ExecuteSQL Abstraction

Because SQL within FileMaker requires learning at least a subset of the SQL-92 standard a developer is moving beyond FileMaker's list of functions.

This page is currently a placeholder for the discussion of adopting a best practice regarding dealing with using SQL within FileMaker calculation.

Further discussion about this topic can be found on the Google Groups discussion about ExecuteSQL

Current proposed snippet which uses Substitution method (similar to printf)

```
Let ( [ ~sql = "
        SELECT t1.~field
        FROM ~table1 t1
        JOIN ~table2 t2
        ON t1.~field = t2.~field
        WHERE ~field LIKE '%~value%'
        AND ~field=?
        ORDER BY ~field";

        $sqlQuery = Substitute ( ~sql ;
                [ "~table1" ; SQLTableName ( Table1::fieldName ) ];
                [ "~table2" ; SQLTableName ( Table2::fieldName ) ];
                [ "~field" ; SQLFieldName ( Table1::fieldName ) ];
                [ "~value" ; Table::field ]
        );

        $sqlResult = ExecuteSQL ( $sqlQuery ; "" ; "" ;
            $value;
            $value[2];
            $value[$n]
        )
];
        //Substitute ( $sqlQuery ; "        " ; "" ) &¶& // sql preview
        If ( $sqlResult = "?" ;
                Let ( ~debug = False ; If ( ~debug ; SQLDebugResult ( $sqlResult ) ; False ) );
                $sqlResult
        )
)
```

In order to see any SQL errors, the above calculation must be used within FileMaker's Data Viewer. The SQLDebugResult custom function (below) is very simple and was a hidden "feature".

SQLDebugResult (custom function)

```
If ( sql = "?" ; "" ; sql )

//If passing in ExecuteSQL results in an error, return empty so the error will be returned within the Data
Viewer.
```