; Example usage: Type "April 2004", "2024", " 1 April 2004", or " April 1, 2004" and press Ctrl+Alt+D to get "2024-04-01"

```
^!d::
    ; Save the currently selected text (date) to a variable
    Clipboard := ""
    Send, ^c
    ClipWait
    DateParts := clipboard
    ; Remove leading spaces and tabs from the front of the DateParts variable
    temp := ltrim(DateParts, " \t")
    Clipboard := temp
    RegExReplace(DateParts,"\b[\w\-\']+\b","",DatePartCount)
/*
```

Let’s break down the RegExReplace(DateParts, "\b[\w\-\']+\b", "", DatePartCount) command:

RegExReplace: This function replaces occurrences of a pattern (regular expression) inside a string.
DateParts: The first parameter is the input string (in this case, the variable DateParts).
"\b[\w\-\']+\b": The second parameter is the regular expression pattern to search for. Let’s dissect it:
\b: This represents a word boundary. It ensures that the match occurs at the beginning or end of a word.
[\w\-\']+: Inside the square brackets, we have a character class that matches any of the following:
\w: Represents any word character (letters, digits, or underscores).
\-\': Represents a hyphen or an apostrophe.
+: Indicates that one or more of these characters should be matched.
\b: Again, a word boundary to ensure the match occurs at the end of a word.
"": The third parameter is the replacement string. In this case, it’s an empty string, effectively removing the matched substrings.
DatePartCount: The fourth parameter is a variable where the number of replacements made will be stored.
*/

```
if (DatePartCount = 1) {
    FormattedDate := DateParts "-01-01"
} Else If (DatePartCount = 2) {
    StringSplit, DateParts, Clipboard, %A_Space%
    Month := DateParts1
    Year := DateParts2
    ; Convert month name to numeric value
    MonthNum := Format("{:02d}", MonthToNumber(Month))
    FormattedDate := Year "-" MonthNum "-01"
```

```
} Else If (DatePartCount = 3) {
   StringSplit, DateParts, Clipboard, %A_Space%
   If (StrLen(DateParts1) < 3) {
         Day := DateParts1
               Month := DateParts2
               StringSplit, Daypart, Day, `,
         sleep, 500
   } Else {
         Month := DateParts1
         Day := DateParts2
         StringSplit, Daypart, Day, `,
         Day := Daypart1
         sleep, 500
   }


   ; Append a 0 infront of Day if the length is 1 (I.E. day is 9 or less), making a 2 digit day
   Day := (StrLen(Day) = 1) ? "0" Day : Day
   Year := DateParts3

   ; Convert month name to numeric value
   MonthNum := Format("{:02d}", MonthToNumber(Month))

   ; Format the date as YYYY-MM-DD
   FormattedDate := Year "-" MonthNum "-" Day

}
;MsgBox, % DatePartCount
; Replace the clipboard content with the formatted date
Clipboard := FormattedDate

; Function to convert month name to numeric value
MonthToNumber(MonthName) {
   switch (MonthName) {
         case "January": return 01
         case "February": return 02
         case "March": return 03
         case "April": return 04
         case "May": return 05
         case "June": return 06
         case "July": return 07
         case "August": return 08
         case "September": return 09
         case "October": return 10
```

```
            case "November": return 11
            case "December": return 12
    }

            return 0 ; Invalid month name
}
```