**StagingData**

1. We are executing **two different queries for Pass and fail** staging test cases as we want each fail stage to be counted irrespective of its stage count (as per discussion – ideally if max stage is failing we shouldn’t be executing rest stages for it but in our data this we could found such cases) while in case of pass count we will increase total and pass count only if stages are passed till stage 1.

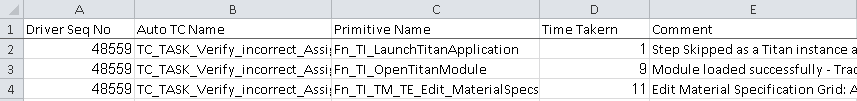
For example if we take combined output for pass and fail testcases,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 21-APR-17 06.00.49.520573000 | 27 | Passed | TestCase Execution Time\_Stage4 | 2:1:1:4:653 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 20-APR-17 14.45.20.057364000 | 105 | Passed | TestCase Execution Time\_Stage3 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 20-APR-17 09.57.03.815976000 | 89 | Passed | TestCase Execution Time\_Stage2 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 20-APR-17 08.08.48.119445000 | 432 | Passed | TestCase Execution Time\_Stage1 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 20-APR-17 06.57.55.081007000 | 140 | Failed | TestCase Execution Time\_Stage3 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 20-APR-17 06.01.56.552034000 | 99 | Passed | TestCase Execution Time\_Stage2 |
| TC\_EV\_CNI\_Logistic\_Update\_Split\_and\_Undo\_split | 19-APR-17 14.46.32.810346000 | 439 | Passed | TestCase Execution Time\_Stage1 |

Then in above case it will consider pass count as 2 which is incorrect as second time full execution is not completed. Hence we need to fire two separate queries and paste data in macro excel one below the other before running macro.

1. “StageAll” and “Invalid\_Status” instances in DB against Exec\_Comment column are ignored as it is difficult to get exact stage count with these comments
2. For failed test cases time will not get counted

**ExecutionPerDriverSequence**



**The primitive count per driver sequence is evaluated using the below flow:**

1. Compare 2 consecutive Primitive Name {say, i(C2) and i+1(C3), for first iteration and C3 and C4 for the next iteration and so on}
2. Primitive Names are same, GoTo2
3. Primitive Names are different, GoTo 4
4. Compare the respective Auto Test case name,

a) If Test case name is same, GoTo 3

b) If Test case name is not same, GoTo 4

1. Compare the execution time,

a) Time Taken for the first primitive(i) < next primitive(i+1), then GoTo 1

b) Time Taken for the first primitive(i) > next primitive(i+1), then GoTo 4

1. Fetch the respective Driver Sequence number of the first Primitive(i.e “i”)
2. Check if the driver sequence number exists in Dictionary

a) If Driver Sequence number exists, then, increment the Primitive name count against the Driver sequence.

b) If Driver Sequence number does not exist, add as a new Key in dictionary with default value as “0:1”. (Test Case Count : Primitive Count)

**The Auto Test case count per driver sequence is evaluated using the below flow:**

1. In Column “C”,i.e. Primitive Name, verify using InStr function if the text in the cell starts with string “TC\_”
2. If it starts with string “TC\_”, then, verify if the Comments column “E” contains the string “TestCase Execution Time”.
3. If Comment contain the string “TestCase Execution Time”, Fetch the respective Driver Sequence number
4. Check if the driver sequence number exists in Dictionary

a) If Driver Sequence number exists, then, increment the Primitive name count against the Driver sequence.

b) If Driver Sequence number does not exist, add as a new Key in dictionary with default value as “1:0”. (Test Case Count : Primitive Count)

**PrimitiveData**

The primitive count per driver sequence is evaluated using the below flow:

1. Compare 2 consecutive Primitive Name
2. Compare 2 consecutive Primitive Name Primitive Names are same, GoTo2
3. Primitive Names are different, GoTo 4
4. Compare the respective Auto Test case name,
5. If Test case name is same, GoTo 3.
6. If Test case name is not same, GoTo 4.
7. Compare the execution time,
8. Time Taken for the first primitive(i) < next primitive(i+1), then GoTo 1
9. Time Taken for the first primitive(i) > next primitive(i+1), then GoTo 4
10. Fetch the respective Primitive name as KEY of the first Primitive(i.e “i”)
11. Check if the Primitive name exists in Dictionary
12. If Primitive name exists, then, fetch the status “Pass” or “Fail” and increment the count and also add the time take for execution. (Order🡪 Pass:Fall:TimeTaken)
13. If Primitive name does not exist, add as a new Key in dictionary with default value.