SQL Query –

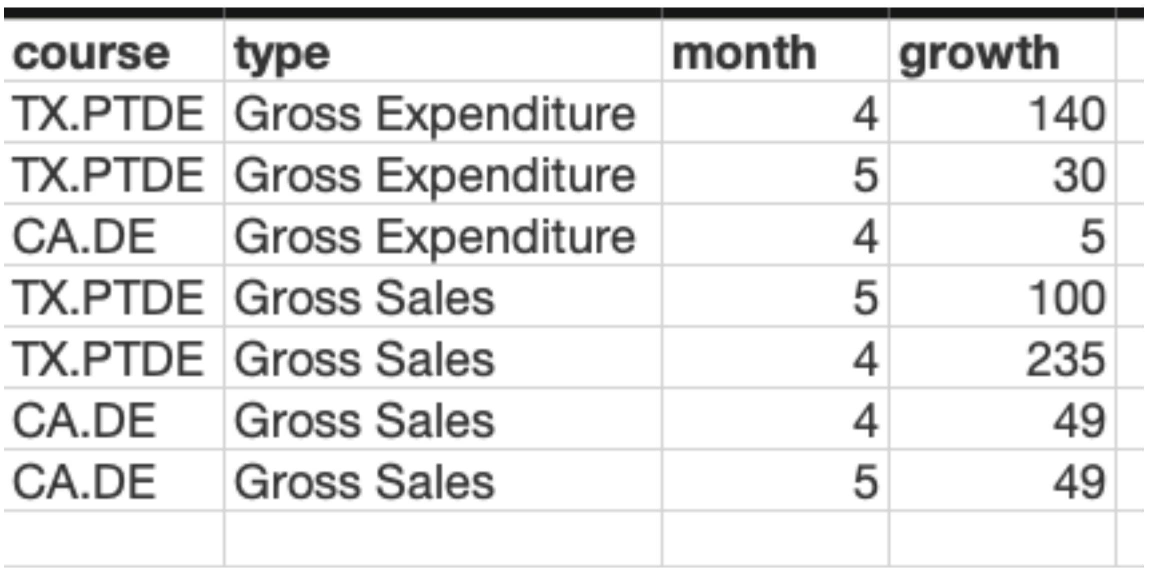
Calculate the monthly year over year growth for sales and cost by product.

Syntax (PostgreSQL) –

```
WITH
temp_view AS(
 (select
 r.course,
 to_char(r.date, 'MM') as month,
 to_char(r.date, 'YY') as c_year,
 SUM(r.gross_charge - COALESCE(r.refund_amount,0)) as amount,
 'Gross Sales'::text as type
from course_revenue r
group by r.course, month, c_year)
 UNION ALL
 (select
 r.course,
 to_char(r.date, 'MM') as month,
 to_char(r.date, 'YY') as c_year,
 SUM(r.total_spend) as amount,
 'Gross Expenditure'::text as type
from course_spend r
group by r.course, month, c_year))

SELECT course, type, month,
       SUM(CASE c_year
             WHEN '18' THEN amount
             WHEN '17' THEN -amount
           END) AS growth
FROM  temp_view
GROUP BY course, type, month;
```

Result Snapshot –



course	type	month	growth
TX.PTDE	Gross Expenditure	4	140
TX.PTDE	Gross Expenditure	5	30
CA.DE	Gross Expenditure	4	5
TX.PTDE	Gross Sales	5	100
TX.PTDE	Gross Sales	4	235
CA.DE	Gross Sales	4	49
CA.DE	Gross Sales	5	49

Trigger –

This trigger was part of a database I worked on. The database represented the inventory and customer management of the bookstore at Syracuse University.
This trigger updates the OrderSummary table to reflect total orders and the total amount spent by each customer at the bookstore.

Syntax (T-SQL) –

```
DROP TRIGGER total_date

CREATE TRIGGER total_order ON OrderItem
AFTER INSERT
AS
BEGIN
       DECLARE @curr_customer int;
       SET @curr_customer = (SELECT CustomerID FROM INSERTED)

       DECLARE @new_count int;
       SET @new_count = (SELECT DISTINCT CONVERT(int,os.NoOfUnits)
       FROM OrderSummary os
       INNER JOIN OrderItem oi ON
       os.CustomerID = oi.CustomerID
       WHERE oi.CustomerID = @curr_customer)

       DECLARE @new_sum int;
       SET @new_sum = (SELECT CONVERT(int,RIGHT(Price, LEN(Price) - 1)) FROM INSERTED) +
(SELECT SUM(CONVERT(int,RIGHT(os.TotalPrice, LEN(os.TotalPrice) - 1)))
       FROM OrderSummary os
       INNER JOIN OrderItem oi ON
       os.CustomerID = oi.CustomerID
       WHERE oi.CustomerID = @curr_customer)

       UPDATE OrderSummary
       SET NoOfUnits = @new_count,
           TotalPrice = @new_sum
       WHERE CustomerID = @curr_customer
END
GO
```