**1.Fabric Sales Per Year Per Month**

SELECT T.fabric\_id, f.fabric\_name, T.total\_price, T.month, T.year

FROM

(

select pfm.fabric\_id,

DATE\_PART(month, transaction\_date) as month,

DATE\_PART(year, transaction\_date) as year,

SUM(t.price) as total\_price

from transactions t

join articles a on t.article\_id = a.article\_id

join product\_fabric\_map pfm on pfm.product\_code = a.product\_code

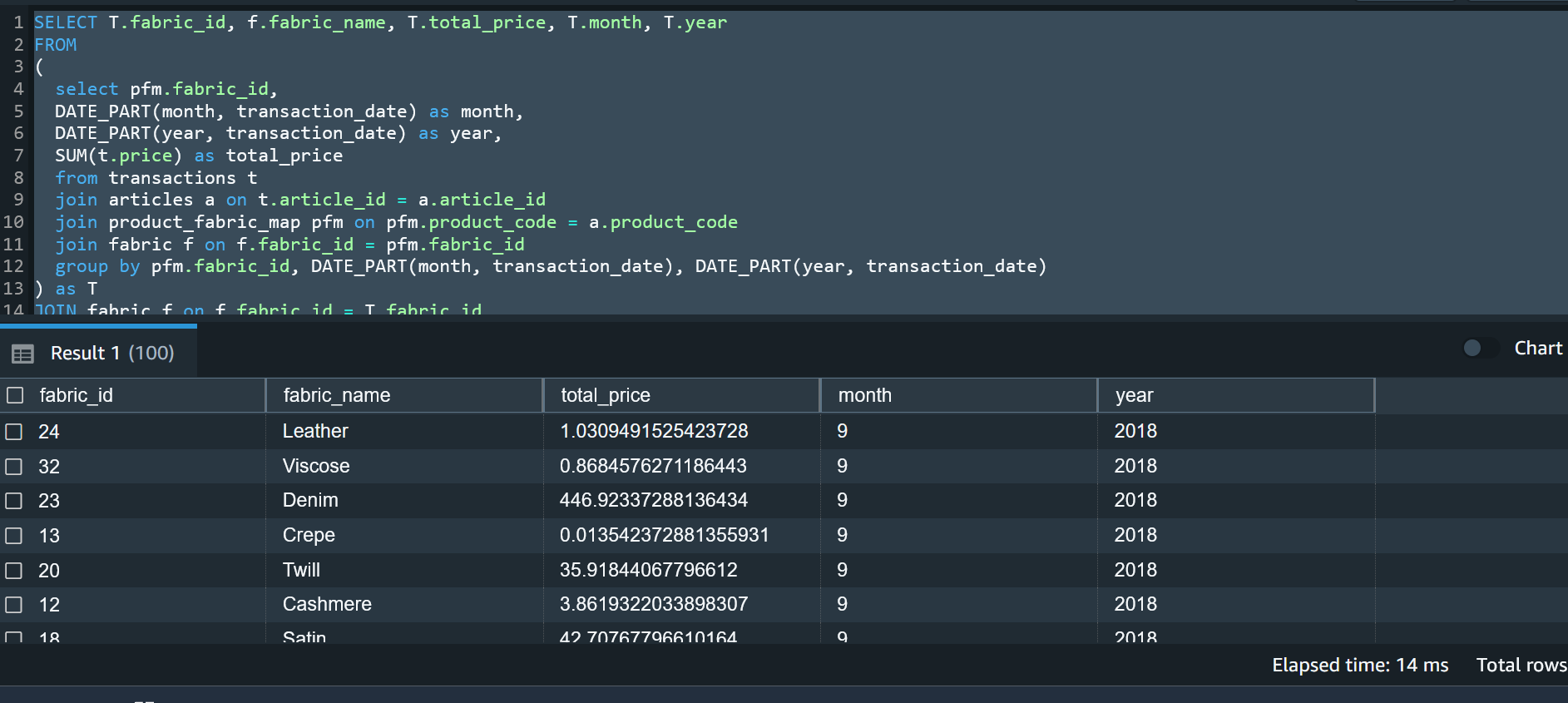
join fabric f on f.fabric\_id = pfm.fabric\_id

group by pfm.fabric\_id, DATE\_PART(month, transaction\_date), DATE\_PART(year, transaction\_date)

) as T

JOIN fabric f on f.fabric\_id = T.fabric\_id

order by T.year, T.month ASC;

****

**2.Overall most sold fabric with total sales**

SELECT T.fabric\_id, f.fabric\_name, T.total\_price

FROM

(

select pfm.fabric\_id, SUM(t.price) as total\_price

from transactions t

join articles a on t.article\_id = a.article\_id

join product\_fabric\_map pfm on pfm.product\_code = a.product\_code

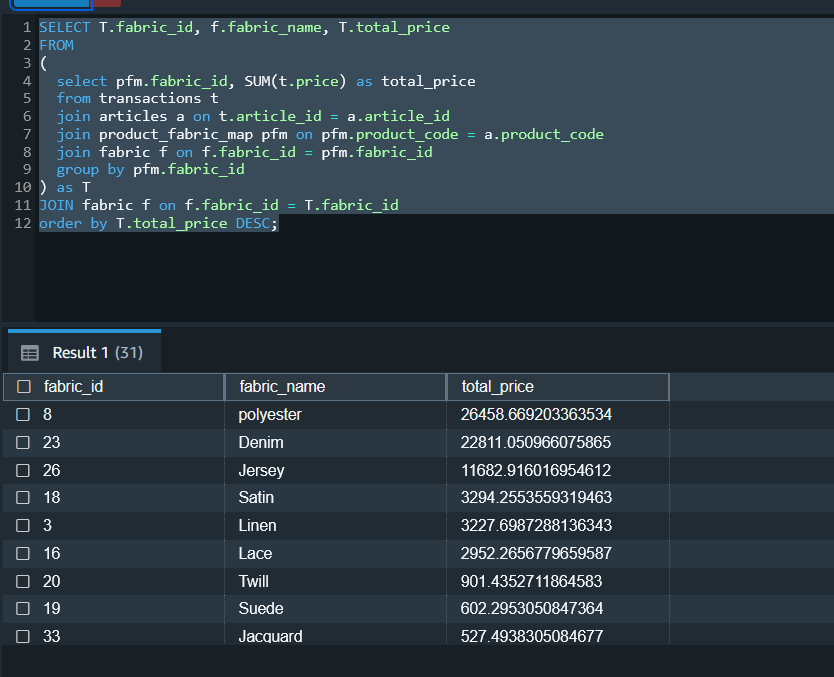
join fabric f on f.fabric\_id = pfm.fabric\_id

group by pfm.fabric\_id

) as T

JOIN fabric f on f.fabric\_id = T.fabric\_id

order by T.total\_price DESC;



**3.Most sold fabric with total sales Per Year Per Month**

SELECT S.fabric\_id, f.fabric\_name, S.month, S.year, S.total\_price

FROM (

select T.fabric\_id, T.month, T.year, T.total\_price,

MAX(T.total\_price) over (partition by T.month, T.year) as max\_sale

from

(

select pfm.fabric\_id,

DATE\_PART(month, transaction\_date) as month,

DATE\_PART(year, transaction\_date) as year,

SUM(t.price) as total\_price

from transactions t

join articles a on t.article\_id = a.article\_id

join product\_fabric\_map pfm on pfm.product\_code = a.product\_code

join fabric f on f.fabric\_id = pfm.fabric\_id

group by pfm.fabric\_id, DATE\_PART(month, transaction\_date), DATE\_PART(year, transaction\_date)

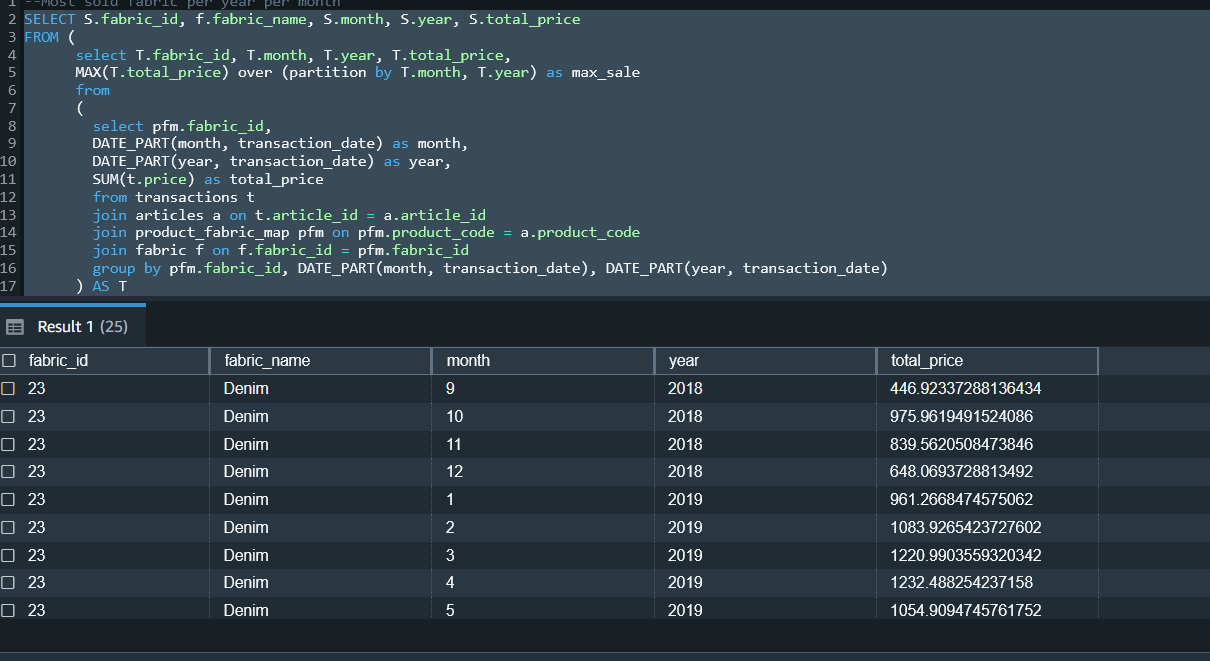
) AS T

) AS S

JOIN fabric f on f.fabric\_id = S.fabric\_id

WHERE S.total\_price = S.max\_sale

order by S.year, S.month;

****

**4. List the number of transactions that happened in each quarter of all the years in order of minimum transactions. Excluding 2018, since it had data for only 4th quarter, we can see that sales were least in 2020 first and third quarter.**

select T.year, T.quarter, min(T.number\_of\_transactions) as MINIMUM\_TRANS\_CNT

from

(select extract(YEAR from transaction\_date) as year,

extract(QUARTER from transaction\_date) as quarter,

count(price) as number\_of\_transactions

from transactions

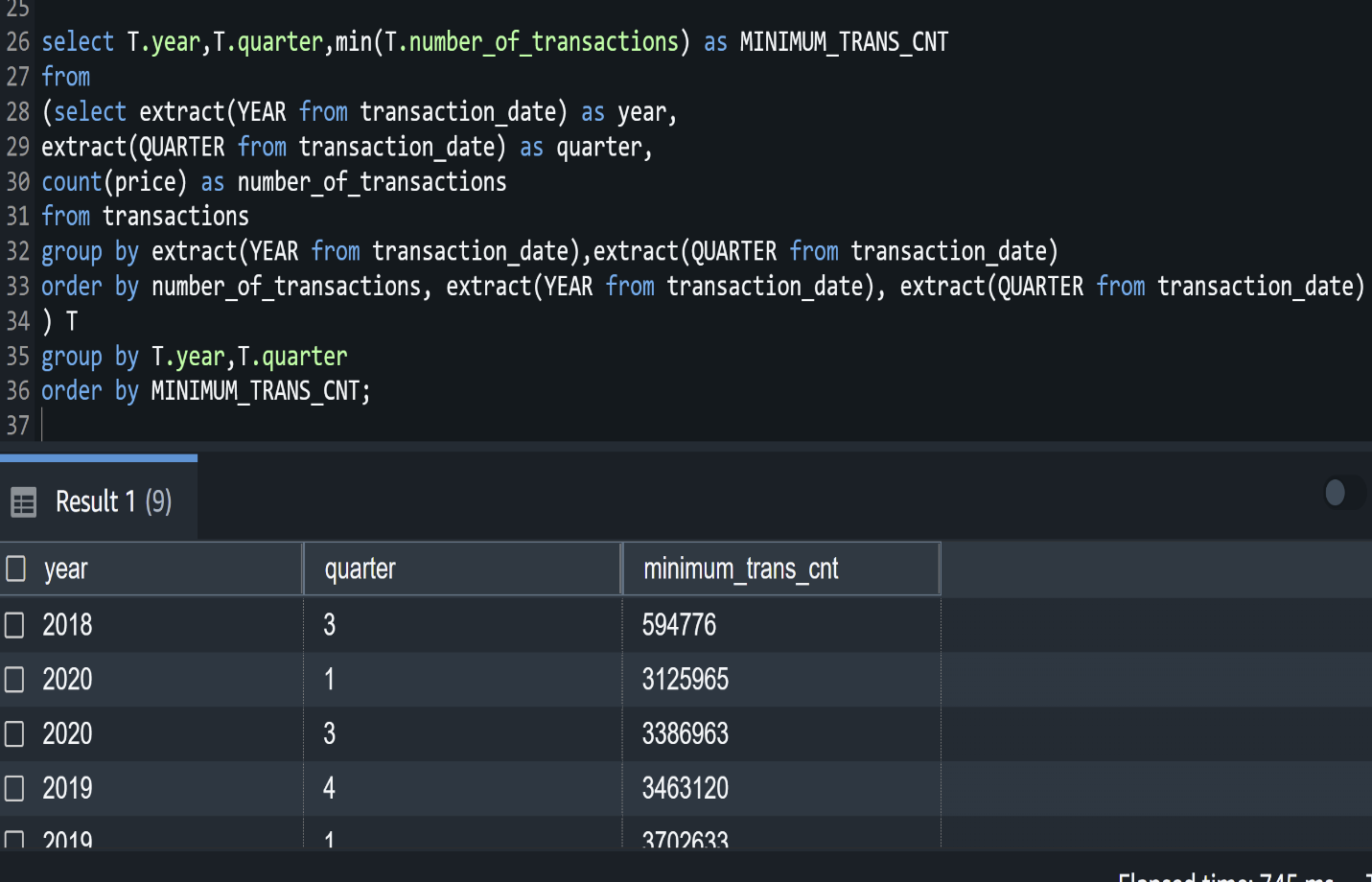
group by extract(YEAR from transaction\_date),extract(QUARTER from transaction\_date)

order by number\_of\_transactions, extract(YEAR from transaction\_date), extract(QUARTER from transaction\_date)

) T

group by T.year,T.quarter

order by MINIMUM\_TRANS\_CNT;

****

**5. List Per day transaction running amount per product type for year 2020.**

select distinct(t.transaction\_date) , pt.type\_name,

sum(t.price) over (PARTITION BY p.prod\_name ORDER BY t.transaction\_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as transaction\_runnin\_total

from transactions t,product p,product\_color\_map pcm,

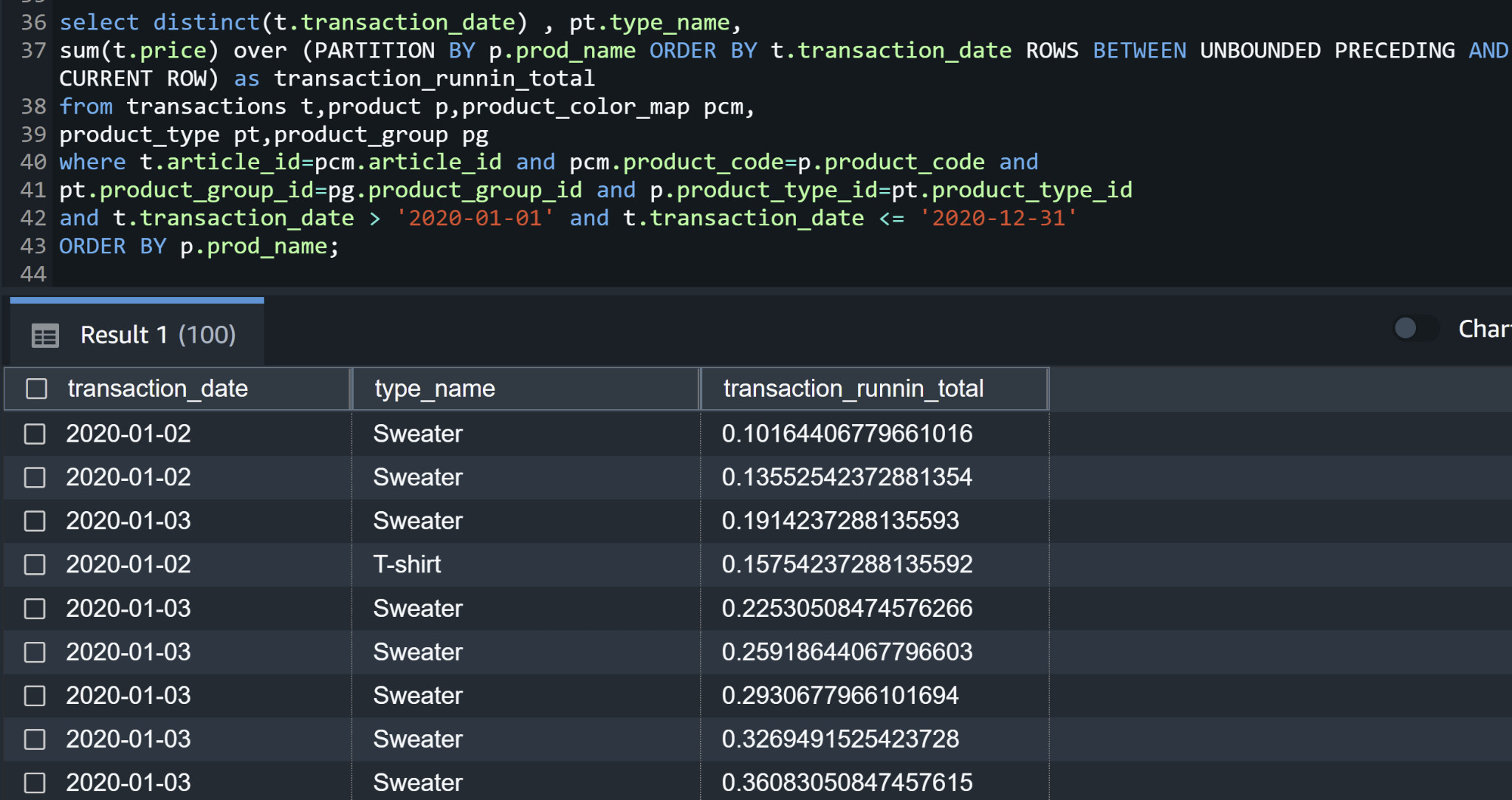
product\_type pt,product\_group pg

where t.article\_id=pcm.article\_id and pcm.product\_code=p.product\_code and

pt.product\_group\_id=pg.product\_group\_id and p.product\_type\_id=pt.product\_type\_id

and t.transaction\_date > '2020-01-01' and t.transaction\_date <= '2020-12-31'

ORDER BY p.prod\_name;



**6. List the first quarter of 2020 with daily sum price and daily price difference**.

select t.transaction\_date , sum(t.price) as daily\_sum\_price,

(sum(t.price) - lag(sum(t.price)) over ( order by t.transaction\_date))

as daily\_difference

from transactions t,product p,product\_color\_map pcm,

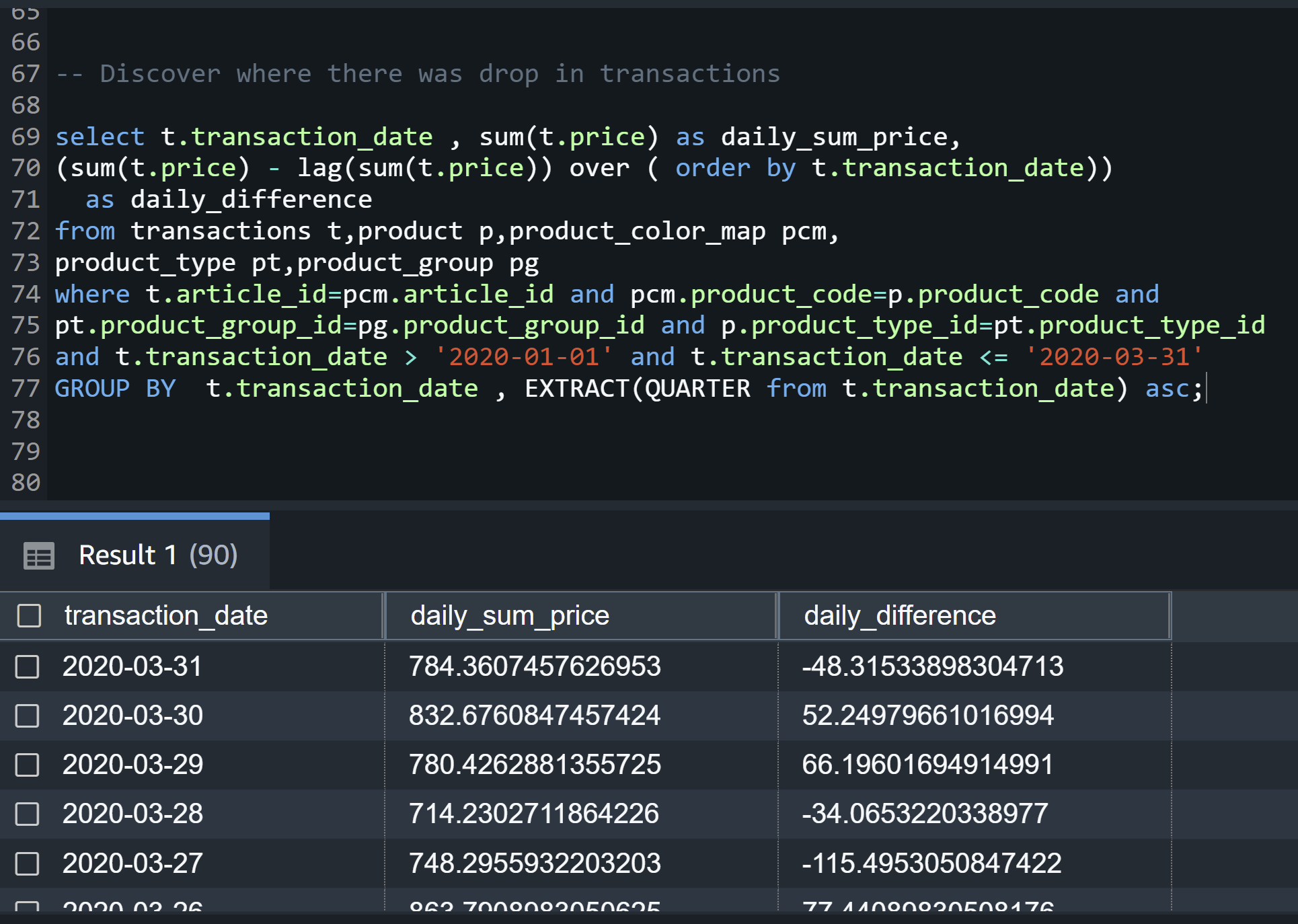
product\_type pt,product\_group pg

where t.article\_id=pcm.article\_id and pcm.product\_code=p.product\_code and

pt.product\_group\_id=pg.product\_group\_id and p.product\_type\_id=pt.product\_type\_id

and t.transaction\_date > '2020-01-01' and t.transaction\_date <= '2020-03-31'

GROUP BY t.transaction\_date , EXTRACT(QUARTER from t.transaction\_date) asc;



**7. ROLL UP for average age of customer along with their skin condition. ROLL UP and CUBE are not supported in AWS REDSHIFT. So, we use the alternative here.**

**select customer\_id, skin\_condition\_level, avg(age) as total\_age**

**from customer**

**group by customer\_id, skin\_condition\_level**

**union all**

**select customer\_id, null as skin\_cond, avg(age) as total\_age**

**from customer**

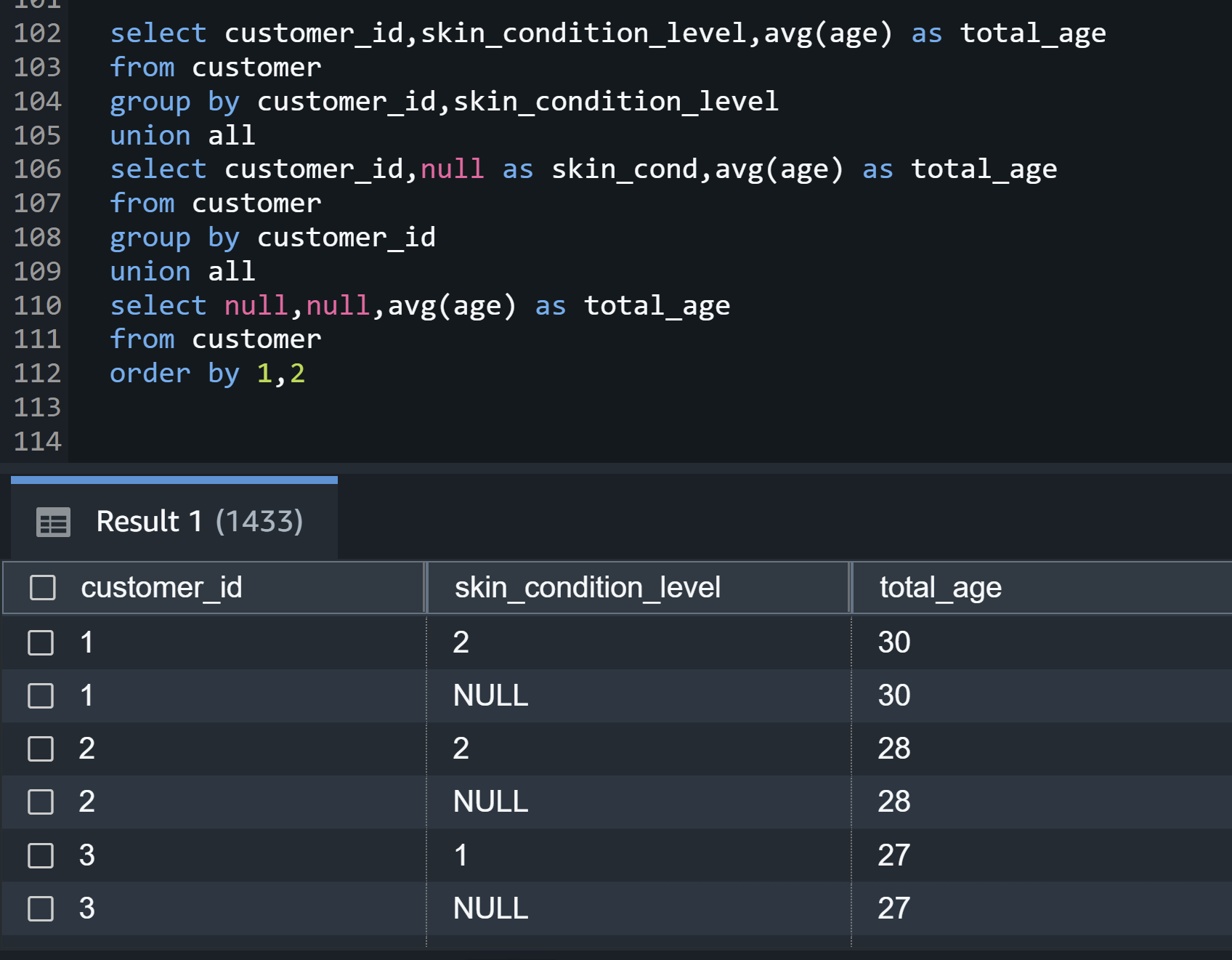
**group by customer\_id**

**union all**

**select null, null, avg(age) as total\_age**

**from customer**

**order by 1,2**

****

**8. Customer sales for the year 2019 and 2020. Sales for 2020 has declined for all customers, most probably due to the pandemic.**

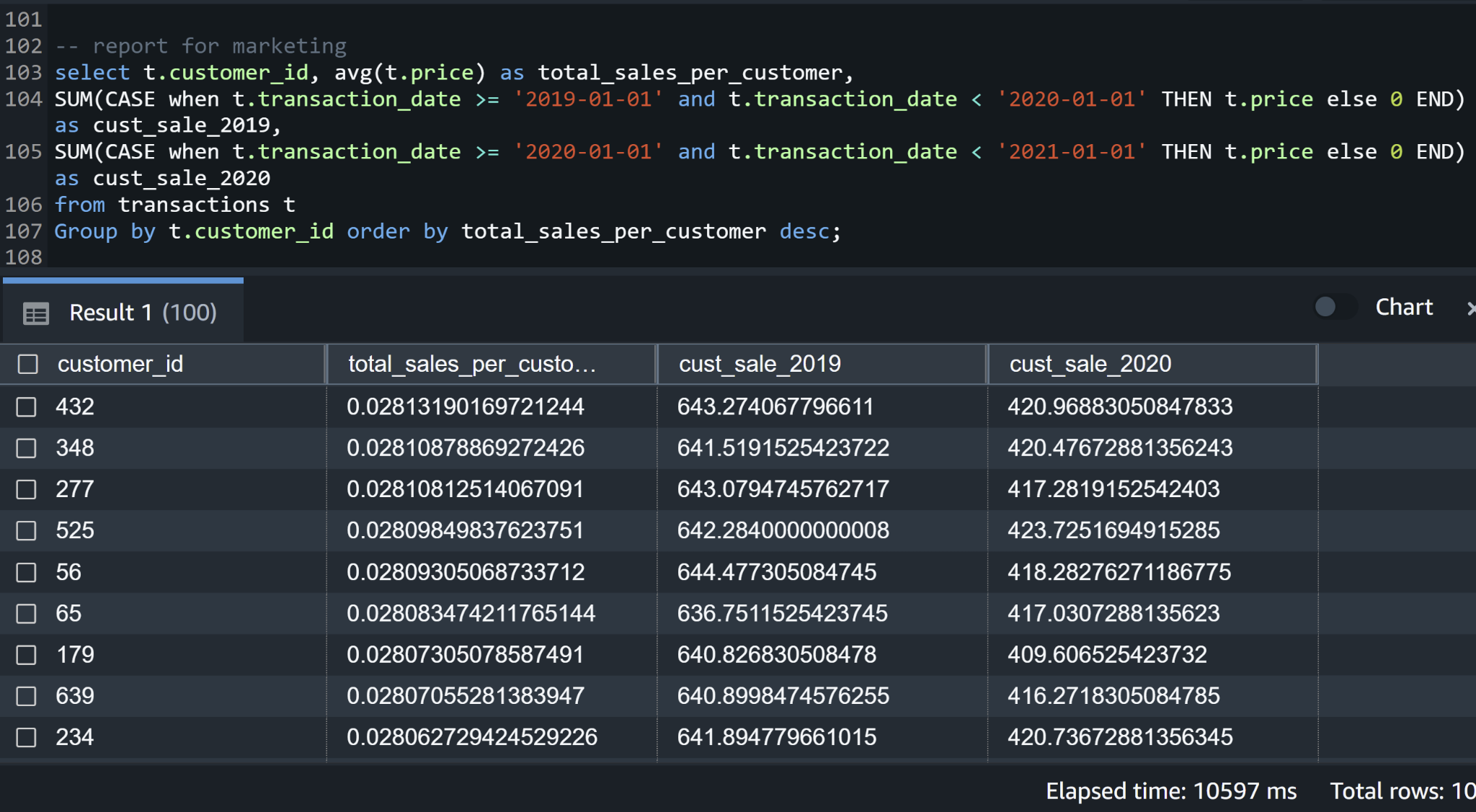
**select t.customer\_id, avg(t.price) as total\_sales\_per\_customer,**

**SUM(CASE when t.transaction\_date >= '2019-01-01' and t.transaction\_date < '2020-01-01' THEN t.price else 0 END) as cust\_sale\_2019,**

**SUM(CASE when t.transaction\_date >= '2020-01-01' and t.transaction\_date < '2021-01-01' THEN t.price else 0 END) as cust\_sale\_2020**

**from transactions t**

**Group by t.customer\_id order by total\_sales\_per\_customer desc;**



**9. Find the cumulative distribution of customer transactions**

select EXTRACT(MONTH from t.transaction\_date) as Month, t.customer\_id, t.price ,

cume\_dist() OVER(

PARTITION BY EXTRACT(MONTH from t.transaction\_date)

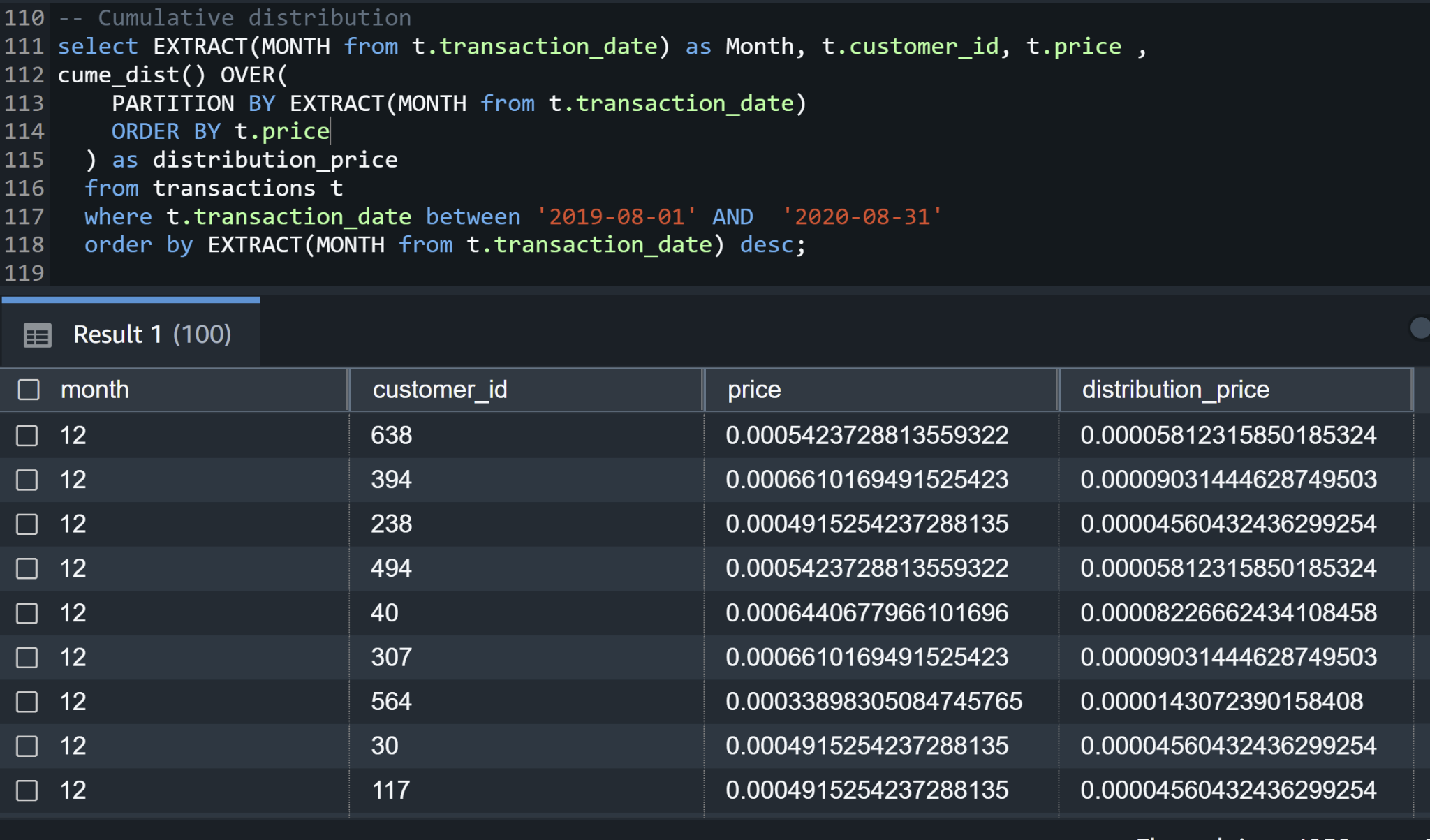
ORDER BY t.price

) as distribution\_price

from transactions t

where t.transaction\_date between '2019-01-01' AND '2020-08-31'

order by EXTRACT(MONTH from t.transaction\_date) desc;



**10. Average pricing per product group.**

select distinct pg.group\_name,avg(price) over (partition by pt.product\_group\_id) AvgPrice

from transactions t,product p,product\_color\_map pcm,

product\_type pt,product\_group pg

where t.article\_id=pcm.article\_id and pcm.product\_code=p.product\_code and

pt.product\_group\_id=pg.product\_group\_id and p.product\_type\_id=pt.product\_type\_id;

