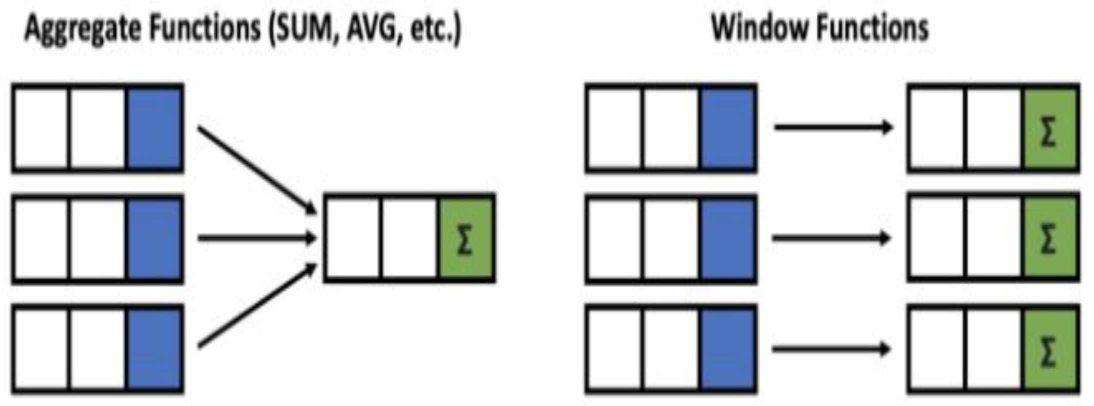
Data Bhau

Class 6
Window Functions

What is a WINDOW FUNCTION

Window functions applies aggregate and ranking functions over a particular window (set of rows)



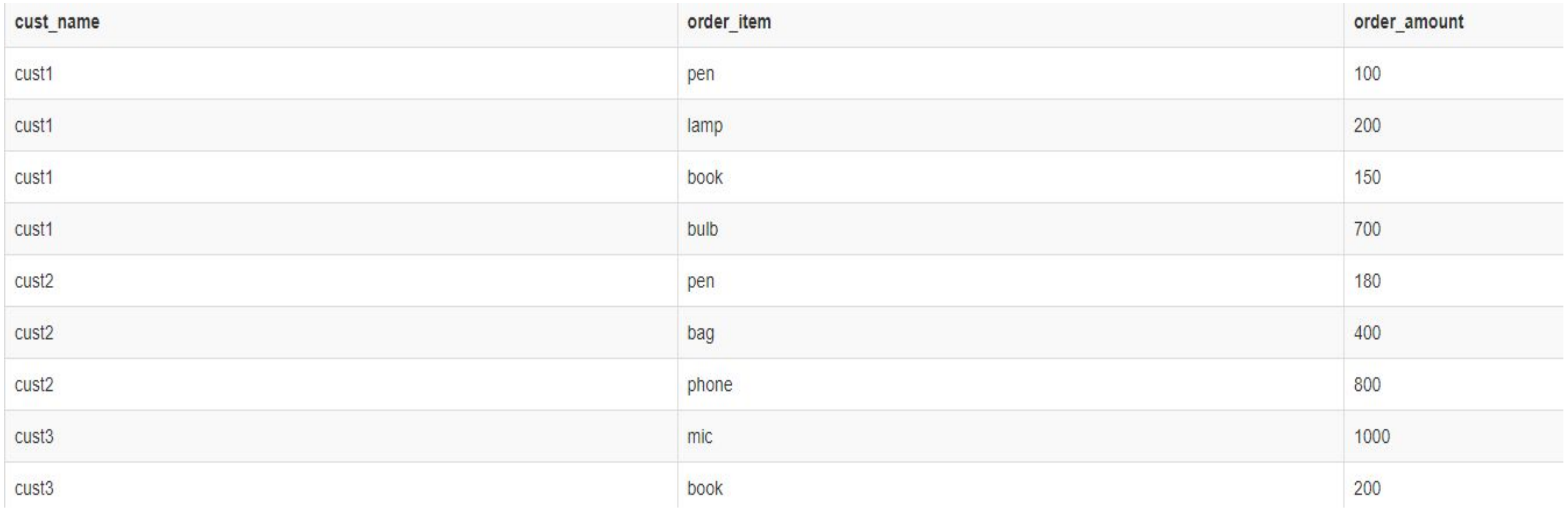
Window functions applies aggregate and ranking functions over a particular window (set of rows). OVER clause is used with window functions to define that window. OVER clause does two things :

- Partitions rows into form set of rows. (PARTITION BY clause is used)
- Orders rows within those partitions into a particular order. (ORDER BY clause is used)

GROUP BY vs WINDOW FUNCTION

We want to create a new column having average order amount per customer

Table sample

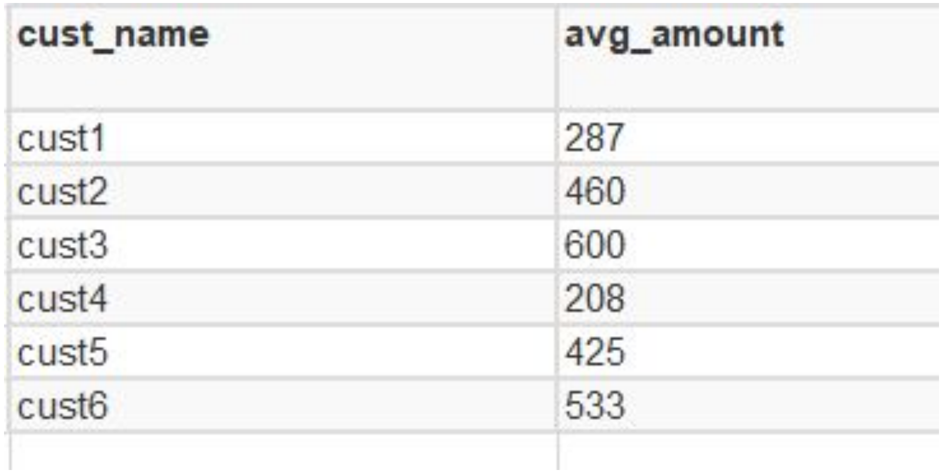


cust_name	order_item	order_amount
cust1	pen	100
cust1	lamp	200
cust1	book	150
cust1	bulb	700
cust2	pen	180
cust2	bag	400
cust2	phone	800
cust3	mic	1000
cust3	book	200

GROUP BY vs WINDOW FUNCTION

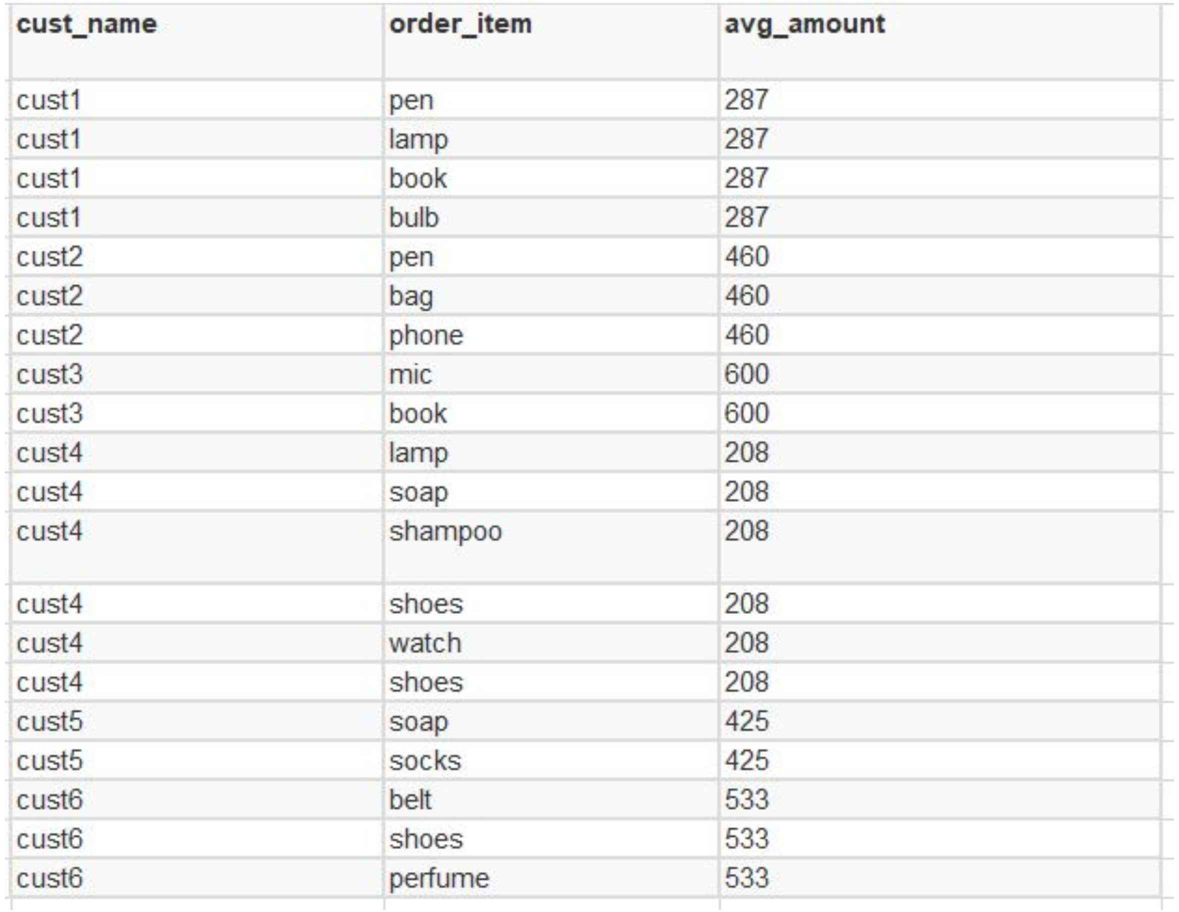
RESULTS

GROUP BY



cust_name	avg_amount
cust1	287
cust2	460
cust3	600
cust4	208
cust5	425
cust6	533

WINDOW FUNCTION



cust_name	order_item	avg_amount
cust1	pen	287
cust1	lamp	287
cust1	book	287
cust1	bulb	287
cust2	pen	460
cust2	bag	460
cust2	phone	460
cust3	mic	600
cust3	book	600
cust4	lamp	208
cust4	soap	208
cust4	shampoo	208
cust4	shoes	208
cust4	watch	208
cust4	shoes	208
cust5	soap	425
cust5	socks	425
cust6	belt	533
cust6	shoes	533
cust6	perfume	533

Why use window Functions?

1. **Allows you to work on both aggregated and unaggregated data at the same time.**

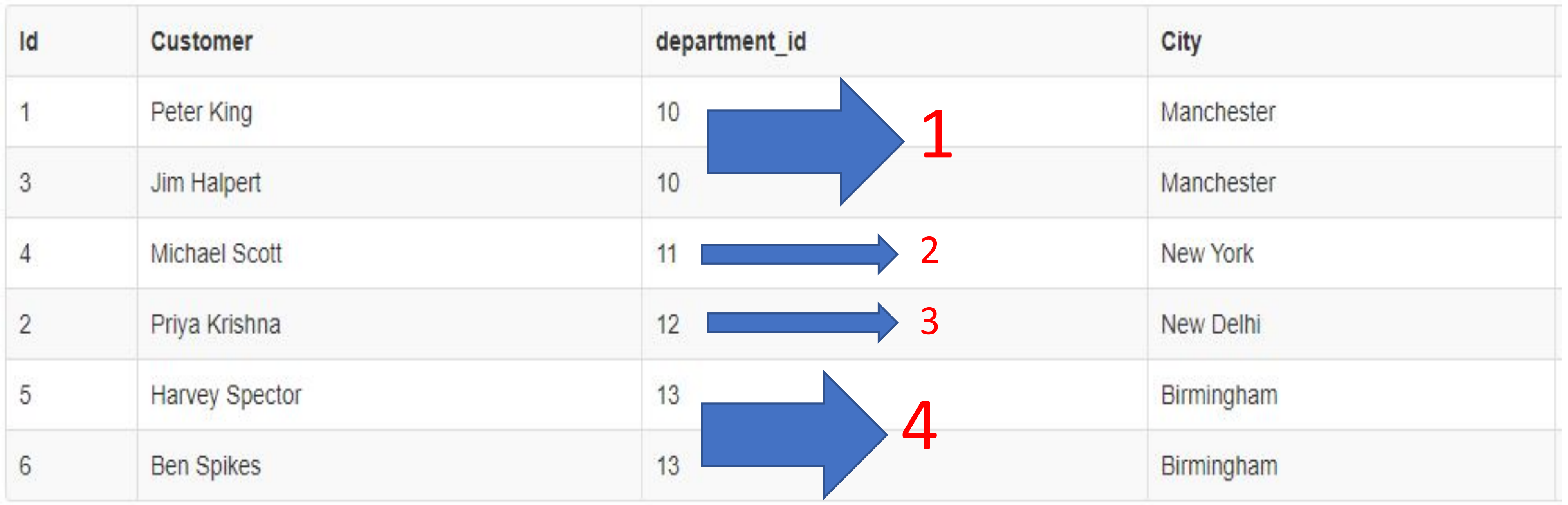
2. **They help with performance issues, as now you might not have to do a join to get any aggregated data into the main table or result**

SYNTAX

```
Select col1, col2, col3,
<window_function> OVER (PARTITION BY <col list>
ORDER BY <col list>)
FROM
Table_name1
```

PARTITION

PARTITION BY department_id



Id	Customer	department_id	City
1	Peter King	10	Manchester
3	Jim Halpert	10	Manchester
4	Michael Scott	11	New York
2	Priya Krishna	12	New Delhi
5	Harvey Spector	13	Birmingham
6	Ben Spikes	13	Birmingham

Basically

PARTITION BY = GROUP BY

We use partition by to group the column for which we want the aggregate of

Window Functions

AGGREGATE BASED

1) Avg()
2) SUM()
3) COUNT()
4) MAX()
5) MIN()

RANK BASED

1) RANK()
2) DENSE_RANK()
3) ROW_NUMBER()

Window Functions

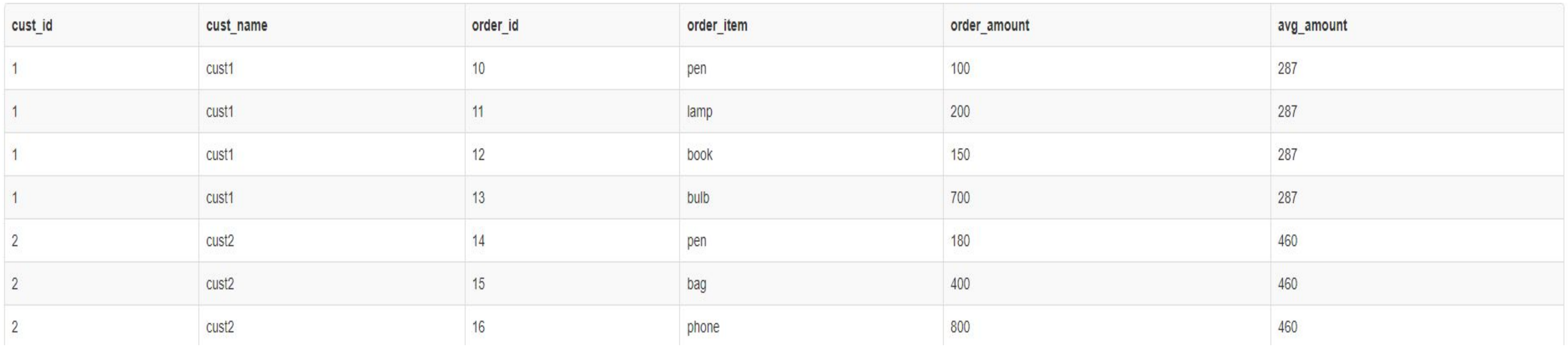
AGGREGATE
BASED

AVG() window function

```
select a.*,

avg(order_amount) OVER(PARTITION BY cust_id) as
avg_amount

from

orders as a
```

Average order amount per customer

Result sample



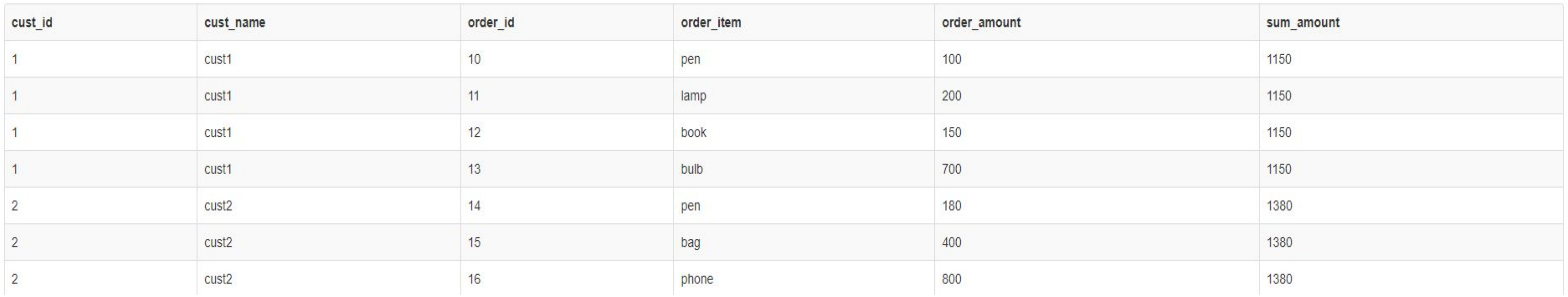
cust_id	cust_name	order_id	order_item	order_amount	avg_amount
1	cust1	10	pen	100	287
1	cust1	11	lamp	200	287
1	cust1	12	book	150	287
1	cust1	13	bulb	700	287
2	cust2	14	pen	180	460
2	cust2	15	bag	400	460
2	cust2	16	phone	800	460

SUM() window function

```
select a.*,

SUM(order_amount) OVER(PARTITION BY cust_id) as
sum_amount

from

Orders as a
```

Sum of order amount per customer

Result sample



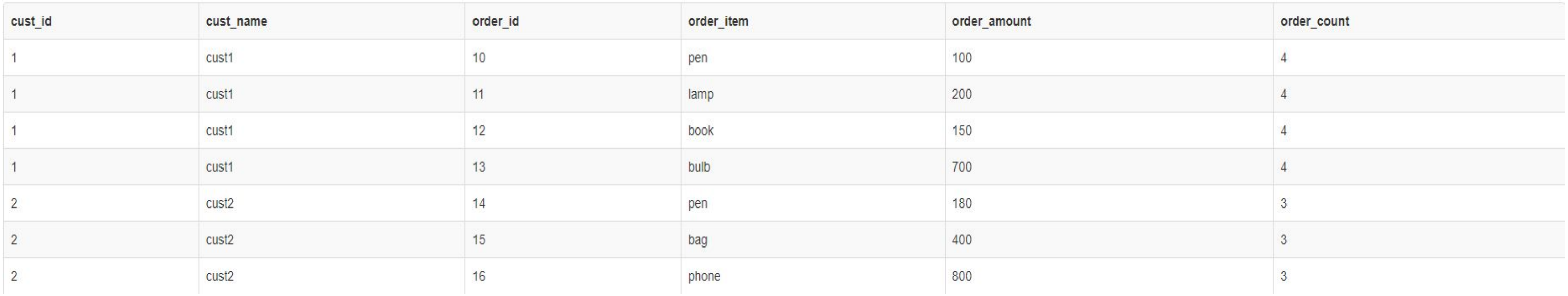
cust_id	cust_name	order_id	order_item	order_amount	sum_amount
1	cust1	10	pen	100	1150
1	cust1	11	lamp	200	1150
1	cust1	12	book	150	1150
1	cust1	13	bulb	700	1150
2	cust2	14	pen	180	1380
2	cust2	15	bag	400	1380
2	cust2	16	phone	800	1380

COUNT() window function

```
select a.*,

COUNT(order_id) OVER(PARTITION BY cust_id) as
order_count

from

Orders as a
```

No. of order amount per customer

Result sample



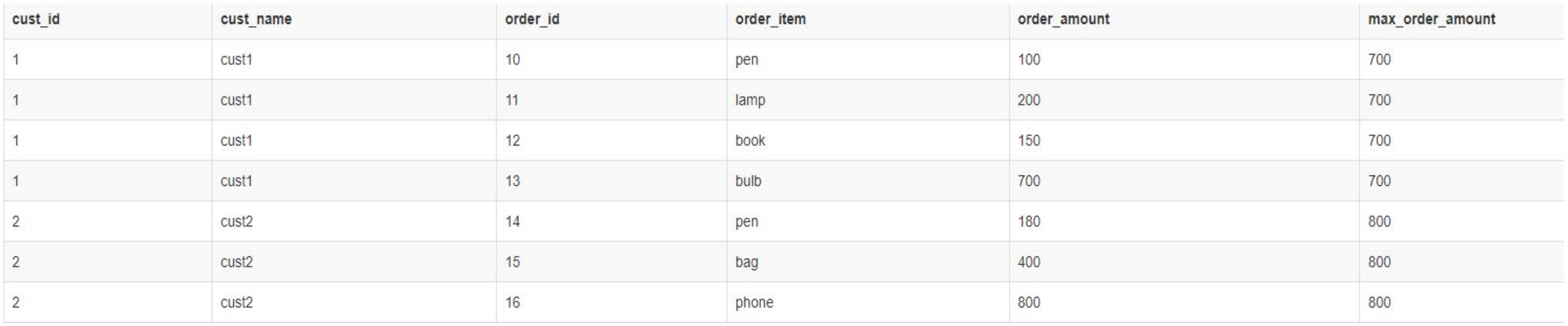
cust_id	cust_name	order_id	order_item	order_amount	order_count
1	cust1	10	pen	100	4
1	cust1	11	lamp	200	4
1	cust1	12	book	150	4
1	cust1	13	bulb	700	4
2	cust2	14	pen	180	3
2	cust2	15	bag	400	3
2	cust2	16	phone	800	3

MAX() window function

```
select a.*,

MAX(order_amount) OVER(PARTITION BY cust_id) as
max_order_amount

from

Orders as a
```

MAX order amount per customer

Result sample



cust_id	cust_name	order_id	order_item	order_amount	max_order_amount
1	cust1	10	pen	100	700
1	cust1	11	lamp	200	700
1	cust1	12	book	150	700
1	cust1	13	bulb	700	700
2	cust2	14	pen	180	800
2	cust2	15	bag	400	800
2	cust2	16	phone	800	800

Window Functions

RANK BASED

ROW_NUMBER() window function

```
select a.*,

ROW_NUMBER() OVER(order by cust_id) as
row_number

from

orders a
```

Add row number column for entire table

Result sample

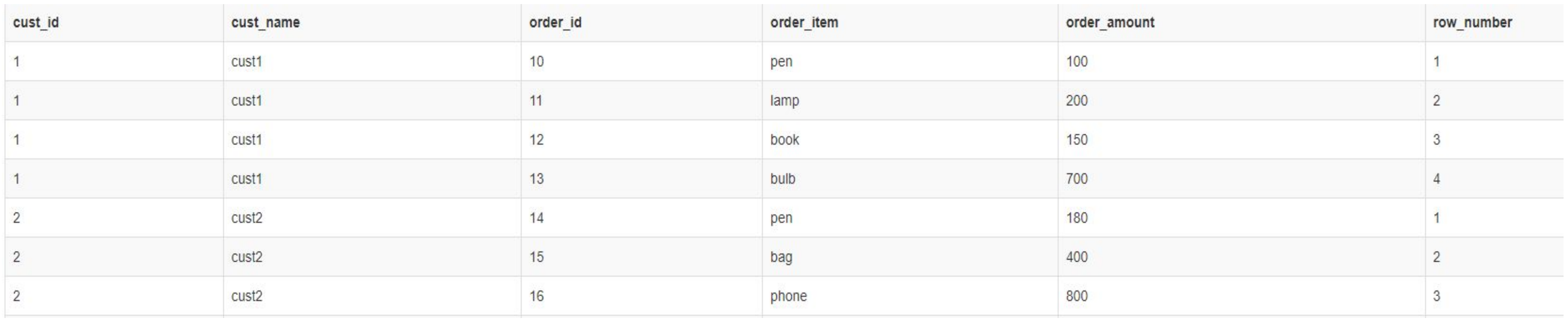
cust_id	cust_name	order_id	order_item	order_amount	row_number
1	cust1	10	pen	100	1
1	cust1	11	lamp	200	2
1	cust1	12	book	150	3
1	cust1	13	bulb	700	4
2	cust2	14	pen	180	5
2	cust2	15	bag	400	6
2	cust2	16	phone	800	7

ROW_NUMBER() window function

```
select a.*,

ROW_NUMBER() OVER(partition by cust_id order by
cust_id) as row_number

from

orders a
```

Add row number column per customer

Result sample



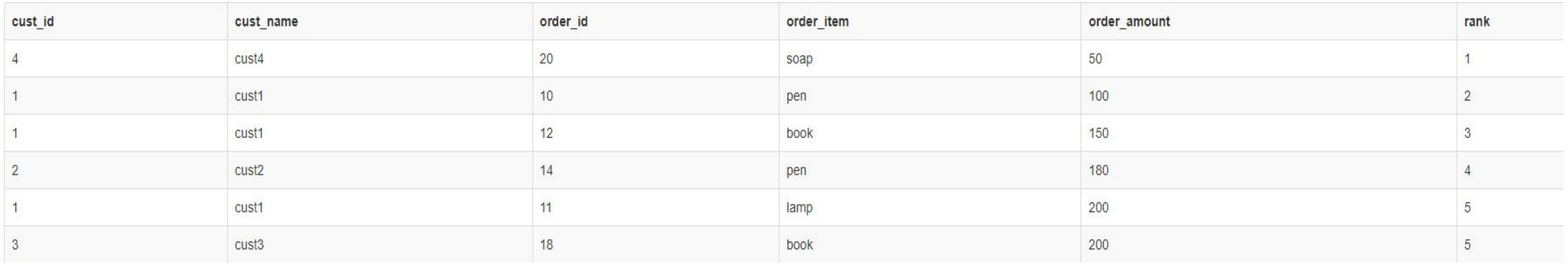
cust_id	cust_name	order_id	order_item	order_amount	row_number
1	cust1	10	pen	100	1
1	cust1	11	lamp	200	2
1	cust1	12	book	150	3
1	cust1	13	bulb	700	4
2	cust2	14	pen	180	1
2	cust2	15	bag	400	2
2	cust2	16	phone	800	3

RANK() window function

```
select a.*,

RANK() OVER(order by order_amount) as
row_number

from

orders a
```

Add rank column based on order amount

Result sample



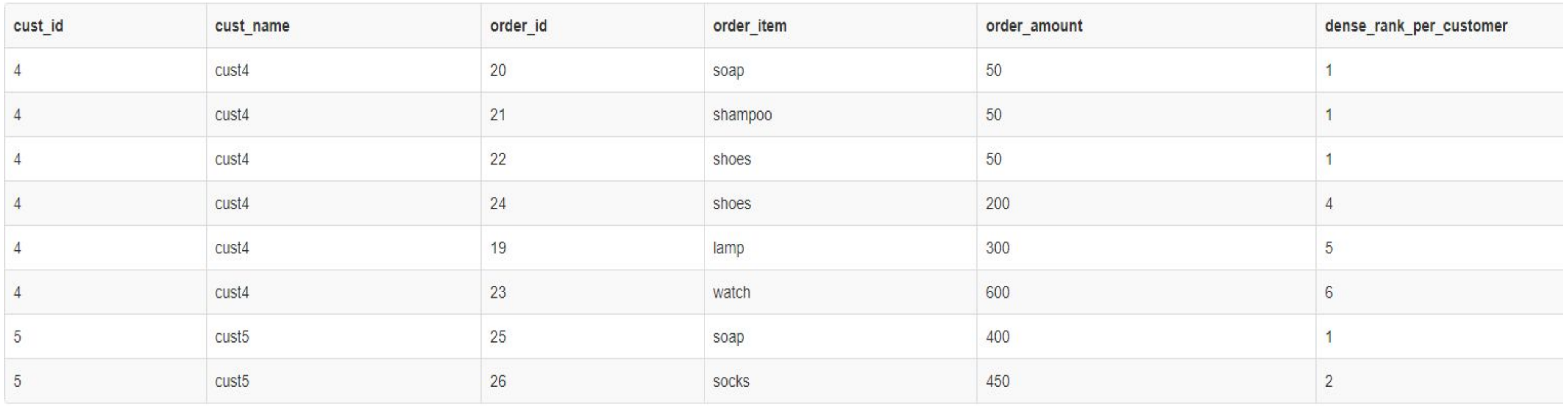
cust_id	cust_name	order_id	order_item	order_amount	rank
4	cust4	20	soap	50	1
1	cust1	10	pen	100	2
1	cust1	12	book	150	3
2	cust2	14	pen	180	4
1	cust1	11	lamp	200	5
3	cust3	18	book	200	5

RANK() window function

```
select a.*,

RANK() OVER(partition by cust_id order by
order_amount) as row_number

from

orders a
```

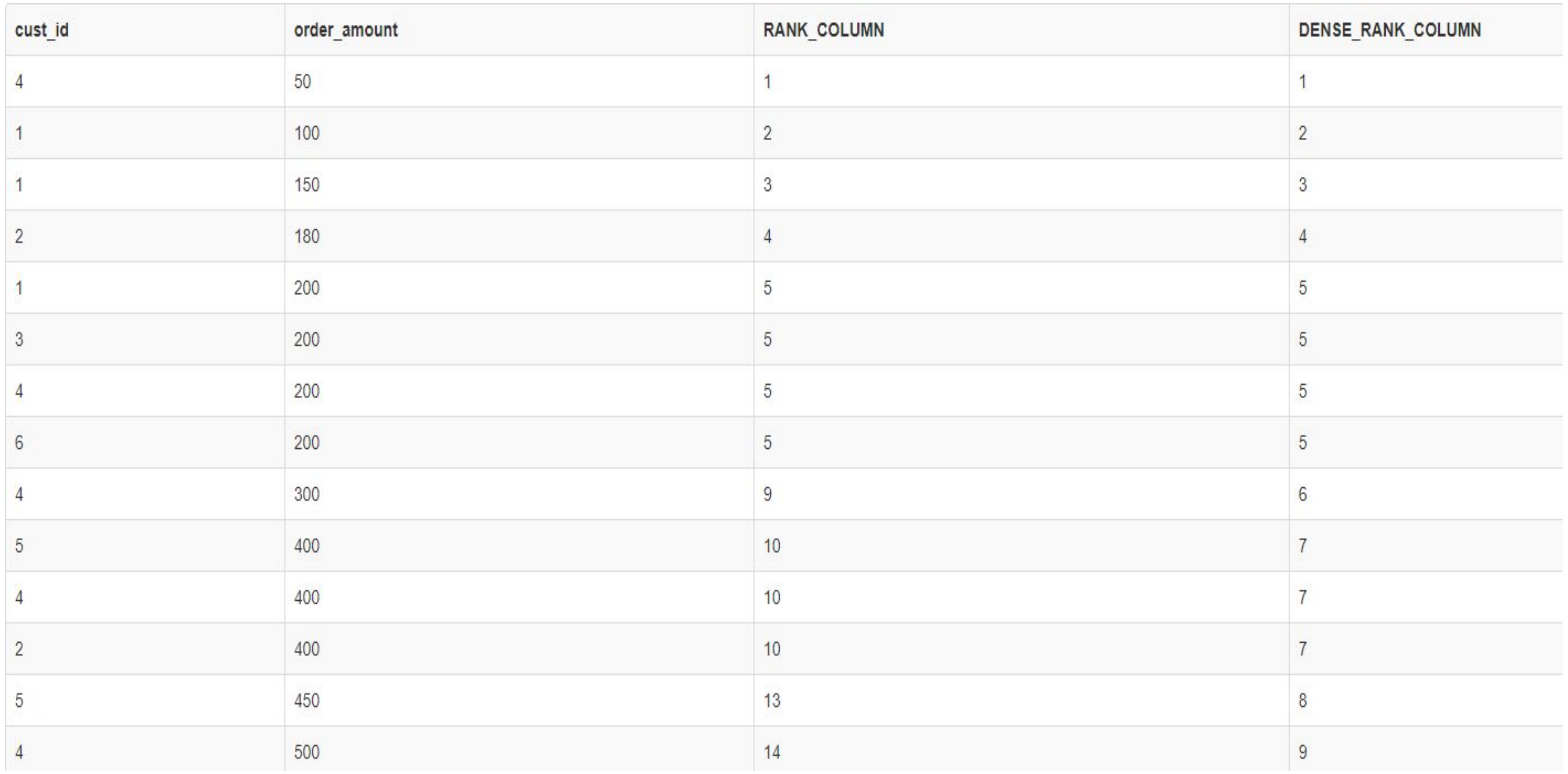
Add rank column based on order amount per customer

Result sample



cust_id	cust_name	order_id	order_item	order_amount	dense_rank_per_customer
4	cust4	20	soap	50	1
4	cust4	21	shampoo	50	1
4	cust4	22	shoes	50	1
4	cust4	24	shoes	200	4
4	cust4	19	lamp	300	5
4	cust4	23	watch	600	6
5	cust5	25	soap	400	1
5	cust5	26	socks	450	2

RANK vs DENSE RANK



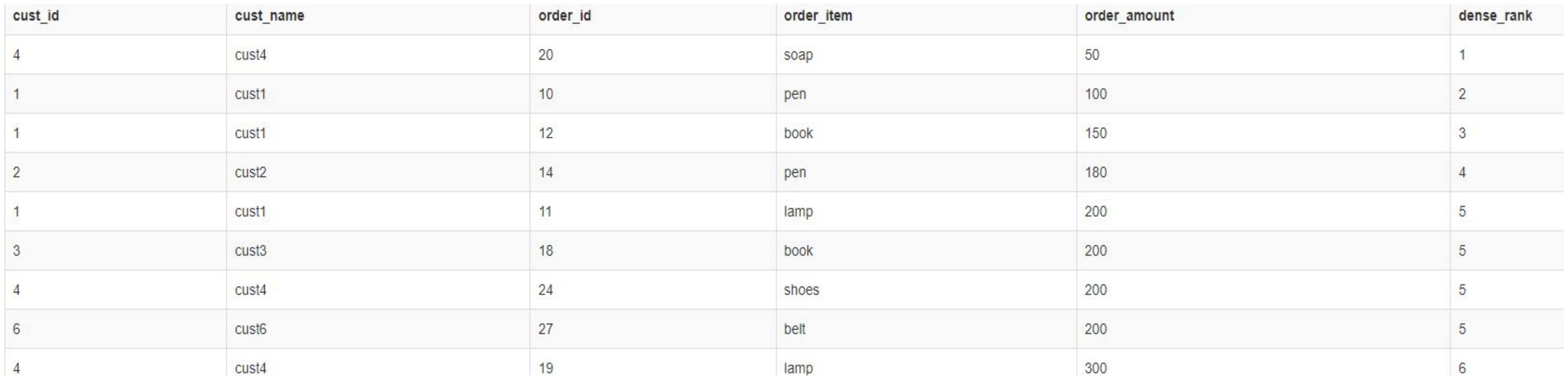
cust_id	order_amount	RANK_COLUMN	DENSE_RANK_COLUMN
4	50	1	1
1	100	2	2
1	150	3	3
2	180	4	4
1	200	5	5
3	200	5	5
4	200	5	5
6	200	5	5
4	300	9	6
5	400	10	7
4	400	10	7
2	400	10	7
5	450	13	8
4	500	14	9

DENSE_RANK() window function

```
select a.*,

DENSE_RANK() OVER(order by order_amount) as
row_number

from

orders a
```

Add dense rank column based on order amount

Result sample



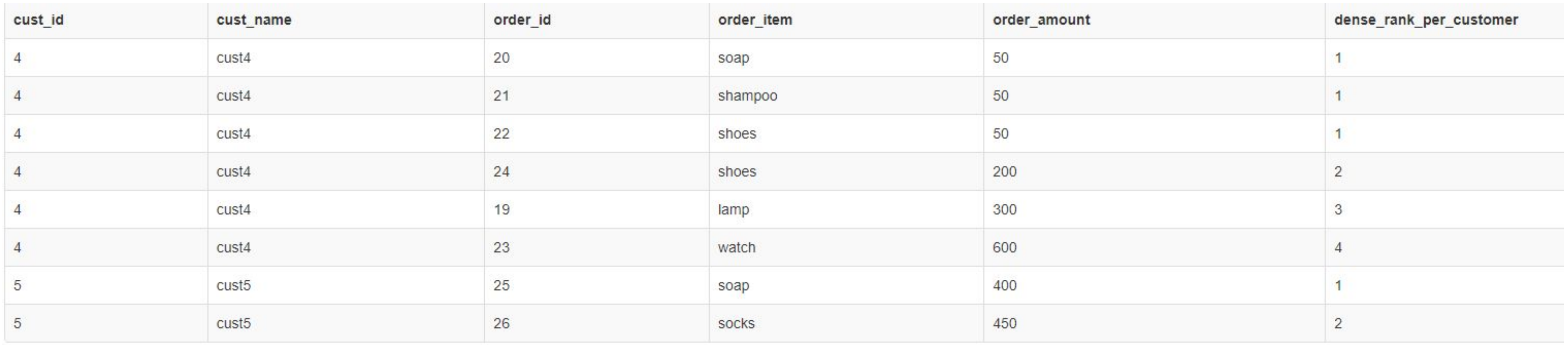
cust_id	cust_name	order_id	order_item	order_amount	dense_rank
4	cust4	20	soap	50	1
1	cust1	10	pen	100	2
1	cust1	12	book	150	3
2	cust2	14	pen	180	4
1	cust1	11	lamp	200	5
3	cust3	18	book	200	5
4	cust4	24	shoes	200	5
6	cust6	27	belt	200	5
4	cust4	19	lamp	300	6

DENSE_RANK() window function

```
select a.*,

RANK() OVER(partition by cust_id order by
order_amount) as row_number

from

orders a
```

Add dense rank column based on order amount per customer

Result sample



cust_id	cust_name	order_id	order_item	order_amount	dense_rank_per_customer
4	cust4	20	soap	50	1
4	cust4	21	shampoo	50	1
4	cust4	22	shoes	50	1
4	cust4	24	shoes	200	2
4	cust4	19	lamp	300	3
4	cust4	23	watch	600	4
5	cust5	25	soap	400	1
5	cust5	26	socks	450	2

Practice Question 1

Find the order with maximum order amount per customer

```
with cte_harsh as

(

select a.*,

RANK() OVER(partition by cust_id order by order_amount desc) as rank_per_customer

from

orders a

)

select * from cte_harsh

where rank_per_customer = 1
```

Practice Question 1

Find customers with more than 3 orders

```
with cte_harsh as

(

select a.*,

COUNT(order_id) OVER(partition by cust_id) as order_count

from

orders a

)

select * from cte_harsh

where order_count > 3
```

Practice Question 1

Find customer with the maximum order amount

```
with cte_harsh as

(

select a.*,

MAX(order_amount) OVER() as max_order_amount

from

orders a

)

select * from cte_harsh

where order_amount = max_order_amount
```

Thanks for attending

DataBhau communication platforms:

- **Whatsapp :** https://chat.whatsapp.com/BaP2CAajm9597J8LwzYE3j
- **Linkedin Handle:** https://www.linkedin.com/company/databhau/
- **Instructors Linkedin Handles:**
    - Shrey Jain : https://www.linkedin.com/in/shrey-jain-74a90b13b/
    - Harsh Katyayan : https://www.linkedin.com/in/harsh-katyayan-a2248316b/