WeCloudData

SQL Window Function

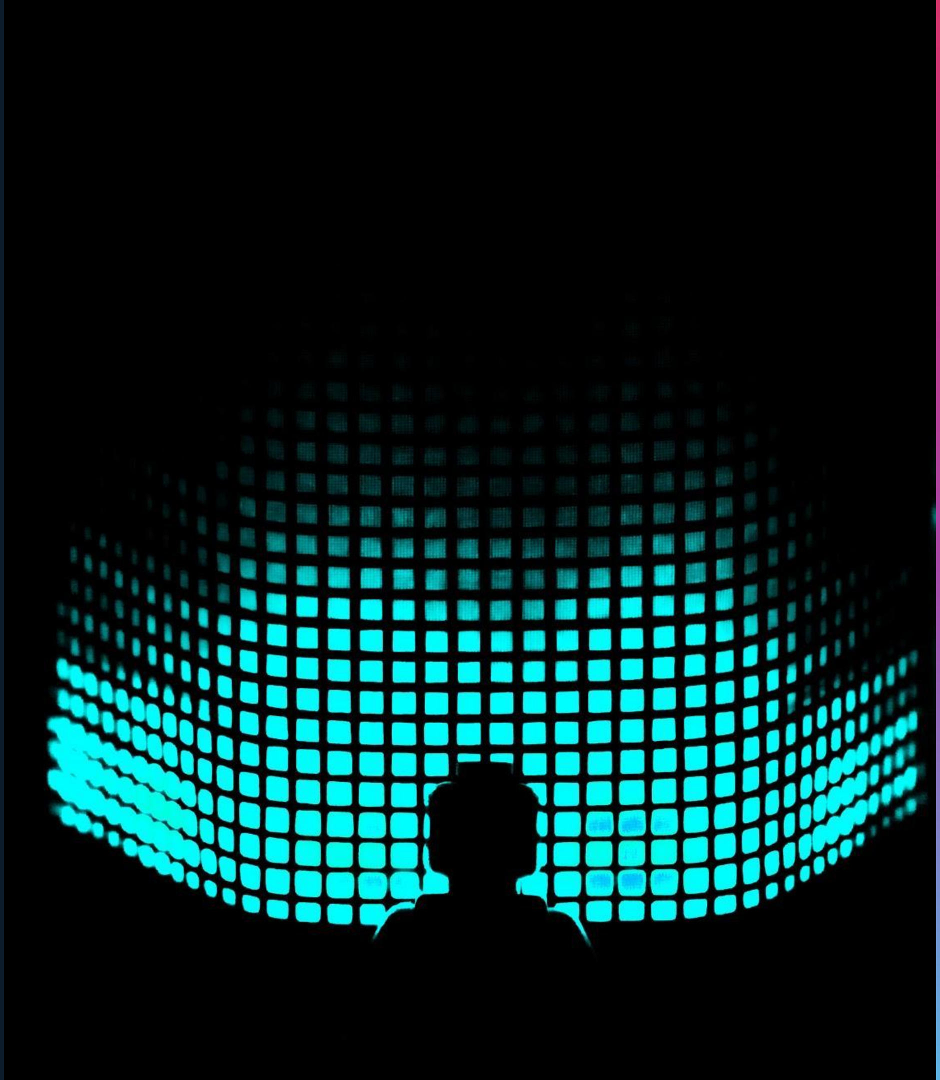
Kick off your career in data engineering & analytics

Analytics Functions

Learning Objectives

In this module, we will introduce the window function in SQL. Specifically, we will share with you:

- Intro to Window Functions
- Syntax
- Bounds



What is a Window Function?

Window Function: Syntax

Window Function: Bounds

Agenda.

What is a Window Function?

Window Functions

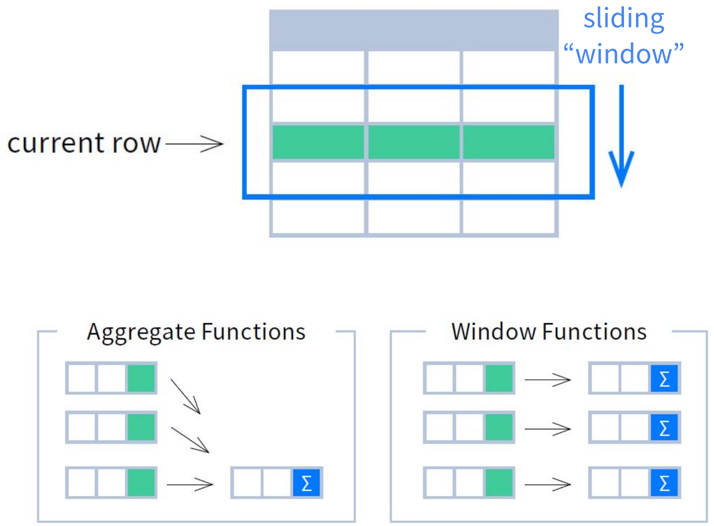


Image: learnsql

A **window function**:

- Performs a calculation based on a sliding set of rows that are related to the current row
    - This set of rows is called the "window"

- Type of calculation is comparable to aggregate functions
    - But unlike aggregate functions, window functions do not cause the rows to be collapsed into a single output row

List of Window Functions

Window Functions

Ranking Functions	Distribution Functions	Analytic Functions	Aggregate Functions
• row_number() • rank() • dense_rank()	• percent_rank() • cume_dist()	• lead() • lag() • ntile() • first_value() • last_value() • nth_value()	• avg() • count() • max() • min() • sum()

Window Functions Cheat Sheet:

- https://learnsql.com/blog/sql-window-functions-cheat-sheet/

What is a Window Function?

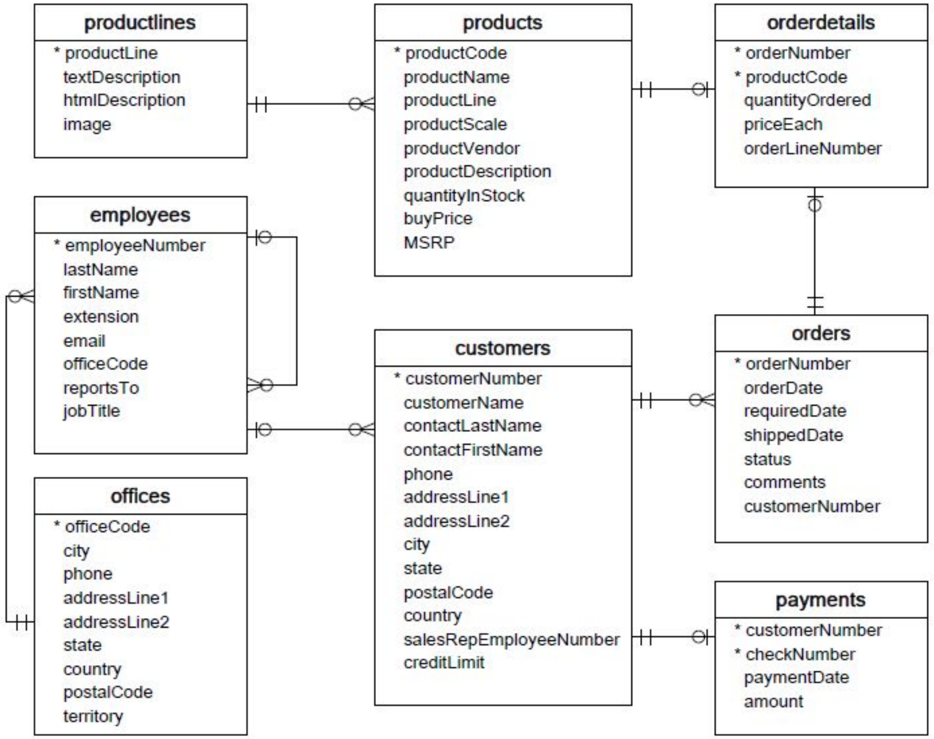
Window Function: Syntax

Window Function: Bounds

Agenda.

ClassicModels Database (Automotive Factory)

Window Functions

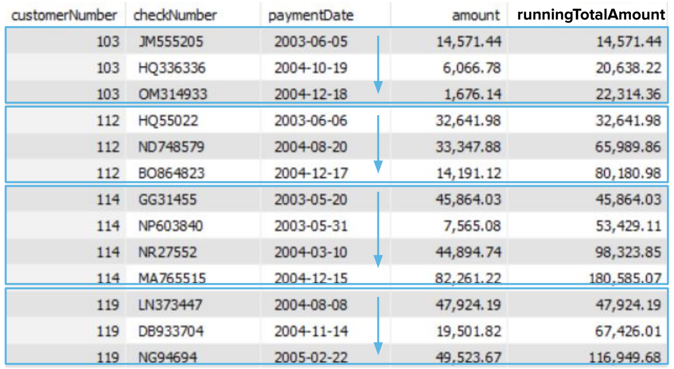


Window Function: Syntax

Window Functions

```
-- Example
# Find the customers' running total for amount spent, ordered
by the date paid

SELECT *, sum(amount) OVER (
                PARTITION BY customerNumber
                ORDER BY paymentDate) as
        runningTotalAmount
FROM payments;
```



customerNumber	checkNumber	paymentDate	amount	runningTotalAmount
103	JM555205	2003-06-05	14,571.44	14,571.44
103	HQ336336	2004-10-19	6,066.78	20,638.22
103	OM314933	2004-12-18	1,676.14	22,314.36
112	HQ55022	2003-06-06	32,641.98	32,641.98
112	ND748579	2004-08-20	33,347.88	65,989.86
112	BO864823	2004-12-17	14,191.12	80,180.98
114	GG31455	2003-05-20	45,864.03	45,864.03
114	NP603840	2003-05-31	7,565.08	53,429.11
114	NR27552	2004-03-10	44,894.74	98,323.85
114	MA765515	2004-12-15	82,261.22	180,585.07
119	LN373447	2004-08-08	47,924.19	47,924.19
119	DB933704	2004-11-14	19,501.82	67,426.01
119	NG94694	2005-02-22	49,523.67	116,949.68

...

Syntax

```
SELECT column1, column2,
        window_function(...) OVER (
                PARTITION BY ...
                ORDER BY ...) as window_name
FROM db.table_name;
```

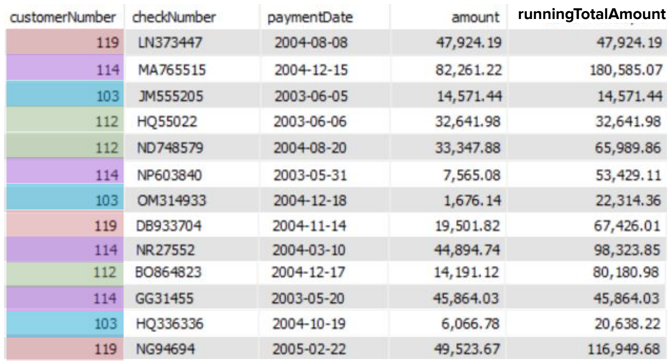
Note:

- You can read this query as, "take the sum over the partitioned groups, in order by *paymentDate*, and label it as *currentTotalSpent*"
- ORDER BY in window function queries work the same way as it would normally but treats each partition (in this case, the *customerNumber*) as separate

Taking a Closer Look: Partition By

Window Functions

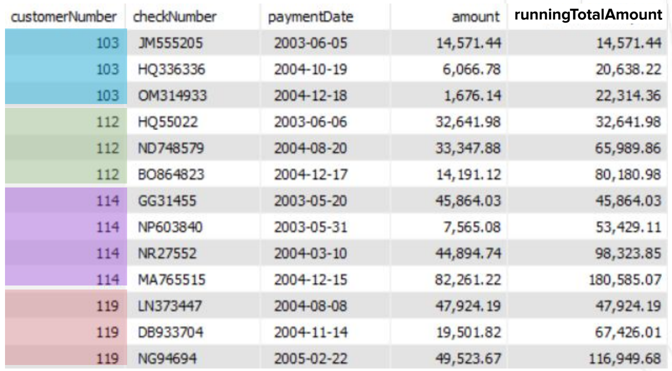
PARTITION BY organizes rows into multiple groups (called partitions).
If a PARTITION BY clause is not defined then the entire set is the partition.



customerNumber	checkNumber	paymentDate	amount	runningTotalAmount
119	LN373447	2004-08-08	47,924.19	47,924.19
114	MA765515	2004-12-15	82,261.22	180,585.07
103	JM555205	2003-06-05	14,571.44	14,571.44
112	HQ55022	2003-06-06	32,641.98	32,641.98
112	ND748579	2004-08-20	33,347.88	65,989.86
114	NP603840	2003-05-31	7,565.08	53,429.11
103	OM314933	2004-12-18	1,676.14	22,314.36
119	DB933704	2004-11-14	19,501.82	67,426.01
114	NR27552	2004-03-10	44,894.74	98,323.85
112	BO864823	2004-12-17	14,191.12	80,180.98
114	GG31455	2003-05-20	45,864.03	45,864.03
103	HQ336336	2004-10-19	6,066.78	20,638.22
119	NG94694	2005-02-22	49,523.67	116,949.68

...

PARTITION BY *customerNumber*



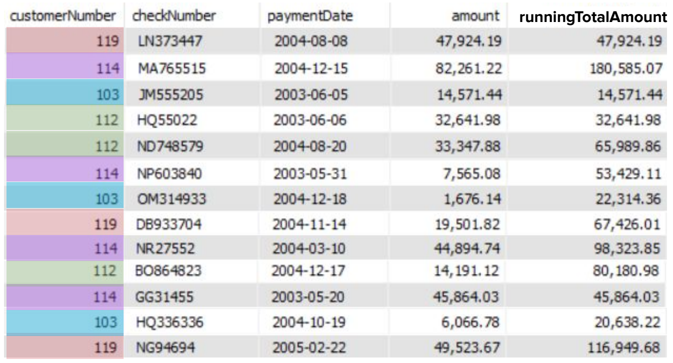
customerNumber	checkNumber	paymentDate	amount	runningTotalAmount
103	JM555205	2003-06-05	14,571.44	14,571.44
103	HQ336336	2004-10-19	6,066.78	20,638.22
103	OM314933	2004-12-18	1,676.14	22,314.36
112	HQ55022	2003-06-06	32,641.98	32,641.98
112	ND748579	2004-08-20	33,347.88	65,989.86
112	BO864823	2004-12-17	14,191.12	80,180.98
114	GG31455	2003-05-20	45,864.03	45,864.03
114	NP603840	2003-05-31	7,565.08	53,429.11
114	NR27552	2004-03-10	44,894.74	98,323.85
114	MA765515	2004-12-15	82,261.22	180,585.07
119	LN373447	2004-08-08	47,924.19	47,924.19
119	DB933704	2004-11-14	19,501.82	67,426.01
119	NG94694	2005-02-22	49,523.67	116,949.68

...

Taking a Closer Look: Order By

Window Functions

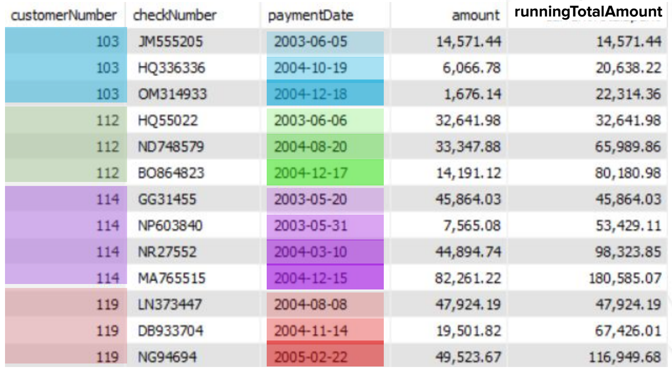
ORDER BY orders the specified column(s) in each partition.
If an ORDER BY clause is not defined
then the order of rows within each partition is arbitrary.



customerNumber	checkNumber	paymentDate	amount	runningTotalAmount
119	LN373447	2004-08-08	47,924.19	47,924.19
114	MA765515	2004-12-15	82,261.22	180,585.07
103	JM555205	2003-06-05	14,571.44	14,571.44
112	HQ55022	2003-06-06	32,641.98	32,641.98
112	ND748579	2004-08-20	33,347.88	65,989.86
114	NP603840	2003-05-31	7,565.08	53,429.11
103	OM314933	2004-12-18	1,676.14	22,314.36
119	DB933704	2004-11-14	19,501.82	67,426.01
114	NR27552	2004-03-10	44,894.74	98,323.85
112	BO864823	2004-12-17	14,191.12	80,180.98
114	GG31455	2003-05-20	45,864.03	45,864.03
103	HQ336336	2004-10-19	6,066.78	20,638.22
119	NG94694	2005-02-22	49,523.67	116,949.68

...

PARTITION BY *customerNumber*
ORDER BY *paymentDate*



customerNumber	checkNumber	paymentDate	amount	runningTotalAmount
103	JM555205	2003-06-05	14,571.44	14,571.44
103	HQ336336	2004-10-19	6,066.78	20,638.22
103	OM314933	2004-12-18	1,676.14	22,314.36
112	HQ55022	2003-06-06	32,641.98	32,641.98
112	ND748579	2004-08-20	33,347.88	65,989.86
112	BO864823	2004-12-17	14,191.12	80,180.98
114	GG31455	2003-05-20	45,864.03	45,864.03
114	NP603840	2003-05-31	7,565.08	53,429.11
114	NR27552	2004-03-10	44,894.74	98,323.85
114	MA765515	2004-12-15	82,261.22	180,585.07
119	LN373447	2004-08-08	47,924.19	47,924.19
119	DB933704	2004-11-14	19,501.82	67,426.01
119	NG94694	2005-02-22	49,523.67	116,949.68

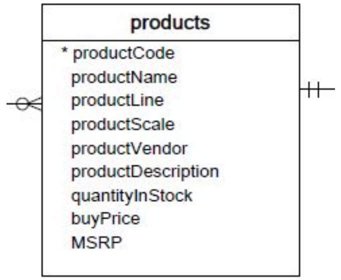
...

Lab #1

Window Functions

An automotive factory has a list of all their vehicles in a table named **products**. The vehicles must be ordered first by *product line* and then by *product name*. After, they will be sent to a separate area to prepare for shipment.

The factory is interested in knowing what is the running total *quantity in stock* of the vehicles.

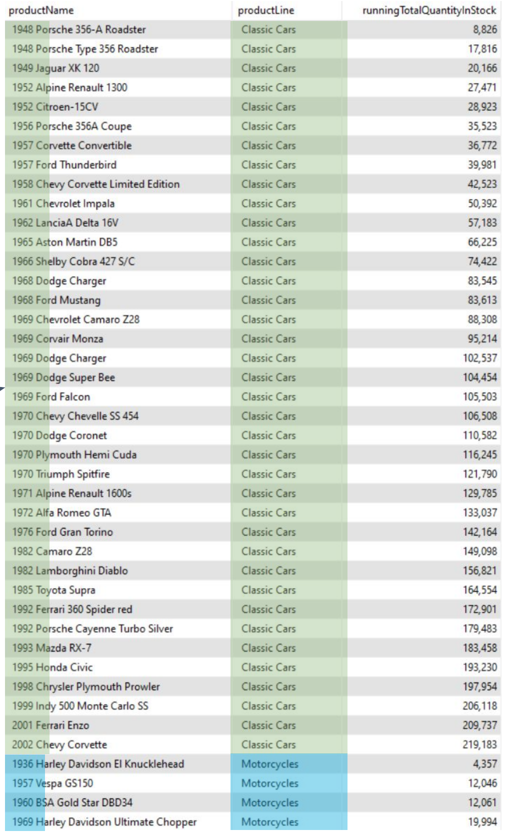


Return: *product name, product line, running total quantity in stock*
Order by: *product line, product name*

Write a query that follows the specifications given above.

Lab #1: Snippet of Desired Output

Window Functions



productName	productLine	runningTotalQuantityInStock
1948 Porsche 356-A Roadster	Classic Cars	8,826
1948 Porsche Type 356 Roadster	Classic Cars	17,816
1949 Jaguar XK 120	Classic Cars	20,166
1952 Alpine Renault 1300	Classic Cars	27,471
1952 Citroen-15CV	Classic Cars	28,923
1956 Porsche 356A Coupe	Classic Cars	35,523
1957 Corvette Convertible	Classic Cars	36,772
1957 Ford Thunderbird	Classic Cars	39,981
1958 Chevy Corvette Limited Edition	Classic Cars	42,523
1961 Chevrolet Impala	Classic Cars	50,392
1962 LanciaA Delta 16V	Classic Cars	57,183
1965 Aston Martin DB5	Classic Cars	66,225
1966 Shelby Cobra 427 S/C	Classic Cars	74,422
1968 Dodge Charger	Classic Cars	83,545
1968 Ford Mustang	Classic Cars	83,613
1969 Chevrolet Camaro Z28	Classic Cars	88,308
1969 Corvair Monza	Classic Cars	95,214
1969 Dodge Charger	Classic Cars	102,537
1969 Dodge Super Bee	Classic Cars	104,454
1969 Ford Falcon	Classic Cars	105,503
1970 Chevy Chevelle SS 454	Classic Cars	106,508
1970 Dodge Coronet	Classic Cars	110,582
1970 Plymouth Hemi Cuda	Classic Cars	116,245
1970 Triumph Spitfire	Classic Cars	121,790
1971 Alpine Renault 1600s	Classic Cars	129,785
1972 Alfa Romeo GTA	Classic Cars	133,037
1976 Ford Gran Torino	Classic Cars	142,164
1982 Camaro Z28	Classic Cars	149,098
1982 Lamborghini Diablo	Classic Cars	156,821
1985 Toyota Supra	Classic Cars	164,554
1992 Ferrari 360 Spider red	Classic Cars	172,901
1992 Porsche Cayenne Turbo Silver	Classic Cars	179,483
1993 Mazda RX-7	Classic Cars	183,458
1995 Honda Civic	Classic Cars	193,230
1998 Chrysler Plymouth Prowler	Classic Cars	197,954
1999 Indy 500 Monte Carlo SS	Classic Cars	206,118
2001 Ferrari Enzo	Classic Cars	209,737
2002 Chevy Corvette	Classic Cars	219,183
1936 Harley Davidson El Knucklehead	Motorcycles	4,357
1957 Vespa GS150	Motorcycles	12,046
1960 BSA Gold Star DBD34	Motorcycles	12,061
1969 Harley Davidson Ultimate Chopper	Motorcycles	19,994

...

The product names are sorted by what it starts with

Notice how the cumulative sum is updated with each row in its partition

What is a Window Function?

Window Function: Syntax

Window Function: Bounds

Agenda.

Window Function: Bounds

Window Functions

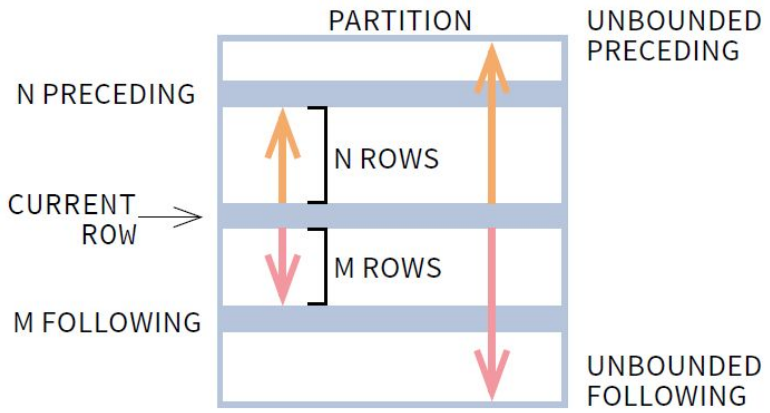


Image: learnsql

A **window bound** has five options:

- Unbounded preceding
- n preceding
- Current row
- n following
- Unbounded following

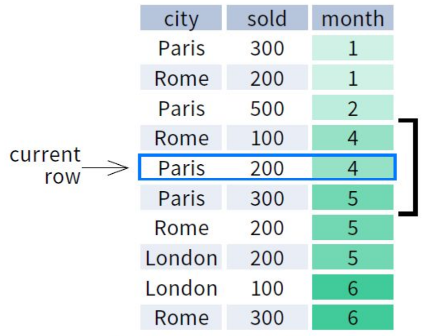
Window Function: Bounds (Cont'd)

Window Functions

To use a **bound**, follow this syntax within :

ROWS | RANGE | GROUPS BETWEEN lower_bound AND upper_bound

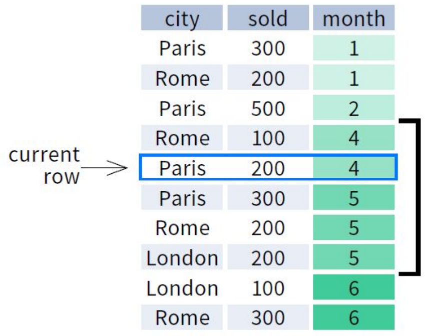
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING



city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

1 row before the current row and 1 row after the current row

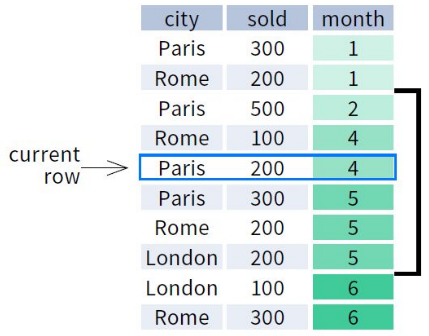
RANGE BETWEEN 1 PRECEDING AND 1 FOLLOWING



city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

values in the range between 3 and 5
ORDER BY must contain a single expression

GROUPS BETWEEN 1 PRECEDING AND 1 FOLLOWING



city	sold	month
Paris	300	1
Rome	200	1
Paris	500	2
Rome	100	4
Paris	200	4
Paris	300	5
Rome	200	5
London	200	5
London	100	6
Rome	300	6

1 group before the current row and 1 group after the current row regardless of the value

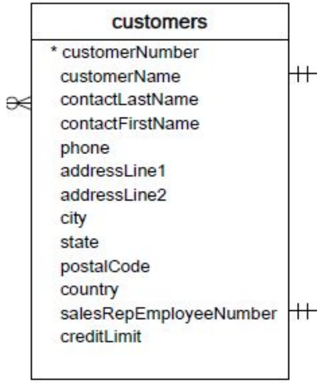
learnsql

Lab #2

Window Functions

The French branch of the automotive factory would like to send all of their customers a gift as part of their anniversary celebration. They have organized their list of **customers** by *customer name* in ascending order and would like to update the list to keep track of the running total count.

If customers appear multiple times, the branch will just make a note on the side (you don't need to do anything).

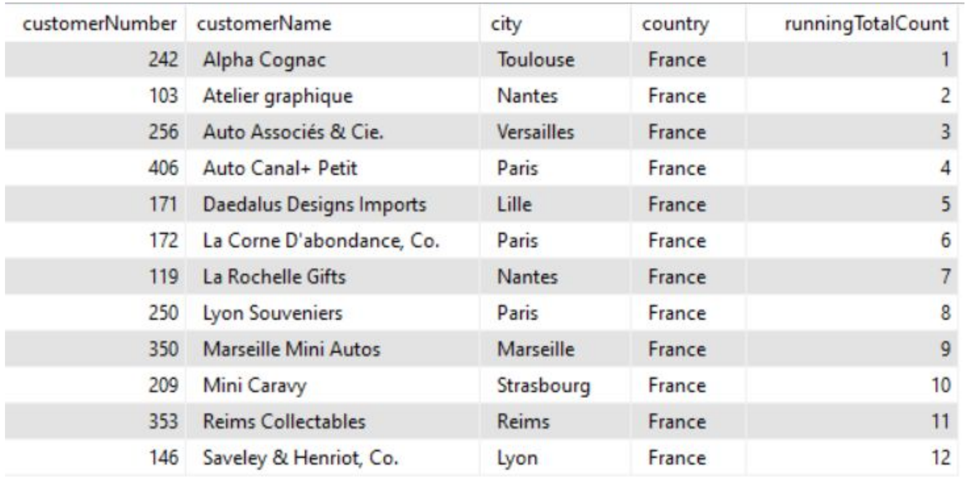


Return: *customer number, customer name, city, country, running total count*
Order by: *customer name, running total count*
Where: country is *France*

Write a query that follows the specifications given above.

Lab #2: Desired Output

Window Functions



customerNumber	customerName	city	country	runningTotalCount
242	Alpha Cognac	Toulouse	France	1
103	Atelier graphique	Nantes	France	2
256	Auto Associés & Cie.	Versailles	France	3
406	Auto Canal+ Petit	Paris	France	4
171	Daedalus Designs Imports	Lille	France	5
172	La Corne D'abondance, Co.	Paris	France	6
119	La Rochelle Gifts	Nantes	France	7
250	Lyon Souveniers	Paris	France	8
350	Marseille Mini Autos	Marseille	France	9
209	Mini Caravy	Strasbourg	France	10
353	Reims Collectables	Reims	France	11
146	Saveley & Henriot, Co.	Lyon	France	12