Rolling Revenue Report Future Years

Last Modified on 23/03/2026 3:25 pm EDT

/*

This query provides monthly operator-level summary of bookings, WH Out, invoiced revenue, averages, closure ratios, lost revenue, and prior-year revenue for trend analysis over a rolling multi-year period. The report starts at 2022 and is forward facing to the current date

*/

/*

Operator Booking Report (From 2022 – Current Year)

Overview

The stored procedure `usp_OperatorBookingReport_From2023_Auto_AR9` generates a monthly operator performance report combining booking activity, invoicing, and year-over-year comparisons.

The report provides visibility into:

- Booking revenue (created vs operational)
- Invoiced revenue
- Conversion ratios (booked → invoiced)
- Revenue loss
- Prior year comparisons

Date Range Logic

The procedure dynamically calculates its reporting period:

- **Start Date:** January 1, 2022
- **End Date:** Current date (`GETDATE()`)

This ensures:

- Historical data is included from 2022 onward
- The current year is included up to today (Year-To-Date)

Previous Year Comparison

A shifted date range is used to calculate prior-year revenue:

- Previous period = current period minus 1 year
- Used for monthly year-over-year comparisons

Data Source

The procedure pulls data from:

- `vwBookAndHist` (current and archived bookings)
- `tbloperators` (operator details)

> Note: `vwBookAndHist` may include both active and historical bookings.

Key Components

1. RevenueLastYear (CTE)

Calculates invoiced revenue from the previous year:

- Filters:
  - `invoiced = 'Y'`
  - Excludes booking types: 5 - Sub Rentals, 11 - Location Transfers, 13 - Sundry Transfers
- Groups by:
  - Operator
  - Month

---

2. CreatedData (CTE)

Captures bookings based on **entry date** (when the booking was created).

Metrics:

- `BKTotal-C` : Total booking value (excluding taxes)
- `QtyBkd-C` : Number of bookings created

3. BookedData (CTE)

Captures bookings based on **ddate** (operational date).

Metrics:

- `BookingTotal-O` : Total booked revenue
- `QtyBooked-O` : Total bookings

Invoicing Metrics (Conditional Aggregation):

- `InvoiceTotal` : Revenue where `invoiced = 'Y'`
- `QtyInvoiced` : Count of invoiced bookings

> Uses `CASE` inside `SUM()` to ensure accurate aggregation.

---

4. Data Merge

The procedure combines Created and Booked datasets using a:

- `FULL OUTER JOIN`

This ensures:

- All months are included, even if data exists in only one dataset

A temporary table `#tmpresults` stores the merged results.

Final Output Columns

Column	Description
Operator	Operator name
Year / Month	Reporting period
BooKing Total-C	Revenue from created bookings
Quantity Booked-C	Count of created bookings
Average Booking-C	Avg value per created booking
Booking Total-O	Revenue from operational bookings
Quantity Booked-O	Count of operational bookings
Average Booking-O	Avg value per operational booking
Invoice Total	Revenue from invoiced bookings

Column	Description
Quantity Invoiced	Count of invoiced bookings
Average Invoice	Avg invoiced value
Ratio Closed Qty	% of bookings invoiced (count)
Ratio Closed Revenue	% of revenue invoiced
Revenue Lost	Booked revenue not invoiced
Revenue Last Year	Prior year invoiced revenue

Business Logic Notes

Revenue Calculation

All revenue is calculated as:

```
price_quoted - tax1 - tax2
```

- Taxes are excluded
- NULL taxes are treated as 0

Booking Type Exclusions

The following booking types are excluded:

- 5 - Sub Rentals
- 11 - Location Transfers
- 13 - Sundry Transfers

Invoicing Logic

- A booking is considered invoiced when:

  ```
  invoiced = 'Y'
  ```

- Invoice totals are calculated using conditional aggregation

---

Key Metrics Explained

Ratio Closed Qty

```
QtyInvoiced / QtyBooked-O
```

Indicates operational conversion rate.

Ratio Closed Revenue

```
InvoiceTotal / BookingTotal-O
```

Indicates revenue realization efficiency.

Revenue Lost

```
BookingTotal-O - InvoiceTotal
```

Represents potential revenue not invoiced.

Use Cases

This report is commonly used for:

- Operator performance tracking
- Revenue conversion analysis
- Year-over-year comparisons
- Identifying invoicing gaps

Summary

This stored procedure provides a **comprehensive operator-level monthly report** combining:

- Booking activity
- Invoicing performance
- Historical comparison

It is designed for **operational reporting and financial analysis**, with dynamic date handling to ensure it always reflects current data.

---

*/

Running the Report

/* Complete step 2 before trying to use this report.

Add the code below to EXCEL Query Builder

*/

```
EXEC dbo.usp_OperatorBookingReport_From2023_Auto;
```

BEFORE Running the Report

/* Open SQL Server Management Studio and execute the code below to create a stored procedure */

```
CREATE PROCEDURE [dbo].[usp_OperatorBookingReport_From2023_Auto]

AS

BEGIN

    SET NOCOUNT ON;



    DECLARE @StartYear INT = 2022;

    DECLARE @EndYear INT = YEAR(GETDATE());



    DECLARE @StartDate DATE = DATEFROMPARTS(@StartYear, 1, 1);

    DECLARE @EndDate DATE   = GETDATE();



    DECLARE @PrevStartDate DATE = DATEADD(YEAR, -1, @StartDate);

    DECLARE @PrevEndDate DATE   = DATEADD(YEAR, -1, @EndDate);



    IF OBJECT_ID('tempdb..#tmpresults') IS NOT NULL

        DROP TABLE #tmpresults;
```

```
------------------------------------------------------------
-- Revenue Last Year (based on ddate)
------------------------------------------------------------
;WITH RevenueLastYear AS (
    SELECT
        b.operatorsid,
        YEAR(DATEADD(YEAR,1,b.ddate)) AS [Year],
        MONTH(b.ddate) AS MonthNum,
        SUM(b.price_quoted - ISNULL(b.tax1,0) - ISNULL(b.tax2,0)) AS RevenueLastYear
    FROM vwBookAndHist b
    WHERE b.ddate BETWEEN @PrevStartDate AND @PrevEndDate
      AND b.invoiced = 'Y'
      AND b.booking_type_v32 NOT IN (5,11,13)
    GROUP BY b.operatorsid, YEAR(DATEADD(YEAR,1,b.ddate)), MONTH(b.ddate)
),

------------------------------------------------------------
-- Created (ENTRYDATE)
------------------------------------------------------------
CreatedData AS (
    SELECT
        ISNULL(o.id, b.operatorsid) AS OperatorID,
        ISNULL(o.firstname,'Unknown') AS Operator,
        YEAR(b.entrydate) AS RYear,
        MONTH(b.entrydate) AS MonthNum,

        SUM(b.price_quoted - ISNULL(b.tax1,0) - ISNULL(b.tax2,0)) AS [BKTotal-C],
        COUNT(CASE WHEN b.entrydate IS NOT NULL THEN 1 END) AS [QtyBkd-C]
    FROM vwBookAndHist b
    LEFT JOIN dbo.tbloperators o
        ON b.operatorsid = o.id
    WHERE b.entrydate BETWEEN @StartDate AND @EndDate
      AND b.booking_type_v32 NOT IN (5,11,13)
    GROUP BY ISNULL(o.id, b.operatorsid), o.firstname, YEAR(b.entrydate), MONTH(b.entrydate)
),

------------------------------------------------------------
-- Booked + Invoiced (DDATE)
------------------------------------------------------------
BookedData AS (
    SELECT
        ISNULL(o.id, b.operatorsid) AS OperatorID,
        ISNULL(o.firstname,'Unknown') AS Operator,
        YEAR(b.ddate) AS RYear,
        MONTH(b.ddate) AS MonthNum,

        SUM(b.price_quoted - ISNULL(b.tax1,0) - ISNULL(b.tax2,0)) AS [BookingTotal-O],
        COUNT(*) AS [QtyBooked-O],

        SUM(CASE
               WHEN b.invoiced = 'Y'
               THEN b.price_quoted - ISNULL(b.tax1,0) - ISNULL(b.tax2,0)
               ELSE 0
          END) AS [InvoiceTotal],
        SUM(CASE

            WHEN b.invoiced = 'Y'

            THEN 1

             ELSE 0

         END) AS [QtyInvoiced]


    FROM vwBookAndHist b

    LEFT JOIN dbo.tbloperators o

        ON b.operatorsid = o.id

    WHERE (b.ddate BETWEEN @StartDate AND @EndDate )

      AND (b.booking_type_v32 NOT IN (5,11,13))

    GROUP BY ISNULL(o.id, b.operatorsid), o.firstname, YEAR(b.ddate), MONTH(b.ddate)

)


------------------------------------------------------------

-- Merge datasets

------------------------------------------------------------

SELECT

    COALESCE(c.Operator, b.Operator) AS Operator,

    COALESCE(c.RYear, b.RYear) AS RYear,


    DATENAME(MONTH, DATEFROMPARTS(

        COALESCE(c.RYear, b.RYear),

        COALESCE(c.MonthNum, b.MonthNum),

        1

    )) AS RMonth,


    COALESCE(c.MonthNum, b.MonthNum) AS MonthNum,
    ISNULL(c.[BKTotal-C],0) AS [BKTotal-C],

    ISNULL(c.[QtyBkd-C],0) AS [QtyBkd-C],


    ISNULL(b.[BookingTotal-O],0) AS [BookingTotal-O],

    ISNULL(b.[QtyBooked-O],0) AS [QtyBooked-O],


    ISNULL(b.[InvoiceTotal],0) AS [InvoiceTotal],

    ISNULL(b.[QtyInvoiced],0) AS [QtyInvoiced],


    MAX(ISNULL(ry.RevenueLastYear,0)) AS [RevLastYr]


INTO #tmpresults


FROM CreatedData c

FULL OUTER JOIN BookedData b

    ON c.OperatorID = b.OperatorID

    AND c.RYear = b.RYear

    AND c.MonthNum = b.MonthNum


LEFT JOIN RevenueLastYear ry

    ON ry.operatorsid = COALESCE(c.OperatorID, b.OperatorID)

    AND ry.[Year] = COALESCE(c.RYear, b.RYear)

    AND ry.MonthNum = COALESCE(c.MonthNum, b.MonthNum)


GROUP BY

    COALESCE(c.Operator, b.Operator),

    COALESCE(c.RYear, b.RYear),
        COALESCE(c.MonthNum, b.MonthNum),

        c.[BKTotal-C], c.[QtyBkd-C],

        b.[BookingTotal-O], b.[QtyBooked-O],

        b.[InvoiceTotal], b.[QtyInvoiced];


    ------------------------------------------------------------

    -- Final output (UNCHANGED column names)

    ------------------------------------------------------------

    SELECT

        Operator,

        RYear as [Year],

        RMonth as [Month],


        [BKTotal-C] as [BooKing Total-C],

        [QtyBkd-C] as [Quantity Booked-C],

        [BKTotal-C]/NULLIF([QtyBkd-C],0) as [Average Booking-C],


        [BookingTotal-O] as [Booking Total-O],

        [QtyBooked-O] as [Quantity Booked-O],

        [BookingTotal-O]/NULLIF([QtyBooked-O],0) as [Average Booking-O],


        [InvoiceTotal] as [Invoice Total],

        [QtyInvoiced] as [Quantity Invoiced],

        [InvoiceTotal]/NULLIF([QtyInvoiced],0) as [Average Invoice],


        CAST(100.0 * [QtyInvoiced] / NULLIF([QtyBooked-O],0) AS DECIMAL(6,2)) as [Ratio Closed
Qty],

        CAST(100.0 * [InvoiceTotal] / NULLIF([BookingTotal-O],0) AS DECIMAL(6,2)) as [Ratio Closed
Revenue],
        [BookingTotal-O] - [InvoiceTotal] as [Revenue Lost],

        [RevLastYr] as [Revenue Last Year]


    FROM #tmpresults

    ORDER BY

        [Year],

        MonthNum,

        [Operator];


END;
```