Ask Andy: User Level Grade (UserID ; Level)

#Purpose: Given a user ID and a level, return the grade (percentage correct on most recent answer) for that user at that level.
Parameters: UserID: The primary key of the user to get the grade for
Level: The level to check the grade for
Return: Let notation dictionary with one key, `~grade`, a percentage of questions most recently answered correctly
Version: 1.0.0 - Charles Ross - 19-01-25
Notes: There might be a single SQL query that will do this, but I haven't come up with it yet.

Allow User Abort [Off]

If [not script.AssignParams]
    Show Custom Dialog [Title: "Invalid Parameters"; Message: "Invalid parameters were passed to the " & Quote (Get (ScriptName)) & ". Please contact the developer."; Default Button: “Halt”, Commit: “Yes”]
    Halt Script
End If

Set Error Capture [On]
Freeze Window

Perform Script [“window: New Utility (Layout {; WindowID })”; Parameter: script.Param ("Layout" ; "z_CHCraw")]

Enter Find Mode [ ]
Set Field [CHC::_id_USR; $UserID]
Set Field [chc|QST::level; $Level]
Perform Find [ ]
Set Variable [$_answered_count; Value:Get (FoundCount)]
If [Get (FoundCount) = 0]
    Close Window [Current Window]
    Exit Script [Result: let.Set ("~grade" ; 0) &
        let.Set ("~answered_count" ; 0) &
        let.Set ("~correctly_answered_count" ; 0) &
        let.Set ("~incorrectly_answered_count" ; 0) &
        let.Set ("~recently_correctly_answered_count" ; 0) &
        let.Set ("~recently_incorrectly_answered_count" ; 0)]
End If
Sort Records [Keep records in sorted order; Specified Sort Order: CHC::_id_QST; ascending
    CHC::z_creationTimestamp; descending]
    [Restore; No dialog]

Set Variable [$_question_id; Value:dev.Nil]
Go to Record/Request/Page
    [First]
Loop
    Set Variable [$_correctly_answered_count; Value:$_correctly_answered_count + CHC::is_correct]
    Set Variable [$_incorrectly_answered_count; Value:$_incorrectly_answered_count + (not CHC::is_correct)]
    If [(CHC::_id_QST = $_question_id)
        and
        (Get (RecordNumber) ≠ Get (FoundCount))]
        Omit Record
    Else
        Set Variable [$_question_id; Value:CHC::_id_QST]
        Go to Record/Request/Page
            [Next; Exit after last]
    End If

End Loop

Set Variable [$_recently_answered_count; Value:Get (FoundCount)]

Enter Find Mode [ ]
Set Field [CHC::is_correct; True]
Constrain Found Set [ ]

Set Variable [$_recently_correctly_answered_count; Value:Get (FoundCount)]
Set Variable [$_recently_incorrectly_answered_count; Value:$_recently_answered_count - $_recently_correctly_answered_count]
Set Variable [$_question_count; Value:Let (

```
[
  _sql = List (
    "SELECT COUNT(*)" ;
    "FROM   _questions_table" ;
    "WHERE  _level_field = ?" ;
    "AND    _status_field = ?" ;
    "AND    _answer_id_field IS NOT NULL"
  ) ;

  _questions_table = sql.QuotedTableName ( QST::_id ) ;
  _level_field = sql.QuotedFieldName ( QST::level ) ;
  _status_field = sql.QuotedFieldName ( QST::status ) ;
  _answer_id_field = sql.QuotedFieldName ( QST::_id_ANS~correct ) ;


  _sql = Substitute (
    _sql ;
    [ "_questions_table" ; _questions_table ] ;
    [ "_level_field" ; _level_field ] ;
    [ "_status_field" ; _status_field ] ;
    [ "_answer_id_field" ; _answer_id_field ]
  ) ;

  _result = ExecuteSQL ( _sql ; dev.Nil ; dev.Nil ; $Level ; "Approved" ) ;

  _ = ""
] ;

_result
```

)]
Set Variable [$_grade; Value:$_recently_correctly_answered_count / $_question_count]

Close Window [Current Window]
Exit Script [Result: let.Set ("~grade" ; $_grade) &
let.Set ("~answered_count" ; $_answered_count) &
let.Set ("~correctly_answered_count" ; $_correctly_answered_count) &
let.Set ("~incorrectly_answered_count" ; $_incorrectly_answered_count) &
let.Set ("~recently_correctly_answered_count" ; $_recently_correctly_answered_count) &
let.Set ("~recently_incorrectly_answered_count" ; $_recently_incorrectly_answered_count)]