**Billing Widget Enhancement**

# Purpose and Summary

The purpose of this functional specification is to guide in the development of a series of enhancements and additions to the Billing Widget in RealTrack:

New cumulated curves to be added to the two already existing in the **Billing Chart**, after which there will be 7 curves in total, which can be shown / hidden at will:

1. Forecast Billing: **existing** *(even if source is different between TPS and OFE we keep the original mapping and set up)*
2. Actual Billing: **existing** *(renamed from current “Cumulated Actual Billing”)*
3. Current Due Invoices: **new**
4. Past Due Invoices: **new**
5. Invoice Payments: **new**
6. Oracle Blank Forecast date: **existing**
7. Actual+P6 Forecast billing: **new** and TPS only

Addition of 4 new **Billing Reports** and modification of the existing one. All of them should be downloadable to excel:

1. Modification to the existing “Milestones” report
2. Addition of an “Invoices” report
3. Addition of a “Current Dues” report
4. Addition of a “Past Dues” report
5. Addition of a “Payments” report
6. Addition of a “Billing Look Ahead” report

Addition of 7 **KPIs** to the widget header and to the “KEY INDICATORS” in the Portfolio section of RealTrack, as well as a “Trends” tab to display how each of these KPIs are trending through monthly storage:

1. Weighted Avg Cycle Time to Invoice
2. Weighted Avg Delinquent Days
3. # Overdue Milestones
4. # Completed but not Billed Milestones
5. Not Billed balance
6. Current Due balance
7. Past Due balance
8. Collected balance
9. Billed Balance

# Applicability to the Product Companies

|  |  |  |
| --- | --- | --- |
| **BILLING CHART** | **OFE** | **TPS** |
| Forecast Billing: **existing** *renamed from current “Cumulated Forecast Billing”* | Y | Y |
| Actual Billing: **existing** *(renamed from current “Cumulated Actual Billing”)* | Y | Y |
| Current Due Invoices: **new** | Y | N |
| Past Due Invoices: **new** | Y | N |
| Invoice Payments: **new** | Y | Y\* |
| Oracle Blank Forecast date: **existing** | N | Y |
| Actual+P6 Forecast billing: **new** | N | Y |
|  |  |  |
| **BILLING REPORTS/TABS** | **OFE** | **TPS** |
| 1.       Modification to the existing “Milestones” report | Y | Y |
| 2.       Addition of an “Invoices” report | Y | N |
| 3.       Addition of a “Current Dues” report | Y | N |
| 4.       Addition of a “Past Dues” report | Y | N |
| 5.       Addition of a “Payments” report | Y | Y\* |
| 6.       Addition of a “Billing Look Ahead” report | Y° | Y |
|  |  |  |
| **KPIs and TRENDS** | **OFE** | **TPS** |
| 1.       Weighted Avg Cycle Time to Invoice | Y | Y |
| 2.       Weighted Avg Delinquent Days | Y | Y |
| 3.       # Overdue Milestones | Y | N |
| 4.       # Completed but not Billed Milestones | Y | N |
| 5.       Not Billed balance | Y | ? |
| 6.       Current Due balance | Y | Y\* |
| 7.       Past Due balance (***existing*** *in the KPI widget*) | Y | N |
| 8.       Collected balance | Y | Y |
| 9. Billed Balance | tbc | Y |

Y\* = implement only if GECARS data will be available

Y° = keep it hidden for OFE as data quality will be too poor at first

# Detailed Description of the Billing Chart

Fig. 1 below shows the existing Billing Chart in RealTrack, with only the first 2 cumulated curves out of the 5 required ones, while Fig. 2 shows an excel version with all 5 of them:

1. Forecast Billing: **existing.** Even if source is different between TPS (Oracle) and OFE (P6) we keep the original mapping and set up. Changes: Name from Cum. Forecast Billing to Forecast Billing for TPS and OFE.
2. Actual Billing: **existing** . Changes: Name from Cum. Actual Billing to Actual Billing for TPS and OFE.
3. Current Due Invoices: **new** and OFE only
4. Past Due Invoices: **new** and OFE only
5. Invoice Payments: **new**
6. Oracle Blank Forecast date: **existing.** Changes: remove from OFE as it is not applicable.
7. P6 Forecast billing: **new** and TPS only *(despite the simplified name it is a combination of Actual, P6 and Oracle dates for Milestones)*

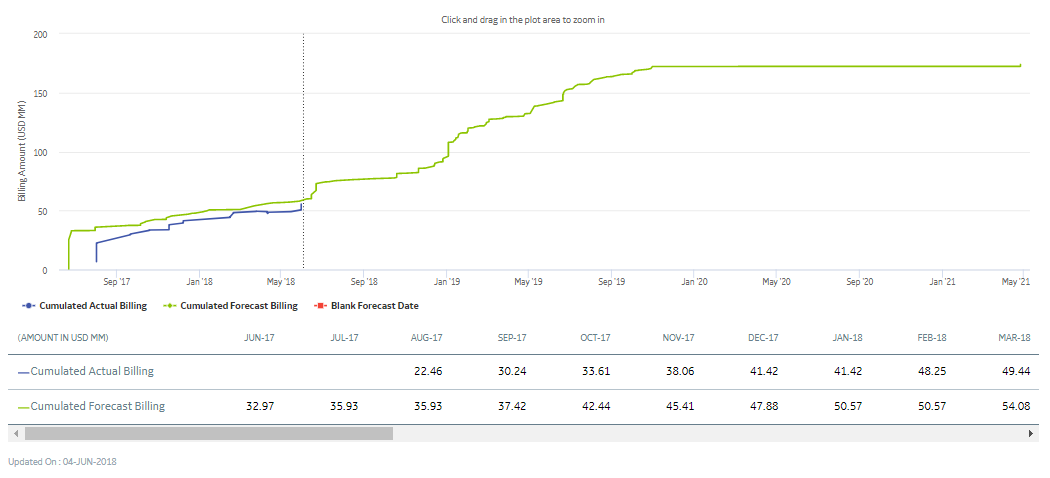


Fig. 1 – Existing Billing Chart in RealTrack

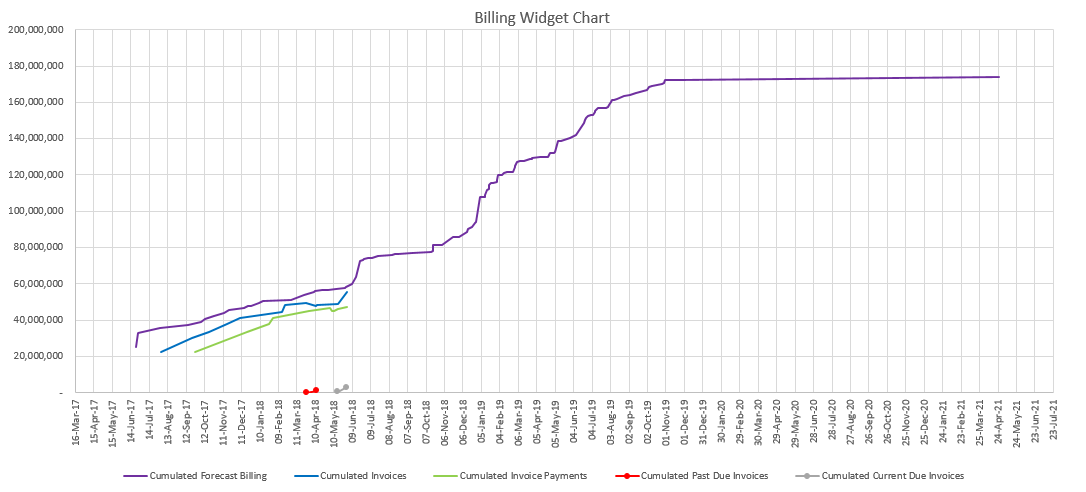


Fig. 2 – **OFE** - Excel version of the To-Be Billing Chart in RealTrack

**TPS - P6 Forecast billing curve**

The curve is built by plotting all milestones captured from Oracle on the chart and using the Actual Invoice Date for milestones where the invoice exists, and the P6 Forecast Date for milestones where the invoice does not exist. In addition, if milestones do not have a P6 Forecast Date the Oracle Forecast date shall be used for plotting the curve.

By sorting from oldest to newest date, the milestone amount is accumulated and drawn on a chart similarly to the existing curve.

The curve shall be dotted; the Actual curve shall be in front of the P6 curve.

Refer to the file for an example : 

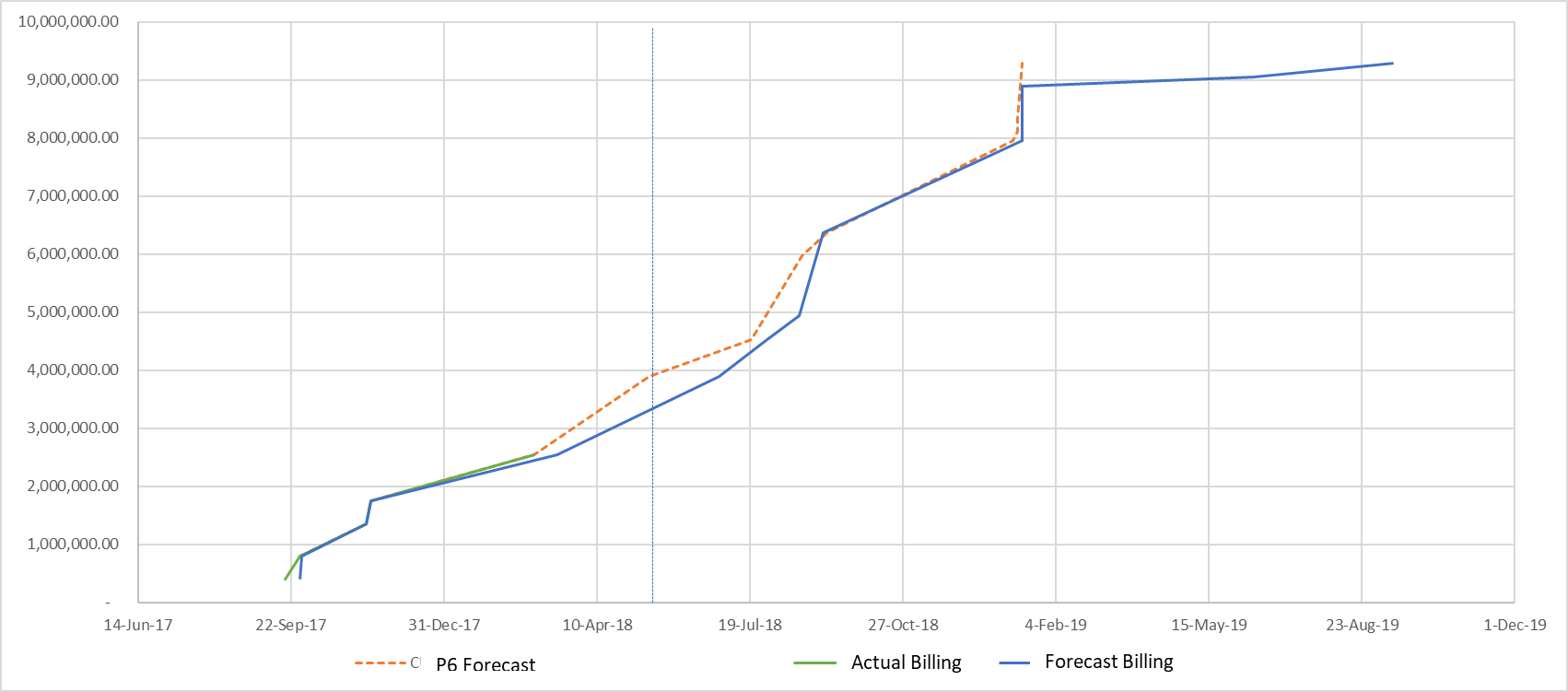


Fig. 3 – **TPS** - Excel version of the To-Be Billing Chart in RealTrack (no Blank forecast for the project)

**CURRENCY** of the curves:

TPS:

All curves, small and large pop-ups must be expressed in USD, this is a change from current.

One Project (Mother Job. M\_xxxxx) has many Contract numbers below it and can have different currencies. We will need to apply the Swap rate if currency <> EUR and all amounts will be converted into USD at the OP rate of the current year (as it is for the contract amount). In the existing Billing Report the value is already converted into EURO at the Swap rate, that amount needs to be converted once more at the OP rate.

Milestone Amount in USD: new data point ‘MILESTONE\_AMNT\_USD’ 🡪 if CURRENCY=EUR -> 'UNIT\_PRICE'\*OP RATE, if CURRENCY<>EUR -> 'UNIT\_PRICE'&'CURRENCY'\*Swap Rate\*OP rate

OFE:

All curves, small and large pop-ups must be expressed in the P6 CONTRACT\_CURRENCY, which is the currency of the Cumulated Forecast Billing chart. This means that, just like RealTrack currently does for the Cumulated Invoices curve, the values of all the following curves must be converted to the P6 CONTRACT\_CURRENCY using GE MOR Rates: (*Note: the queries shared in this specification do not contain this conversion, but it is expected that such conversion be implemented*)

* + Cumulated Invoices @ INVOICE\_DATE
  + Cumulated Current Due Invoices @ Inv DT
  + Cumulated Past Due Invoices @ Inv DT
  + Cumulated Invoice Payments @ Inv DT
* Being able to mouse over the curves and see the marker move to the next closest step change, showing date and cumulated value on a small pop-up.
* Being able to click on the marker and see a larger pop-up with more information.
* Other chart features: vertical line marking the last update date and zoom / reset capacity.

New feature to be considered if feasible: Find a way to graphically link or highlight the points in the various charts that are “logically connected” to each other, when the user mouses over (or clicks, as an alternative) one specific point in any of the 5 curves. Example: if the user mouses over (or clicks, as an alternative) a specific billing milestone on the Cumulated Forecast Billing curve, the connection to the corresponding invoice / current due / past due / payment(s) would be shown somehow (through marked up dots or connecting dashed lines, or any other possible means).

**DATA TABLE:**

Maintain as it is. Build the additional one for the P6 Forecast curve.

**POP UPS:**

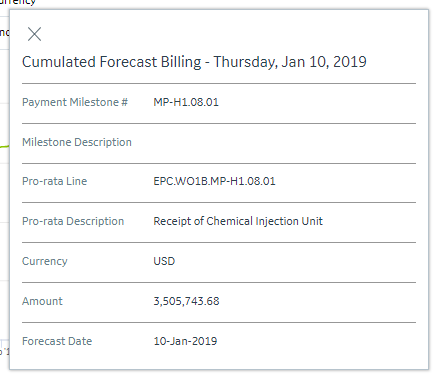
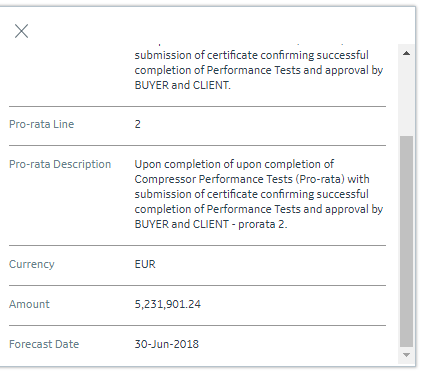
All current features of the 2 cumulated curves are to be maintained, and to be extended to the new ones:

Small pop-ups: all small pop-ups, which appear on mouse over, will show the date and the cumulated value of the curve being moused over. This will be available in all 5 curves.

Large pop-ups: what follows is a list of the fields to be displayed in large pop-ups upon clicking on markers of each of the 5 curves. Please refer to the next section and to the Data Sources section for a description of the following reports: Milestones / Invoices / Current Dues / Past Dues / Payments.

1. Forecast Billing: show as currently the name of the curve and the date, plus these 7 fields:

Current OFE: Current TPS:

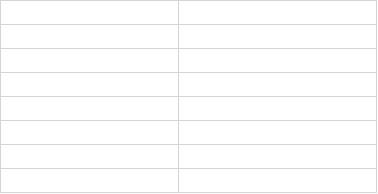
 

|  |  |  |  |
| --- | --- | --- | --- |
| Caption | Current RT label | Field Name in “Milestones” report OFE | TPS |
| Contract ID | N/A | n/a | CONTRACT\_NUMBER |
| Milestone identification | Payment Milestone # | Milestone Id & “ – “ & Milestone | LINE\_NUMBER |
| Milestone identification | Milestone Description | n/a | MILESTONE\_DESCRIPTION |
| Milestone identification | Pro-rata Line | n/a | PAYMENT\_MILESTONE\_LINE |
| Milestone identification | Pro-rata Description | Activity Id | PAYMENT\_MILESTONE\_DESCRIPTION |
| Currency | Currency |  | Currency |
| Value | Amount | Milestone Value | UNIT\_PRICE |
| Amount in USD | N/A | n/a | MILESTONE\_AMNT\_USD |
| Completion Date | ACTUAL INVOICE DATE in the Milestones Tab | Milestone Completion | NA |
| Status | N/A – This is a calculated field | Milestone Status | 'if INVOICE<>"", then 'Billed'. if INVOICE="", then if PLANNED\_FINISH\_DATE > INVOICE\_FORECAST\_DATE = 'Pull in' and green; else 'Delay and yellow |
| P6 Forecast date | N/A | n/a | PLANNED\_FINISH\_DATE , show only if Status = ‘To Bill’, else “” |
| No. of days | N/A | n/a | PLANNED\_FINISH\_DATE - INVOICE\_FORECAST\_DATE |
| Days to prepare Invoice | N/A | Days to prepare Invoice | n/a |
| Invoice | N/A | Inv & “ – “ & Inv Line & “ – “ & Inv Desc | INVOICE |

Example new Large pop-up OFE: Example New Large pop-up TPS:

New Large pop-up OFE w/data points and logic: New Large pop-up TPS with data point and logic:

1. Actual billing: show as currently the name of the curve and the date, plus the below fields:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caption | OFE | OFE comments | TPS | Current RT label |
| Invoice | Inv & “ – “ & Inv Line | Now is Invoice no. though it lacks the Inv Line, from the Cumulated Actual Billing large pop-up) | INVOICE\_NUMBER | Invoice Number |
| Inv Description | Inv Desc | (please note this is actually the Invoice Description, but it is shown as Milestone Description, it is mislabeled) | n/a | Milestone Description |
| Date | Inv Date |  | INVOICE\_DATE | Invoice Date |
| Inv Amount | Inv Amount |  | UNIT\_PRICE | Amount |
| Currency |  |  | CURRENCY |  |
| Amount in USD |  |  | MILESTONE\_AMNT\_USD | N/A |
| Status | Inv Outs Status & “ – “ & Paym Status |  | Milestone Status | N/A |
| Outs Amount | Outs Amount |  | N/A tbc | N/A |
| Milestone # | Milestone Id & “ – “ & Milestone |  | LINE\_NUMBER | Payment Milestone # |
| Milestone Descr. |  | n/a | MILESTONE\_DESCRIPTION | Milestone Description |
| Pro-rata Line |  | n/a | PAYMENT\_MILESTONE\_LINE | Pro-rata Line |
| Pro-rata descr. |  | n/a | PAYMENT\_MILESTONE\_DESCRIPTION | Pro-rata descr. |

1. Current Due Invoices: **OFE only**. Show the name of the curve and the date, plus these 7 fields:

|  |  |  |
| --- | --- | --- |
| Caption | Field Name in “Current Dues” report | Current RT label |
| Invoice | Inv & “ – “ & Inv Line & “ – “ & Inv Desc | Invoice Number (though it lacks the Inv Line, from the Cumulated Actual Billing large pop-up) |
| Due Date | Due Date | N/A |
| Inv Amount | Inv Amount | Amount (from the Cumulated Actual Billing large pop-up) |
| Outs Amount | Outs Amount | N/A |
| Days PD | Days PD | N/A |
| Open Stat Desc | Open Stat Desc | N/A |
| Milestone | Milestone Id & “ – “ & Milestone | Payment Milestone # concatenated with Milestone Description, from the Milestones Tab |

1. Cumulated Past Due Invoices: **OFE only** . show the name of the curve and the date, plus these 7 fields:

|  |  |  |
| --- | --- | --- |
| Caption | Field Name in “Past Dues” report | Current RT label |
| Invoice | Inv & “ – “ & Inv Desc | Invoice Number (though it lacks the Inv Line) |
| Due Date | Due Date | N/A |
| Inv Amount | Inv Amount | Amount (from the Cumulated Actual Billing large pop-up) |
| Outs Amount | Outs Amount | N/A |
| Days PD | Days PD | N/A |
| Open Stat Desc | Open Stat Desc | N/A |
| Milestone | Milestone Id & “ – “ & Milestone | Payment Milestone # concatenated with Milestone Description, from the Milestones Tab |

1. Cumulated Invoice Payments: show the name of the curve and the date, plus these 7 fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Caption | Field Name in “Payments” report OFE | Field Name TPS | Current RT label |
| Invoice | Inv & “ – “ & Inv Desc | INVOICE | Invoice Number (though it lacks the Inv Line) |
| Due Date | Due Date | DUE DATE | N/A |
| Inv Amount | Inv Amount | n/a | Amount (from the Cumulated Actual Billing large pop-up) |
| Amount in USD |  | MILESTONE\_AMNT\_USD | N/A |
| Paid Amount | Paid Amount | *Same data as OFE, but all to be in USD at OP rate* | N/A |
| Rem Amount | Rem Amount | *Same data as OFE, but all to be in USD at OP rate* | N/A |
| Paym Days PD | Paym Days PD | *Same data as OFE, but all to be in USD at OP rate* | N/A |
| Milestone identification | Milestone Id & “ – “ & Milestone | LINE\_NUMBER | Payment Milestone # |
| Milestone identification | n/a | MILESTONE\_DESCRIPTION | Milestone Description |
| Milestone identification | n/a | PAYMENT\_MILESTONE\_LINE | Pro-rata Line |
| Milestone identification | n/a | PAYMENT\_MILESTONE\_DESCRIPTION | Pro-rata Description |

# Detailed Description of the Billing Tabs

These are the five Billing Tabs to show in the Billing Widget, in tabs next to the Billing Chart. Please note all of them should allow downloads to excel:

1. “Milestones” tab
2. “Invoices” tab
3. “Current Dues” tab
4. “Past Dues” tab
5. “Payments” tab

We will go over each of them .

1. “Milestones”: – the Milestone table can be used for data mapping. For OFE please refer to the “MilestonesReport” query in the Data Sources section. Below is a description of the columns in this report

| **Column** | **OFE** | **TPS** | **TPS comments** | **Current RT label** |
| --- | --- | --- | --- | --- |
| Contract number | n/a | CONTRACT\_NUMBER | **New to add** | N/A |
| Milestone Id | IPP\_PAYMENT\_MILESTONE\_CODE in the Milestones table | LINE\_NUMBER |  | PAYM. MILESTONE # in the Milestones Tab |
| Milestone Descr | IPP\_PAYMENT\_MILESTONE\_DESC in the Milestones table | MILESTONE\_DESCRIPTION |  | MILESTONE DESCRIPTION in the Milestones Tab |
| Milestone Deliverable (Activity) | ACTIVITY\_ID in the Milestones table | PAYMENT\_MILESTONE\_LINE |  | PRO-RATA LINE in the Milestones Tab |
| Milestone Deliverable Descr (Activity descr) | n/a | PAYMENT\_MILESTONE \_DESCRIPTION |  | Pro-rata Description |
| P6 Milestone ID | n/a | CASH\_ MILESTONE \_ACTIVITY\_ID | In P6 the activity ID is obtained by concatenating Milestones Ids and Contract number from Oracle: CONTRACT\_NUMBER-LINE\_NUMBER-PAYMENT\_MILESTONE\_LINE. So, unify the 3 fields for obtaining a unique reference between Oracle data and P6 data. Not to show on UI. | N/A |
| Milestone Value | IPP\_PAYMENT\_MILESTONE\_VALUE in the Milestones table | UNIT\_PRICE |  | AMOUNT in the Milestones Tab |
| Milestone Curr | CONTRACT\_CURRENCY in the Milestones table | CURRENCY |  | CURR. in the Milestones Tab |
| Milestone Amount in USD | na | MILESTONE\_AMNT\_USD | **New to add** . Calculated: if CURRENCY=EUR -> 'UNIT\_PRICE'\*OP RATE, if CURRENCY<>EUR -> 'UNIT\_PRICE' & 'CURRENCY'\*Swap Rate\*OP rate | N/A |
| Milestone Forecast | FINISH\_DATE in the Milestones table | INVOICE\_FORCAST\_DATE |  | FORECAST INVOICING DATE on the Milestones Tab |
| P6 Milestone Forecast | n/a | PLANNED\_FINISH\_DATE | **New to add** . Show it only for milestones where Invoice = " " | N/A |
| Combined Actual+Oracle+P6 Forecast | n/a | Calculated: if INVOICE\_DATE<>"", INVOICE\_DATE, else If CASH\_MILESTONE\_ACTIVITY\_ID does not exist, INVOICE\_FORECAST\_DATE, else PLANNED\_FINISH\_DATE | Not to show on UI. use it for P6 Forecast Billing | N/A |
| Milestone Y-Q | Calculated field based on FINISH\_DATE from the Milestones table:  Year([FINISH\_DATE]) & "-" & IIf(Month([FINISH\_DATE])>9, "Q4", IIf(Month([FINISH\_DATE])>6 ,"Q3",IIf(Month([FINISH\_DATE]) >3,"Q2","Q1"))) | ??? | Not sure how this field is used | N/A |
| Milestone Completion | ACTUAL\_FINISH in the Milestones table | n/a | maybe the current mapping is pointing to the Invoice date here for TPS, please check | N/A |
| Milestone Status | Calculated field based on ACTUAL\_FINISH from the Milestones table and Inv from InvoicesQuery:  IIf(IsNull([Inv]),IIf(Not IsNull([ACTUAL\_FINISH]),"Completed, not Billed", IIf([FINISH\_DATE]<[LAST\_UPDATE\_DATE], "Overdue","Forecasted")), "Billed") | Calculated: keep the current logic for TPS |  | Calculated , logic is different between OFE and TPS |
| Days Not Billed | Calculated field based on ACTUAL\_FINISH and LAST\_UPDATE\_DATE from the Milestones table and Inv from InvoicesQuery:  IIf(Not IsNull([ACTUAL\_FINISH]) And IsNull([Inv]),DateDiff("d" ,[ACTUAL\_FINISH], [LAST\_UPDATE\_DATE]),Null) | n/a |  | N/A |
| Days to prepare Invoice | Calculated field based on FINISH\_DATE, ACTUAL\_FINISH and LAST\_UPDATE\_DATE from the Milestones table and Inv from InvoicesQuery:  IIf(IsNull([Inv]) And IsNull([ACTUAL\_FINISH]), DateDiff("d", [LAST\_UPDATE\_DATE],[FINISH\_DATE]),Null) | PLANNED\_FINISH\_DATE – sys\_date , where INVOICE = " " | Not to show on UI. P6 Forecast Date - Sys date for unbilled milestones | N/A |
| Days taken to issue Invoice | Calculated field based on ACTUAL\_FINISH from the Milestones table and Inv Date from InvoicesQuery:  IIf(Not IsNull([ACTUAL\_FINISH]) And Not IsNull([Inv]),DateDiff("d" ,[ACTUAL\_FINISH],[Inv Date]),Null) | If Invoice<>"" , PLANNED\_FINISH\_DATE - INVOICE\_DATE | Not to show on UI. Use it for KPIs only | N/A |
| Inv | Inv from InvoicesQuery | INVOICE | Invoice number | Invoice Nr |
| Inv Line | Inv Line from InvoicesQuery | na |  | N/A |
| Inv Date | Inv Date from InvoicesQuery | INVOICE\_DATE |  | Actual Invoice Date |
| Inv Outs Status | Inv Outs Status from InvoicesQuery | n/a |  | N/A |
| Inv Term | Inv Term from InvoicesQuery |  | NEED TO GATHER THE DATA FROM Oracle ? Or does it exist in GECARS | N/A |
| Due Date | Due Date from InvoicesQuery |  | NEED TO GATHER THE DATA FROM Oracle ? Or does it exist in GECARS | N/A |
| Inv Amount | Inv Amount from InvoicesQuery | na |  | OFE: Amount from Cumulated Actual Billing |
| Inv Desc | Inv Desc from InvoicesQuery | na |  | Milestone Description (for OFE this is a wrong labeling, as this is not a Milestone, but an Invoice) |
| Paym Status | Paym Status from InvoicesQuery |  |  | N/A |
| Outs Amount | Outs Amount from InvoicesQuery |  |  | N/A |
| Days PD | Days PD from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| Last Paym Date | Last Paym Date from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| Last Paym Type | Last Paym Type from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| Last Paym Days PD | Last Paym Days PD from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| Last Paid Amount | Last Paid Amount from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| TotalPaymentToDate | TotalPaymentToDate from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |
| Last Rem Amount | Last Rem Amount from InvoicesQuery | if source is GECARS Pedro's query should be valid for TPS as well |  | N/A |

1. “Invoices” tab: **OFE only**. Please refer to the “InvoicesReport” query in the Data Sources section. Below is a description of the columns in this report:

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Current RT label** |
| Inv | Inv from InvoicesQuery | Invoice Number from Cumulated Actual Billing |
| Inv Line | Inv Line from InvoicesQuery | N/A |
| Inv Desc | Inv Desc from InvoicesQuery | Milestone Description from Cumulated Actual Billing (please note this is a wrong labeling, as this is not a Milestone, but an Invoice) |
| Days taken to issue Invoice | Calculated field based on ACTUAL\_FINISH from the Milestones table and Inv Date from InvoicesQuery:  IIf(Not IsNull([ACTUAL\_FINISH]) And Not IsNull([Inv]),DateDiff("d",[ACTUAL\_FINISH],[Inv Date]),Null) | Calculated field based on the ACTUAL INVOICE DATE on the Milestones Tab (please be aware this label is incorrect for this field, it is not an Invoice Date but the completion date of the Milestone, not the invoice), the Invoice Number from Cumulated Actual Billing, and the Invoice Date from Cumulated Actual Billing |
| Inv Date | Inv Date from InvoicesQuery | Invoice Date from Cumulated Actual Billing |
| Inv Y-Q | Inv Y-Q from InvoicesQuery | Calculated field based on Invoice Date from Cumulated Actual Billing |
| Inv Term | Inv Term from InvoicesQuery | N/A |
| Due Date | Due Date from InvoicesQuery | N/A |
| Inv Amount | Inv Amount from InvoicesQuery | Amount from Cumulated Actual Billing |
| Inv Outs Status | Inv Outs Status from InvoicesQuery | N/A |
| Paym Status | Paym Status from InvoicesQuery | N/A |
| Outs Amount | Outs Amount from InvoicesQuery | N/A |
| Days PD | Days PD from InvoicesQuery | N/A |
| Last Paym Date | Last Paym Date from InvoicesQuery | N/A |
| Last Paym Days PD | Last Paym Days PD from InvoicesQuery | N/A |
| Last Paid Amount | Last Paid Amount from InvoicesQuery | N/A |
| TotalPaymentToDate | TotalPaymentToDate from InvoicesQuery | N/A |
| Last Paym Type | Last Paym Type from InvoicesQuery | N/A |
| Last Rem Amount | Last Rem Amount from InvoicesQuery | N/A |
| Milestone Id | IPP\_PAYMENT\_MILESTONE\_CODE from Milestones table | PAYM. MILESTONE # in the Milestones Tab |
| Milestone | IPP\_PAYMENT\_MILESTONE\_DESC from Milestones table | MILESTONE DESCRIPTION in the Milestones Tab |
| Activity Id | ACTIVITY\_ID from Milestones table | PRO-RATA LINE in the Milestones Tab |
| Milestone Value | IPP\_PAYMENT\_MILESTONE\_VALUE from Milestones table | AMOUNT in the Milestones Tab |
| Milestone Curr | CONTRACT\_CURRENCY from Milestones table | CURR. in the Milestones Tab |
| Milestone Forecast | FINISH\_DATE from Milestones table | FORECAST INVOICING DATE on the Milestones Tab |
| Milestone Completion | ACTUAL\_FINISH from Milestones table | ACTUAL INVOICE DATE on the Milestones Tab (please be aware this label is incorrect for this field, it is not an Invoice Date but the completion date of the Milestone, not the invoice) |

1. “Current Dues” tab: **OFE only**. Please refer to the “CurrentDuesReport” query in the Data Sources section. Below is a description of the columns in this report:

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Current RT label** |
| Inv | INVOICE NUMBER from OutsInvoices | Invoice Number from Cumulated Actual Billing |
| Inv Desc | Inv Desc from Invoices | Milestone Description from Cumulated Actual Billing (please note this is a wrong labeling, as this is not a Milestone, but an Invoice) |
| Inv Date | Inv DT from OutsInvoices | Invoice Date from Cumulated Actual Billing |
| Inv Y-Q | Calculated field based on Inv DT from OutsInvoices:  Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))) | N/A (please note it should be based on the Inv DT as read from OutsInvoices, not on Invoice Date from Cumulated Actual Billing) |
| Inv Term | TERM\_NAME from Invoices | N/A |
| Due Date | Due DT from OutsInvoices | N/A |
| Days PD | Calculated field based on Due DT and LOAD\_DT from OutsInvoices:  DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) | N/A |
| Open Stat Desc | Open Stat Desc from OutsInvoices | N/A |
| Open Stat Dt | Open Stat Dt from OutsInvoices | N/A |
| Inv Amount | INV\_AMOUNT from Invoices | Amount from Cumulated Actual Billing |
| Outs Amount | GECARS AMOUNT from OutsInvoices | N/A |
| Paym Status | Calculated field based on RemainingAmount and Inv Amt from LastPayments:  IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)) | N/A |
| Last Paym Date | PaymentDate from LastPayments | N/A |
| Last Paym Days PD | GECARS Days PD from LastPayments | N/A |
| Last Paid Amount | PaymentAmount from LastPayments | N/A |
| TotalPaymentToDate | TotalPaymentToDate from LastPayments | N/A |
| Last Paym Type | PaymentType from LastPayments | N/A |
| Last Rem Amount | RemainingAmount from LastPayments | N/A |

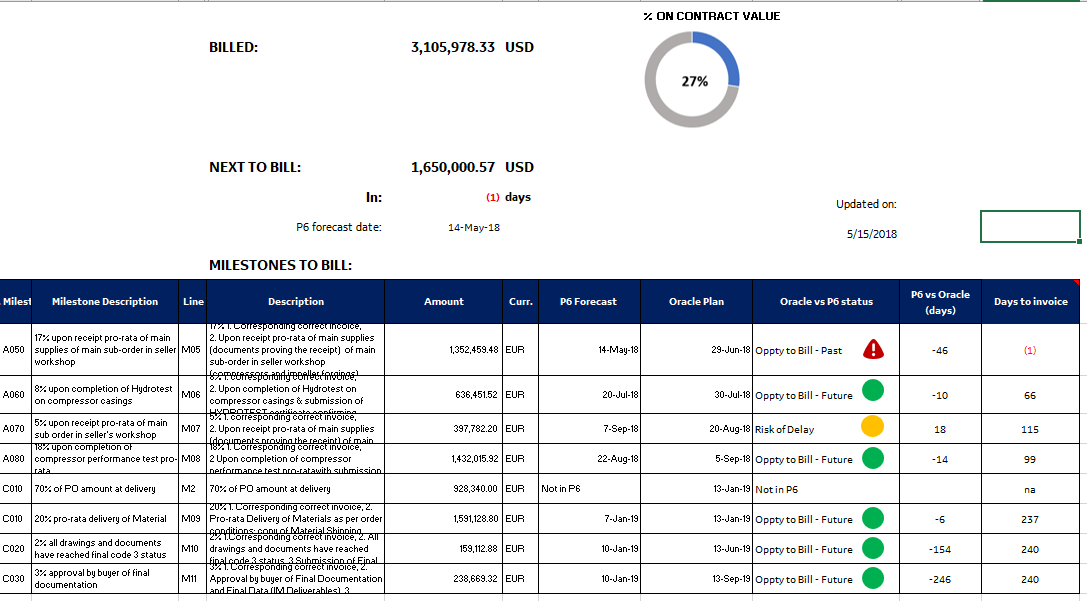
1. “Past Dues” report: **OFE only**. Please refer to the “PastDuesReport” query in the Data Sources section. Below is a description of the columns in this report:

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Current RT label** |
| Inv | INVOICE NUMBER from OutsInvoices | Invoice Number from Cumulated Actual Billing |
| Inv Desc | Inv Desc from Invoices | Milestone Description from Cumulated Actual Billing (please note this is a wrong labeling, as this is not a Milestone, but an Invoice) |
| Inv Date | Inv DT from OutsInvoices | Invoice Date from Cumulated Actual Billing |
| Inv Y-Q | Calculated field based on Inv DT from OutsInvoices:  Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))) | N/A (please note it should be based on the Inv DT as read from OutsInvoices, not on Invoice Date from Cumulated Actual Billing) |
| Inv Term | TERM\_NAME from Invoices | N/A |
| Due Date | Due DT from OutsInvoices | N/A |
| Days PD | Calculated field based on Due DT and LOAD\_DT from OutsInvoices:  DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) | N/A |
| Open Stat Desc | Open Stat Desc from OutsInvoices | N/A |
| Open Stat Dt | Open Stat Dt from OutsInvoices | N/A |
| Inv Amount | INV\_AMOUNT from Invoices | Amount from Cumulated Actual Billing |
| Outs Amount | GECARS AMOUNT from OutsInvoices | N/A |
| Paym Status | Calculated field based on RemainingAmount and Inv Amt from LastPayments:  IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)) | N/A |
| Last Paym Date | PaymentDate from LastPayments | N/A |
| Last Paym Days PD | GECARS Days PD from LastPayments | N/A |
| Last Paid Amount | PaymentAmount from LastPayments | N/A |
| TotalPaymentToDate | TotalPaymentToDate from LastPayments | N/A |
| Last Paym Type | PaymentType from LastPayments | N/A |
| Last Rem Amount | RemainingAmount from LastPayments | N/A |

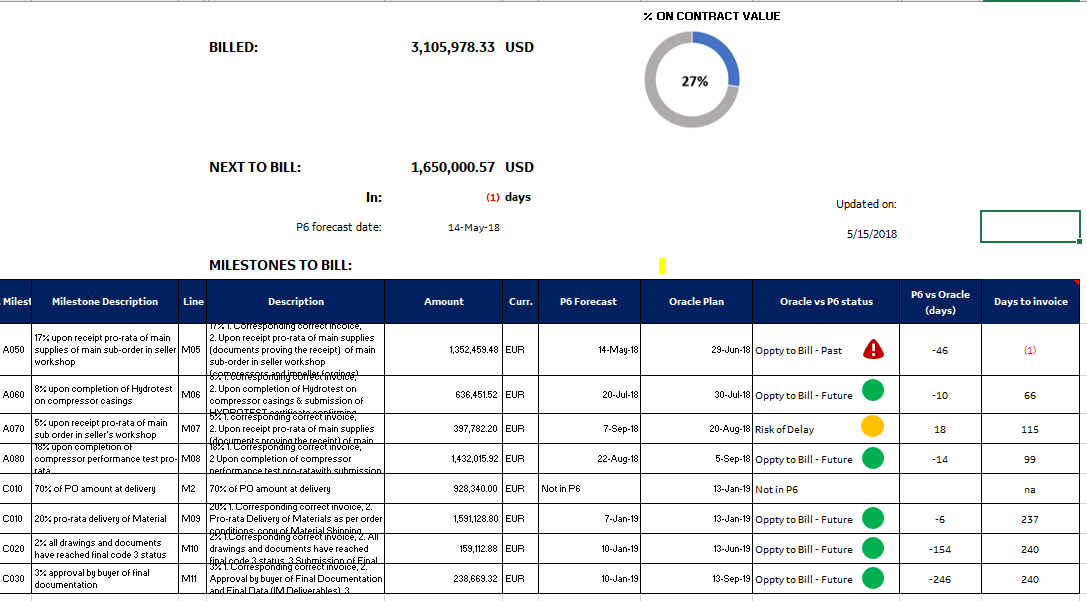
1. “Payments” report: **TPS tbc**. Please refer to the “PaymentsReport” query in the Data Sources section. Below is a description of the columns in this report:

|  |  |  |
| --- | --- | --- |
| **Column** | **Description** | **Current RT label** |
| Inv | INVOICE NUMBER from Payments | Invoice Number from Cumulated Actual Billing |
| Inv Desc | Inv Desc from Invoices | Milestone Description from Cumulated Actual Billing (please note this is a wrong labeling, as this is not a Milestone, but an Invoice) |
| Inv Amount | INV\_AMOUNT from Invoices | Amount from Cumulated Actual Billing |
| Paym Status | Calculated field based on Remaining Amount and Inv Amt from Payments:  IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[Payments].[Inv Amt],"Partially Paid",Null)) | N/A |
| Inv Outs Status | OutsStatus from OutsInvoices | N/A |
| Outs Amount | GECARS AMOUNT from OutsInvoices | N/A |
| Days PD | Calculated field based on Due DT and LOAD\_DT from OutsInvoices:  DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) | N/A |
| Inv DT | Inv DT from Payments | N/A (please note this is not Invoice Date from Cumulated Actual Billing, as it should be read from Payments) |
| Inv Term | TERM\_NAME from Invoices | N/A |
| Due Date | Due DT from Payments | N/A |
| PaymentId | PaymentId from Payments | N/A |
| Paym Date | PaymentDate from Payments | N/A |
| Paym Days PD | Calculated field based on Due DT and PaymentDate from Payments:  DateDiff("d",[Payments].[Due DT],[PaymentDate]) | N/A |
| Paym Type | PaymentType from Payments | N/A |
| Paid Amount | PaymentAmount from Payments | N/A |
| Rem Amount | RemainingAmount from Payments | N/A |

1. “Billing Look Ahead” tab: **OFE tbc**. It is a summary view of the current and next billing plus a table as subset of the Milestone table and some additional calculated fields. The table needs to be immediately below the summary view.



SUMMARY VIEW:



|  |  |  |  |
| --- | --- | --- | --- |
| **Column/Data point** | **Description** | **OFE** | **TPS** |
| Billed | Amount billed as of today (sys date) in USD |  | Sum USD AMOUNT (OP RATE) where INVOICE\_DATE <>”” |
| CURRENCY |  |  | USD AMOUNT (OP RATE) |
| % on Contract Value |  |  | Billed / Contract Value USD % |
| NEXT TO BILL | Next unbilled milestone amount based on P6 forecast |  | UNIT\_PRICE where min P6 Forecast date and Invoice date = “” |
| In: | How many days between P6 date for next billing milestone and sys date |  | (min P6 Forecast date where Invoice date = “”) – sys\_date.  *Show negative values in red color* |
| P6 Forecast date | Forecast date pushed by the predecessors in the Primavera plan. |  | min P6 Forecast date where Invoice date = “” |
| Updated on | Data date of the source |  |  |

MILESTONES TO BILL TABLE



|  |  |  |  |
| --- | --- | --- | --- |
|  | **Column/Data point** | **OFE** | **TPS** |
| 1 | CONTRACT\_NUMBER |  | CONTRACT\_NUMBER |
| 2 | LINE\_NUMBER |  | LINE\_NUMBER |
| 3 | MILESTONE\_DESCRIPTION |  | MILESTONE\_DESCRIPTION |
| 4 | PAYMENT\_MILESTONE\_LINE |  | PAYMENT\_MILESTONE\_LINE |
| 5 | PAYMENT\_MILESTONE\_DESCRIPTION |  | PAYMENT\_MILESTONE\_DESCRIPTION |
| 6 | Amount |  | UNIT\_PRICE |
| 7 | Curr. |  | CURRENCY |
| 8 | Amount in USD (OP Rate) |  | MILESTONE\_AMNT\_USD |
| 9 | P6 Forecast |  | PLANNED\_FINISH\_DATE. If Milestone does not exist in P6, then ‘Not in P6’ |
| 10 | Oracle Forecast |  | INVOICE\_FORECAST\_DATE |
| 11 | Oracle vs P6 Status |  | * If PLANNED\_FINISH\_DATE = ‘Not in P6’, ‘Not in P6’ * If PLANNED\_FINISH\_DATE < INVOICE\_FORECAST\_DATE and PLANNED\_FINISH\_DATE>sys\_date , ‘Oppty to Bill – Future’ . *Green bullet* * If PLANNED\_FINISH\_DATE < INVOICE\_FORECAST\_DATE and PLANNED\_FINISH\_DATE < sys\_date, ‘Oppty to Bill – Past’ . *Red Alert icon* * If PLANNED\_FINISH\_DATE > INVOICE\_FORECAST\_DATE and PLANNED\_FINISH\_DATE > sys\_date, ‘Risk of Delay – Yellow’ . *Yellow bullet.* * If PLANNED\_FINISH\_DATE = INVOICE\_FORECAST\_DATE = sys\_date, ‘Oppty to Bill – Today’ . *Red Alert icon* |
| 12 | Oracle vs P6 days |  | PLANNED\_FINISH\_DATE - INVOICE\_FORECAST\_DATE, where PLANNED\_FINISH\_DATE<>’Not in P6’ |
| 13 | Days to Prepare Invoice |  | PLANNED\_FINISH\_DATE – sys\_date . If negative show in red color. |

In the default view the table has to be sorted by the P6 Forecast Date field , oldest to newest.

# Detailed Description of the Billing KPIs

The following 7 **KPIs** will be added to the widget header and to the “KEY INDICATORS” in the Portfolio section of RealTrack:

1. Weighted Avg Cycle Time to Invoice
2. Weighted Avg Delinquent Days
3. # Overdue Milestones
4. # Completed but not Billed Milestones
5. Not Billed balance
6. Current Due balance
7. Past Due balance
8. Collected balance
9. Billed Balance

We will go over each of them:

1. Weighted Avg Cycle Time to Invoice: This KPI indicates how many days it has taken, on an average weighted by the Milestone Values, to issue the Invoice to the customer in ERP from the Completion of the Billing Milestone in P6. Same logic for TPS and OFE, grouping is by Project ID as it is in RealTrack;

Sum([Days taken to issue Invoice]\*[Milestone Value])/Sum([Milestone Value])

For OFE:

Please follow this SQL in its calculation, which is based on the InvoicesReport query:

SELECT InvoicesReport.MASTER\_PROJECT\_NAME, Sum([Days taken to issue Invoice]\*[Milestone Value])/Sum([Milestone Value]) AS [Weighted Avg Cycle Time to Invoice]

FROM InvoicesReport

GROUP BY InvoicesReport.MASTER\_PROJECT\_NAME

ORDER BY InvoicesReport.MASTER\_PROJECT\_NAME;

1. Weighted Avg Delinquent Days: This KPI indicates how many Past Due days (i.e. days beyond the invoice’s Due Date) it has taken the customer, on an average weighted by the Inv Amount, to pay BHGE. Same logic for TPS and OFE, grouping is by Project ID as it is in RealTrack;

Sum([Paym Days PD]\*[Inv Amount])/Sum([Inv Amount])

For OFE:

Please follow this SQL in its calculation, which is based on the PaymentsReport query:

SELECT PaymentsReport.MASTER\_PROJECT\_NAME, Sum([Paym Days PD]\*[Inv Amount])/Sum([Inv Amount]) AS [Weighted Avg Delinquent Days]

FROM PaymentsReport

GROUP BY PaymentsReport.MASTER\_PROJECT\_NAME

ORDER BY PaymentsReport.MASTER\_PROJECT\_NAME;

1. # Overdue Milestones: **OFE only**. This KPI indicates how many P6 Billing Milestones have gone beyond their Forecast Date without being completed. Please follow this SQL in its calculation, which is based on the MilestonesReport query:

SELECT MilestonesReport.MASTER\_PROJECT\_NAME, Count(MilestonesReport.[Milestone Id]) AS [Milestones - Overdue]

FROM MilestonesReport

WHERE (((MilestonesReport.[Milestone Status])="Overdue"))

GROUP BY MilestonesReport.MASTER\_PROJECT\_NAME

ORDER BY MilestonesReport.MASTER\_PROJECT\_NAME;

1. # Completed but not Billed Milestones: **OFE only**. This KPI indicates how many P6 Billing Milestones have been Completed but not Billed yet. Please follow this SQL in its calculation, which is based on the MilestonesReport query:

SELECT MilestonesReport.MASTER\_PROJECT\_NAME, Count(MilestonesReport.[Milestone Id]) AS [Milestones - Completed, not Billed]

FROM MilestonesReport

WHERE (((MilestonesReport.[Milestone Status])="Completed, not Billed"))

GROUP BY MilestonesReport.MASTER\_PROJECT\_NAME

ORDER BY MilestonesReport.MASTER\_PROJECT\_NAME;

1. Not Billed balance: **OFE only**. This KPI is the balance of completed Milestones that have not been billed yet:

SELECT MilestoneCalcs.MASTER\_PROJECT\_NAME, MilestoneCalcs.CompletedMilestonesValue, InvoicedTotals.InvoicedTotal, [CompletedMilestonesValue]-IIf(IsNull([InvoicedTotal]),0,[InvoicedTotal]) AS NotBilledTotal

FROM InvoicedTotals RIGHT JOIN MilestoneCalcs ON InvoicedTotals.MASTER\_PROJECT\_NAME = MilestoneCalcs.MASTER\_PROJECT\_NAME

ORDER BY MilestoneCalcs.MASTER\_PROJECT\_NAME;

1. Current Due balance: This KPI is the balance of Invoices issued and not due yet. Same logic for TPS and OFE. Grouping by Project ID as it is identified in RealTrack.

SELECT CurrentDuesReport.MASTER\_PROJECT\_NAME, Sum(CurrentDuesReport.[Outs Amount]) AS CurrentDueTotal

FROM CurrentDuesReport

GROUP BY CurrentDuesReport.MASTER\_PROJECT\_NAME

ORDER BY CurrentDuesReport.MASTER\_PROJECT\_NAME;

1. Past Due balance: Already in the KPI widget. This KPI is the balance of Invoices already past their due date.

SELECT PastDuesReport.MASTER\_PROJECT\_NAME, Sum(PastDuesReport.[Outs Amount]) AS PastDueTotal

FROM PastDuesReport

GROUP BY PastDuesReport.MASTER\_PROJECT\_NAME

ORDER BY PastDuesReport.MASTER\_PROJECT\_NAME;

1. Collected balance: it is the total amount of the paid invoices. Need to verify data availability.
2. Billed balance: **OFE tbc**. The actual amount billed present in the Financial Summary needs to be reported on the KPI widget.

# Data Sources - OFE

The 5 tables on which this widget is built are:

1. **Milestones**: P6 Payment Milestones
2. **Invoices**: ERP Invoices that have already been issued
3. **OutsInvoices**: GECARS Outstanding (open) Invoices. Some are current due, some are past due.
4. **OutsInvoicesHistory**: GECARS history of (almost) daily snapshots of Outstanding Invoices.
5. **Payments**: Payments received from the customer

The first four tables are available in the ARGO Datalake, SQLs are provided further below, while the 5th table, Payments, is not currently available and must be generated based on Invoices and OutsInvoicesHistory.

SQLs to generate tables 1, 2, 3 & 4 from the ARGO datalake:

1. **P6 Payment Milestones leveraging Project Registry: Milestones**

SELECT ProjRegistry.MASTER\_PROJECT\_NAME, Milestones.IPP\_PROJECT\_NAME, Milestones.PROJECT\_ID, Milestones.ACTIVITY\_ID,

Milestones.ACTIVITY\_NAME, Milestones.ACTIVITY\_STATUS, Milestones.ACT\_PERCENT\_COMPLETE, Milestones.BL\_PROJECT\_FINISH,

Milestones.FINISH\_DATE, Milestones.ACTUAL\_FINISH, Milestones.IPP\_PAYMENT\_MILESTONE\_CODE, Milestones.IPP\_PAYMENT\_MILESTONE\_DESC,

Milestones.IPP\_PAYMENT\_MILESTONE\_VALUE, Milestones.CONTRACT\_CURRENCY,

ProjRegistry.MASTER\_PROJECT\_NAME || ' - ' || Milestones.ACTIVITY\_ID AS Project\_Milestone,

Milestones.LAST\_UPDATE\_DATE

FROM **RT\_P6\_REALTRACK\_WIDGET** Milestones, (SELECT b.MASTER\_PROJECT\_NAME, a.PROJECT\_ID from CUST\_GPO\_DW.GE\_OG\_ALIAS\_LIST a, CUST\_GPO\_DW.GE\_OG\_ANAGRAPHICS b

WHERE a.master\_project\_id = b.master\_project\_id) ProjRegistry

WHERE Milestones.PROJECT\_ID = ProjRegistry.PROJECT\_ID

AND Milestones.IPP\_PAYMENT\_MILESTONE\_CODE IS NOT NULL

AND NVL(Milestones.IPP\_PAYMENT\_MILESTONE\_VALUE,0) <> 0

ORDER BY ProjRegistry.MASTER\_PROJECT\_NAME, Milestones.FINISH\_DATE;

**2) Invoices Leveraging Project Registry - Invoices**

SELECT

ProjRegistry.MASTER\_PROJECT\_NAME, Invoices.PROJECT\_NUMBER, Invoices.INVOICE\_DATE, Invoices.RA\_INVOICE\_NUMBER, Invoices.LINE\_NUM,

Invoices.TEXT, Invoices.TERM\_NAME, Invoices.INV\_AMOUNT, Invoices.INV\_CURRENCY\_CODE, Invoices.EVENT\_ID,

CASE

WHEN INSTR(Invoices.PM\_EVENT\_REFERENCE, '[')-1 > 0

THEN SUBSTR(Invoices.PM\_EVENT\_REFERENCE,1,INSTR(Invoices.PM\_EVENT\_REFERENCE,'[')-1)

ELSE Invoices.PM\_EVENT\_REFERENCE

END

AS PM\_EVENT\_REFERENCE,

Invoices.CANCELED\_FLAG, Invoices.DRAFT\_INVOICE\_NUM\_CREDITED, Invoices.DRAFT\_INVOICE\_NUM,

CASE

WHEN Invoices.DRAFT\_INVOICE\_NUM\_CREDITED IS NOT NULL OR Invoices.CANCELED\_FLAG = 'Y'

THEN 'Y'

ELSE 'N'

END

AS "CancelingOut",

ProjRegistry.MASTER\_PROJECT\_NAME || ' - ' ||

(CASE

WHEN INSTR(Invoices.PM\_EVENT\_REFERENCE, '[')-1 > 0

THEN SUBSTR(Invoices.PM\_EVENT\_REFERENCE,1,INSTR(Invoices.PM\_EVENT\_REFERENCE,'[')-1)

ELSE Invoices.PM\_EVENT\_REFERENCE

END)

AS Project\_Milestone,

ProjRegistry.MASTER\_PROJECT\_NAME || ' - ' || Invoices.PROJECT\_NUMBER || ' - ' || Invoices.RA\_INVOICE\_NUMBER AS Project\_Invoice,

IMPORT\_DATE

FROM **WC\_PRJ\_INVOICES** Invoices, (SELECT b.MASTER\_PROJECT\_NAME, a.PROJECT\_ID from CUST\_GPO\_DW.GE\_OG\_ALIAS\_LIST a, CUST\_GPO\_DW.GE\_OG\_ANAGRAPHICS b

WHERE a.master\_project\_id = b.master\_project\_id AND a.METRIC LIKE '%Billing') ProjRegistry

WHERE Invoices.PROJECT\_NUMBER = ProjRegistry.PROJECT\_ID

ORDER BY ProjRegistry.MASTER\_PROJECT\_NAME, Invoices.INVOICE\_DATE;

**3) GECARS Outstanding Invoices leveraging Project Registry: OutsInvoices**

SELECT

ProjRegistry.MASTER\_PROJECT\_NAME, GECARS."Project Num", GECARS."INVOICE NUMBER",

TO\_CHAR(GECARS.DTS\_LOAD\_DT,'yyyy-mm-dd') AS "LOAD\_DT", GECARS."Cust Name", GECARS."PO Num",

GECARS."Inv DT", GECARS."Due DT", GECARS."Currency Name", GECARS."Days PD" AS "GECARS Days PD",

CASE

WHEN GECARS.DTS\_LOAD\_DT > GECARS."Due DT"

THEN 'Past Due'

ELSE 'Current Due'

END

AS "OutsStatus",

GECARS."Inv Amt", GECARS."GECARS AMOUNT", GECARS."OUTS AMT (in USD MOR)",

"Open Stat Desc", "Open Stat Dt",

ProjRegistry.MASTER\_PROJECT\_NAME || ' - ' || GECARS."Project Num" || ' - ' || GECARS."INVOICE NUMBER" AS Project\_Invoice

FROM **STG\_DTS\_GECARS\_PASTDUE\_ALL** GECARS, (SELECT b.MASTER\_PROJECT\_NAME, a.PROJECT\_ID

FROM CUST\_GPO\_DW.GE\_OG\_ALIAS\_LIST a, CUST\_GPO\_DW.GE\_OG\_ANAGRAPHICS b

WHERE a.master\_project\_id = b.master\_project\_id AND a.METRIC LIKE '%Billing') ProjRegistry

WHERE GECARS."Project Num" = ProjRegistry.PROJECT\_ID

ORDER BY ProjRegistry.MASTER\_PROJECT\_NAME, GECARS."Project Num", GECARS."INVOICE NUMBER", LOAD\_DT;

**4) GECARS History of Outstanding Invoices leveraging Project Registry: OutsInvoicesHistory**

SELECT

ProjRegistry.MASTER\_PROJECT\_NAME, GECARS."Project Num", GECARS."INVOICE NUMBER",

TO\_CHAR(GECARS.DTS\_LOAD\_DT,'yyyy-mm-dd') AS "LOAD\_DT", GECARS."Cust Name", GECARS."PO Num",

GECARS."Inv DT", GECARS."Due DT", GECARS."Currency Name", GECARS."Days PD" AS "GECARS Days PD",

CASE

WHEN GECARS.DTS\_LOAD\_DT > GECARS."Due DT"

THEN 'Past Due'

ELSE 'Current Due'

END

AS "OutsStatus",

GECARS."Inv Amt", GECARS."GECARS AMOUNT", GECARS."OUTS AMT (in USD MOR)",

"Open Stat Desc", "Open Stat Dt",

ProjRegistry.MASTER\_PROJECT\_NAME || ' - ' || GECARS."Project Num" || ' - ' || GECARS."INVOICE NUMBER" AS Project\_Invoice

FROM **STG\_DTS\_GECARS\_PASTDUE\_ALL\_BKP** GECARS, (SELECT b.MASTER\_PROJECT\_NAME, a.PROJECT\_ID

FROM CUST\_GPO\_DW.GE\_OG\_ALIAS\_LIST a, CUST\_GPO\_DW.GE\_OG\_ANAGRAPHICS b

WHERE a.master\_project\_id = b.master\_project\_id AND a.METRIC LIKE '%Billing') ProjRegistry

WHERE GECARS."Project Num" = ProjRegistry.PROJECT\_ID

ORDER BY ProjRegistry.MASTER\_PROJECT\_NAME, GECARS."Project Num", GECARS."INVOICE NUMBER", LOAD\_DT;

**5) Payments table generated through an algorithm**

The Python code below, **UpdatePayments.py**, generates the 5th table, Payments, based on the 2nd (Invoices) and the 4th (OutsInvoicesHistory). It assumes the existence of an SQLite database, **BillingWidget.sqlite**, where to find the 2nd and 4th tables and where to create and populate the 5th. I leave it up to your development experience to choose how you follow the algorithm executed by this code. Please note I have left as comments a few print sentences that I used while I was developing it, as I thought they would help:

import sqlite3

conn = sqlite3.connect('BillingWidget.sqlite')

cur1 = conn.cursor()

cur2 = conn.cursor()

cur3 = conn.cursor()

# First we put all the distinct Invoices of OutsInvoicesHistory in a new Table called InvoicesWithHistory

# By default we populate the field CancelingOut with a 'N' --> This is a field we will populate later

print('Creating the table InvoicesWithHistory')

cur1.execute('DROP TABLE IF EXISTS InvoicesWithHistory')

cur1.execute('''CREATE TABLE InvoicesWithHistory (MASTER\_PROJECT\_NAME TEXT, [Project Num] TEXT, [INVOICE NUMBER] TEXT, CancelingOut TEXT)''')

for row in cur1.execute('''SELECT DISTINCT MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER] FROM OutsInvoicesHistory GROUP BY MASTER\_PROJECT\_NAME,

[Project Num], [INVOICE NUMBER] ORDER BY MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER]'''):

cur2.execute('''INSERT INTO InvoicesWithHistory (MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER], CancelingOut) VALUES (?,?,?,?)''',

(row[0],row[1],row[2],'N'))

conn.commit()

# We will now exclude all Invoices that have CancelingOut = "Y" in Invoices

# Those are Canceled Invoices and their Credit Notes, which, besides netting out to zero, are

# inconsistently tracked in GECARS (some are tracked, and some are not, which generates unreliable Payments)

# To do that, we first create the Table InvoicesCancelingOut, which has all the Invoices that we must exclude

print('Creating the table InvoicesCancelingOut')

cur1.execute('DROP TABLE IF EXISTS InvoicesCancelingOut')

cur1.execute('''CREATE TABLE InvoicesCancelingOut (MASTER\_PROJECT\_NAME TEXT, [Project Num] TEXT, [INVOICE NUMBER] TEXT)''')

for row in cur1.execute('''SELECT DISTINCT MASTER\_PROJECT\_NAME, PROJECT\_NUMBER AS [Project Num], RA\_INVOICE\_NUMBER AS [INVOICE NUMBER] FROM Invoices

WHERE CancelingOut = "Y"

GROUP BY MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER]

ORDER BY MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER]'''):

cur2.execute('''INSERT INTO InvoicesCancelingOut (MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER]) VALUES (?,?,?)''',

(row[0],row[1],row[2]))

conn.commit()

# Now we can use this newly created table, InvoicesCancelingOut, to update the field CancelingOut in InvoicesWithHistory

print('Marking CancelingOut Invoices in InvoicesWithHistory')

for row in cur1.execute('SELECT \* FROM InvoicesCancelingOut'):

cur2.execute('''UPDATE InvoicesWithHistory SET CancelingOut = "Y"

WHERE MASTER\_PROJECT\_NAME = ? AND [Project Num] = ? AND [INVOICE NUMBER] = ?''',(row[0],row[1],row[2]))

conn.commit()

# Now we delete and recreate the table Payments

# Important Note: this is a non-recursive procedure, we create All Payments every time we run it

# Reasons: The time it takes to run it is not long enough (< 2 minutes) to require a recursive procedure, which would be much more complex and

# potentially not that much faster (because of all the checks it would entail)

print('Creating the table Payments')

cur1.execute('DROP TABLE IF EXISTS Payments')

cur1.execute ('''CREATE TABLE "Payments" ( `PaymentId` integer primary key autoincrement,`MASTER\_PROJECT\_NAME` TEXT, `Project Num` TEXT, `INVOICE NUMBER` TEXT,

`Currency Name` TEXT, `Inv DT` TEXT, `Due DT` TEXT, `Inv Amt` REAL, `PaymentDate` TEXT, `GECARS Days PD` REAL,`PaymentAmount` REAL, `PaymentType` TEXT,

`RemainingAmount` REAL, 'PROJECT\_INVOICE' TEXT, 'LAST\_LOAD\_DT' TEXT)''')

conn.commit()

# We need as a reference the latest LOAD\_DT of OutsInvoicesHistory:

cur1.execute('SELECT max(LOAD\_DT) FROM OutsInvoicesHistory')

row = cur1.fetchone()

LAST\_LOAD\_DT = row[0] # lAST\_LOAD\_DT now has the latest LOAD\_DT of OutsInvoicesHistory

print('As a reference, LAST\_LOAD\_DT is:',LAST\_LOAD\_DT)

print('Starting the big loop through all InvoicesWithHistory without FullPayment to include the new payments in the table Payments')

# Now is the time to go through all of the InvoicesWithHistory that are NOT CancelingOut and Update their Payments in the Payments table

for Invoice in cur1.execute('SELECT \* FROM InvoicesWithHistory WHERE CancelingOut = "N"'):

print(Invoice[0],Invoice[1],Invoice[2])

counter = 1

PriorGECARSAmount = 0.0

# Let's find out how many records are there in OutsInvoicesHistory for this Invoice

cur2.execute('''SELECT count(MASTER\_PROJECT\_NAME) FROM OutsInvoicesHistory WHERE MASTER\_PROJECT\_NAME = ? AND [Project Num] = ?

AND [INVOICE NUMBER] = ? ORDER BY LOAD\_DT''',(Invoice[0],Invoice[1],Invoice[2]))

row = cur2.fetchone()

rows = row[0] # rows now has the number of records in OutsInvoicesHistory for this Invoice

# And now we can start going through each of the records in OutsInvoicesHistory for this Invoice

for row in cur2.execute('''SELECT \* FROM OutsInvoicesHistory WHERE MASTER\_PROJECT\_NAME = ? AND [Project Num] = ?

AND [INVOICE NUMBER] = ? ORDER BY LOAD\_DT''',(Invoice[0],Invoice[1],Invoice[2])):

if counter > 1 and row[12] != PriorGECARSAmount:

# We are past the first record and there has been a change in GECARS Amount --> We need to record the partial payment

# Note that the payment has not been Total because that only can happen on the last record

#print('Partial Payment to be stored with these details:')

#print(' MASTER\_PROJECT\_NAME:',Invoice[0])

#print(' Project Num:',Invoice[1])

#print(' INVOICE NUMBER:',Invoice[2])

#print(' Currency Name:',PriorCurrencyName)

#print(' Inv DT:',PriorInvDT)

#print(' Due DT:',PriorDueDT)

#print(' Inv Amt:',PriorInvAmt)

#print(' Payment Date:',PriorLOAD\_DT)

#print(' GECARS Days PD:',PriorDaysPD)

#print(' PaymentAmount:',PriorGECARSAmount-row[12])

#print(' PaymentType:','Partial')

#print(' RemainingAmount:',row[12])

#print(' PROJECT\_INVOICE: ,row[16]')

cur3.execute('''INSERT INTO Payments (MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER],

[Currency Name], [Inv DT], [Due DT], [Inv Amt], PaymentDate, [GECARS Days PD], PaymentAmount,

PaymentType, RemainingAmount, PROJECT\_INVOICE, LAST\_LOAD\_DT) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',

(Invoice[0],Invoice[1],Invoice[2],PriorCurrencyName,PriorInvDT,PriorDueDT,PriorInvAmt,

PriorLOAD\_DT,PriorGECARSDaysPD,PriorGECARSAmount-row[12],'Partial',row[12],row[16],LAST\_LOAD\_DT))

if counter == rows:

if row[3] != LAST\_LOAD\_DT:

# We are at the last record and the LOAD\_DT is not equal to the lAST\_LOAD\_DT, which means

# there has been a payment in full of the GECARS Amount remaining at that point

# Note that this could be the total invoiced amount or only the last remaining amount after

# a series of one or more partial payments. In both cases the remaining amount afterwards is 0

# Note that if LOAD\_DT is equal to lAST\_LOAD\_DT the invoice remains open, and we must not record

# any payment in that case (this is the case where the if statement above is not true, which leads

# to no action)

# --> We need to record the payment in full

#print('Payment in Full to be stored with these details:')

#print(' MASTER\_PROJECT\_NAME:',Invoice[0])

#print(' Project Num:',Invoice[1])

#print(' INVOICE NUMBER:',Invoice[2])

#print(' Currency Name:',row[8])

#print(' Inv DT:',row[6])

#print(' Due DT:',row[7])

#print(' Inv Amt:',row[11])

#print(' Payment Date:',row[3])

#print(' GECARS Days PD:',row[9])

#print(' PaymentAmount:',row[12])

#if (row[12]==row[11]):

# print(' PaymentType:','Total')

#else:

# print(' PaymentType:','Partial')

#print(' RemainingAmount:',0)

#print(' PROJECT\_INVOICE: ,row[16]')

if (row[12]==row[11]):

cur3.execute('''INSERT INTO Payments (MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER],

[Currency Name], [Inv DT], [Due DT], [Inv Amt], PaymentDate, [GECARS Days PD], PaymentAmount,

PaymentType, RemainingAmount, PROJECT\_INVOICE, LAST\_LOAD\_DT) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',

(Invoice[0],Invoice[1],Invoice[2],row[8],row[6],row[7],row[11],row[3],row[9],row[12],

'Total',0,row[16], LAST\_LOAD\_DT))

else:

cur3.execute('''INSERT INTO Payments (MASTER\_PROJECT\_NAME, [Project Num], [INVOICE NUMBER],

[Currency Name], [Inv DT], [Due DT], [Inv Amt], PaymentDate, [GECARS Days PD], PaymentAmount,

PaymentType, RemainingAmount, PROJECT\_INVOICE, LAST\_LOAD\_DT) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)''',

(Invoice[0],Invoice[1],Invoice[2],row[8],row[6],row[7],row[11],row[3],row[9],row[12],

'Partial',0,row[16], LAST\_LOAD\_DT))

# We record the prior values before advancing in the loop, as we will need them:

PriorLOAD\_DT = row[3]

PriorInvDT = row[6]

PriorDueDT = row[7]

PriorCurrencyName = row[8]

PriorGECARSDaysPD = row[9]

PriorIsPastDue = row[10]

PriorInvAmt = row[11]

PriorGECARSAmount = row[12]

counter = counter + 1

conn.commit()

# We do not need the table InvoicesWithHistory anymore:

cur1.execute('DROP TABLE IF EXISTS InvoicesWithHistory')

conn.commit()

# We do not need the table InvoicesCancelingOut anymore:

cur1.execute('DROP TABLE IF EXISTS InvoicesCancelingOut')

conn.commit()

# Now we are done!

print('Process complete')

cur1.close()

cur2.close()

cur3.close()

# Queries supporting the Billing Reports

Below we will describe the various queries that are used to support the Billing Reports. Some of them are used as part of others, so we will present them in proper order:

LastPaymentIds: this query provides the Id of the last payment received for each project invoice:

SELECT Payments.PROJECT\_INVOICE, Max(Payments.PaymentId) AS LastPaymentId

FROM Payments

GROUP BY Payments.PROJECT\_INVOICE

ORDER BY Payments.PROJECT\_INVOICE;

LastPayments: this query uses “LastPaymentIds” and “Payments” to produce a complete, all fields query of the last payment received for each project invoice, adding two calculated fields, “Last Paym Days PD”, which indicates how many days Past Due was the invoice when that last payment was received, and “TotalPaymentToDate”, which holds all cumulated payments received to date for that project invoice:

SELECT Payments.\*, DateDiff("d",[Due DT],[PaymentDate]) AS [Last Paym Days PD], [Inv Amt]-[RemainingAmount] AS TotalPaymentToDate

FROM Payments INNER JOIN LastPaymentIds ON (LastPaymentIds.LastPaymentId = Payments.PaymentId) AND (Payments.PROJECT\_INVOICE = LastPaymentIds.PROJECT\_INVOICE);

InvoicesQuery: this is an important query that will be used throughout all Billing Reports. It leverages the “Invoices” table as its primary source, as well as “OutsInvoices” and “LastPayments”:

SELECT Invoices.MASTER\_PROJECT\_NAME, Invoices.RA\_INVOICE\_NUMBER AS Inv, Invoices.LINE\_NUM AS [Inv Line], Invoices.Text AS [Inv Desc], Invoices.INVOICE\_DATE AS [Inv Date], Year([INVOICE\_DATE]) & "-" & IIf(Month([INVOICE\_DATE])>9,"Q4",IIf(Month([INVOICE\_DATE])>6,"Q3",IIf(Month([INVOICE\_DATE])>3,"Q2","Q1"))) AS [Inv Y-Q], Invoices.TERM\_NAME AS [Inv Term], IIf(IsNull([OutsInvoices].[Due DT]),[LastPayments].[Due DT],[OutsInvoices].[Due DT]) AS [Due Date], Invoices.INV\_AMOUNT AS [Inv Amount], Invoices.INV\_CURRENCY\_CODE AS [Inv Curr], IIf(IsNull([OutsStatus]),"Not Outstanding",[OutsStatus]) AS [Inv Outs Status], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)) AS [Paym Status], OutsInvoices.[GECARS AMOUNT] AS [Outs Amount], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) AS [Days PD], LastPayments.PaymentDate AS [Last Paym Date], LastPayments.[Last Paym Days PD], LastPayments.PaymentAmount AS [Last Paid Amount], LastPayments.TotalPaymentToDate, LastPayments.PaymentType AS [Last Paym Type], LastPayments.RemainingAmount AS [Last Rem Amount], Invoices.PROJECT\_INVOICE, Invoices.PROJECT\_MILESTONE

FROM (Invoices LEFT JOIN OutsInvoices ON Invoices.PROJECT\_INVOICE = OutsInvoices.PROJECT\_INVOICE) LEFT JOIN LastPayments ON Invoices.PROJECT\_INVOICE = LastPayments.PROJECT\_INVOICE

WHERE (((Invoices.CancelingOut)="N" Or (Invoices.CancelingOut) Is Null))

ORDER BY Invoices.MASTER\_PROJECT\_NAME, Invoices.INVOICE\_DATE;

MilestonesReport: this is the basis for the first of the Billing Reports. It leverages the “Milestones” table as primary source, as well as “InvoicesQuery”:

SELECT Milestones.MASTER\_PROJECT\_NAME, Milestones.IPP\_PAYMENT\_MILESTONE\_CODE AS [Milestone Id], Milestones.IPP\_PAYMENT\_MILESTONE\_DESC AS Milestone, Milestones.ACTIVITY\_ID AS [Activity Id], Milestones.IPP\_PAYMENT\_MILESTONE\_VALUE AS [Milestone Value], Milestones.CONTRACT\_CURRENCY AS [Milestone Curr], Milestones.FINISH\_DATE AS [Milestone Forecast], Year([FINISH\_DATE]) & "-" & IIf(Month([FINISH\_DATE])>9,"Q4",IIf(Month([FINISH\_DATE])>6,"Q3",IIf(Month([FINISH\_DATE])>3,"Q2","Q1"))) AS [Milestone Y-Q], Milestones.ACTUAL\_FINISH AS [Milestone Completion], IIf(IsNull([Inv]),IIf(Not IsNull([ACTUAL\_FINISH]),"Completed, not Billed",IIf([FINISH\_DATE]<[LAST\_UPDATE\_DATE],"Overdue","Forecasted")),"Billed") AS [Milestone Status], IIf(Not IsNull([ACTUAL\_FINISH]) And IsNull([Inv]),DateDiff("d",[ACTUAL\_FINISH],[LAST\_UPDATE\_DATE]),Null) AS [Days Unbilled], IIf(IsNull([Inv]) And IsNull([ACTUAL\_FINISH]),DateDiff("d",[LAST\_UPDATE\_DATE],[FINISH\_DATE]),Null) AS [Days to prepare Invoice], IIf(Not IsNull([ACTUAL\_FINISH]) And Not IsNull([Inv]),DateDiff("d",[ACTUAL\_FINISH],[Inv Date]),Null) AS [Days taken to issue Invoice], InvoicesQuery.Inv, InvoicesQuery.[Inv Line], InvoicesQuery.[Inv Date], InvoicesQuery.[Inv Outs Status], InvoicesQuery.[Inv Term], InvoicesQuery.[Due Date], InvoicesQuery.[Inv Amount], InvoicesQuery.[Inv Curr], InvoicesQuery.[Inv Desc], InvoicesQuery.[Paym Status], InvoicesQuery.[Outs Amount], InvoicesQuery.[Days PD], InvoicesQuery.[Last Paym Date], InvoicesQuery.[Last Paym Type], InvoicesQuery.[Last Paym Days PD], InvoicesQuery.[Last Paid Amount], InvoicesQuery.TotalPaymentToDate, InvoicesQuery.[Last Rem Amount]

FROM Milestones LEFT JOIN InvoicesQuery ON Milestones.PROJECT\_MILESTONE = InvoicesQuery.PROJECT\_MILESTONE

ORDER BY Milestones.MASTER\_PROJECT\_NAME, Milestones.FINISH\_DATE, InvoicesQuery.Inv, InvoicesQuery.[Inv Line];

InvoicesReport: this is the basis for the second of the Billing Reports. It leverages “InvoicesQuery” as primary source, as well as the “Milestones” table:

SELECT InvoicesQuery.MASTER\_PROJECT\_NAME, InvoicesQuery.Inv, InvoicesQuery.[Inv Line], InvoicesQuery.[Inv Desc], IIf(Not IsNull([ACTUAL\_FINISH]),DateDiff("d",[ACTUAL\_FINISH],[Inv Date]),Null) AS [Days taken to issue Invoice], InvoicesQuery.[Inv Date], InvoicesQuery.[Inv Y-Q], InvoicesQuery.[Inv Term], InvoicesQuery.[Due Date], InvoicesQuery.[Inv Amount], InvoicesQuery.[Inv Curr], InvoicesQuery.[Inv Outs Status], InvoicesQuery.[Paym Status], InvoicesQuery.[Outs Amount], InvoicesQuery.[Days PD], InvoicesQuery.[Last Paym Date], InvoicesQuery.[Last Paym Days PD], InvoicesQuery.[Last Paid Amount], InvoicesQuery.TotalPaymentToDate, InvoicesQuery.[Last Paym Type], InvoicesQuery.[Last Rem Amount], Milestones.IPP\_PAYMENT\_MILESTONE\_CODE AS [Milestone Id], Milestones.IPP\_PAYMENT\_MILESTONE\_DESC AS Milestone, Milestones.ACTIVITY\_ID AS [Activity Id], Milestones.IPP\_PAYMENT\_MILESTONE\_VALUE AS [Milestone Value], Milestones.CONTRACT\_CURRENCY AS [Milestone Curr], Milestones.FINISH\_DATE AS [Milestone FC], Milestones.ACTUAL\_FINISH AS [Milestone Completion]

FROM InvoicesQuery LEFT JOIN Milestones ON InvoicesQuery.PROJECT\_MILESTONE = Milestones.PROJECT\_MILESTONE

ORDER BY InvoicesQuery.MASTER\_PROJECT\_NAME, InvoicesQuery.[Inv Date];

CurrentDuesReport: this is the basis for the third of the Billing Reports. It leverages “OutsInvoices” as primary source, as well as the “Invoices” table and the “LastPayments” query:

SELECT OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[INVOICE NUMBER] AS Inv, First(Invoices.Text) AS [Inv Desc], OutsInvoices.[Inv DT] AS [Inv Date], Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))) AS [Inv Y-Q], First(Invoices.TERM\_NAME) AS [Inv Term], OutsInvoices.[Due DT] AS [Due Date], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) AS [Days PD], OutsInvoices.[Open Stat Desc], OutsInvoices.[Open Stat Dt], First(Invoices.INV\_AMOUNT) AS [Inv Amount], OutsInvoices.[GECARS AMOUNT] AS [Outs Amount], OutsInvoices.[Currency Name] AS [Inv Curr], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)) AS [Paym Status], LastPayments.PaymentDate AS [Last Paym Date], LastPayments.[GECARS Days PD] AS [Last Paym Days PD], LastPayments.PaymentAmount AS [Last Paid Amount], LastPayments.TotalPaymentToDate, LastPayments.PaymentType AS [Last Paym Type], LastPayments.RemainingAmount AS [Last Rem Amount]

FROM (Invoices RIGHT JOIN OutsInvoices ON Invoices.PROJECT\_INVOICE = OutsInvoices.PROJECT\_INVOICE) LEFT JOIN LastPayments ON OutsInvoices.PROJECT\_INVOICE = LastPayments.PROJECT\_INVOICE

WHERE (((OutsInvoices.OutsStatus)="Current Due"))

GROUP BY OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[INVOICE NUMBER], OutsInvoices.[Inv DT], Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))), OutsInvoices.[Due DT], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]), OutsInvoices.[Open Stat Desc], OutsInvoices.[Open Stat Dt], OutsInvoices.[GECARS AMOUNT], OutsInvoices.[Currency Name], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)), LastPayments.PaymentDate, LastPayments.[GECARS Days PD], LastPayments.PaymentAmount, LastPayments.TotalPaymentToDate, LastPayments.PaymentType, LastPayments.RemainingAmount, Invoices.CancelingOut

HAVING (((Invoices.CancelingOut)="N" Or (Invoices.CancelingOut) Is Null))

ORDER BY OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[Inv DT];

PastDuesReport: this is the basis for the fourth of the Billing Reports. It leverages “OutsInvoices” as primary source, as well as the “Invoices” table and the “LastPayments” query:

SELECT OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[INVOICE NUMBER] AS Inv, First(Invoices.Text) AS [Inv Desc], OutsInvoices.[Inv DT] AS [Inv Date], Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))) AS [Inv Y-Q], First(Invoices.TERM\_NAME) AS [Inv Term], OutsInvoices.[Due DT] AS [Due Date], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) AS [Days PD], OutsInvoices.[Open Stat Desc], OutsInvoices.[Open Stat Dt], First(Invoices.INV\_AMOUNT) AS [Inv Amount], OutsInvoices.[GECARS AMOUNT] AS [Outs Amount], OutsInvoices.[Currency Name] AS [Inv Curr], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)) AS [Paym Status], LastPayments.PaymentDate AS [Last Paym Date], LastPayments.[GECARS Days PD] AS [Last Paym Days PD], LastPayments.PaymentAmount AS [Last Paid Amount], LastPayments.TotalPaymentToDate, LastPayments.PaymentType AS [Last Paym Type], LastPayments.RemainingAmount AS [Last Rem Amount]

FROM (Invoices RIGHT JOIN OutsInvoices ON Invoices.PROJECT\_INVOICE = OutsInvoices.PROJECT\_INVOICE) LEFT JOIN LastPayments ON OutsInvoices.PROJECT\_INVOICE = LastPayments.PROJECT\_INVOICE

WHERE (((OutsInvoices.OutsStatus)="Past Due"))

GROUP BY OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[INVOICE NUMBER], OutsInvoices.[Inv DT], Year([OutsInvoices].[Inv DT]) & "-" & IIf(Month([OutsInvoices].[Inv DT])>9,"Q4",IIf(Month([OutsInvoices].[Inv DT])>6,"Q3",IIf(Month([OutsInvoices].[Inv DT])>3,"Q2","Q1"))), OutsInvoices.[Due DT], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]), OutsInvoices.[Open Stat Desc], OutsInvoices.[Open Stat Dt], OutsInvoices.[GECARS AMOUNT], OutsInvoices.[Currency Name], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[LastPayments].[Inv Amt],"Partially Paid",Null)), LastPayments.PaymentDate, LastPayments.[GECARS Days PD], LastPayments.PaymentAmount, LastPayments.TotalPaymentToDate, LastPayments.PaymentType, LastPayments.RemainingAmount, Invoices.CancelingOut

HAVING (((Invoices.CancelingOut)="N" Or (Invoices.CancelingOut) Is Null))

ORDER BY OutsInvoices.MASTER\_PROJECT\_NAME, OutsInvoices.[Inv DT];

PaymentsReport: this is the basis for the fifth of the Billing Reports. It leverages the “Payments” table as primary source, as well as the “Invoices” and “OutsInvoices” tables:

SELECT Payments.MASTER\_PROJECT\_NAME, Payments.[INVOICE NUMBER] AS Inv, First(Invoices.Text) AS [Inv Desc], First(Invoices.INV\_AMOUNT) AS [Inv Amount], Payments.[Currency Name] AS [Inv Curr], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[Payments].[Inv Amt],"Partially Paid",Null)) AS [Paym Status], OutsInvoices.OutsStatus AS [Inv Outs Status], OutsInvoices.[GECARS AMOUNT] AS [Outs Amount], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]) AS [Days PD], Payments.[Inv DT], First(Invoices.TERM\_NAME) AS [Inv Term], Payments.[Due DT] AS [Due Date], Payments.PaymentId, Payments.PaymentDate AS [Paym Date], DateDiff("d",[Payments].[Due DT],[PaymentDate]) AS [Paym Days PD], Payments.PaymentType AS [Paym Type], Payments.PaymentAmount AS [Paid Amount], Payments.RemainingAmount AS [Rem Amount]

FROM (Payments LEFT JOIN Invoices ON Payments.PROJECT\_INVOICE = Invoices.PROJECT\_INVOICE) LEFT JOIN OutsInvoices ON Payments.PROJECT\_INVOICE = OutsInvoices.PROJECT\_INVOICE

GROUP BY Payments.MASTER\_PROJECT\_NAME, Payments.[INVOICE NUMBER], Payments.[Currency Name], IIf([RemainingAmount]=0,"Fully Paid",IIf([RemainingAmount]<[Payments].[Inv Amt],"Partially Paid",Null)), OutsInvoices.OutsStatus, OutsInvoices.[GECARS AMOUNT], DateDiff("d",[OutsInvoices].[Due DT],[OutsInvoices].[LOAD\_DT]), Payments.[Inv DT], Payments.[Due DT], Payments.PaymentId, Payments.PaymentDate, DateDiff("d",[Payments].[Due DT],[PaymentDate]), Payments.PaymentType, Payments.PaymentAmount, Payments.RemainingAmount, Invoices.CancelingOut

HAVING (((Invoices.CancelingOut)="N" Or (Invoices.CancelingOut) Is Null))

ORDER BY Payments.MASTER\_PROJECT\_NAME, Payments.PaymentId;