

SQL technique:

Use **DISTINCT** *and*
GROUP BY to avoid
using CTEs or sub-
queries

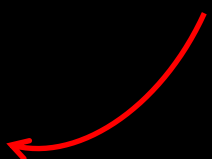
Problem statement:

“Return the maximum single-order sales amount by customer”

We have one row per product/order/customer

```
1  SELECT
2      CustomerKey,
3      OrderDateKey,
4      SalesOrderLineNumber,
5      ProductKey,
6      SalesAmount
7  FROM FactInternetSales fis
8  WHERE CustomerKey = 18005
9  ORDER BY
10     CustomerKey,
11     OrderDateKey,
12     SalesOrderLineNumber;
```

Use a single customer to
explore the problem



	CustomerKey	OrderDateKey	SalesOrderLineNumber	ProductKey	SalesAmount
1	18005	20121220	1	371	2181.5625
2	18005	20130511	1	560	1214.85
3	18005	20130511	2	222	34.99
4	18005	20131119	1	363	2294.99
5	18005	20131119	2	478	9.99
6	18005	20131119	3	477	4.99
7	18005	20131119	4	472	63.50

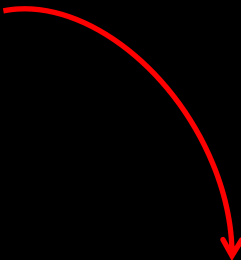
This order has
the largest sum
of SalesAmount



Because each order can have more than one line, we must sum by order first

```
1  SELECT
2      CustomerKey,
3      OrderDateKey,
4      SUM(SalesAmount) AS OrderSalesAmount
5  FROM FactInternetSales fis
6  WHERE CustomerKey = 18005
7  GROUP BY CustomerKey, OrderDateKey
8  ORDER BY CustomerKey, OrderDateKey;
```

	CustomerKey	OrderDateKey	SalesOrderLineNumber	ProductKey	SalesAmount
1	18005	20121220	1	371	2181.5625
2	18005	20130511	1	560	1214.85
3	18005	20130511	2	222	34.99
4	18005	20131119	1	363	2294.99
5	18005	20131119	2	478	9.99
6	18005	20131119	3	477	4.99
7	18005	20131119	4	472	63.50




	CustomerKey	OrderDateKey	OrderSalesAmount
1	18005	20121220	2181.5625
2	18005	20130511	1249.84
3	18005	20131119	2373.47

We can use a CTE for the maximum of the OrderSalesAmount values

```
1  WITH orders
2  AS
3  (
4      SELECT
5          CustomerKey,
6          OrderDateKey,
7          SUM(SalesAmount) AS OrderSalesAmount
8      FROM FactInternetSales fis
9      WHERE CustomerKey = 18005
10     GROUP BY CustomerKey, OrderDateKey
11 )
12 SELECT
13     CustomerKey,
14     MAX(OrderSalesAmount) AS MaxOrderSalesAmount
15 FROM orders
16 GROUP BY CustomerKey;
```

	CustomerKey	OrderDateKey	OrderSalesAmount
1	18005	20121220	2181.5625
2	18005	20130511	1249.84
3	18005	20131119	2373.47

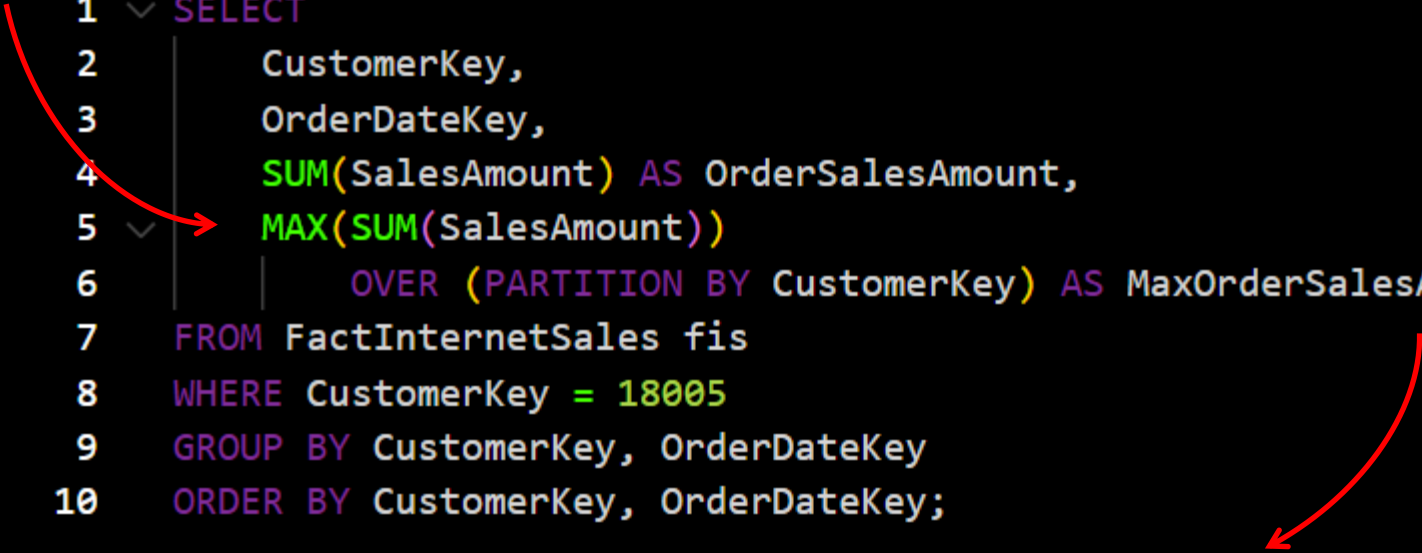
	CustomerKey	MaxOrderSalesAmount
1	18005	2373.47



But, we can also get the MAX *without* a CTE

We can wrap the SUM with a MAX window function, partitioned on CustomerKey

```
1  SELECT
2      CustomerKey,
3      OrderDateKey,
4      SUM(SalesAmount) AS OrderSalesAmount,
5      MAX(SUM(SalesAmount))
6          OVER (PARTITION BY CustomerKey) AS MaxOrderSalesAmount
7  FROM FactInternetSales fis
8  WHERE CustomerKey = 18005
9  GROUP BY CustomerKey, OrderDateKey
10 ORDER BY CustomerKey, OrderDateKey;
```



	CustomerKey	OrderDateKey	OrderSalesAmount	MaxOrderSalesAmount
1	18005	20121220	2181.5625	2373.47
2	18005	20130511	1249.84	2373.47
3	18005	20131119	2373.47	2373.47

And if we remove OrderDateKey from the SELECT clause but leave it in GROUP BY...

```
1  SELECT
2      CustomerKey,
3      MAX(SUM(SalesAmount))
4      OVER (PARTITION BY CustomerKey) AS MaxOrderSalesAmount
5  FROM FactInternetSales fis
6  WHERE CustomerKey = 18005
7  GROUP BY CustomerKey, OrderDateKey
8  ORDER BY CustomerKey;
```

OrderDateKey remains in GROUP BY so that the SUMs are still correct

	CustomerKey	MaxOrderSalesAmount
1	18005	2373.47
2	18005	2373.47
3	18005	2373.47

We now have duplicates!

And how do we get rid of duplicates?

We add DISTINCT!

```
1 SELECT DISTINCT
2     CustomerKey,
3     MAX(SUM(SalesAmount))
4     OVER (PARTITION BY CustomerKey) AS MaxOrderSalesAmount
5 FROM FactInternetSales fis
6 GROUP BY CustomerKey, OrderDateKey
7 ORDER BY CustomerKey;
```

We returned the max single-order sales amount by customer without sub-queries and without CTEs

	CustomerKey	MaxOrderSalesAmount
1	18001	31.27
2	18002	27.28
3	18003	63.97
4	18004	819.48
5	18005	2373.47
6	18006	38.98
7	18007	38.98
8	18008	1759.97
9	18009	1735.98
10	18010	1795.97

