Start-Tech Academy

Windows Functions

Introduction

Row functions

Quarter	Store	Sales	Sales*0.2
1	A	40	8
2	A	60	12
3	A	80	16
2	B	100	20
1	B	60	12

Aggregate functions

Store	Total
A	180
B	160

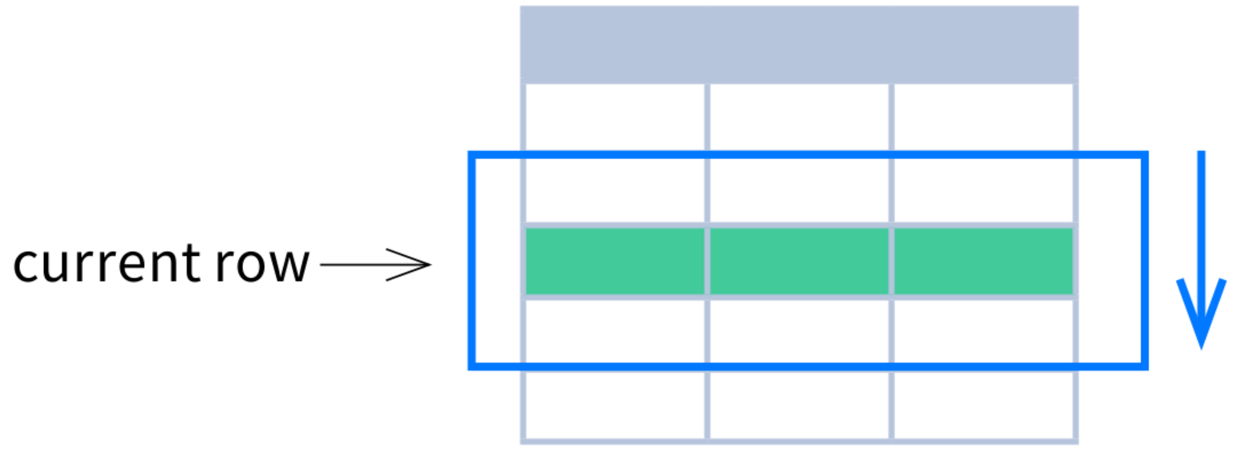
Windows functions

Quarter	Store	Sales	Total
1	A	40	180
2	A	60	180
3	A	80	180
2	B	100	160
1	B	60	160

Windows Functions

Introduction

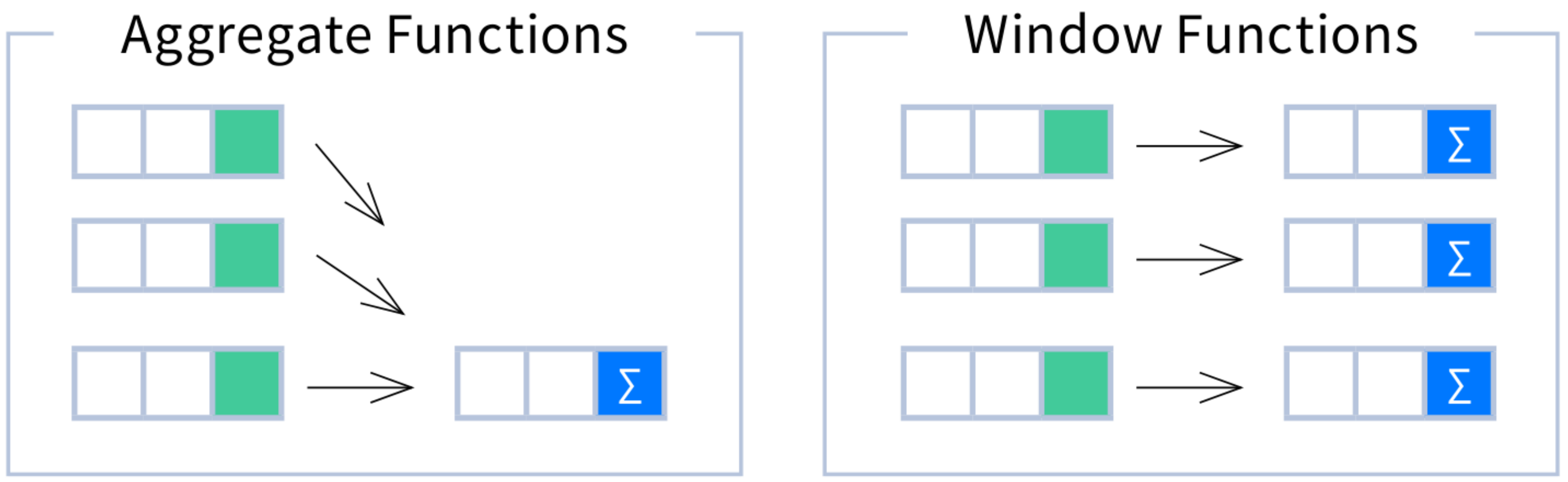
1. Window functions provide the ability to perform calculations across sets of rows that are related to the current query row
2. Window Functions compute their result based on a sliding window frame, a set of rows that are somehow related to the current row.



Windows Functions

How it is different

1. Use of a window function does not cause rows to become grouped into a single output row — the rows retain their separate identities.
2. Behind the scenes, the window function can access more than just the current row of the query result.



Windows Functions

Syntax

```
SELECT <column_1>, <column_2>,
<window_function>() OVER (
PARTITION BY <...>
ORDER BY <...>
FROM <table_name>;
```

Customer	Store	Orders
C-1	A	3
C-2	B	5
C-3	B	4
C-4	B	2
C-5	A	6
C-6	B	4
C-7	A	2

```
SELECT Cust, Store, Orders,
row_number() OVER (
PARTITION BY Store
ORDER BY orders desc) as row
FROM <table_name>;
```

Cust	Store	Orders	Row
C-5	A	6	1
C-7	A	3	2
C-1	A	2	3
C-2	B	5	1
C-3	B	4	2
C-6	B	4	3
C-4	B	2	4

Windows Functions

Row Number

Customer	State	Orders
C-1	A	3
C-2	B	5
C-3	B	4
C-4	B	2
C-5	A	6
C-6	B	4
C-7	A	2

Cust	State	Orders	Row
C-5	A	6	1
C-1	A	3	2
C-7	A	2	3
C-2	B	5	1
C-3	B	4	2
C-6	B	4	3
C-4	B	2	4

Unique number for each row within partition, with different numbers for tied values

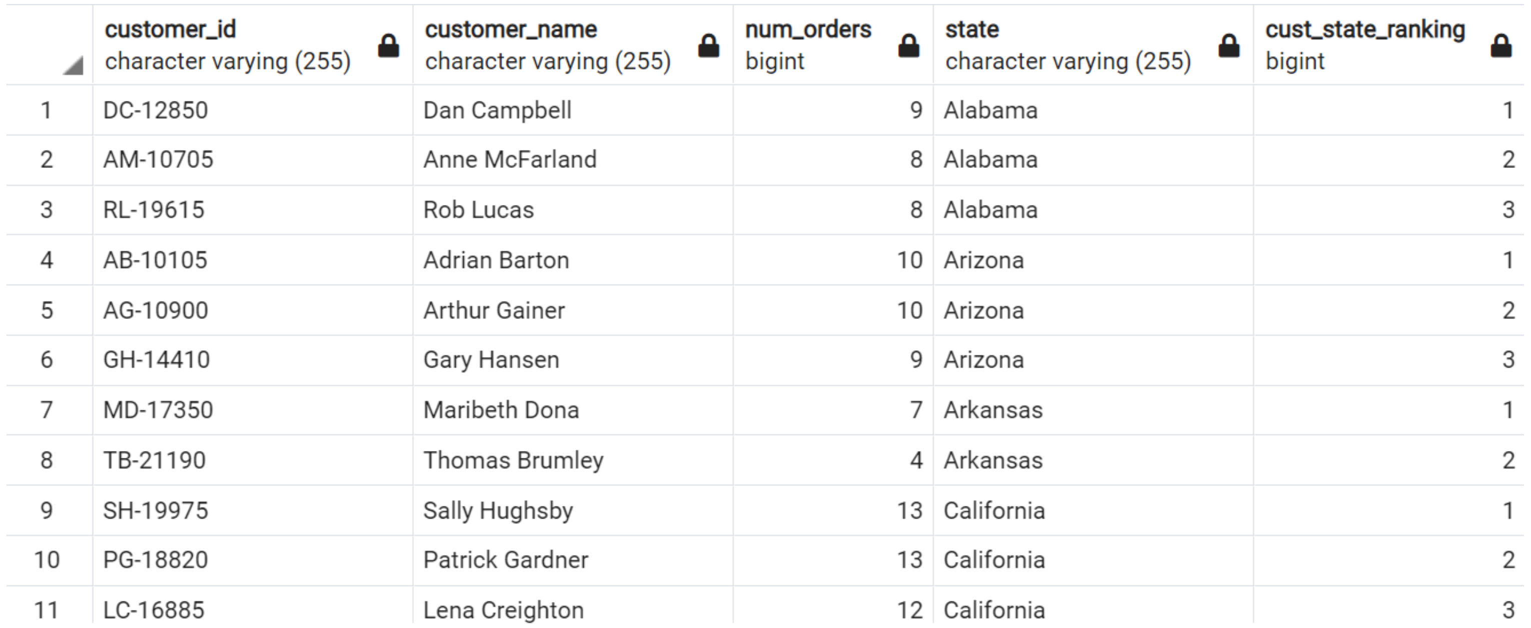
```
SELECT Cust, Store, Orders,
row_number() OVER (
PARTITION BY Store
ORDER BY orders desc) as row
FROM <table_name>;
```

Windows Functions

Example

Suppose we need to create a list of top 3 customers with maximum orders

from each state ?



	customer_id character varying (255)	customer_name character varying (255)	num_orders bigint	state character varying (255)	cust_state_ranking bigint
1	DC-12850	Dan Campbell	9	Alabama	1
2	AM-10705	Anne McFarland	8	Alabama	2
3	RL-19615	Rob Lucas	8	Alabama	3
4	AB-10105	Adrian Barton	10	Arizona	1
5	AG-10900	Arthur Gainer	10	Arizona	2
6	GH-14410	Gary Hansen	9	Arizona	3
7	MD-17350	Maribeth Dona	7	Arkansas	1
8	TB-21190	Thomas Brumley	4	Arkansas	2
9	SH-19975	Sally Hughsby	13	California	1
10	PG-18820	Patrick Gardner	13	California	2
11	LC-16885	Lena Creighton	12	California	3

Windows Functions

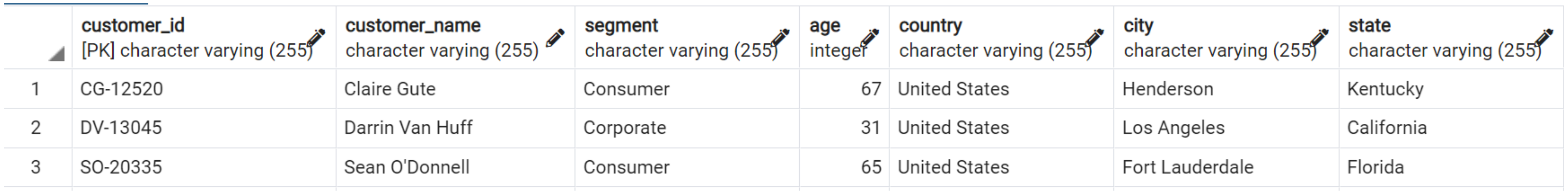
Example

Suppose we need to create a list of top 3 customers with maximum orders from each state ?

1. Combine the customer and orders table
2. Add row numbers within state
3. Filter row_number less than equal to 3

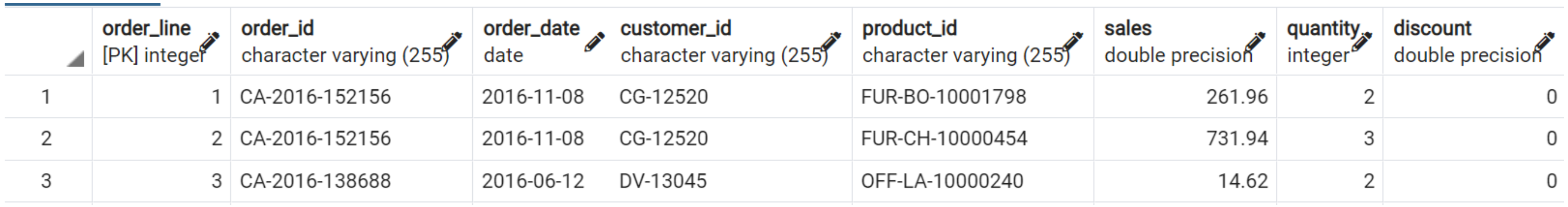
Windows Functions

Customer



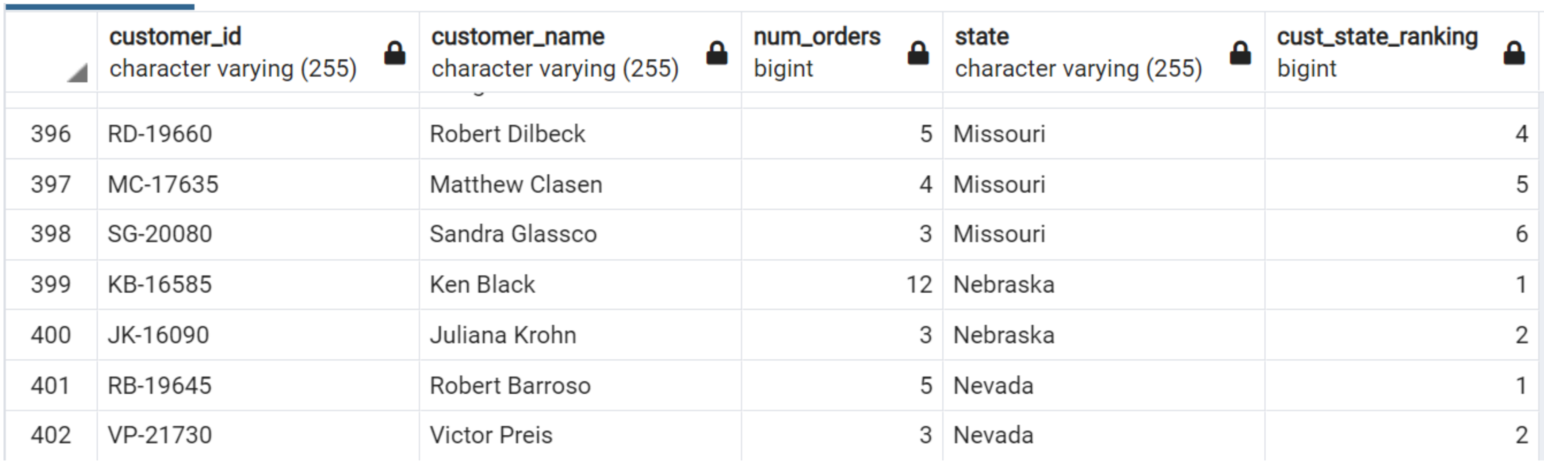
	customer_id [PK] character varying (255)	customer_name character varying (255)	segment character varying (255)	age integer	country character varying (255)	city character varying (255)	state character varying (255)
1	CG-12520	Claire Gute	Consumer	67	United States	Henderson	Kentucky
2	DV-13045	Darrin Van Huff	Corporate	31	United States	Los Angeles	California
3	SO-20335	Sean O'Donnell	Consumer	65	United States	Fort Lauderdale	Florida

Sales



	order_line [PK] integer	order_id character varying (255)	order_date date	customer_id character varying (255)	product_id character varying (255)	sales double precision	quantity integer	discount double precision
1	1	CA-2016-152156	2016-11-08	CG-12520	FUR-BO-10001798	261.96	2	0
2	2	CA-2016-152156	2016-11-08	CG-12520	FUR-CH-10000454	731.94	3	0
3	3	CA-2016-138688	2016-06-12	DV-13045	OFF-LA-10000240	14.62	2	0

Combined



	customer_id character varying (255)	customer_name character varying (255)	num_orders bigint	state character varying (255)	cust_state_ranking bigint
396	RD-19660	Robert Dilbeck	5	Missouri	4
397	MC-17635	Matthew Clasen	4	Missouri	5
398	SG-20080	Sandra Glassco	3	Missouri	6
399	KB-16585	Ken Black	12	Nebraska	1
400	JK-16090	Juliana Krohn	3	Nebraska	2
401	RB-19645	Robert Barroso	5	Nevada	1
402	VP-21730	Victor Preis	3	Nevada	2

Windows Functions

Rank

Customer	Store	Orders
C-1	A	3
C-2	B	5
C-3	B	4
C-4	B	2
C-5	A	6
C-6	B	4
C-7	A	2

Cust	Store	Orders	Row
C-5	A	6	1
C-7	A	3	2
C-1	A	2	3
C-2	B	5	1
C-3	B	4	2
C-6	B	4	2
C-4	B	2	4

Ranking within partition, with gaps and same ranking for tied values

```
SELECT Cust, Store, Orders,
rank() OVER (
PARTITION BY Store
ORDER BY Orders desc) as row
FROM <table_name>;
```

Windows Functions

Dense Rank

Customer	Orders	Orders
C-1	A	3
C-2	B	5
C-3	B	4
C-4	B	2
C-5	A	6
C-6	B	4
C-7	A	2

Cust	Store	Orders	Row
C-5	A	6	1
C-7	A	3	2
C-1	A	2	3
C-2	B	5	1
C-3	B	4	2
C-6	B	4	2
C-4	B	2	3

Ranking within partition, with gaps and same ranking for tied values

```
SELECT Cust, Store, Orders,
dense_rank() OVER (
PARTITION BY Store
ORDER BY Orders desc) as row
FROM <table_name>;
```

Windows Functions

Ntile

Customer	Store	Orders
C-1	A	3
C-2	B	5
C-3	B	4
C-4	B	2
C-5	A	6
C-6	B	4
C-7	A	2

Cust	Store	Orders	group
C-5	A	6	1
C-7	A	3	1
C-1	A	2	2
C-2	B	5	1
C-3	B	4	1
C-6	B	4	2
C-4	B	2	2

divide rows within a partition as equally as possible into n groups, and assign each row its group number

```
SELECT Cust, Store, Orders,
ntile(2) OVER (PARTITION BY Store
ORDER BY Orders desc) as group
FROM <table_name>;
```

Windows Functions - Aggregate

Average

Customer	Store	Revenue
C-1	A	100
C-2	B	300
C-3	B	300
C-4	B	200
C-5	A	200
C-6	B	400
C-7	A	300

Cust	Store	Revenue	Avg_r
C-1	A	100	**200**
C-5	A	200	**200**
C-7	A	300	**200**
C-2	B	300	**300**
C-3	B	300	**300**
C-4	B	200	**300**
C-6	B	400	**300**

average value for rows within the window frame

```
SELECT Cust, Store, Revenue,
avg(revenue) OVER (PARTITION BY Store)
as Avg_r
FROM <table_name>;
```

Windows Functions - Aggregate

Count

Customer	Store
C-1	A
C-2	B
C-3	B
C-4	B
C-5	A
C-6	B
C-7	A

Cust	Store	N_Cust
C-1	A	3
C-5	A	3
C-7	A	3
C-2	B	4
C-3	B	4
C-4	B	4
C-6	B	4

count of values for rows within the window frame

```
SELECT Cust, Store,
count(Customer) OVER (PARTITION BY Store)
as N_Cust
FROM <table_name>;
```

Windows Functions - Aggregate

Total

Cust	Date	Revenue
C-1	01-01-22	100
C-2	30-03-22	300
C-2	21-04-22	300
C-2	10-05-22	200
C-1	11-05-22	200
C-2	12-06-22	400
C-1	25-08-22	300

Cust	Date	Revenue	Total
C-1	01-01-22	100	600
C-1	11-05-22	200	600
C-1	25-08-22	300	600
C-2	30-03-22	300	1200
C-2	21-04-22	300	1200
C-2	10-05-22	200	1200
C-2	12-06-22	400	1200

sum of values within the window frame

```
SELECT Cust, Date, Revenue,
sum(revenue) OVER (PARTITION BY Cust )
as Total
FROM <table_name>;
```

Windows Functions - Aggregate

Running Total

Cust	Date	Revenue
C-1	01-01-22	100
C-2	30-03-22	300
C-2	21-04-22	300
C-2	10-05-22	200
C-1	11-05-22	200
C-2	12-06-22	400
C-1	25-08-22	300

Cust	Date	Revenue	Total
C-1	01-01-22	100	100
C-1	11-05-22	200	300
C-1	25-08-22	300	600
C-2	30-03-22	300	300
C-2	21-04-22	300	600
C-2	10-05-22	200	800
C-2	12-06-22	400	1200

sum of values within the window frame

```
SELECT Cust, Date, Revenue,
sum(revenue) OVER (PARTITION BY Cust
ORDER BY Date desc) as Total
FROM <table_name>;
```

Windows Functions - Aggregate

Lag/Lead

Cust	Date	Revenue
C-1	01-01-22	100
C-2	30-03-22	300
C-2	21-04-22	300
C-2	10-05-22	200
C-1	11-05-22	200
C-2	12-06-22	400
C-1	25-08-22	300

Cust	Date	Revenue	Last_r
C-1	01-01-22	100	
C-1	11-05-22	200	100
C-1	25-08-22	300	200
C-2	30-03-22	300	
C-2	21-04-22	300	300
C-2	10-05-22	200	300
C-2	12-06-22	400	200

sum of values within the window frame

```
SELECT Cust, Date, Revenue,
lag(revenue,1) OVER (PARTITION BY Cust
ORDER BY Date desc) as Last_r
FROM <table_name>;
```

COALESCE

COALESCE

S.No.	First	Middle	Last
1	Paul	Van	Hugh
2	David		Flashing
3		Lena	Radford
4	Henry		Goldwyn
5			Holden
6	Erin	T	Mull

S.No.	First	Middle	Last	Name_Col
1	Paul	Van	Hugh	Paul
2	David		Flashing	David
3		Lena	Radford	Lena
4	Henry		Goldwyn	Henry
5			Holden	Holden
6	Erin	T	Mull	Erin

COALESCE is a function that returns the first non- NULL value in a list of values.

```
SELECT Sno, First, Middle, Last ,
COALESCE(First, Middle, Last) as Name_col
FROM <table_name>;
```

COALESCE

S.No.	First	Middle	Last
1	Paul	Van	Hugh
2	David		Flashing
3		Lena	Radford
4	Henry		Goldwyn
5			Holden
6	Erin	T	Mull

S.No.	First	Middle	Last	Combined
1	Paul	Van	Hugh	Paul Van Hugh
2	David		Flashing	David Flashing
3		Lena	Radford	Lena Radford
4	Henry		Goldwyn	Henry Goldwyn
5			Holden	Holden
6	Erin	T	Mull	Erin T Mull