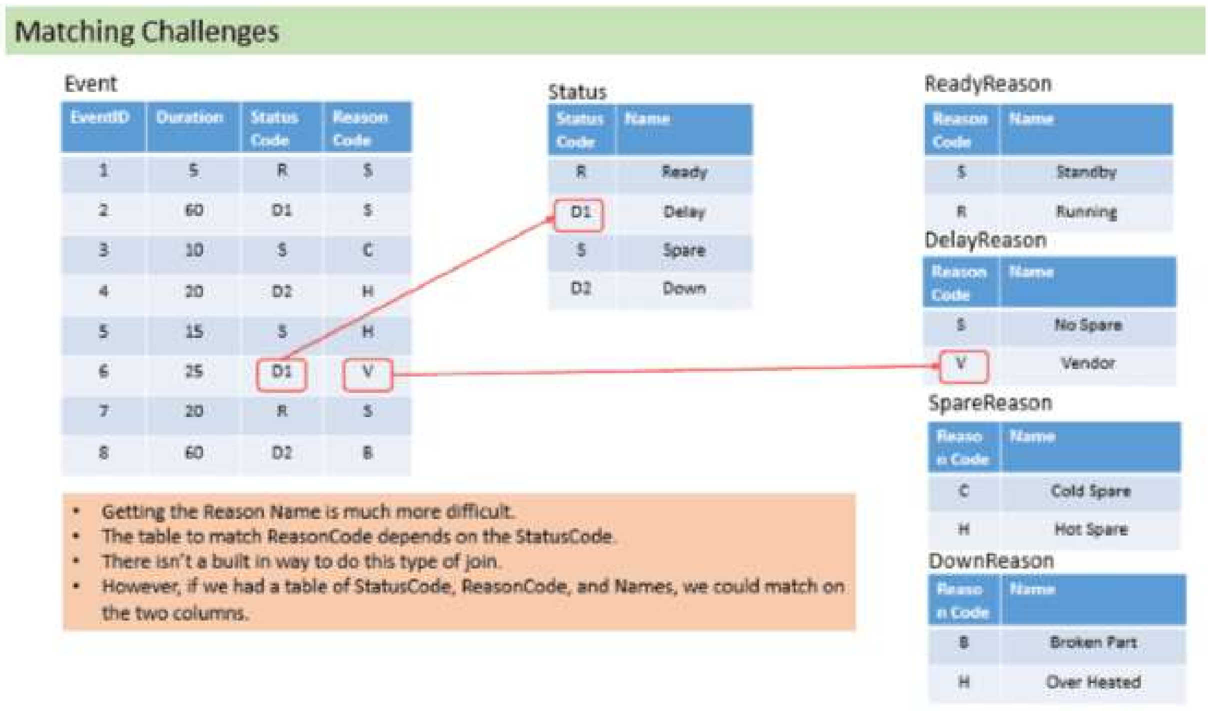
EventID	Duration	Status Name	Reason Name
1	5	Ready	Standby
2	60	Delay	No Spare
3	10	Spare	Cold Spare
4	20	Down	Over Heated
5	15	Spare	Hot Spare
6	25	Delay	Vendor
7	20	Ready	Standby
8	60	Down	Broken Part

Problem:
- Each status code has a corresponding table of reason codes.
- To get the name of the reason code, you first need to use the status code to determine which reason table to match.
- These tables aren't normalized, but sometimes you get what you get and just have to cope…

Case Study Goal

Here is an overall view of the various tables and the matching challenge, which is to “conditionally match” on one of the four “ready” tables depending on the value of Status:

Matching Challenge:



Matching Challenges

Event

EventID	Duration	Status Code	Reason Code
1	5	R	S
2	60	D1	S
3	10	S	C
4	20	D2	H
5	15	S	H
6	25	D1	V
7	20	R	S
8	60	D2	B

Status

Status Code	Name
R	Ready
D1	Delay
S	Spare
D2	Down

ReadyReason

Reason Code	Name
S	Standby
R	Running

DelayReason

Reason Code	Name
S	No Spare
V	Vendor

SpareReason

Reaso n Code	Name
C	Cold Spare
H	Hot Spare

DownReason

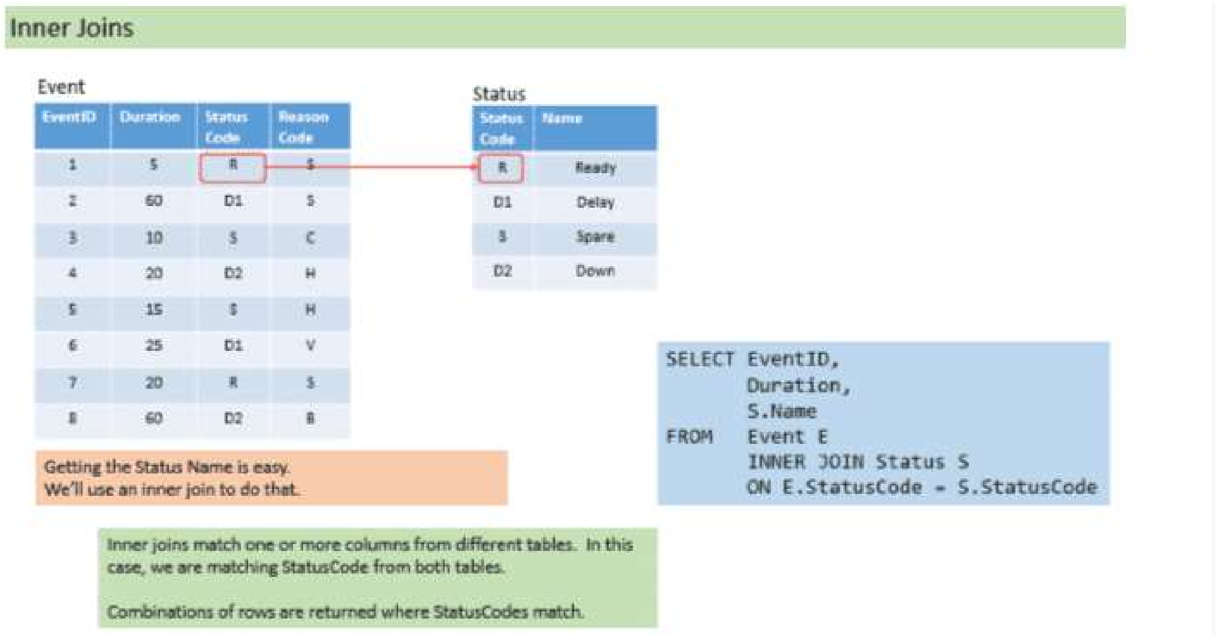
Reaso n Code	Name
B	Broken Part
H	Over Heated

- Getting the Reason Name is much more difficult.
- The table to match ReasonCode depends on the StatusCode.
- There isn’t a built in way to do this type of join.
- However, if we had a table of StatusCode, ReasonCode, and Names, we could match on the two columns.

Let’s go through each of these sub problems in order.

1. Using INNER JOIN to return Status Name:

Since we need to obtain the Status Name from the status table corresponding to the status code in events we can use an INNER JOIN.



Inner Joins

Event

EventID	Duration	Status Code	Reason Code
1	5	R	S
2	60	D1	S
3	10	S	C
4	20	D2	H
5	15	S	H
6	25	D1	V
7	20	R	S
8	60	D2	B

Status

Status Code	Name
R	Ready
D1	Delay
S	Spare
D2	Down

```
SELECT EventID,
       Duration,
       S.Name
FROM   Event E
       INNER JOIN Status S
       ON E.StatusCode = S.StatusCode
```

Getting the Status Name is easy.
We'll use an inner join to do that.

Inner joins match one or more columns from different tables. In this case, we are matching StatusCode from both tables.

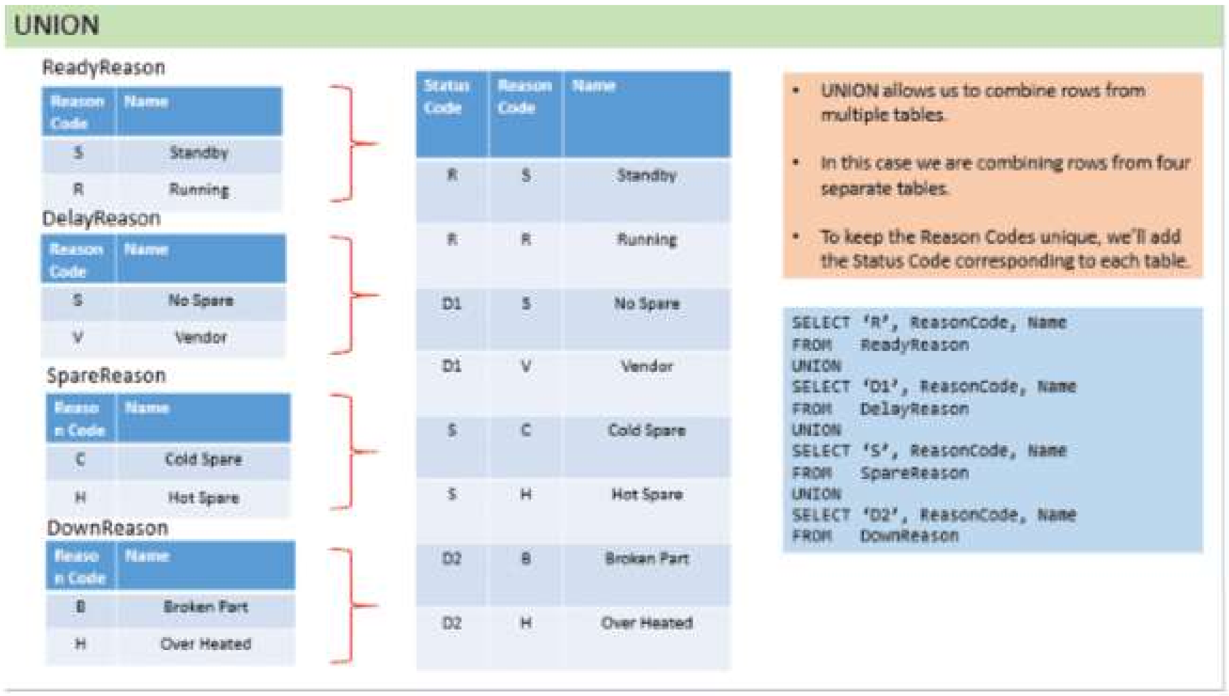
Combinations of rows are returned where StatusCodes match.

Here is the code used to create the unified results.

SELECT EventID, Duration, S.Name FROM Event E INNER JOIN Status S ON E.StatusCode = S.StatusCode

2. Utilizing UNION to Combine Reason Table Row:

UNION is called a set operator. The UNION operator is used to combine rows from several tables into a single result. Whereas a join is meant to combine columns from different tables into a single row the UNION operator is adding rows from each table.



UNION

ReadyReason

Reason Code	Name
S	Standby
R	Running

DelayReason

Reason Code	Name
S	No Spare
V	Vendor

SpareReason

Reaso n Code	Name
C	Cold Spare
H	Hot Spare

DownReason

Reaso n Code	Name
B	Broken Part
H	Over Heated

Status Code	Reason Code	Name
R	S	Standby
R	R	Running
D1	S	No Spare
D1	V	Vendor
S	C	Cold Spare
S	H	Hot Spare
D2	B	Broken Part
D2	H	Over Heated

- UNION allows us to combine rows from multiple tables.
- In this case we are combining rows from four separate tables.
- To keep the Reason Codes unique, we'll add the Status Code corresponding to each table.

```
SELECT 'R', ReasonCode, Name
FROM   ReadyReason
UNION
SELECT 'D1', ReasonCode, Name
FROM   DelayReason
UNION
SELECT 'S', ReasonCode, Name
FROM   SpareReason
UNION
SELECT 'D2', ReasonCode, Name
FROM   DownReason
```

Here is the code used to create the unified results.

SELECT 'R', ReasonCode, Name FROM   ReadyReason UNION SELECT 'D1', ReasonCode, Name FROM   DelayReason UNION SELECT 'S', ReasonCode, Name FROM   SpareReason UNION SELECT 'D2', ReasonCode, Name FROM   DownReason

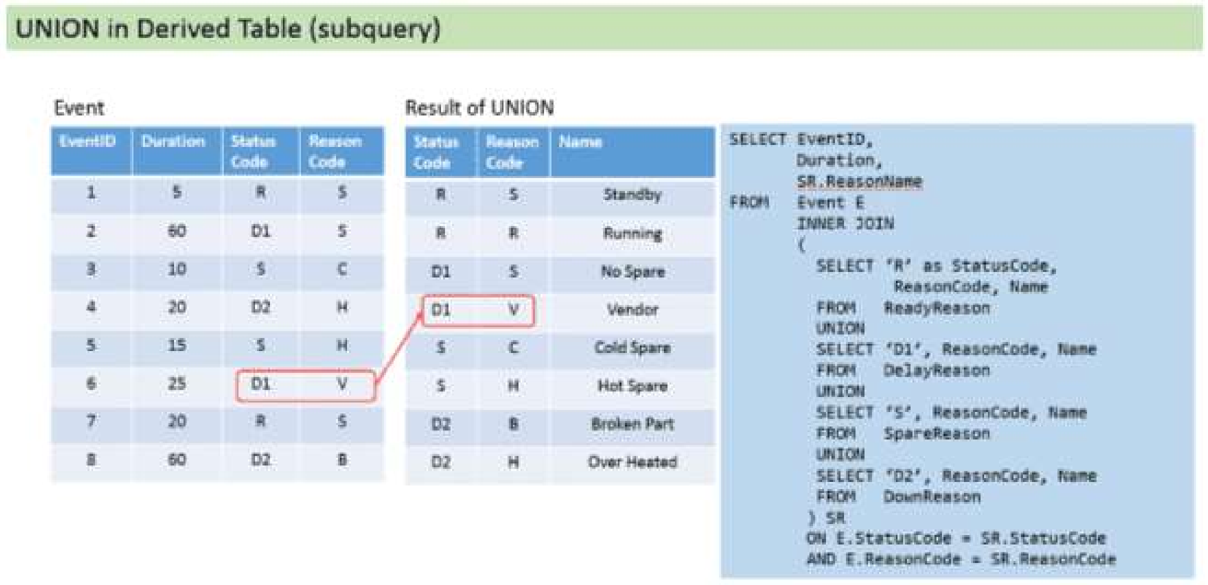
3. Using Subqueries to Include a Derived Table in Final Result:

Once the union is created we are now able to use it to match and pull in the reason names.

The matching becomes much easier. We no longer have to inspect the status code, then decide which one of the four tables to use before matching on reason code to get the name.

Instead we can now use a standard INNER JOIN to match both the status code and reason code to the result of the union.

Derived tables are enclosed in parenthesis, like sub queries, but they are also given a name.



UNION in Derived Table (subquery)

Event

EventID	Duration	Status Code	Reason Code
1	5	R	S
2	60	D1	S
3	10	S	C
4	20	D2	H
5	15	S	H
6	25	D1	V
7	20	R	S
8	60	D2	B

Result of UNION

Status Code	Reason Code	Name
R	S	Standby
R	R	Running
D1	S	No Spare
D1	V	Vendor
S	C	Cold Spare
S	H	Hot Spare
D2	B	Broken Part
D2	H	Over Heated

```
SELECT EventID,
       Duration,
       SR.ReasonName
FROM   Event E
       INNER JOIN
       (
         SELECT 'R' as StatusCode,
                ReasonCode, Name
         FROM   ReadyReason
         UNION
         SELECT 'D1', ReasonCode, Name
         FROM   DelayReason
         UNION
         SELECT 'S', ReasonCode, Name
         FROM   SpareReason
         UNION
         SELECT 'D2', ReasonCode, Name
         FROM   DownReason
        ) SR
        ON E.StatusCode = SR.StatusCode
        AND E.ReasonCode = SR.ReasonCode
```

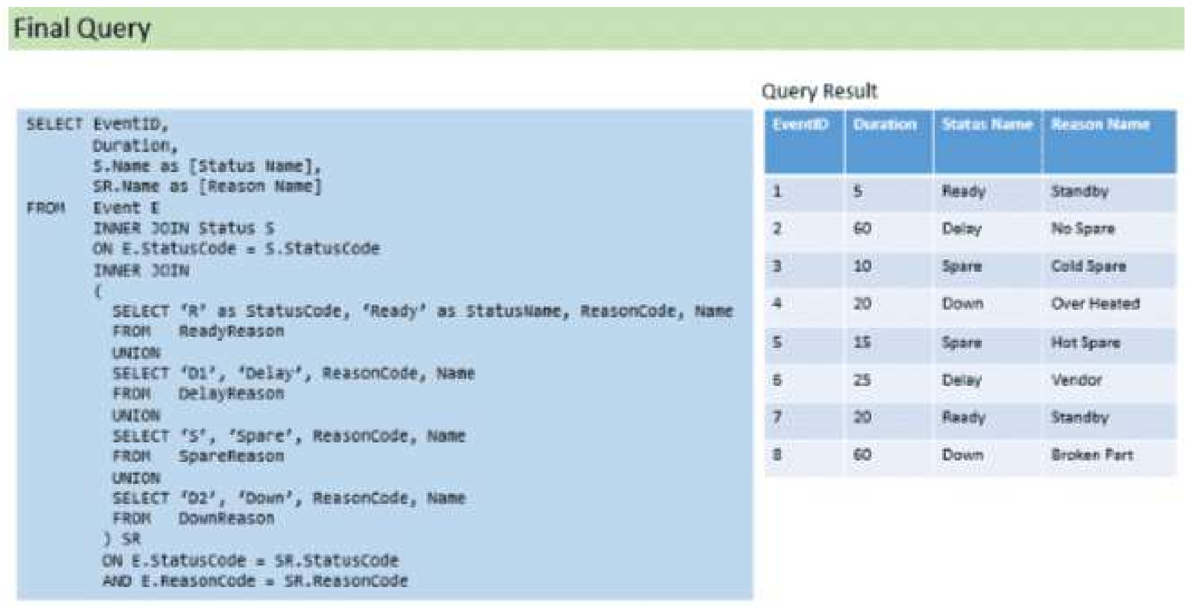
If you look closely at the SQL you see the UNION result is given the name SR.

In the sample below I've color coded the UNION green and it use in the INNER JOIN blue..

SELECT EventID, Duration, SR.ReasonName FROM Event E INNER JOIN (SELECT 'R' as StatusCode, ReasonCode, Name FROM ReadyReason UNION SELECT 'D1', ReasonCode, Name FROM DelayReason UNION SELECT 'S', ReasonCode, Name FROM SpareReason UNION SELECT 'D2', ReasonCode, Name FROM DownReason) SR ON E.StatusCode = SR.StatusCode AND E.ReasonCode = SR.ReasonCode

Final Query:

To create the final result we combine the three sub solutions together. From the section above, you can see that each means to do so is relatively simple. Sure, there is syntax to contend with, but I think overall the ideas are straightforward.



Final Query

```
SELECT EventID,
       Duration,
       S.Name as [Status Name],
       SR.Name as [Reason Name]
FROM   Event E
       INNER JOIN Status S
       ON E.StatusCode = S.StatusCode
       INNER JOIN
       (
         SELECT 'R' as StatusCode, 'Ready' as StatusName, ReasonCode, Name
         FROM   ReadyReason
         UNION
         SELECT 'D1', 'Delay', ReasonCode, Name
         FROM   DelayReason
         UNION
         SELECT 'S', 'Spare', ReasonCode, Name
         FROM   SpareReason
         UNION
         SELECT 'D2', 'Down', ReasonCode, Name
         FROM   DownReason
        ) SR
        ON E.StatusCode = SR.StatusCode
        AND E.ReasonCode = SR.ReasonCode
```

Query Result

EventID	Duration	Status Name	Reason Name
1	5	Ready	Standby
2	60	Delay	No Spare
3	10	Spare	Cold Spare
4	20	Down	Over Heated
5	15	Spare	Hot Spare
6	25	Delay	Vendor
7	20	Ready	Standby
8	60	Down	Broken Part

This concludes the case study.

Thank You!