R FORMULA TO COMPUTE PPI AND DQ DIMENSIONS VALUE

The formula reported below have been ideated to compute the PPIs selected in the motivating example.

All the formula (SQL-based) initially work on a DataFrame that is named 'Data1'. A DataFrame is a data structure that organizes data into a 2-dimensional table. It contains the event log obtained by combining the ones related to each subprocess.

1. Average Order Fulfillment Lead Time

***Average order fulfillment lead time*:** time (expressed in days) between the moment in which the request of the order is received from the subscriber and the payment of the service. The aim is to evaluate how long it takes an order to be managed from the moment in which it is received to the payment performed by the subscriber, including the time spent for the physical installation of the DSL line.

$$\frac{sum(SA_{paymentManagement}.payment\ receipt.time - CC_{orderManagement}.receive\ order\ request.time)}{count(SA_{serviceManagement}.receive\ order\ request)}$$

Dimensions associated: consistency, completeness, all.

PPI VALUE:

The first part consists on the identification of the cases in which both the timestamps needed to compute the PPI are not null.

```
notNullOrder= sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
notNullPaym = sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_payment receipt' AND [time] IS NOT NULL")
notNullBoth = sqldf("SELECT [case] FROM data1 WHERE [case] IN notNullOrder AND [case] IN notNullPaym")
```

Selection of the cases related to unavailable orders

```
unOrd = sqldf("SELECT [case] FROM data1 WHERE [event] = 'end_event_notification unavailability'")
```

Selection of the data needed to compute the PPI using the conditions expressed by the previous formulas.

```
ord = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM data1 WHERE [case] NOT IN unOrd AND [event] = 'event_MESSAGE_CC_receive order request' AND [case] IN notNullBoth")
paym = sqldf("SELECT [case] AS pCase, [time] AS pTime FROM data1 WHERe [event] = 'event_MESSAGE_payment receipt' AND [case] IN notNullBoth")
```

Calculation of the PPI Value:

```
diff = sqldf("SELECT AVG(strftime('%s', [pTime]) - strftime('%s', [oTime])) AS average FROM ord,paym WHERE [pCase] = [oCase]")
```

Since the value obtained is expessed in seconds, to find the correspond value in days:

```
PPI = diff/3600/24
```

CONSISTENCY:

The numerators takes the cases in which the order of timestamps is coherent with the flow of the process.

```
numerator  = sqldf("SELECT count(*) FROM ord,paym WHERE [pCase] = [oCase] AND [pTime] > [oTime]")
```

The denominator takes all the possible cases that can be used to calculate the consistency dimension

```
denominator = sqldf("SELECT COUNT(distinct [pCase]) FROM ord,paym WHERE [pCase] = [oCase] ")
```

Consistency Value:

```
consistency = numerator/denominator
```

COMPLETENESS:

Selection of the non-null attributes needed to compute the PPI:

```
complROR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
complPR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_payment receipt' AND [time] IS NOT NULL")
```

Selection of the all-possible value (both null and not-null) that can be used to compute the PPI:

```
denominator = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN NotNullBoth")
```

Completeness Value:

```
completeness = numerator/denominator
```

DEGREE OF REMEDIATIONS IN CONSISTENCY-RELATED SCENARIO:

The formula are almost identical to the ones used to calculate the dimension of consistency but – since for this dimension we are interested in the number of wrong data – the numerator considers the cells in the case with timestamps that are not coherent.

```
numerator  = sqldf("SELECT count(*) FROM ord,paym WHERE [pCase] = [oCase] AND [pTime] < [oTime]")
```

The denominator is the consideration of the all-possible value that can be used to compute the PPI:

```
denominator = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN NotNullBoth ")
```

Degree of Remediations Value:

```
deg = numerator/denominator
```

DEGREE OF REMEDIATIONS IN COMPLETENESS-RELATED SCENARIO:

The formula are almost identical to the ones used to calculate the dimension of completeness but – since for this dimension we are interested in the number of wrong data – the numerator considers number of null values.

```
complROR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
complPR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_payment receipt' AND [time] IS NULL")
numerator = complROR+complPR
```

The denominator is the consideration of the all-possible value that can be used to compute the PPI:

```
denominator= sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN NotNullBoth ")
```

Degree of Remediations Value:

```
deg = numerator/denominator
```

In case of application of the remediation 'Average Value from Historical Data', there is the need to consider the inapplicability case that occurs when both the timestamps are null. In this case, the numerator of this dimension should be decreased by the cells of the cases in which there are missing values for both the timestamps:

```
t1 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
t2 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_payment receipt' AND [time] IS NULL")
b2 = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN t1 AND [case] IN t2 ")
numerator = complROR+complPR-b2
```

The denominator is exactly the same of the previous situation.

DEGREE OF REMEDIATIONS IN ALL-RELATED SCENARIO:

The numerator counts all the cells in which there is an error of completeness or consistency.

```
cons  = sqldf("SELECT count(*) FROM ord,paym WHERE [pCase] = [oCase] AND [pTime] < [oTime]")
complROR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
complPR = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_payment receipt' AND [time] IS NULL")
numerator = cons+complROR+complPR
```

The denominator is exactly the same of the previous situation.

```
denominator= sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN NotNullBoth ")
```

Degree of Remediations Value:

```
deg = numerator/denominator
```

As before, in case of application of the remediation 'Average Value from Historical Data', there is the need to consider the inapplicability case.

```
t1 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
t2 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_payment receipt' AND [time] IS NULL")
b2 = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN t1 AND [case] IN t2 ")
numerator = complROR+complPR+cons-b2
```

The denominator is exactly the same of the previous situation.

2. Average fuel for Driving

***Average fuel for driving*:** average fuel used (expressed in liters) for each available order received. The aim is to measure the average quantity of fuel used for each order to evaluate the level of sustainability of the travels of the Field Agent, who moves to install the DSL line at the customer's place.

$$\frac{sum(FA_{InstallationManagement}.manageinstallation\_complete.fuelUsed)}{count(SA_{serviceManagement}.receive\ order\ request)}$$

Dimensions associated: accuracy, completeness, all.

PPI VALUE:

Select the fuel used for each case, then the average is computed.

```
fuelPerCase = sqldf("SELECT SUM([fuelUsed])AS fuel FROM data1 WHERE [fuelUsed] IS NOT NULL GROUP BY [case]")
```

Calculation of the PPI Value:

```
value = sqldf("SELECT AVG([fuel]) from fuelPerCase")
```

ACCURACY:

The numerator counts the value that remains within the correct interval [1,7]

```
accFuel = sqldf("SELECT COUNT(*)/1 FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] BETWEEN 1 AND 7")
```

The denominator counts all the non-value attributes used for the computation of the PPI.

```
denominator = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] IS NOT NULL")
```

Accuracy Value:

```
accur = accFuel/denominator
```

COMPLETENESS:

Selection of the non-null attributes needed to compute the PPI:

```
fuelUsedC = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] IS NOT NULL")
```

Selection of the all-possible value (both null and not-null) that can be used to compute the PPI:

```
denominator = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete'")
```

Completeness Value:

```
fuelUsed/denominator
```

DEGREE OF REMEDIATIONS IN ACCURACY-RELATED SCENARIO:

The formula are almost identical to the ones used to calculate the dimension of accuracy but – since for this dimension we are interested in the number of wrong data – the numerator considers the values out of the correct interval.

```
numerator = sqldf("SELECT COUNT FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] NOT BETWEEN 1 AND 7")
```

The denominator counts the total number of values used to compute the PPI.

```
denominator = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] IS NOT NULL")
```

Degree of Remediations Value:

```
deg = numerator/denominator
```

DEGREE OF REMEDIATIONS IN COMPLETENESS-RELATED SCENARIO:

Since there is no inapplicability scenario, the degree of remediation dimension corresponds to the incompleteness because the remediations apply on all the null-value data.

Degree of Remediations Value:

deg = 1 – completeness

DEGREE OF REMEDIATIONS IN ALL-RELATED SCENARIO:

The numerator counts all the cells in which there is an error of completeness or accuracy.

numerator1 = sqldf("SELECT COUNT FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] NOT BETWEEN 1 AND 7")
numerator2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] IS NULL")

The denominator is exactly the same of the previous situation.

denominator = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'manageinstallation_complete' AND [fuelUsed] IS NOT NULL ")

Degree of Remediations Value:

deg = numerator/denominator

3. Percentage of Successful Orders without payment solicitation

Percentage of successful orders without payment solicitation: percentage of calls turned into successful orders - i.e. with the process of installation completed - and without delay in payment. The aim is to measure the percentage of orders that have been fulfilled without the need to send solicitations due to payment delay.

$$\frac{count\ (SA_{serviceManagement}.receive\ installation\ confirm.status\ = "in\ full" AND\ not\ (SA_{paymentManagement}.send\ solicitation\ with\ new\ amount))}{count\ (CC_{orderManagement}.receive\ order\ request)} \cdot 100$$

Dimensions associated: accuracy, consistency, completeness, all.

PPI VALUE:

The first part counts the number of orders defined as successful, i.e with status in full and without delay in payment.

succOrders = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [status] = 'in full' AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [case] NOT IN (SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_send solicitation with new amount')")

Then, the total number of calls to the call center is taken.

numCalls = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'CC_OM_start_event_' ")

Calculation of the PPI Value:

PPI = succOrders*100/numCalls

ACCURACY:

Since the PPI involves a nominal attribute, the accuracy evaluation requires the calculation of the lexicographic distance between each attribute to any element of the domain. So, the first part takes the values

of the attribute status and then the Jaccard distance is calculated. Finally, the sum of the accuracy of each attribute compose the numerator.

```
statuses = sqldf("SELECT [status] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND  [status] IS NOT NULL")
numerator = 0
for (i in 1:nrow(statuses)) {numerator = numerator+ (1 - min(stringdist(statuses[i,'status'], "in full", method='jaccard', q=2),
stringdist(statuses[i,'status'], "activated", method = 'jaccard', q = 2)))}
```

The denominator counts all the non-value attributes used for the computation of the PPI.

```
denominator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
```

Accuracy Value:

```
Acc = numerator/denominator
```

CONSISTENCY:

The consistency is calculated by evaluating the semantic rule of status that has to be 'in full' for the event of the installation confirm, 'activated' for the event activate the service. So, once the values of status have been taken, the evaluation of the semantic rule is done.

```
eventInst = sqldf("SELECT [case] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm'")
eventAct = sqldf("SELECT [case] FROM data1 WHERE [event] = 'activate the service' ")
yesyes = sqldf("SELECT distinct [case] FROM data1 WHERE [case] IN eventAct AND [case] IN eventInst")
fullCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND ([event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND
[status] = 'in full')")
activatedCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND ([event] = 'activate the service' AND [status] = 'activated')")
completeStatusCheck = sqldf("SELECT COUNT(distinct [case])/1 FROM data1 WHERE [case] IN yesyes AND [case] IN fullCheck AND [case] IN
activatedCheck")
```

The denominator counts the case in which the semantic rule has been checked.

```
denominator = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' OR [event]
= 'activate the service' ")
```

Consistency Value:

```
cons = completeStatusCheck/denominator
```

COMPLETENESS:

Selection of the non-null attributes needed to compute the PPI:

```
numerator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
```

Selection of the all-possible value (both null and not-null) that can be used to compute the PPI:

```
denominator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'event_MESSAGE_receive installation confirm'")
```

Completeness Value:

```
comp = numerator/denominator
```

DEGREE OF REMEDIATIONS IN CONSISTENCY-RELATED SCENARIO:

The formula are almost identical to the ones used to calculate the dimension of consistency but – since for this dimension we are interested in the number of wrong data – the numerator considers the values out of the domain (in full, activated) and therefore violate the semantic rule.

```
eventInst = sqldf("SELECT [case] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm'")
eventAct = sqldf("SELECT [case] FROM data1 WHERE [event] = 'activate the service' ")
yesyes = sqldf("SELECT distinct [case] FROM data1 WHERE [case] IN eventAct AND [case] IN eventInst")
fullCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] <> 'in full' ")
activatedCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND [event] = 'activate the service' AND [status] <> 'activated' ")
```

To make a case violating the semantic rule, it is enough that only one of the attribute is not the right one.

```
checkOr = sqldf("SELECT [case] FROM data1 WHERE [case] IN fullCheck OR [case] IN activatedCheck")

completeStatusCheck = sqldf("SELECT COUNT(distinct [case])/1 FROM data1 WHERE [case] IN yesyes AND [case] IN checkOr")
```

The denominator counts the total number of values used to compute the PPI.

```
denominator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'event_MESSAGE_receive installation confirm'")
```

Degree of Remediations Value:

```
deg = completeStatusCheck/denominator
```

DEGREE OF REMEDIATIONS IN ACCURACY-RELATED SCENARIO:

Since every accuracy error is also a consistency one, the formula is very similar to degree of remediations in case of consistency but in this case checks also that the value of status is not ‘activated’ in receiveInstallationConfirm and not ‘in full’ in activate the service because that are not accuracy errors but only consistency ones.

```
eventInst = sqldf("SELECT [case] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm'")
eventAct = sqldf("SELECT [case] FROM data1 WHERE [event] = 'activate the service' ")
yesyes = sqldf("SELECT distinct [case] FROM data1 WHERE [case] IN eventAct AND [case] IN eventInst")
fullCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND ([event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] <> 'in full' AND [status] <> 'activated'")
activatedCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND [event] = 'activate the service' AND [status] <> 'activated' AND [status] <> 'in full' ")
checkOr = sqldf("SELECT [case] FROM data1 WHERE [case] IN fullCheck OR [case] IN activatedCheck")
completeStatusCheck = sqldf("SELECT COUNT(distinct [case])/1 FROM data1 WHERE [case] IN yesyes AND [case] IN checkOr")
```

The denominator is exactly the same of the previous situation.

```
denominator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
```

Degree of Remediations Value:

```
deg = completeStatusCheck/denominator
```

DEGREE OF REMEDIATIONS IN COMPLETENESS-RELATED SCENARIO:

Since there is no inapplicability scenario, the degree of remediation dimension corresponds to the incompleteness because the remediations apply on all the null-value data.

Degree of Remediations Value:

```
deg = 1 – completeness
```

DEGREE OF REMEDIATIONS IN ALL-RELATED SCENARIO:

The numerator counts all the cells in which there is an error of consistency, completeness or accuracy. Consistency and accuracy part:

```
eventInst = sqldf("SELECT [case] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm'")
eventAct = sqldf("SELECT [case] FROM data1 WHERE [event] = 'activate the service' ")
yesyes = sqldf("SELECT distinct [case] FROM data1 WHERE [case] IN eventAct AND [case] IN eventInst")
fullCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND ([event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND
[status] IS NOT NULL AND [status] <> 'in full') ")
activatedCheck = sqldf("SELECT [case] FROM data1 WHERE [case] IN yesyes AND ([event] = 'activate the service' AND [status] IS NOT NULL AND
[status] <> 'activated')  ")
checkOr = sqldf("SELECT [case] FROM data1 WHERE [case] IN fullCheck OR [case] IN activatedCheck")
completeStatusCheck = sqldf("SELECT COUNT(distinct [case])/1 FROM data1 WHERE [case] IN yesyes AND [case] IN checkOr")
```

Completeness part:

```
numerator2 = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NULL")
```

The denominator is exactly the same of the previous situation.

```
denominator = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL ")
```

Degree of Remediations Value:

```
deg = (completeStatusCheck+numerator1)/denominator
```

4. Percentage of orders with perfect lead time

***Percentage of orders fulfilled with perfect lead time*:** percentage of orders fulfilled - i.e. with the installation completed with success – without any complaint by the subscriber and paid within the payment terms . The aim is to measure the quantity of orders that are linear in their fulfillment, i.e. that do not involve any complaint by the subscriber and solicitation caused by payment delay.

$$\frac{\begin{array}{c} count(order\ fulfillment\ lead\ time < [(FA_{installationManagement}.send\ installation\ confirm.time) - \\ (CC_{orderManagement}.receive\ order\ request.time) + (SA_{paymentManagement}.send\ invoice.paymentTerms) \\ AND\ SA_{serviceManagement}.receive\ installation\ confirm.status = "in\ full"\ AND \\ not\ (CC_{complaintManagement}.complaint)) \end{array}}{\begin{array}{c} count\ (SA_{serviceManagement}.receive\ installation\ confirm.status = "in\ full" \\ AND\ not\ (CC_{complaintManagement}.complaint)) \end{array}}$$

Dimensions associated: accuracy, consistency, completeness, all.

PPI VALUE:

The first part selects the orders that meet the condition expressed in the formula, then takes the elements needed to calculate the average lead time.

```
compRec = sqldf("SELECT [case] FROM data1 WHERE [event] = 'start_event_compliant'")
una = sqldf("SELECT [case] FROM data1 WHERE [event] = 'end_event_notification unavailability'")
statusFull = sqldf("SELECT [case] FROM data1 WHERE [status] = 'in full'")
notNullOrder= sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
notNullPaym = sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_payment receipt' AND [time] IS NOT NULL")
termsNotNull = sqldf("SELECT [case] FROM data1 WHERE [paymentTerms] IS NOT NULL")
notNullInst = sqldf("SELECT [case] FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL")
```

```
rdata1 = sqldf("SELECT * FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [case] IN statusFull AND [case] IN notNullOrder AND [case] IN notNullPaym AND [case] IN termsNotNull")
```

Then, the perfect lead time for these orders is calculated, i.e. orders completed with success – without any complaint by the subscriber and paid within the payment terms.

```
rightOrder = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM rdata1 WHERE [event] = 'event_MESSAGE_CC_receive order request' ")
intConf = sqldf("SELECT [case] AS fCase, [Time] as fTime FROM rdata1 WHERE [event] = 'message_sendInstallationConfirmToSA' ")
terms = sqldf("SELECT [case] AS pCase, [paymentTerms] FROM rdata1 WHERE [event] = 'event_MESSAGE_send invoice' ")
tot = sqldf("SELECT [pCase], strftime('%s', [fTime]) - strftime('%s', [oTime]) + [paymentTerms]*24*60*60 AS lt FROM rightOrder, intConf,terms WHERE [oCase] = [fCase] AND [fCase] = [pCase] GROUP BY [pCase]")
```

Then, the lead time – using the classical formula also expressed for the first KPI - is calculated.

```
ord = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM rdata1 WHERE [event] = 'event_MESSAGE_CC_receive order request'")
paym = sqldf("SELECT [case] AS pCase, [time] AS pTime FROM rdata1 WHERe [event] = 'event_MESSAGE_payment receipt'")
diff = sqldf("SELECT [oCase], (strftime('%s', [pTime]) - strftime('%s', [oTime])) AS leadTime FROM ord,paym WHERE [pCase] = [oCase] GROUP BY [pCase]")
```

Subsequently, the lead time of the orders are compared to the related perfect lead time.

```
compare = sqldf("SELECT COUNT([oCase]) FROM diff,tot WHERE [pCase] = [oCase] AND [leadTime] <= [lt]")
```

Finally, the total number of orders suitable for the calculation are taken:

```
totCase = sqldf("SELECT COUNT(distinct[case]) FROM rdata1 WHERE [case] NOT IN una")
```

Calculation of the PPI Value:

```
PPI = compare/totCase
```

ACCURACY:

Since this PPI comprised a nominal attribute (status) and a numerical attribute (paymentTerms), the accuracy of the PPI is the sum of the accuracy on these 2 attributes.
Accuracy for status:

```
statuses = sqldf("SELECT [status] FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
sum = 0
for (i in 1:nrow(statuses)) {sum = sum + (1 - min(stringdist(statuses[i,'status'], "in full", method='jaccard', q=2), stringdist(statuses[i,'status'], "activated", method = 'jaccard', q = 2)))}
```

Accuracy for paymentTerms:

```
ptacc = sqldf("SELECT COUNT(*)/1 FROM data1 WHERE [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] BETWEEN 3 AND 14")
```

The denominator counts all the non-value attributes used for the computation of the PPI.

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
den2 = sqldf("SELECT COUNT(*) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
denominator = den1+den2
```

Accuracy Value:

```
acc = (sum+ptacc)/denominator
```

COMPLETENESS:

Selection of the non-null attributes needed to compute the PPI:

```
fa_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL")
cc_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
sa_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
comps = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL ")
numerator = fa_compl + cc_compl + sa_compl+comps
```

Selection of the all-possible value (both null and not-null) that can be used to compute the PPI:

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una  AND [event] = 'message_sendInstallationConfirmToSA' ")
den2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' ")
den3 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec  AND [case] NOT IN una AND [event] = 'event_MESSAGE_send invoice' ")
den4 = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' ")
denominator = den1 + den2 + den3 + den4
```

Completeness Value:

```
comp = numerator/denominator
```

CONSISTENCY:

Since this PPI comprises the attribute ‘status’ and two timestamps, the consistency of the PPI is calculated by taking as the average consistency for all the cases involved in the computation of the PPI in which both the semantic rules have been applied.
First, the elements needed to assess the consistency are considered:

```
unOrd = sqldf("SELECT [case] FROM data1 WHERE [event] = 'end_event_notification unavailability'")
ord = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM data1 WHERE [case] NOT IN unOrd AND [event] = 'event_MESSAGE_CC_receive order request'")
noti = sqldf("SELECT [case] AS pCase, [time] AS pTime FROM data1 WHERe [event] = 'message_sendInstallationConfirmToSA'")
st2 = sqldf("SELECT [case] as s2Case, [event] as e2,[status] AS sAct FROM data1 WHERE [event] = 'activate the service'")
st = sqldf("SELECT [case] as sCase, [event] as e1,[status] AS sFull FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm'")
uni = sqldf("SELECT * FROM ord,noti,st,st2 WHERE [oCase] = [pCase] AND [pCase] = [sCase] AND [sCase] = [s2Case]")
```

Then, both the semantic rules are checked in every process case using a loop cycle in R:

```
r1 = 0
        r2 = 0
        d1 = 0
        d2 = 0
        consc = 0
        for(i in 1:nrow(uni)) {
        r1 = 0
        r2 = 0
        d1 = 0
        d2 = 0

        a <- uni[i,'pTime']
        b <- uni[i,'oTime']
        c <- uni[i,'sFull']
        d <- uni[i,'sAct']
         if(!is.na(a) & !is.na(b)) {
        if(a > b) {
        r1 = r1 + 1 }
        d1 = d1 + 1
        }
        if(!is.na(c) & !is.na(d)) {
         if(c == 'in full' & d == 'activated') {
```

```
            r2 = r2+1 } }

            d2 = d2+1
            consc = consc + ((r1+r2)/(d1+d2))
}
```

Consistency Value:

At the denominator, the nrow(uni) contains the rows of the table withthe elements of the case needed to evaluate the consistency. Therefore it represents the number of cases on which the rules have been applied.

```
cons = consc/nrow(uni)
```

DEGREE OF REMEDIATIONS IN COMPLETENESS-RELATED SCENARIO:

The formula are almost identical to the ones used to calculate the dimension of completeness but – since for this dimension we are interested in the number of wrong data – the numerator considers number of null values.

```
fa_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NULL")
cc_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
sa_compl = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] IS NULL")
comps = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NULL ")
numerator = fa_compl + cc_compl + sa_compl+comps
```

The denominator is the consideration of the all-possible value that can be used to compute the PPI:

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una  AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL ")
den2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
den3 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec  AND [case] NOT IN una AND [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
den4 = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
denominator = den1 + den2 + den3 + den4
```

Degree of Remediations Value:

```
deg = numerator/denominator
```

As mentioned for the first PPI, in case of application of the remediation 'Average Value of Historical Data', there is the need to consider the inapplicability case.

```
t1 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
t2 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NULL")
b2 = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN t1 AND [case] IN t2 ")
numerator = fa_compl + cc_compl + sa_compl+comps-b2
```

The denominator is exactly the same of the previous situation.

DEGREE OF REMEDIATIONS IN ACCURACY-RELATED SCENARIO:

The formula for this dimension is quite similar to the one for accuracy but since we are interested in the erroneous value, we count the erroneous value for paymentTerms and Status.

```
ptacc = sqldf("SELECT COUNT(*)/1 FROM data1 WHERE [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] NOT BETWEEN 3 AND 14")
statuses = sqldf("SELECT COUNT([status]) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND  [status] IS NOT
```

```
NULL AND [status] <> 'in full'")
numerator = ptacc+statuses
```

The denominator is exactly the same of the previous situation.

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una  AND [event] =
'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL ")
den2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] =
'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
den3 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec  AND [case] NOT IN una AND [event] =
'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
den4 = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm'
AND [status] IS NOT NULL")
denominator = den1 + den2 + den3 + den4
```

Degree of Remediations Value:

```
PPI = numerator/denominator
```

DEGREE OF REMEDIATIONS IN CONSISTENCY-RELATED SCENARIO:

The formula for this dimension is quite similar to the one for consistency but since we are interested in the erroneous value, we count the elements in which there is an erroneous value for status and the timestamps are not coherent.

```
statuses = sqldf("SELECT COUNT([status]) FROM data1 WHERE [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND  [status] IS NOT
NULL AND [status] <> 'in full'")
compRec = sqldf("SELECT [case] FROM data1 WHERE [event] = 'start_event_compliant'")
notNullOrder= sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
notNullPaym = sqldf("SELECT [case] FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL")
notNullBoth = sqldf("SELECT [case] FROM data1 WHERE [case] IN notNullOrder AND [case] IN notNullPaym")
unOrd = sqldf("SELECT [case] FROM data1 WHERE [event] = 'end_event_notification unavailability'")
ord = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM data1 WHERE [case] NOT IN unOrd AND [case] NOT IN compRec  AND [event] =
'event_MESSAGE_CC_receive order request' AND [case] IN notNullBoth")
paym = sqldf("SELECT [case] AS pCase, [time] AS pTime FROM data1 WHERe [event] = 'message_sendInstallationConfirmToSA' AND [case] NOT IN
compRec  AND [case] IN notNullBoth")
num  = sqldf("SELECT count(*) FROM ord,paym WHERE [pCase] = [oCase] AND [pTime] < [oTime]")
numerator = statuses+num
```

The denominator is exactly the same of the previous situation.

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una  AND [event] =
'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL ")
den2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] =
'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
den3 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec  AND [case] NOT IN una AND [event] =
'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
den4 = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm'
AND [status] IS NOT NULL")
denominator = den1 + den2 + den3 + den4
```

Degree of Remediations Value:

```
PPI = numerator/denominator
```

DEGREE OF REMEDIATIONS IN ALL-RELATED SCENARIO:

The numerator counts all the cells in which there is an error of consistency, completeness or accuracy. Accuracy, consistency and completeness of status and payment terms:

```
ptacc = sqldf("SELECT COUNT(*)/1 FROM data1 WHERE [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] NOT BETWEEN 3 AND 14")
ptnull = sqldf("SELECT COUNT([paymentTerms]) from data1 WHERE [case] NOT IN compRec AND [event] = 'event_MESSAGE_send invoice' AND
```

```
[paymentTerms] IS NULL")
statAccCons = sqldf("SELECT COUNT([status]) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND  [status] IS NOT NULL AND [status] <> 'in full'")
stNull = sqldf("SELECT COUNT([status]) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND  [status] IS NULL ")
```

Consistency of the timestamps:

```
notNullOrder= sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
notNullPaym = sqldf("SELECT [case] FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL")
notNullBoth = sqldf("SELECT [case] FROM data1 WHERE [case] IN notNullOrder AND [case] IN notNullPaym")
unOrd = sqldf("SELECT [case] FROM data1 WHERE [event] = 'end_event_notification unavailability'")
ord = sqldf("SELECT [case] AS oCase, [time] AS oTime FROM data1 WHERE [case] NOT IN unOrd AND [case] NOT IN compRec  AND [event] = 'event_MESSAGE_CC_receive order request' AND [case] IN notNullBoth")
paym = sqldf("SELECT [case] AS pCase, [time] AS pTime FROM data1 WHERe [event] = 'message_sendInstallationConfirmToSA' AND [case] NOT IN compRec  AND [case] IN notNullBoth")
num  = sqldf("SELECT count(*) FROM ord,paym WHERE [pCase] = [oCase] AND [pTime] < [oTime]")


numerator = ptacc+ptnull+statAccCons+stNull+num
```

The denominator is exactly the same of the previous situation.

```
den1 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND  [case] NOT IN una  AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NOT NULL ")
den2 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec AND [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NOT NULL")
den3 = sqldf("SELECT COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN compRec  AND [case] NOT IN una AND [event] = 'event_MESSAGE_send invoice' AND [paymentTerms] IS NOT NULL")
den4 = sqldf("SELECT COUNT(*) FROM data1 WHERE [case] NOT IN compRec AND [event] = 'SA_SM_event_MESSAGE_receive installation confirm' AND [status] IS NOT NULL")
denominator = den1 + den2 + den3 + den4
```

Degree of Remediations Value:

```
PPI = numerator/denominator
```

As mentioned for the first PPI, in case of application of the remediation 'Average Value of Historical Data', there is the need to consider the inapplicability case.

```
t1 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
t2 = sqldf("SELECT [case] FROM data1 WHERE [case] NOT IN una AND [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NULL")
b2 = sqldf("SELECT 2*COUNT(distinct [case]) FROM data1 WHERE [case] NOT IN una AND [case] IN t1 AND [case] IN t2 ")
numerator = ptacc+ptnull+statAccCons+stNull+num-b2
```

The denominator is exactly the same of the previous situation.

5. Remediations

Error: null value for t('event_MESSAGE_payment receipt')
PPI: Average Order Fulfillment Lead Time

Remediation 1: t(SA_payment receipt) = t(CC_receive order request) + avgLeadTime (17)

```
for (i in 1:4999) { a <- data1$time[data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request']
 if(!identical(a,character(0))) { if(!is.na(a)) {
 b = data1$time[data1$case == i & data1$event == 'event_MESSAGE_payment receipt']
 d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") + lubridate::days(17))
 e = as.POSIXct(d, origin = "1970-01-01")
 f = as.character(e)
 if(!identical(b,character(0))) { data1$time[data1$time == '' & data1$case == i & data1$event == 'event_MESSAGE_payment receipt'] <- f }
if(!identical(b,character(0))) { data1$time[is.na(data1$time) & data1$case == i & data1$event == 'event_MESSAGE_payment receipt'] <- f }}}}
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS") }
```

Remediation 2: T(SA_payment receipt) = diff(t(act_succ), t(act_prec))/2 + t(act_prec)

```
tog = sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_payment receipt' AND [time] IS NULL")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case']
rn <- which(data1$case == cs & data1$event =='event_MESSAGE_payment receipt')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'event_MESSAGE_payment receipt'] <- f
}
```

Error: null value for t('event_MESSAGE_CC_receive order request')
PPI: Average Order Fulfillment Lead Time, Percentage of Orders with Perfect Lead Time

Remediation 1: t(CC_receive order request) = t(SA_payment receipt) -avgLeadTime (17)

```
for (i in 1:4999) { a <- data1$time[data1$case == i & data1$event == 'event_MESSAGE_payment receipt']
if(!identical(a,character(0))) { if(!is.na(a)) {
b = data1$time[data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request']
d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") - lubridate::days(17))
e = as.POSIXct(d, origin = "1970-01-01")
f = as.character(e)
if(!identical(b,character(0))) { data1$time[data1$time == '' & data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request'] <- f }
if(!identical(b,character(0))) { data1$time[is.na(data1$time)  & data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request'] <- f
}}}}
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS") }
```

Remediation 2: t('event_MESSAGE_CC_receive order request') = diff(t(act_succ), t(act_prec))/2 + t(act_prec)

```
tog = sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case']
rn <- which(data1$case == cs & data1$event =='event_MESSAGE_CC_receive order request')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'event_MESSAGE_CC_receive order request'] <- f
}
```

Error: wrong order of timestamps between *SA_payment receipt* and *CC_receive order request*: t(*SA_payment receipt*)<t(*CC_receive order request*).
PPI: Average Order Fulfillment Lead

The remediations are applied only to t(SA_payment receipt) based on how the errors has been injected,i.e. making its timestamp precedent than t(CC_receive order request)

Remediation 1: t(SA_payment receipt) = t(CC_receive order request) + avgLeadTime (17)

```
ror = sqldf("SELECT [case] as case1, [time] AS time1 FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request'")
pay = sqldf("SELECT [case] as case2, [time] AS time2 FROM data1 WHERE [event] = 'event_MESSAGE_payment receipt'")
tog = sqldf(" SELECT [case1],[time1],[time2] FROM ror,pay WHERE [case1] = [case2] AND [time1] > [time2]")
for (i in 0:nrow(tog)) {
cs <- tog[i, 'case1']
a <- data1$time[data1$case == cs & data1$event == 'event_MESSAGE_CC_receive order request']
if(!identical(a,character(0))) { if(!is.na(a)) {
 b = data1$time[data1$case == cs & data1$event == 'event_MESSAGE_payment receipt']
lt <- 17
 d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") + lubridate::days(lt))
 e = as.POSIXct(d, origin = "1970-01-01")
 f = as.character(e)
 if(!identical(b,character(0))) { data1$time[data1$case == cs & data1$event == 'event_MESSAGE_payment receipt'] <- f } } }  }
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS") }
```

Remediation 2: T(SA_payment receipt) = diff(t(act_succ), t(act_prec))/2 + t(act_prec)

```
p = sqldf("SELECT * FROM data1 WHERE [event] = 'SA_PM_end_event_'")
 for (i in 1:nrow(p)) {
cs <- p[i, 'case']
y <- data1$time[data1$case == cs & data1$event == 'SA_PM_end_event_']
x <- as.POSIXct(y, format = "%Y-%m-%d %H:%M:%OS")
z <- x + lubridate::minutes(3)
e = as.POSIXct(z, origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'SA_PM_end_event_'] <- f
}
ror = sqldf("SELECT [case] as case1, [time] AS time1 FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request'")
pay = sqldf("SELECT [case] as case2, [time] AS time2 FROM data1 WHERE [event] = 'event_MESSAGE_payment receipt'")
tog = sqldf(" SELECT [case1],[time1],[time2] FROM ror,pay WHERE [case1] = [case2] AND [time1] > [time2]")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case1']
rn <- which(data1$case == cs & data1$event =='event_MESSAGE_payment receipt')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'event_MESSAGE_payment receipt'] <- f
}
```

Error: *FuelUsed* does not belong to the tolerance interval [1,7].
PPI: Average fuel for driving

Remediation 1: The average value of *fuelUsed* calculated with historical data is used

```
data1$fuelUsed[data1$event == 'manageinstallation_complete' & data1$fuelUsed<1 | data1$fuelUsed>7 ] <- 3.471
```

Remediation 1: If the value provided is km traveled, fuelUsed is calculated by using data about the average liters consumed per km (about 13 km/l).

```
for (i in 0:4999) {
a <- data1$fuelUsed[data1$case == i & data1$fuelUsed > 7 & data1$event == 'manageinstallation_complete']
if(!identical(a,numeric(0))) { if(!identical(a,logical(0))) { if(!is.na(a)) {
b <- a/13
data1$fuelUsed[data1$fuelUsed > 7 & data1$event == 'manageinstallation_complete'] <- b }
} } }
```

Error: Null value in *fuelUsed*
PPI: Average fuel for driving

Remediation 1: The average value of *fuelUsed* calculated with historical data is used
The formula is the same of Remediation 1 for the error 'FuelUsed does not belong to the tolerance interval [1,7].'

Error: the value of status does not belong to the elements of domain {activated, in full}
PPI: Percentage of successful orders without payment solicitation, Percentage of orders fulfilled with perfect lead time.

Remediation 1: If event = 'activate the service' -> status = 'activated', If event = 'receive installation confirm'-> status = 'in full'

```
data1$status[data1$event == 'activate the service'] <- 'activated'
data1$status[data1$event == 'SA_SM_event_MESSAGE_receive installation confirm'] <- 'in full'
```

Remediation 2: The value of the domain (activated, in full) that has the minimum lexicographic distance with status is taken.

```
data1$status[data1$status == ''] <- NA
wrongStatus = sqldf("SELECT * FROM data1 WHERE [status] IS NOT NULL AND [status] <> 'activated' AND [status] <> 'in full' ")
for (i in 1:nrow(wrongStatus)) {
cs <- wrongStatus[i,'case']
st <- wrongStatus[i,'status']
ev = wrongStatus[i,'event']
dist1 <- 1 - stringdist(st, "in full", method='jaccard', q=2)
dist2 <- 1 - stringdist(st, "activated", method='jaccard', q=2)
if(dist1 > dist2) { st <- "in full" } else { st <- "activated" }
data1$status[data1$case == cs & data1$event == ev] <- st }
```

Error: not expected value, i.e. violation of the *consistentStatus* semantic rule – the activity event *SA_receive installation confirm* has occured but the status is not "in full"; the activity *SA_activate the service* has occurred but the status is not "activated.
PPI: Percentage of successful orders without payment solicitation, Percentage of orders fulfilled with perfect lead time.

Remediation 1: If event = 'activate the service' -> status = 'activated', If event = 'receive installation
The formula is the same of Remediation 1 for the error 'the value of status does not belong to the elements of domain {activated, in full}'

Remediation 2: The value of the domain (activated, in full) that has the minimum lexicographic distance with the incorrect status is taken.
The formula is the same of Remediation 2 for the error 'the value of status does not belong to the elements of domain {activated, in full}'

Error: null value in *status.*
PPI: Percentage of successful orders without payment solicitation, Percentage of orders fulfilled with perfect lead time.

Remediation 1: If event = 'activate the service' -> status = 'activated' , If event = 'receive installation confirm'-> status = 'in full'
The formula is the same of Remediation 1 for the error 'the value of status does not belong to the elements of domain {activated, in full}'

Error: the value of paymentTerms does not belong to the tolerance interval [3,14].
PPI: Percentage of orders fulfilled with perfect lead time.

The value of *paymentTerms* is taken by computing the average of payment terms from historical data (8)

```
data1$paymentTerms[data1$paymentTerms > 14 | data1$paymentTerms < 3] <- 8
```

Error: wrong order of timestamps between FA_send installation confirm and CC_receive order request: t(FA_send installation confirm)<t(CC_receive order request).
PPI: Percentage of orders fulfilled with perfect lead time.
The remediations are applied only to t(*FA_send installation confirm*) based on how the errors has been injected,i.e. making its timestamp precedent than t(CC_receive order request)

Remediation 1: t(FA_send installation confirm) = t(CC_receive order request) + avgDistance where Distance = t(FA_send installation confirm) – t(CC_receive order request) (value = 4)

```
ror = sqldf("SELECT [case] as case1, [time] AS time1 FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request'")
pay = sqldf("SELECT [case] as case2, [time] AS time2 FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA'")
tog = sqldf(" SELECT [case1],[time1],[time2] FROM ror,pay WHERE [case1] = [case2] AND [time1] > [time2]")
for (i in 0:nrow(tog)) {
cs <- tog[i, 'case1']
a <- data1$time[data1$case == cs & data1$event == 'event_MESSAGE_CC_receive order request']
 if(!identical(a,character(0))) { if(!is.na(a)) {
 b = data1$time[data1$case == cs & data1$event == 'message_sendInstallationConfirmToSA']
 d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") + lubridate::days(4))
 e = as.POSIXct(d, origin = "1970-01-01")
 f = as.character(e)
 if(!identical(b,character(0))) { data1$time[data1$case == cs & data1$event == 'message_sendInstallationConfirmToSA'] <- f } } }  }
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS")
```

Remediation 2: T(FA_send installation confirm) = diff(t(act_succ), t(act_prec))/2 + t(act_prec).

```
ror = sqldf("SELECT [case] as case1, [time] AS time1 FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request'")
pay = sqldf("SELECT [case] as case2, [time] AS time2 FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA'")
tog = sqldf(" SELECT [case1],[time1],[time2] FROM ror,pay WHERE [case1] = [case2] AND [time1] > [time2]")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case1']
```

```
rn <- which(data1$case == cs & data1$event =='message_sendInstallationConfirmToSA')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'message_sendInstallationConfirmToSA'] <- f
}
```

Error: Null value in paymentTerms.
PPI: Percentage of orders fulfilled with perfect lead time.

Remediation 1: The value of *paymentTerms* is taken by computing the average of payment terms from historical data.
The formula is the same of Remediation 1 for the error 'the value of paymentTerms does not belong to the tolerance interval [3,14]'.

Error: null value in CC_receive order request or FA_send installation notification.
PPI: Percentage of orders fulfilled with perfect lead time.

Remediation 1: t(FA_send installation notification) = t(CC_receive order request) + avgDistance, t(CC_receive order request) = t(FA_send installation notification) -avgDistance where Distance = t(FA_send installation notification) – t(CC_receive order request) (value = 4).

```
for (i in 1:4999) { a <- data1$time[data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request']
 if(!identical(a,character(0))) { if(!is.na(a)) {
 b = data1$time[data1$case == i & data1$event == 'message_sendInstallationConfirmToSA']
 d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") + lubridate::days(4))
 e = as.POSIXct(d, origin = "1970-01-01")
 f = as.character(e)
 if(!identical(b,character(0))) { data1$time[data1$time == '' & data1$case == i & data1$event == 'message_sendInstallationConfirmToSA'] <- f }
if(!identical(b,character(0))) { data1$time[is.na(data1$time) & data1$case == i & data1$event == 'message_sendInstallationConfirmToSA'] <- f }}}}
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS") }
```

```
for (i in 1:4999) { a <- data1$time[data1$case == i & data1$event == 'event_MESSAGE_payment receipt']
if(!identical(a,character(0))) { if(!is.na(a)) {
b = data1$time[data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request']
d = (as.POSIXct(a, format = "%Y-%m-%d %H:%M:%OS") - lubridate::days(4))
e = as.POSIXct(d, origin = "1970-01-01")
f = as.character(e)
if(!identical(b,character(0))) { data1$time[data1$time == '' & data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request'] <- f }
if(!identical(b,character(0))) { data1$time[is.na(data1$time)  & data1$case == i & data1$event == 'event_MESSAGE_CC_receive order request'] <- f
}}}}
for (i in 1:nrow(data1)) { strptime(data1[i,'time'], "%Y-%m-%d %H:%M:%OS") }
```

Remediation 2: T(FA_send installation notification) = diff(t(act_succ), t(act_prec))/2 + t(act_prec).The same applies for the timestamp of CC_receive order request activity

```
tog = sqldf("SELECT [case] FROM data1 WHERE [event] = 'message_sendInstallationConfirmToSA' AND [time] IS NULL")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case']
```

```
rn <- which(data1$case == cs & data1$event =='message_sendInstallationConfirmToSA')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'message_sendInstallationConfirmToSA'] <- f
}

tog = sqldf("SELECT [case] FROM data1 WHERE [event] = 'event_MESSAGE_CC_receive order request' AND [time] IS NULL")
for (i in 1:nrow(tog)) { strptime(tog[i,'time'], "%Y-%m-%d %H:%M:%OS") }
for (i in 1:nrow(tog)) {
cs <- tog[i, 'case']
rn <- which(data1$case == cs & data1$event =='event_MESSAGE_CC_receive order request')
prec <- data1[rn-1, 'time']
succ <- data1[rn+1, 'time']
x <- as.POSIXct(prec, format = "%Y-%m-%d %H:%M:%OS")
y <- as.POSIXct(succ, format = "%Y-%m-%d %H:%M:%OS")
mid = y-x
interv = (as.integer(mid))
ino <- interv/2
ini <- as.integer(ino)
d <- x + lubridate::seconds(ini)
e = as.POSIXct(d, format = "%Y-%m-%d %H:%M:%OS", origin = "1970-01-01")
f = as.character(e)
data1$time[data1$case == cs & data1$event == 'event_MESSAGE_CC_receive order request'] <- f
}
```