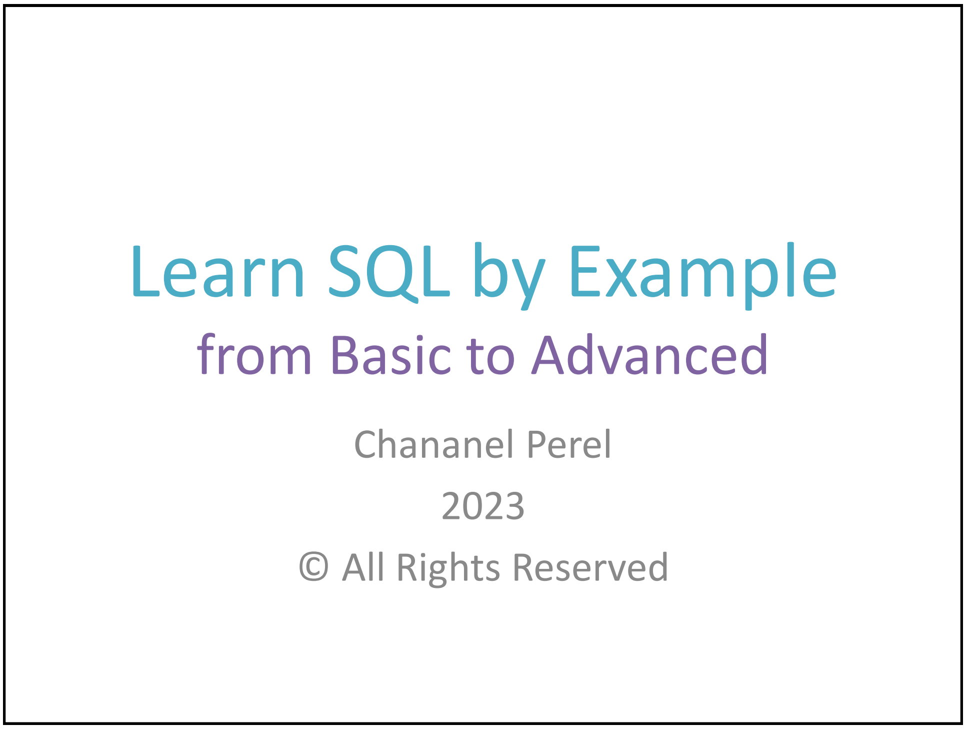


Learn SQL by Example

from Basic to Advanced

Chananel Perel

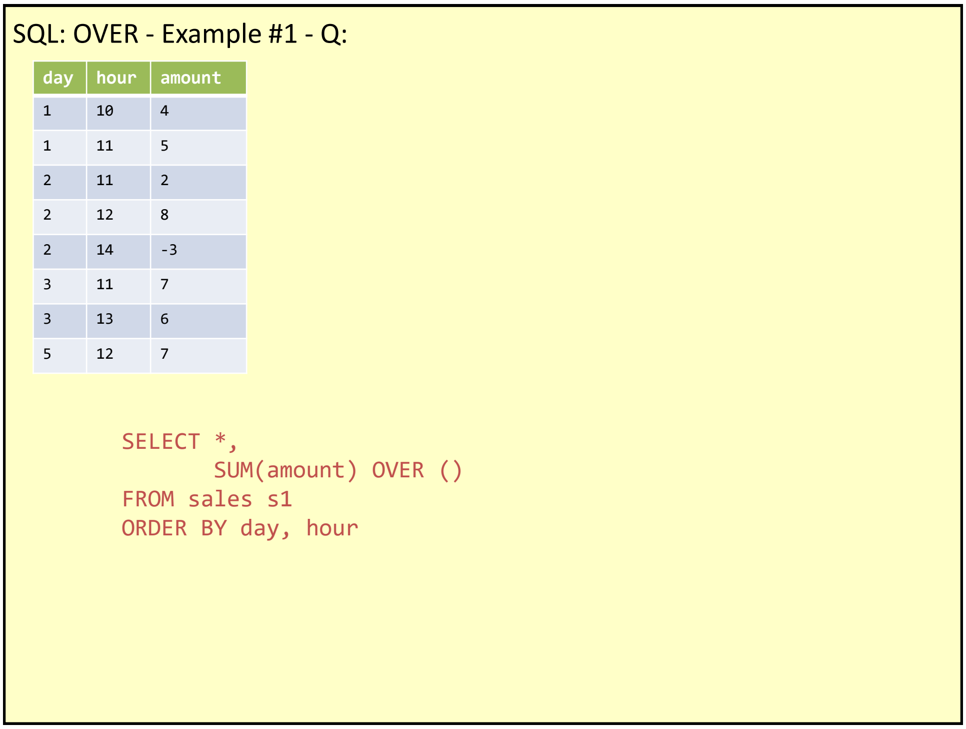
2023

Learn SQL by Example (Chananel Perel) 2024-05-01 15:55:34.299653

WINDOW FUNCTIONS

Learn SQL by Example (Chananel Perel (2023

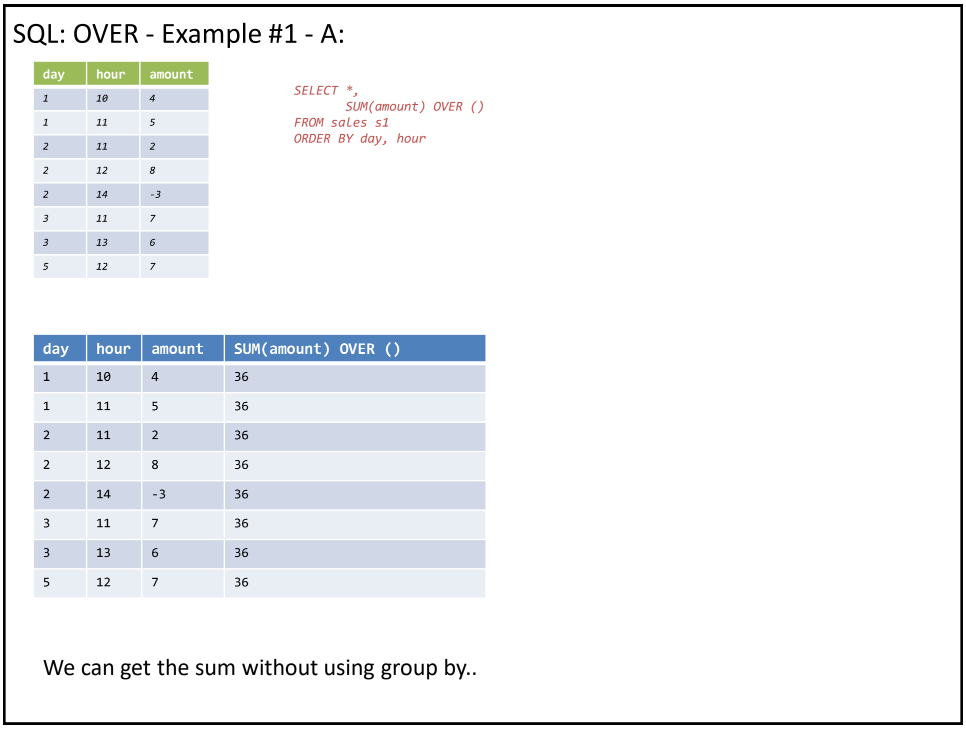
OVER



SQL: OVER - Example #1 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       SUM(amount) OVER ()
FROM sales s1
ORDER BY day, hour
```



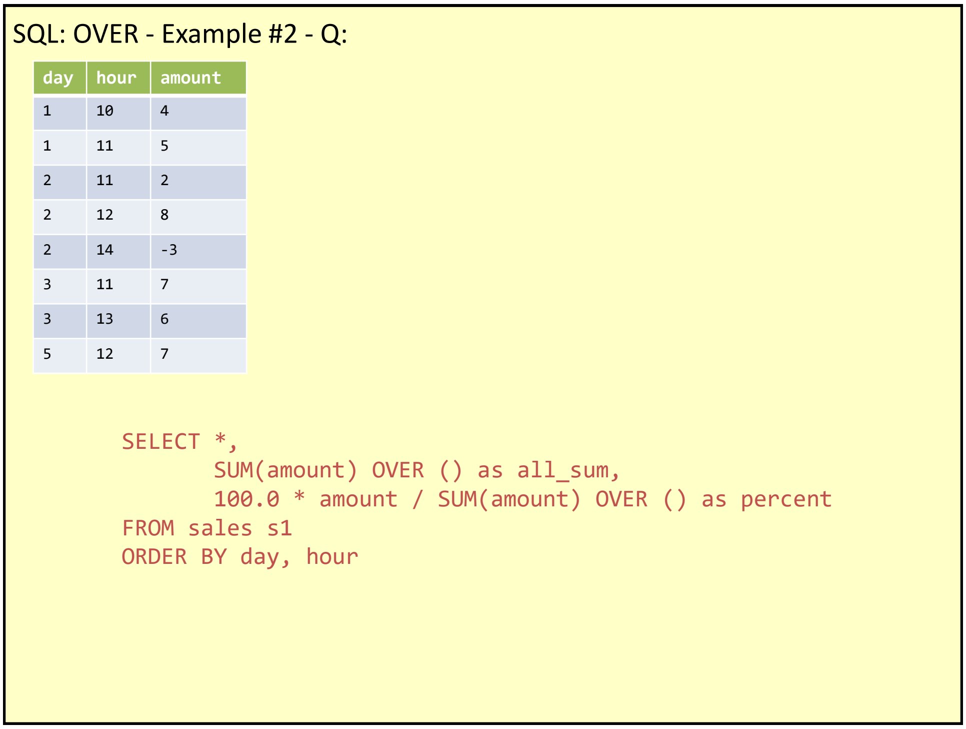
SQL: OVER - Example #1 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       SUM(amount) OVER ()
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	SUM(amount) OVER ()
1	10	4	36
1	11	5	36
2	11	2	36
2	12	8	36
2	14	-3	36
3	11	7	36
3	13	6	36
5	12	7	36

We can get the sum without using group by..



SQL: OVER - Example #2 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       SUM(amount) OVER () as all_sum,
       100.0 * amount / SUM(amount) OVER () as percent
FROM sales s1
ORDER BY day, hour
```

SQL: OVER - Example #2 - A:

day	hour	amount
1	*10*	*4*
1	*11*	*5*
2	*11*	*2*
2	*12*	*8*
2	*14*	*-3*
3	*11*	*7*
3	*13*	*6*
5	*12*	*7*

```
SELECT *,
       SUM(amount) OVER () as all_sum,
       100.0 * amount / SUM(amount) OVER () as percent
FROM sales s1
ORDER BY day, hour
```

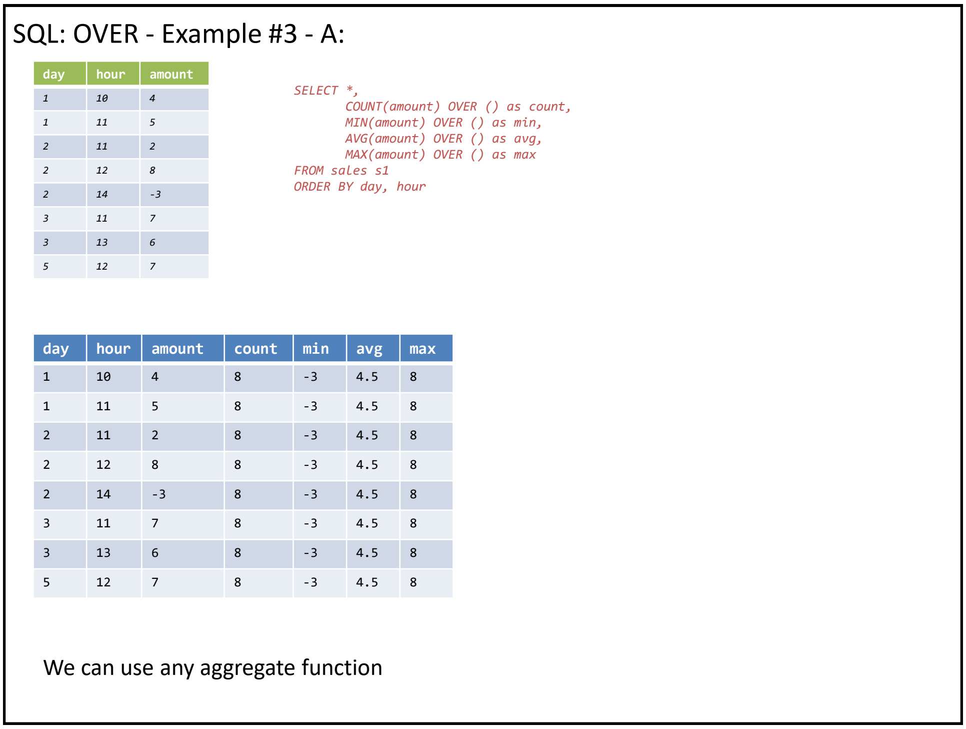
day	hour	amount	all_sum	percent
1	10	4	36	11.11111111111111
1	11	5	36	13.88888888888889
2	11	2	36	5.555555555555555
2	12	8	36	22.22222222222222
2	14	-3	36	-8.333333333333334
3	11	7	36	19.444444444444443
3	13	6	36	16.666666666666668
5	12	7	36	19.444444444444443

Now we can use it to see the percentage out of total sales

SQL: OVER - Example #3 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       COUNT(amount) OVER () as count,
       MIN(amount) OVER () as min,
       AVG(amount) OVER () as avg,
       MAX(amount) OVER () as max
FROM sales s1
ORDER BY day, hour
```



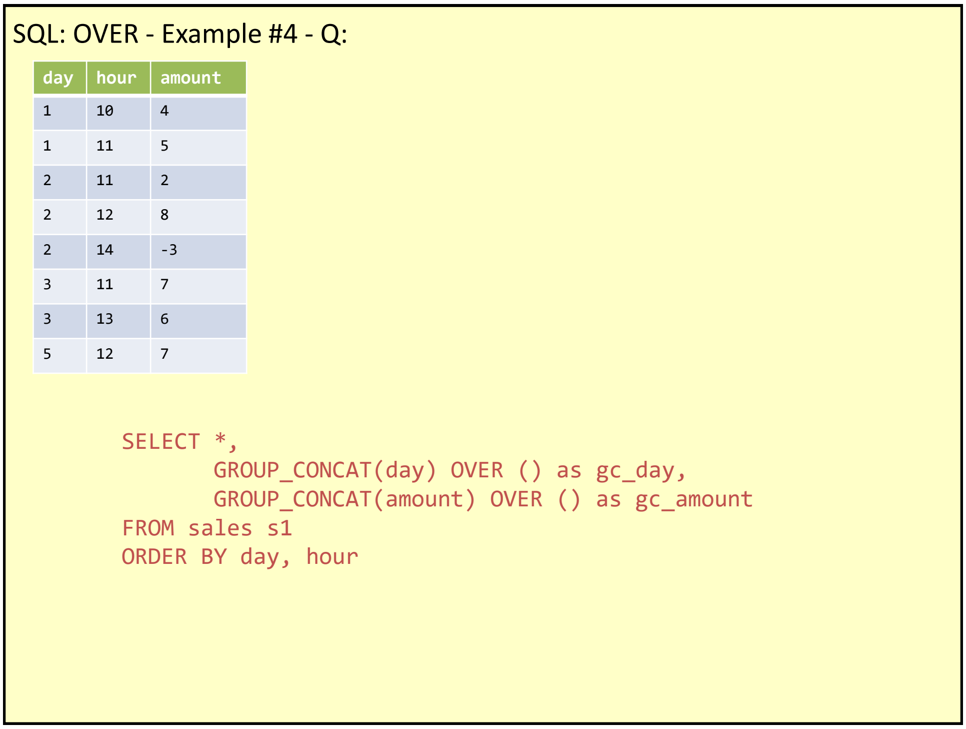
SQL: OVER - Example #3 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       COUNT(amount) OVER () as count,
       MIN(amount) OVER () as min,
       AVG(amount) OVER () as avg,
       MAX(amount) OVER () as max
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	count	min	avg	max
1	10	4	8	-3	4.5	8
1	11	5	8	-3	4.5	8
2	11	2	8	-3	4.5	8
2	12	8	8	-3	4.5	8
2	14	-3	8	-3	4.5	8
3	11	7	8	-3	4.5	8
3	13	6	8	-3	4.5	8
5	12	7	8	-3	4.5	8

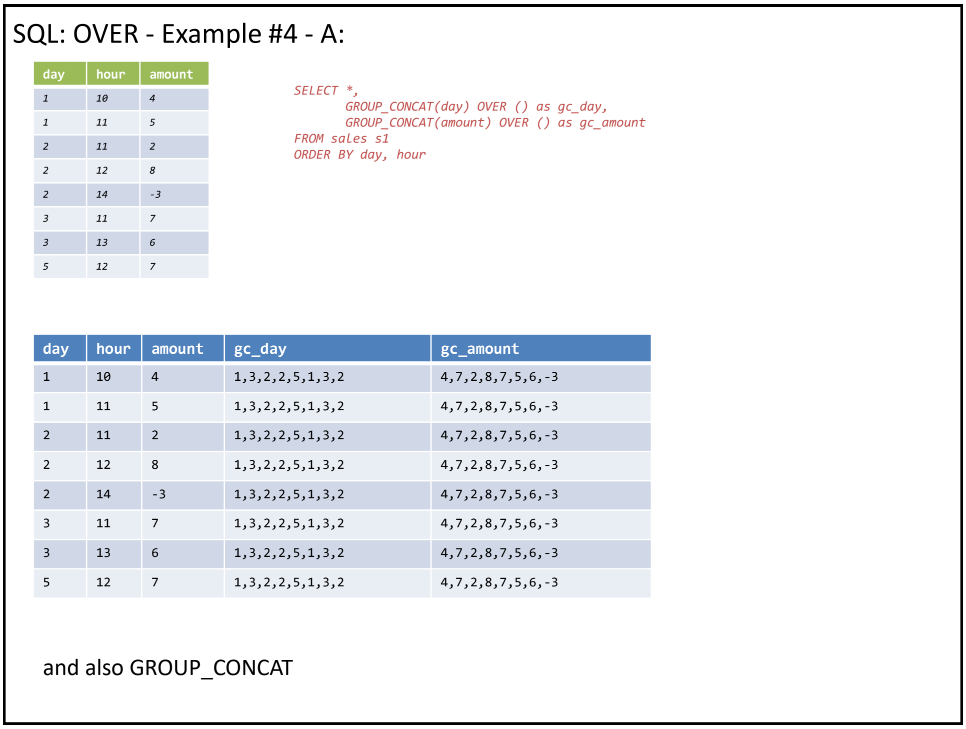
We can use any aggregate function



SQL: OVER - Example #4 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       GROUP_CONCAT(day) OVER () as gc_day,
       GROUP_CONCAT(amount) OVER () as gc_amount
FROM sales s1
ORDER BY day, hour
```



SQL: OVER - Example #4 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       GROUP_CONCAT(day) OVER () as gc_day,
       GROUP_CONCAT(amount) OVER () as gc_amount
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_day	gc_amount
1	10	4	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
1	11	5	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
2	11	2	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
2	12	8	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
2	14	-3	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
3	11	7	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
3	13	6	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3
5	12	7	1,3,2,2,5,1,3,2	4,7,2,8,7,5,6,-3

and also GROUP_CONCAT

Learn SQL by Example (Chananel Perel 2023)

PARTITION BY

SQL: PARTITION BY - Example #5 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       COUNT(amount) OVER (PARTITION BY day) as count_d,
       MIN(amount) OVER (PARTITION BY day) as min_d,
       AVG(amount) OVER (PARTITION BY day) as avg_d,
       MAX(amount) OVER (PARTITION BY day) as max_d
FROM sales s1
ORDER BY day, hour
```

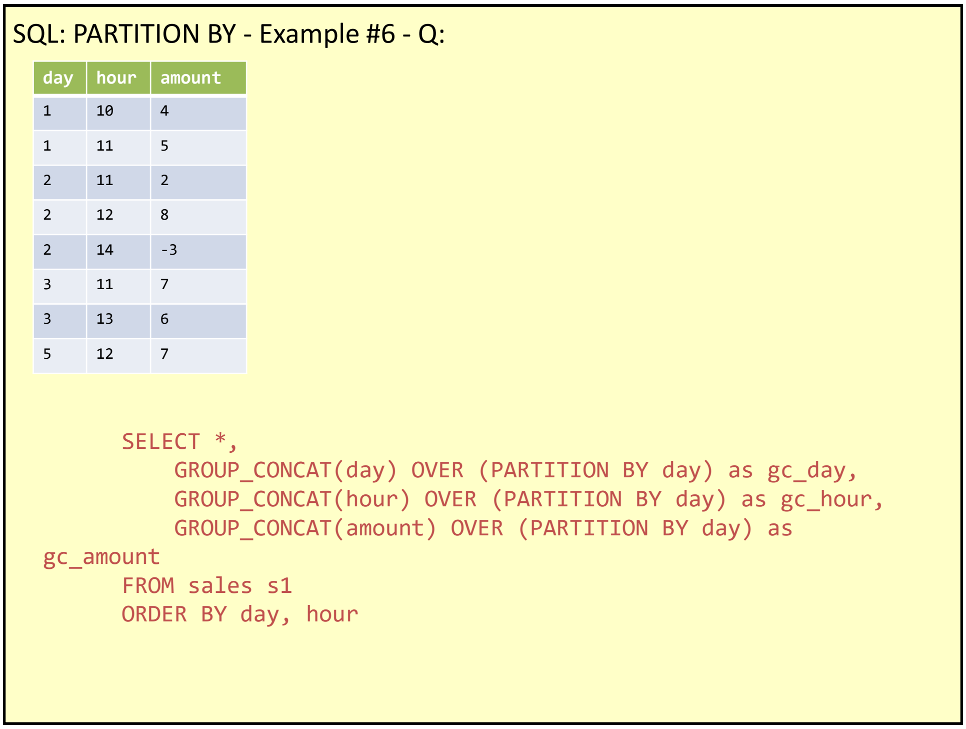
SQL: PARTITION BY - Example #5 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
       COUNT(amount) OVER (PARTITION BY day) as count_d,
       MIN(amount) OVER (PARTITION BY day) as min_d,
       AVG(amount) OVER (PARTITION BY day) as avg_d,
       MAX(amount) OVER (PARTITION BY day) as max_d
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	count_d	min_d	avg_d	max_d
1	10	4	2	4	4.5	5
1	11	5	2	4	4.5	5
2	11	2	3	-3	2.3333333333333335	8
2	12	8	3	-3	2.3333333333333335	8
2	14	-3	3	-3	2.3333333333333335	8
3	11	7	2	6	6.5	7
3	13	6	2	6	6.5	7
5	12	7	1	7	7.0	7

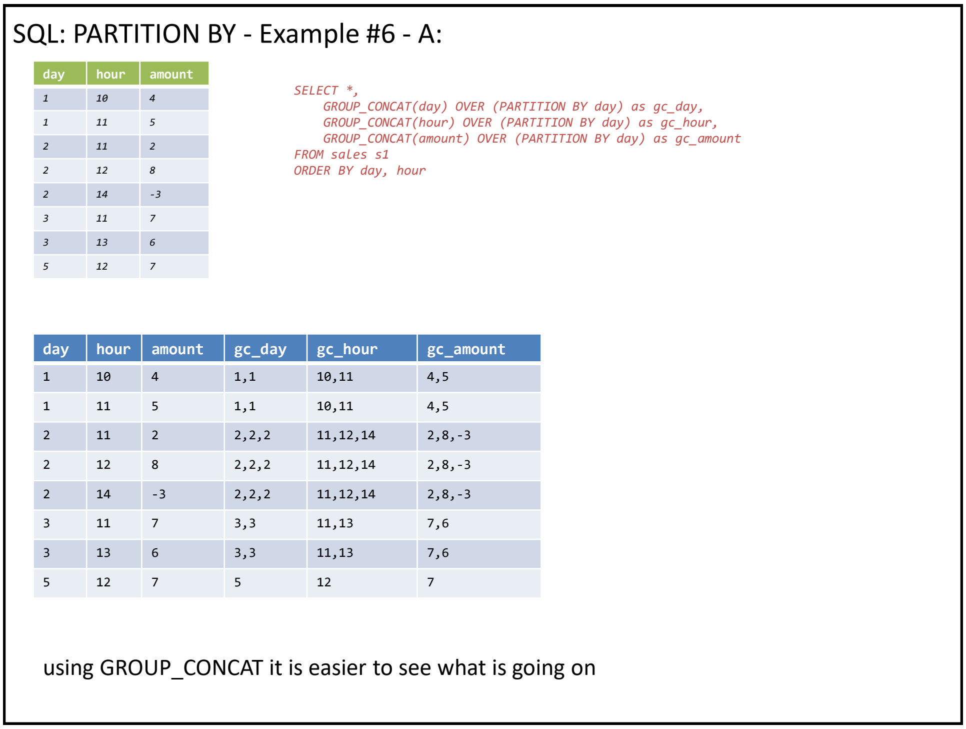
We can do it per day (similar to GROUP BY)



SQL: PARTITION BY - Example #6 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      SELECT *,
          GROUP_CONCAT(day) OVER (PARTITION BY day) as gc_day,
          GROUP_CONCAT(hour) OVER (PARTITION BY day) as gc_hour,
          GROUP_CONCAT(amount) OVER (PARTITION BY day) as
gc_amount
      FROM sales s1
      ORDER BY day, hour
```



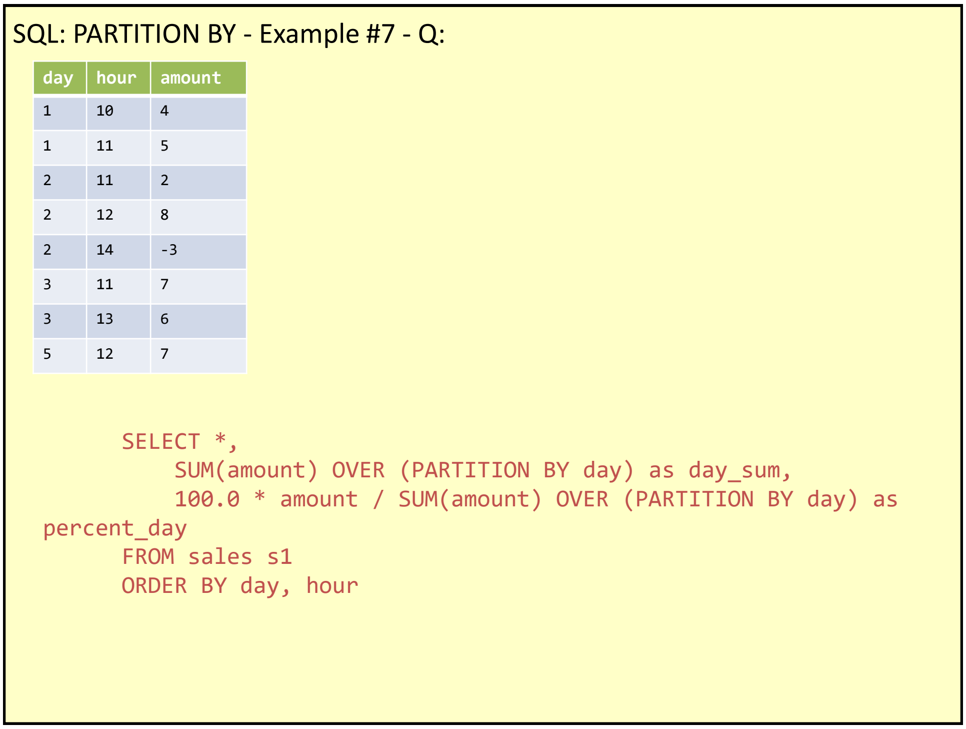
SQL: PARTITION BY - Example #6 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    GROUP_CONCAT(day) OVER (PARTITION BY day) as gc_day,
    GROUP_CONCAT(hour) OVER (PARTITION BY day) as gc_hour,
    GROUP_CONCAT(amount) OVER (PARTITION BY day) as gc_amount
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_day	gc_hour	gc_amount
1	10	4	1,1	10,11	4,5
1	11	5	1,1	10,11	4,5
2	11	2	2,2,2	11,12,14	2,8,-3
2	12	8	2,2,2	11,12,14	2,8,-3
2	14	-3	2,2,2	11,12,14	2,8,-3
3	11	7	3,3	11,13	7,6
3	13	6	3,3	11,13	7,6
5	12	7	5	12	7

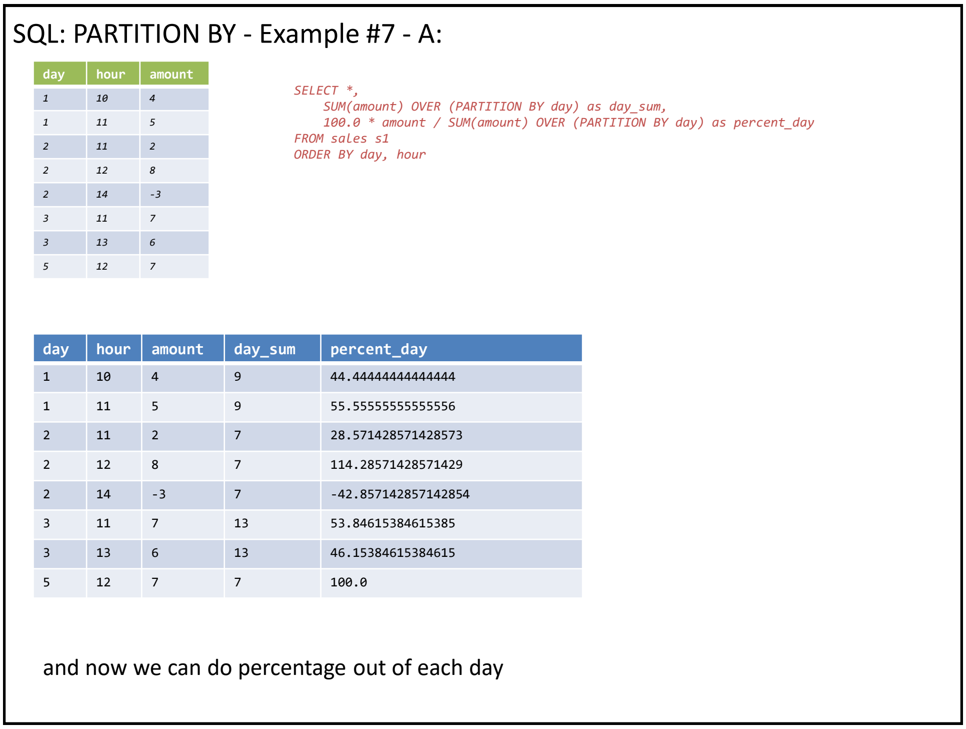
using GROUP_CONCAT it is easier to see what is going on



SQL: PARTITION BY - Example #7 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      SELECT *,
          SUM(amount) OVER (PARTITION BY day) as day_sum,
          100.0 * amount / SUM(amount) OVER (PARTITION BY day) as
percent_day
      FROM sales s1
      ORDER BY day, hour
```



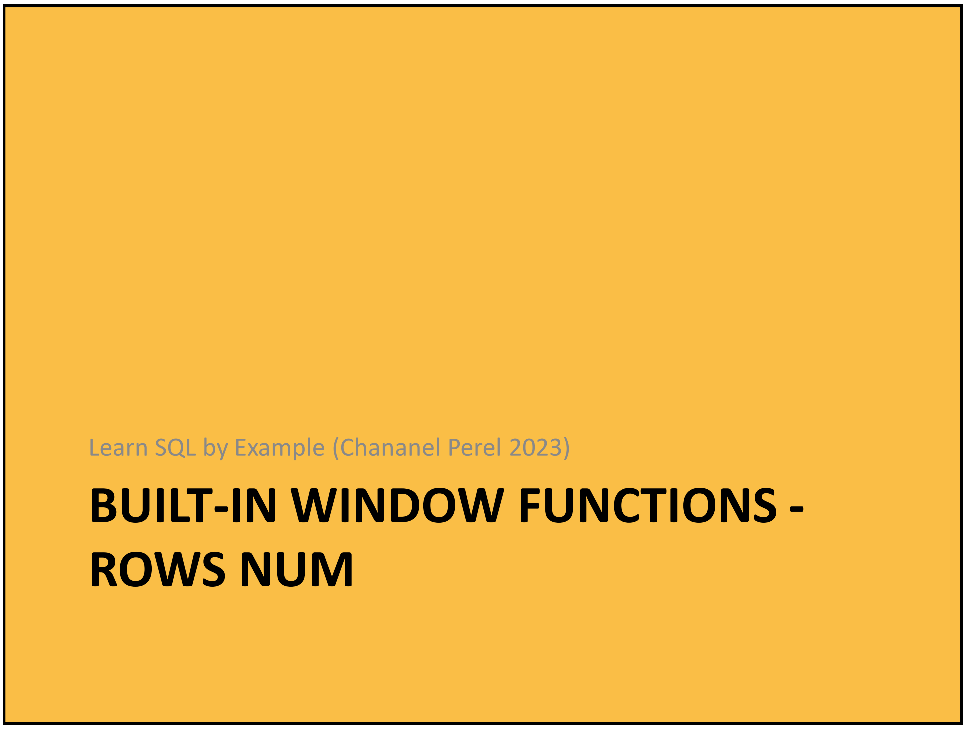
SQL: PARTITION BY - Example #7 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    SUM(amount) OVER (PARTITION BY day) as day_sum,
    100.0 * amount / SUM(amount) OVER (PARTITION BY day) as percent_day
FROM sales s1
ORDER BY day, hour
```

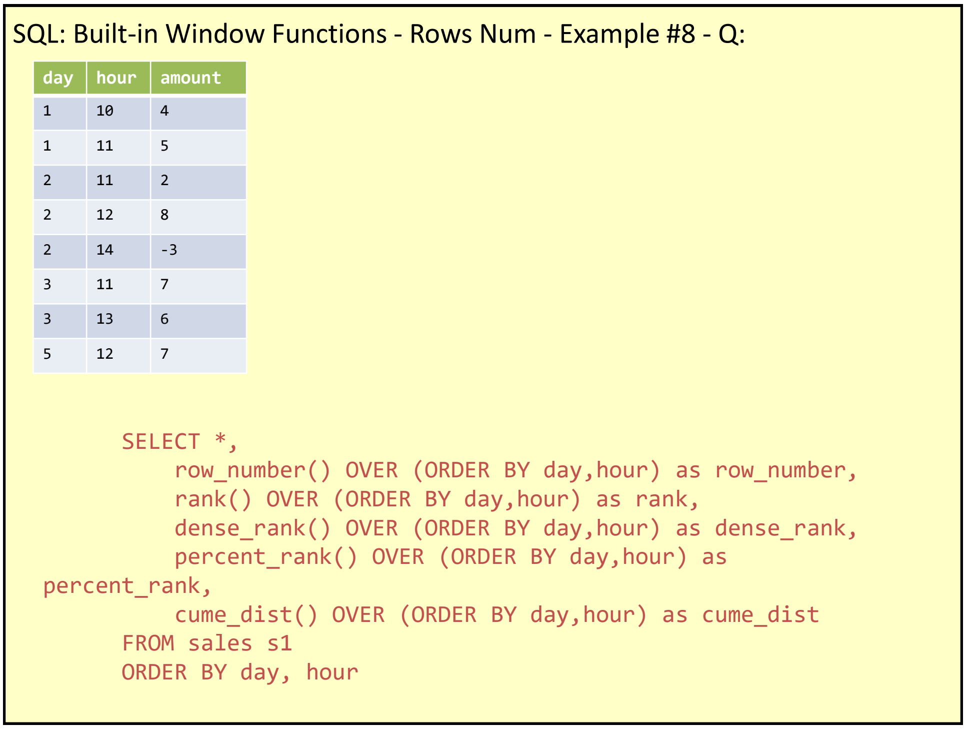
day	hour	amount	day_sum	percent_day
1	10	4	9	44.44444444444444
1	11	5	9	55.55555555555556
2	11	2	7	28.571428571428573
2	12	8	7	114.28571428571429
2	14	-3	7	-42.857142857142854
3	11	7	13	53.84615384615385
3	13	6	13	46.15384615384615
5	12	7	7	100.0

and now we can do percentage out of each day



Learn SQL by Example (Chananel Perel 2023)

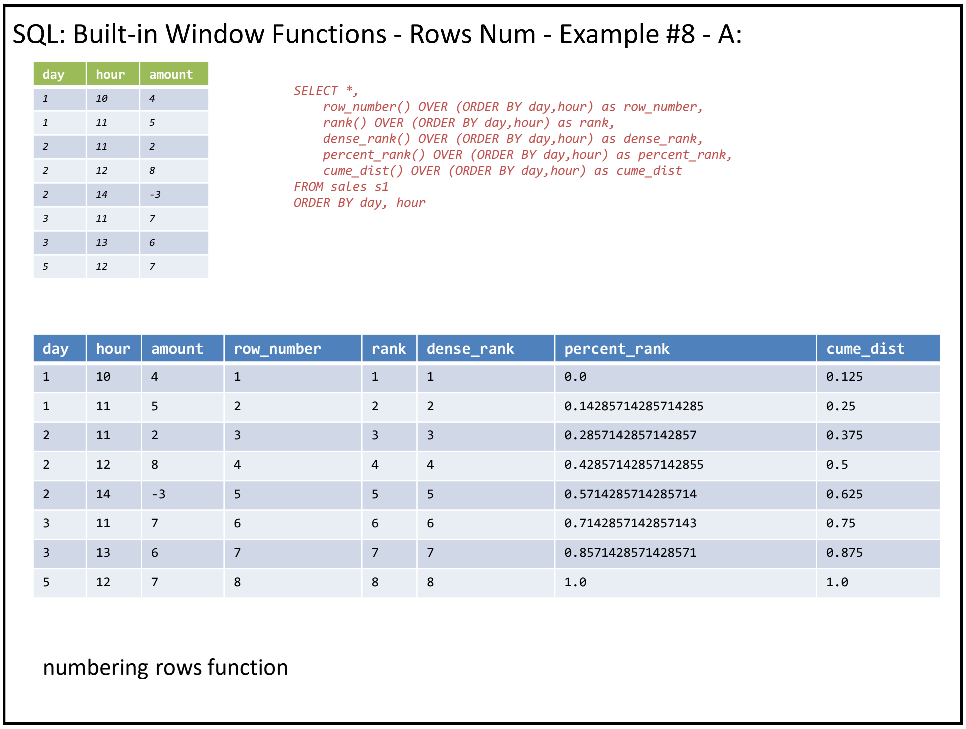
BUILT-IN WINDOW FUNCTIONS - ROWS NUM



SQL: Built-in Window Functions - Rows Num - Example #8 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      SELECT *,
          row_number() OVER (ORDER BY day,hour) as row_number,
          rank() OVER (ORDER BY day,hour) as rank,
          dense_rank() OVER (ORDER BY day,hour) as dense_rank,
          percent_rank() OVER (ORDER BY day,hour) as
percent_rank,
          cume_dist() OVER (ORDER BY day,hour) as cume_dist
      FROM sales s1
      ORDER BY day, hour
```



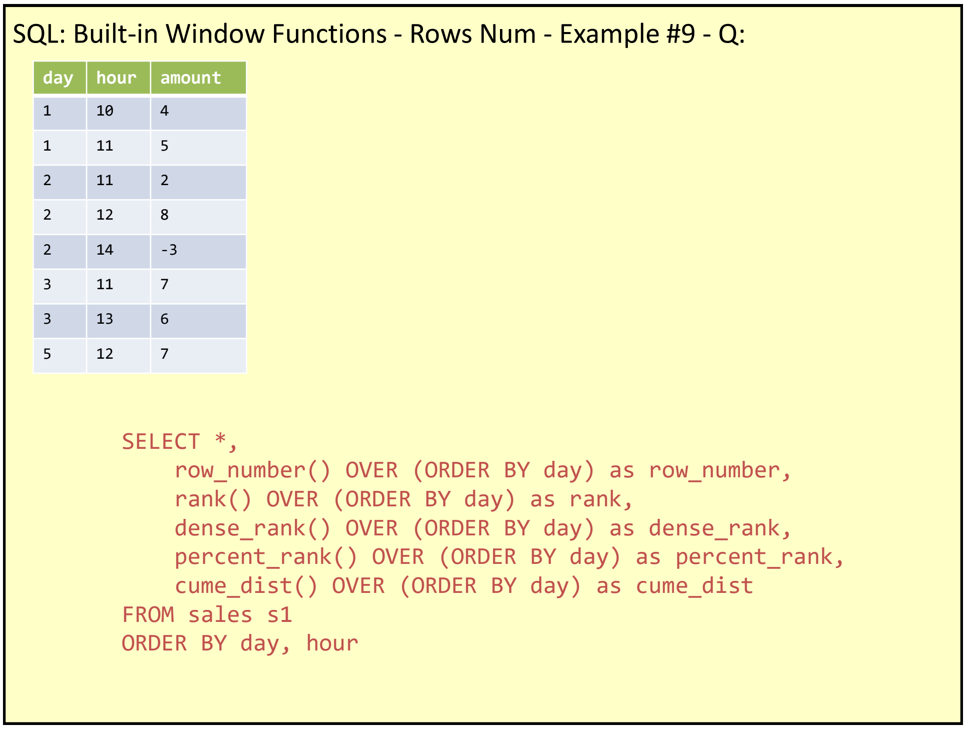
SQL: Built-in Window Functions - Rows Num - Example #8 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    row_number() OVER (ORDER BY day,hour) as row_number,
    rank() OVER (ORDER BY day,hour) as rank,
    dense_rank() OVER (ORDER BY day,hour) as dense_rank,
    percent_rank() OVER (ORDER BY day,hour) as percent_rank,
    cume_dist() OVER (ORDER BY day,hour) as cume_dist
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	row_number	rank	dense_rank	percent_rank	cume_dist
1	10	4	1	1	1	0.0	0.125
1	11	5	2	2	2	0.14285714285714285	0.25
2	11	2	3	3	3	0.2857142857142857	0.375
2	12	8	4	4	4	0.42857142857142855	0.5
2	14	-3	5	5	5	0.5714285714285714	0.625
3	11	7	6	6	6	0.7142857142857143	0.75
3	13	6	7	7	7	0.8571428571428571	0.875
5	12	7	8	8	8	1.0	1.0

numbering rows function



SQL: Built-in Window Functions - Rows Num - Example #9 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    row_number() OVER (ORDER BY day) as row_number,
    rank() OVER (ORDER BY day) as rank,
    dense_rank() OVER (ORDER BY day) as dense_rank,
    percent_rank() OVER (ORDER BY day) as percent_rank,
    cume_dist() OVER (ORDER BY day) as cume_dist
FROM sales s1
ORDER BY day, hour
```

SQL: Built-in Window Functions - Rows Num - Example #9 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    row_number() OVER (ORDER BY day) as row_number,
    rank() OVER (ORDER BY day) as rank,
    dense_rank() OVER (ORDER BY day) as dense_rank,
    percent_rank() OVER (ORDER BY day) as percent_rank,
    cume_dist() OVER (ORDER BY day) as cume_dist
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	row_number	rank	dense_rank	percent_rank	cume_dist
1	10	4	1	1	1	0.0	0.25
1	11	5	2	1	1	0.0	0.25
2	11	2	3	3	2	0.2857142857142857	0.625
2	12	8	4	3	2	0.2857142857142857	0.625
2	14	-3	5	3	2	0.2857142857142857	0.625
3	11	7	6	6	3	0.7142857142857143	0.875
3	13	6	7	6	3	0.7142857142857143	0.875
5	12	7	8	8	4	1.0	1.0

and now we can see the difference.

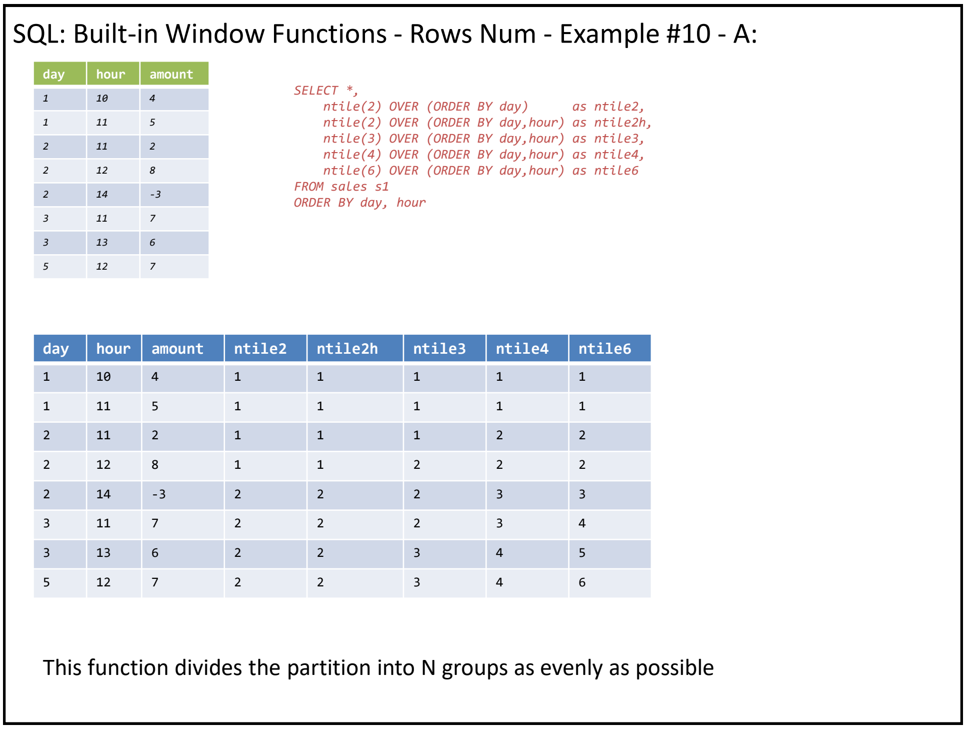
PERCENT_RANK returns the percent of values less than the current score.

CUME_DIST, which stands for cumulative distribution, returns the actual position of the score

SQL: Built-in Window Functions - Rows Num - Example #10 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    ntile(2) OVER (ORDER BY day)      as ntile2,
    ntile(2) OVER (ORDER BY day,hour) as ntile2h,
    ntile(3) OVER (ORDER BY day,hour) as ntile3,
    ntile(4) OVER (ORDER BY day,hour) as ntile4,
    ntile(6) OVER (ORDER BY day,hour) as ntile6
FROM sales s1
ORDER BY day, hour
```



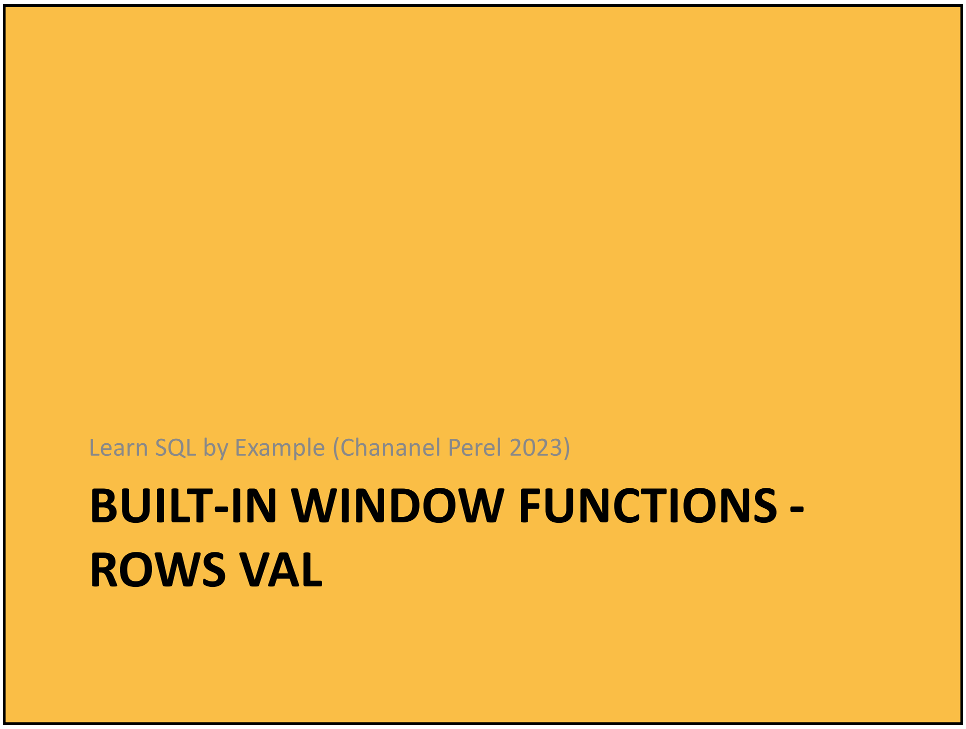
SQL: Built-in Window Functions - Rows Num - Example #10 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
    ntile(2) OVER (ORDER BY day)      as ntile2,
    ntile(2) OVER (ORDER BY day,hour) as ntile2h,
    ntile(3) OVER (ORDER BY day,hour) as ntile3,
    ntile(4) OVER (ORDER BY day,hour) as ntile4,
    ntile(6) OVER (ORDER BY day,hour) as ntile6
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	ntile2	ntile2h	ntile3	ntile4	ntile6
1	10	4	1	1	1	1	1
1	11	5	1	1	1	1	1
2	11	2	1	1	1	2	2
2	12	8	1	1	2	2	2
2	14	-3	2	2	2	3	3
3	11	7	2	2	2	3	4
3	13	6	2	2	3	4	5
5	12	7	2	2	3	4	6

This function divides the partition into N groups as evenly as possible



Learn SQL by Example (Chananel Perel 2023)

BUILT-IN WINDOW FUNCTIONS - ROWS VAL

SQL: Built-in Window Functions - Rows Val - Example #11 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
   LAG(amount) OVER (ORDER BY day,hour) as lag_a,
   LEAD(amount) OVER (ORDER BY day,hour) as lead_a
FROM sales s1
ORDER BY day, hour
```

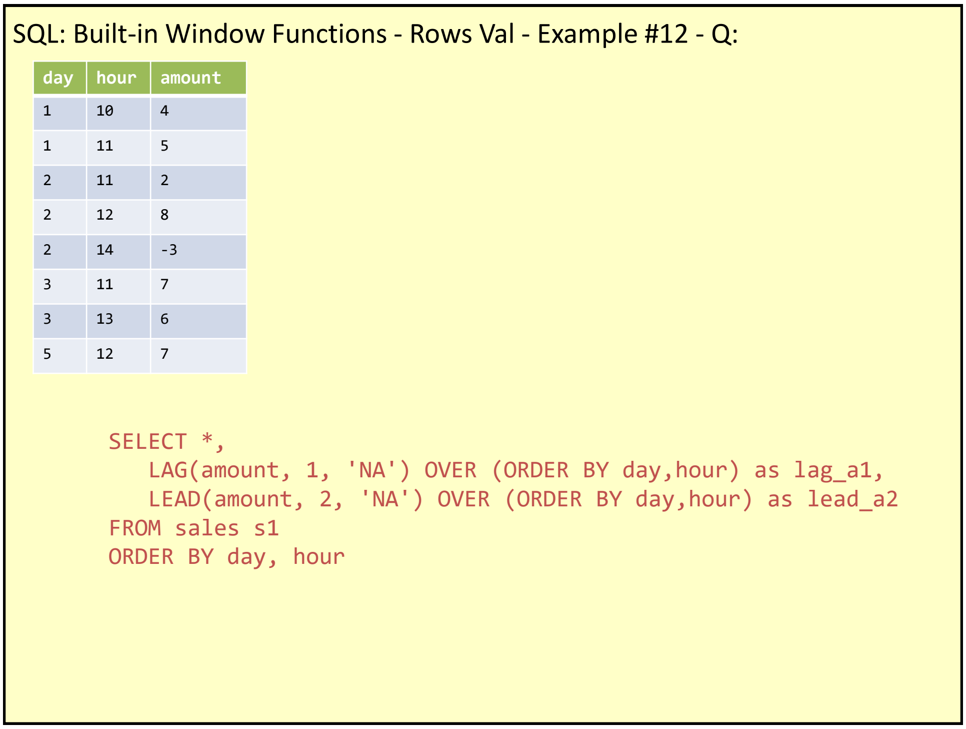
SQL: Built-in Window Functions - Rows Val - Example #11 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
   LAG(amount) OVER (ORDER BY day,hour) as lag_a,
   LEAD(amount) OVER (ORDER BY day,hour) as lead_a
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	lag_a	lead_a
1	10	4	None	5
1	11	5	4	2
2	11	2	5	8
2	12	8	2	-3
2	14	-3	8	7
3	11	7	-3	6
3	13	6	7	7
5	12	7	6	None

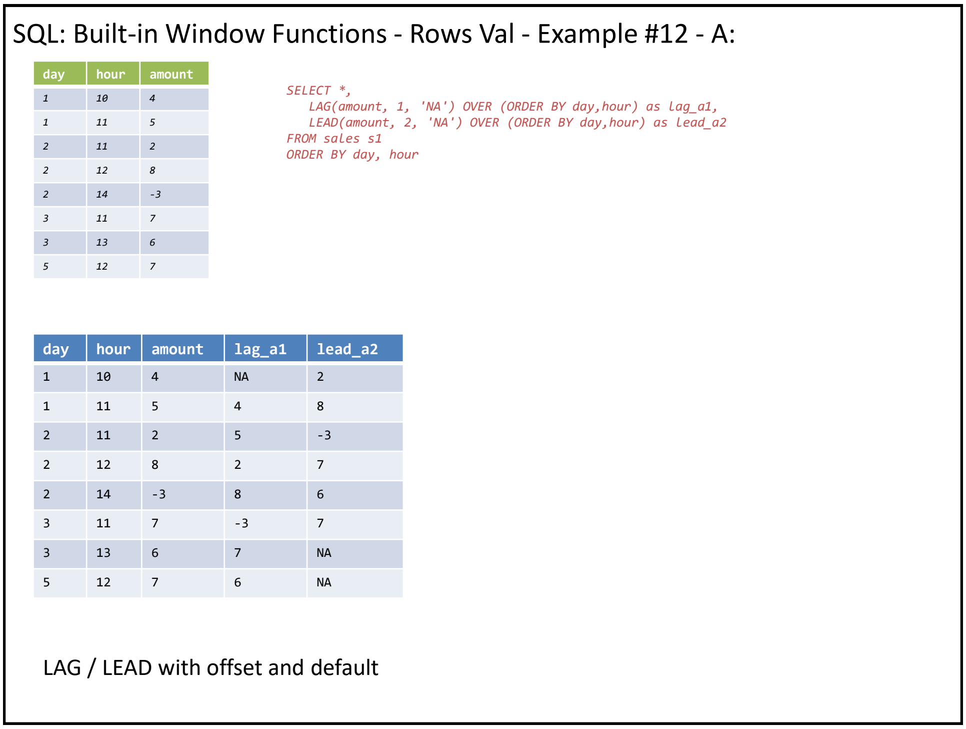
LAG / LEAD



SQL: Built-in Window Functions - Rows Val - Example #12 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
   LAG(amount, 1, 'NA') OVER (ORDER BY day,hour) as lag_a1,
   LEAD(amount, 2, 'NA') OVER (ORDER BY day,hour) as lead_a2
FROM sales s1
ORDER BY day, hour
```



SQL: Built-in Window Functions - Rows Val - Example #12 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

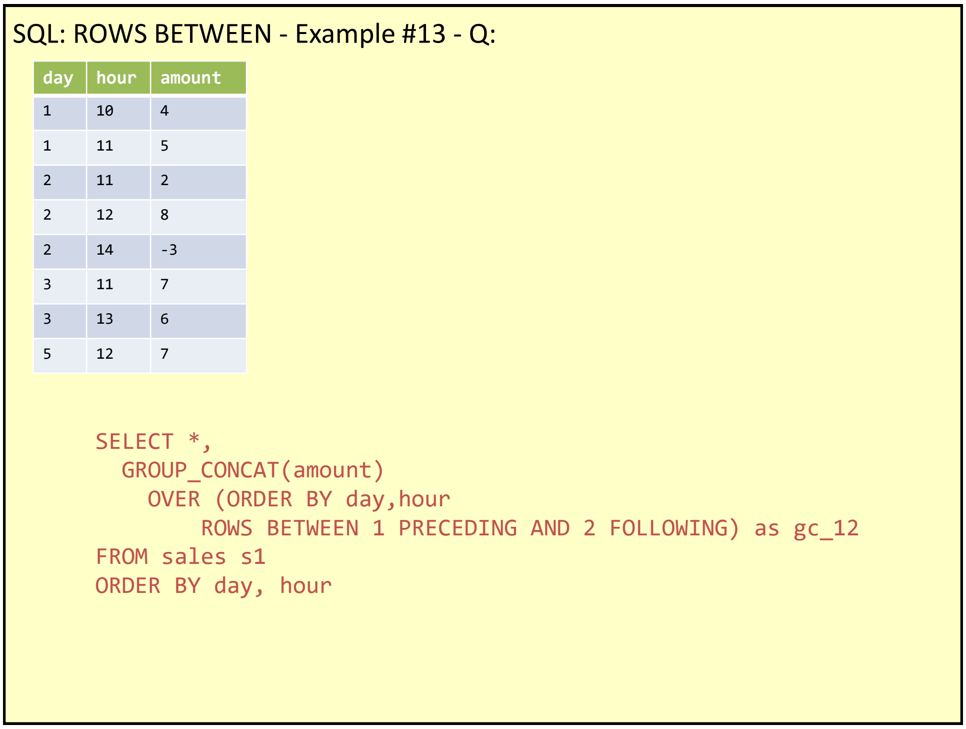
```
SELECT *,
   LAG(amount, 1, 'NA') OVER (ORDER BY day,hour) as lag_a1,
   LEAD(amount, 2, 'NA') OVER (ORDER BY day,hour) as lead_a2
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	lag_a1	lead_a2
1	10	4	NA	2
1	11	5	4	8
2	11	2	5	-3
2	12	8	2	7
2	14	-3	8	6
3	11	7	-3	7
3	13	6	7	NA
5	12	7	6	NA

LAG / LEAD with offset and default

Learn SQL by Example (Chananel Perel 2023)

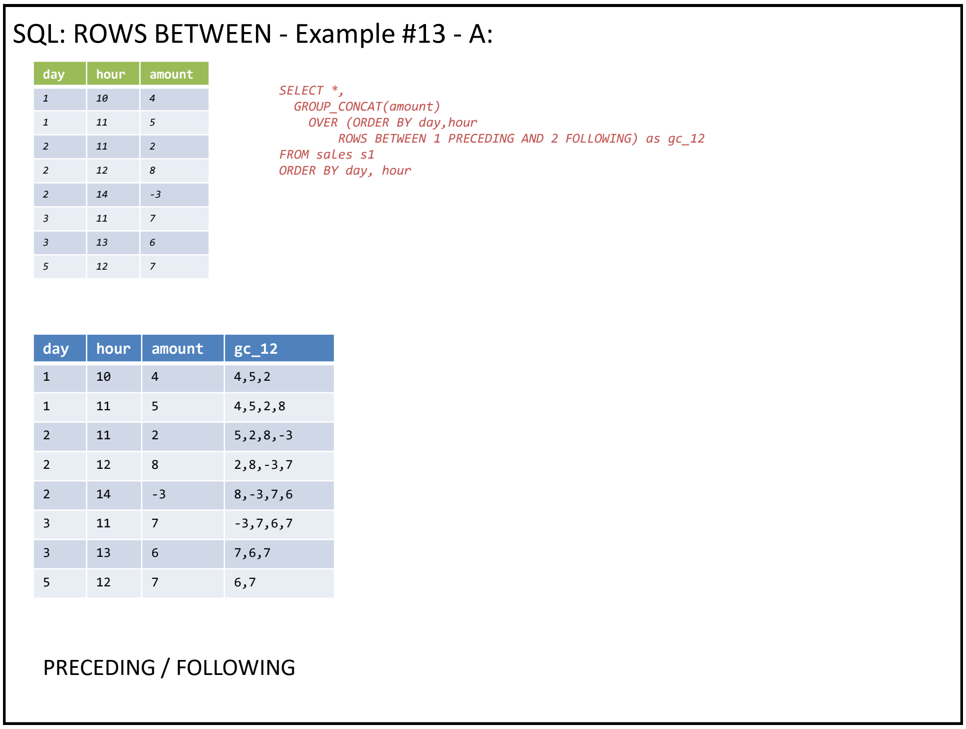
ROWS BETWEEN



SQL: ROWS BETWEEN - Example #13 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND 2 FOLLOWING) as gc_12
FROM sales s1
ORDER BY day, hour
```



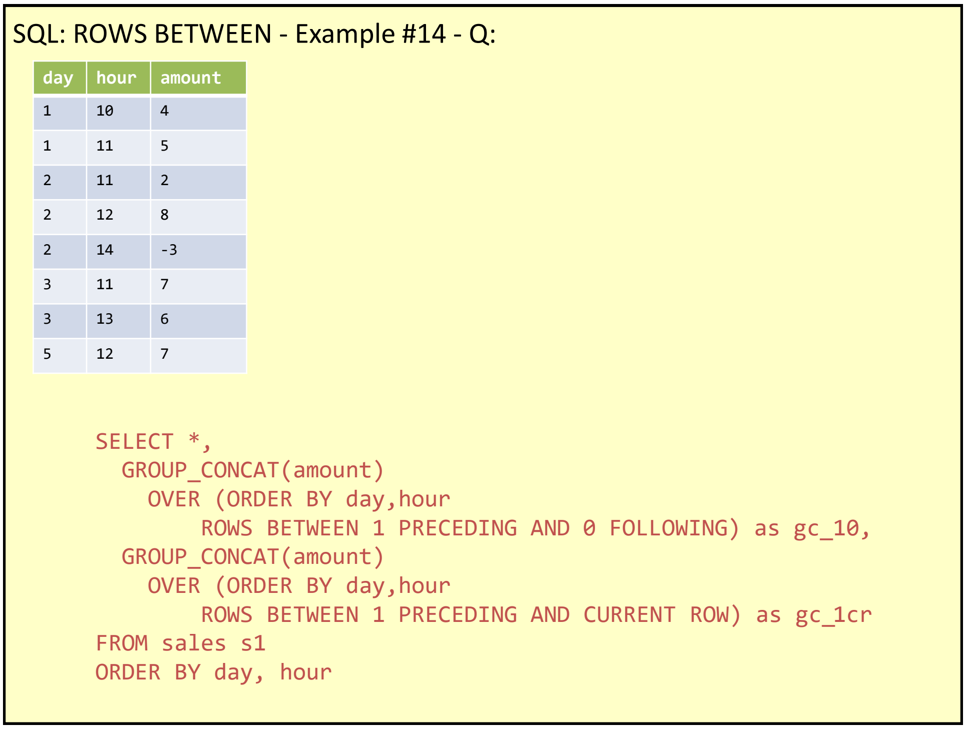
SQL: ROWS BETWEEN - Example #13 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND 2 FOLLOWING) as gc_12
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_12
1	10	4	4,5,2
1	11	5	4,5,2,8
2	11	2	5,2,8,-3
2	12	8	2,8,-3,7
2	14	-3	8,-3,7,6
3	11	7	-3,7,6,7
3	13	6	7,6,7
5	12	7	6,7

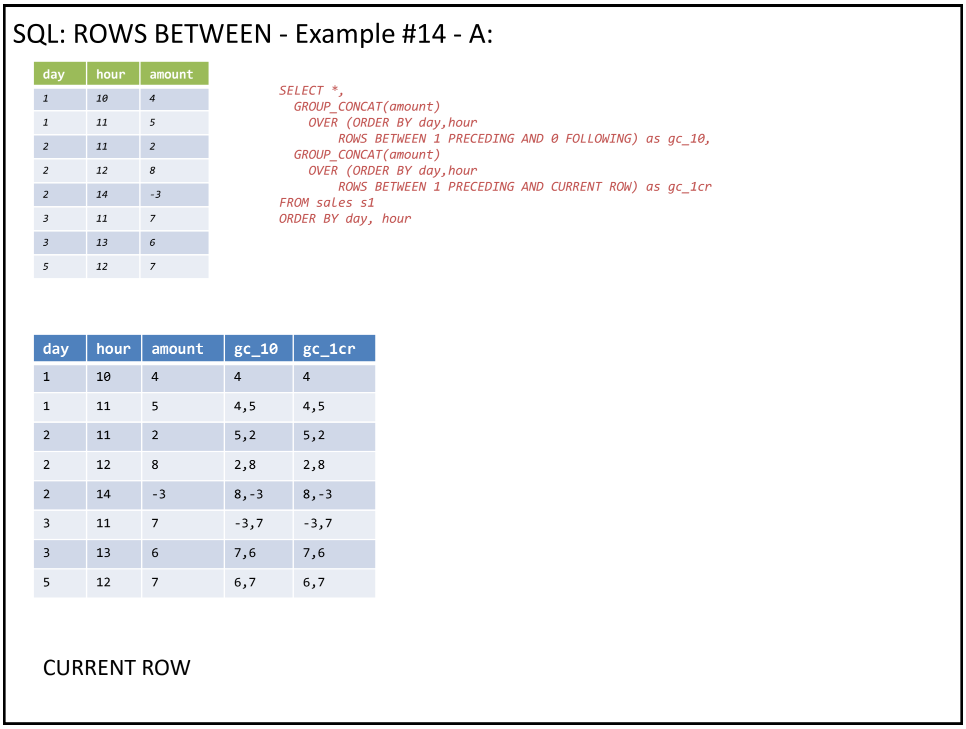
PRECEDING / FOLLOWING



SQL: ROWS BETWEEN - Example #14 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND 0 FOLLOWING) as gc_10,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as gc_1cr
FROM sales s1
ORDER BY day, hour
```



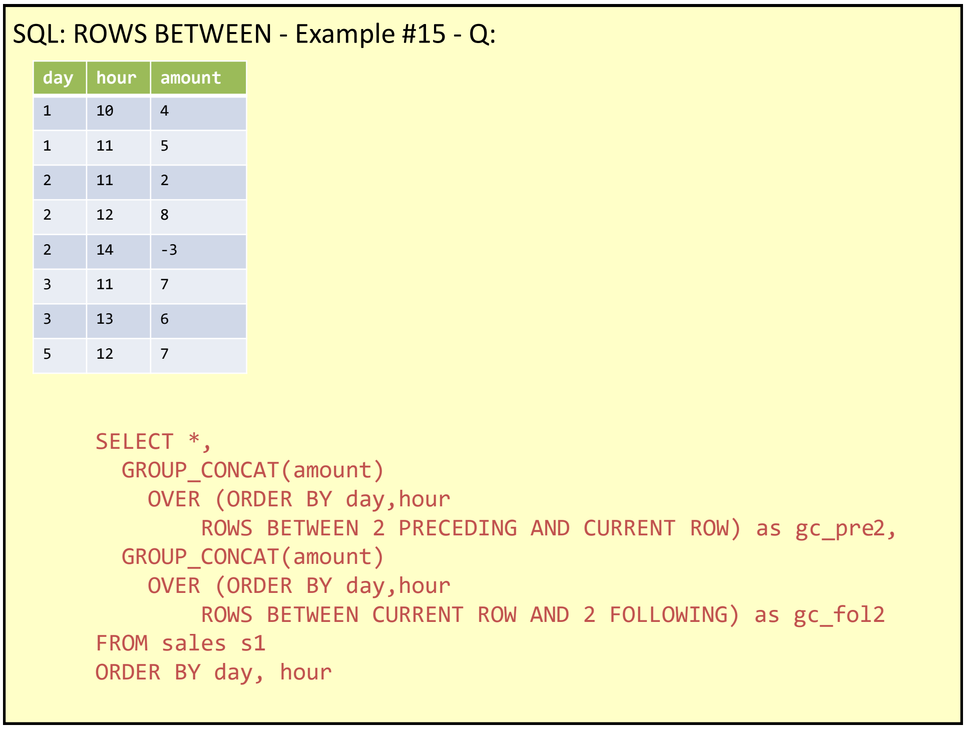
SQL: ROWS BETWEEN - Example #14 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND 0 FOLLOWING) as gc_10,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as gc_1cr
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_10	gc_1cr
1	10	4	4	4
1	11	5	4,5	4,5
2	11	2	5,2	5,2
2	12	8	2,8	2,8
2	14	-3	8,-3	8,-3
3	11	7	-3,7	-3,7
3	13	6	7,6	7,6
5	12	7	6,7	6,7

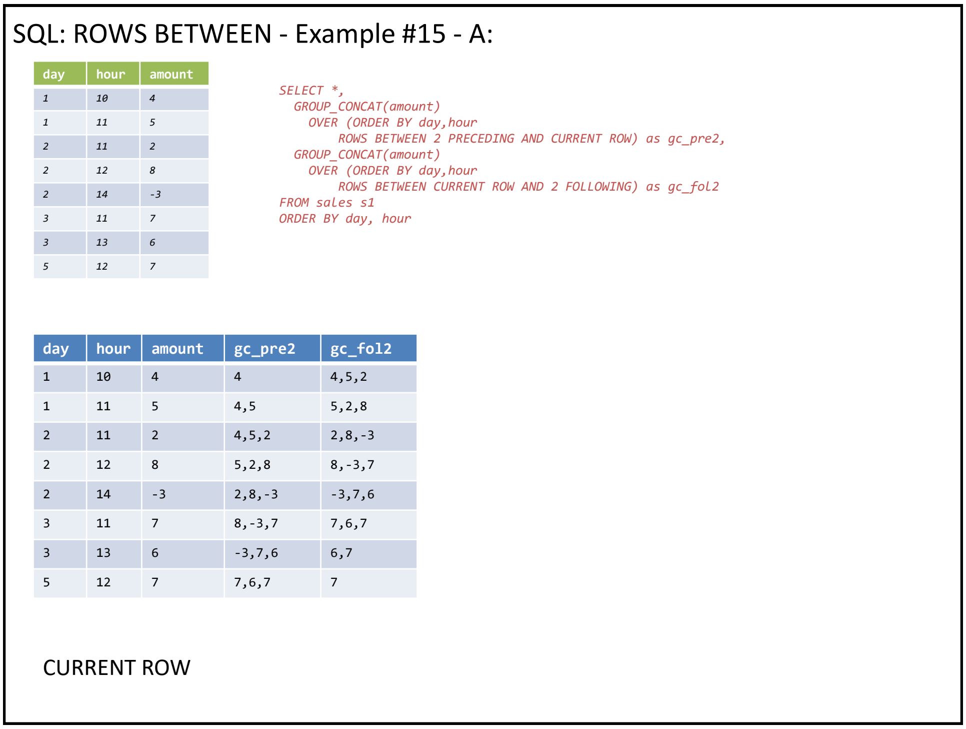
CURRENT ROW



SQL: ROWS BETWEEN - Example #15 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) as gc_pre2,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN CURRENT ROW AND 2 FOLLOWING) as gc_fol2
FROM sales s1
ORDER BY day, hour
```



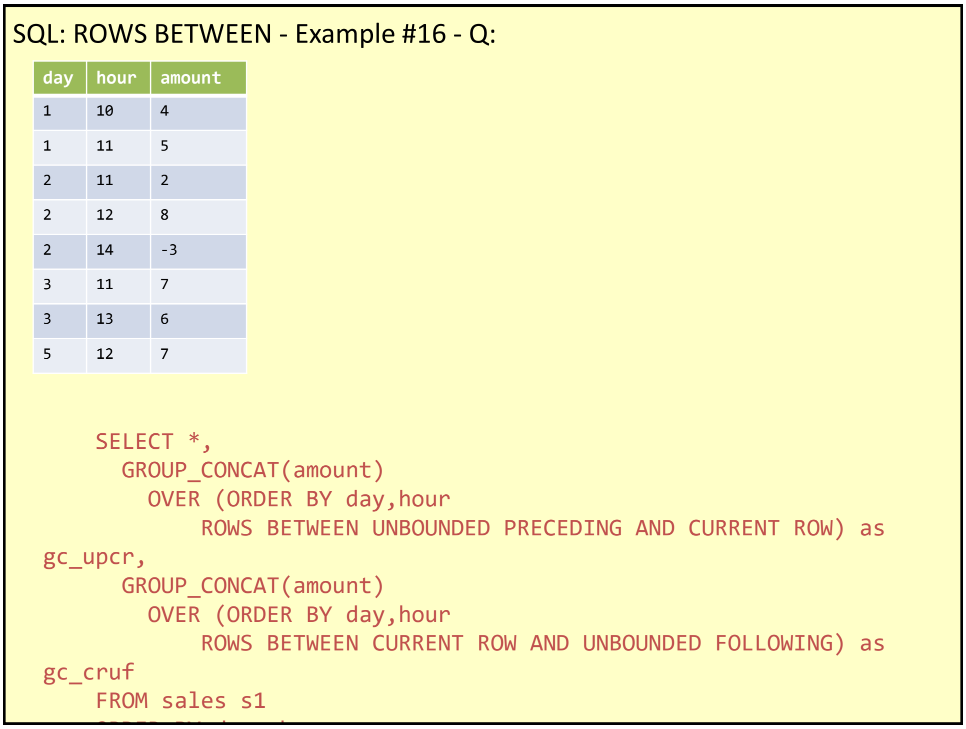
SQL: ROWS BETWEEN - Example #15 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) as gc_pre2,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN CURRENT ROW AND 2 FOLLOWING) as gc_fol2
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_pre2	gc_fol2
1	10	4	4	4,5,2
1	11	5	4,5	5,2,8
2	11	2	4,5,2	2,8,-3
2	12	8	5,2,8	8,-3,7
2	14	-3	2,8,-3	-3,7,6
3	11	7	8,-3,7	7,6,7
3	13	6	-3,7,6	6,7
5	12	7	7,6,7	7

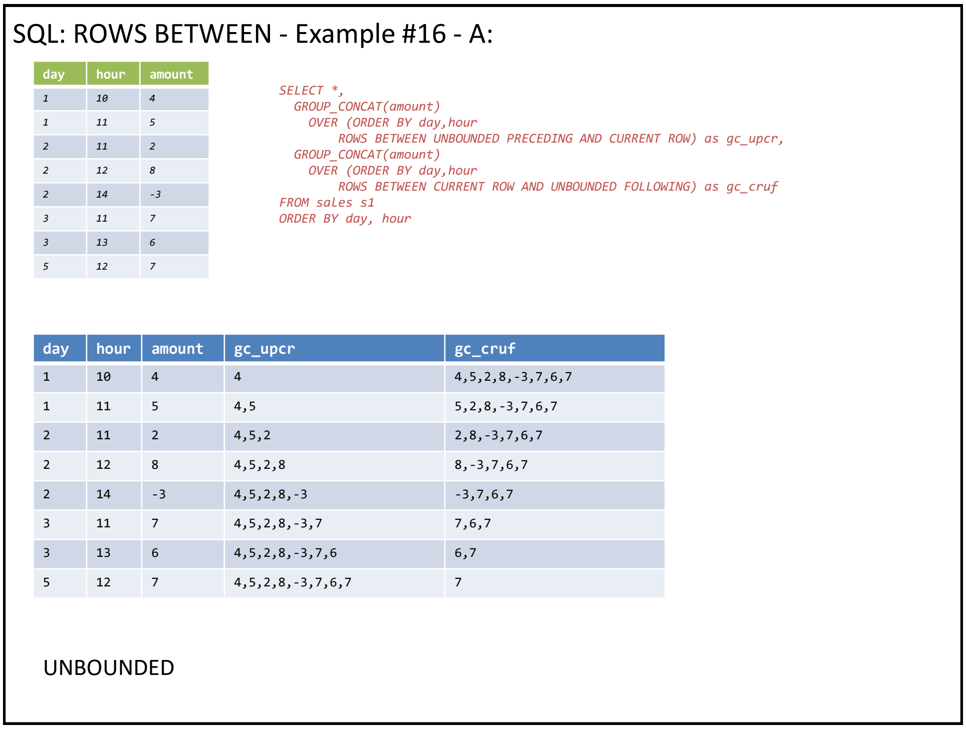
CURRENT ROW



SQL: ROWS BETWEEN - Example #16 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
    SELECT *,
      GROUP_CONCAT(amount)
        OVER (ORDER BY day,hour
            ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as 
gc_upcr,
      GROUP_CONCAT(amount)
        OVER (ORDER BY day,hour
            ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) as 
gc_cruf
    FROM sales s1
```



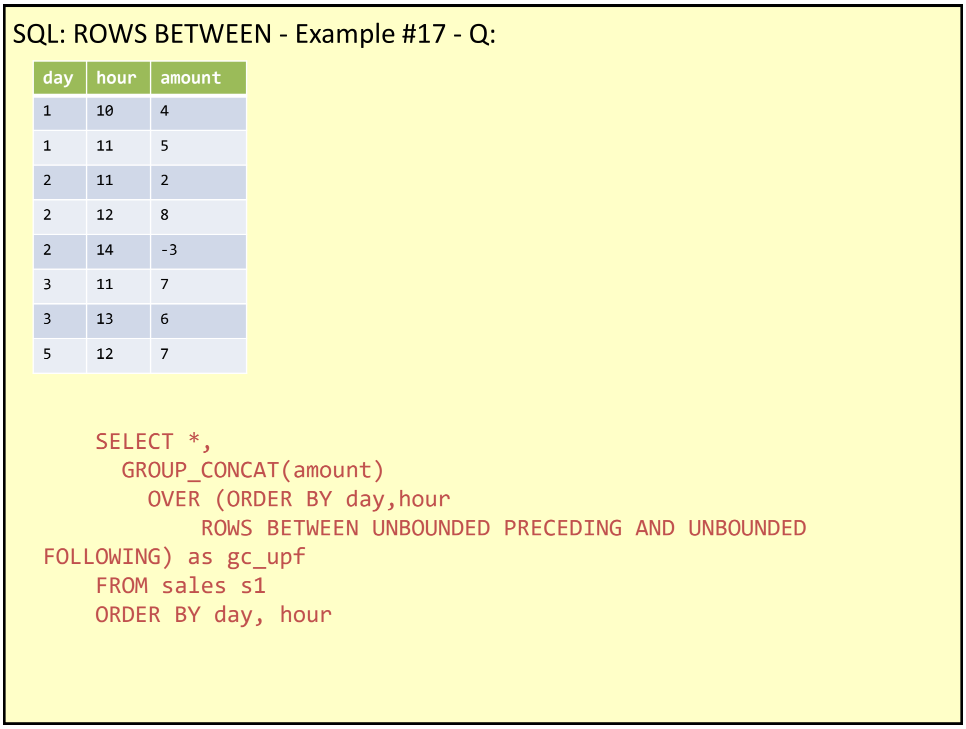
SQL: ROWS BETWEEN - Example #16 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) as gc_upcr,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) as gc_cruf
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_upcr	gc_cruf
1	10	4	4	4,5,2,8,-3,7,6,7
1	11	5	4,5	5,2,8,-3,7,6,7
2	11	2	4,5,2	2,8,-3,7,6,7
2	12	8	4,5,2,8	8,-3,7,6,7
2	14	-3	4,5,2,8,-3	-3,7,6,7
3	11	7	4,5,2,8,-3,7	7,6,7
3	13	6	4,5,2,8,-3,7,6	6,7
5	12	7	4,5,2,8,-3,7,6,7	7

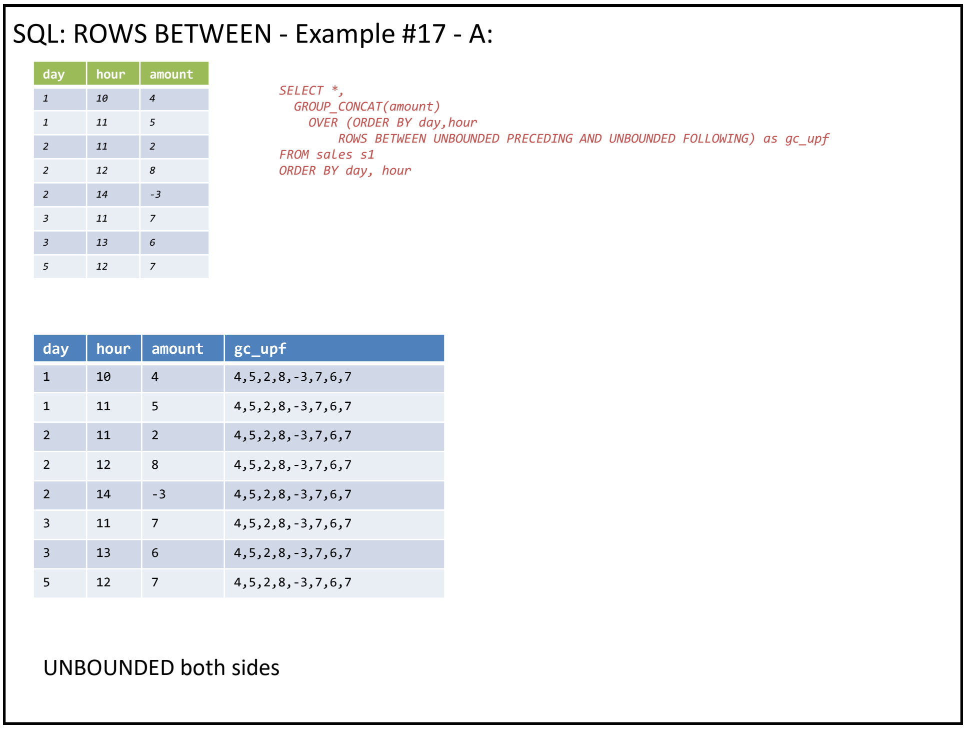
UNBOUNDED



SQL: ROWS BETWEEN - Example #17 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
    SELECT *,
      GROUP_CONCAT(amount)
        OVER (ORDER BY day,hour
            ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED
FOLLOWING) as gc_upf
    FROM sales s1
    ORDER BY day, hour
```



SQL: ROWS BETWEEN - Example #17 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

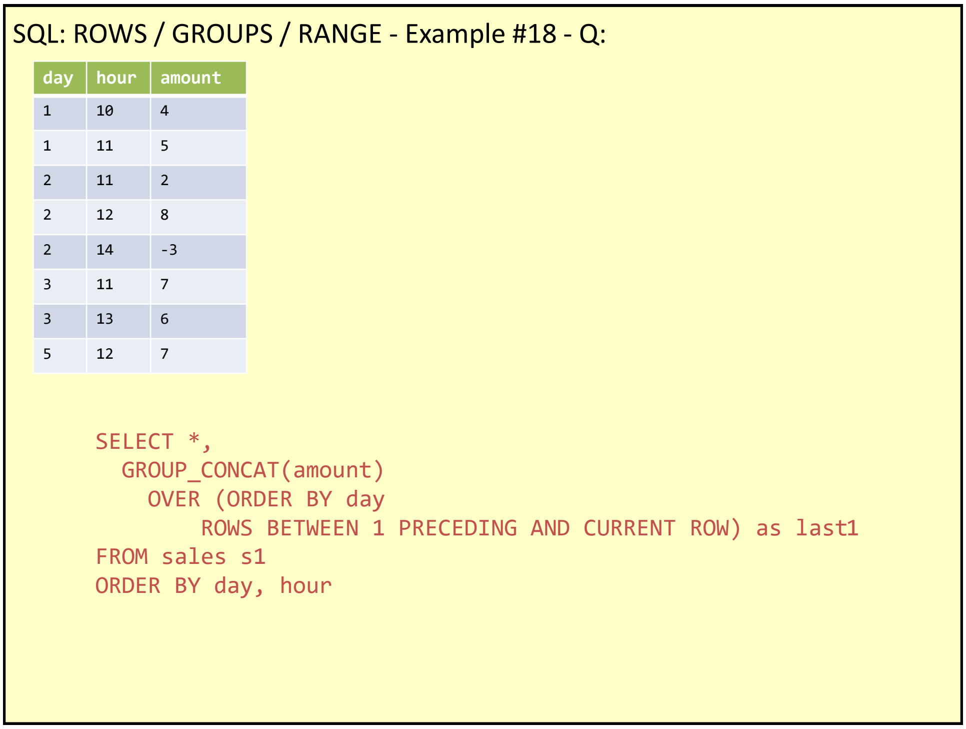
```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day,hour
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as gc_upf
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_upf
1	10	4	4,5,2,8,-3,7,6,7
1	11	5	4,5,2,8,-3,7,6,7
2	11	2	4,5,2,8,-3,7,6,7
2	12	8	4,5,2,8,-3,7,6,7
2	14	-3	4,5,2,8,-3,7,6,7
3	11	7	4,5,2,8,-3,7,6,7
3	13	6	4,5,2,8,-3,7,6,7
5	12	7	4,5,2,8,-3,7,6,7

UNBOUNDED both sides

Learn SQL by Example (Chananel Perel 2023)

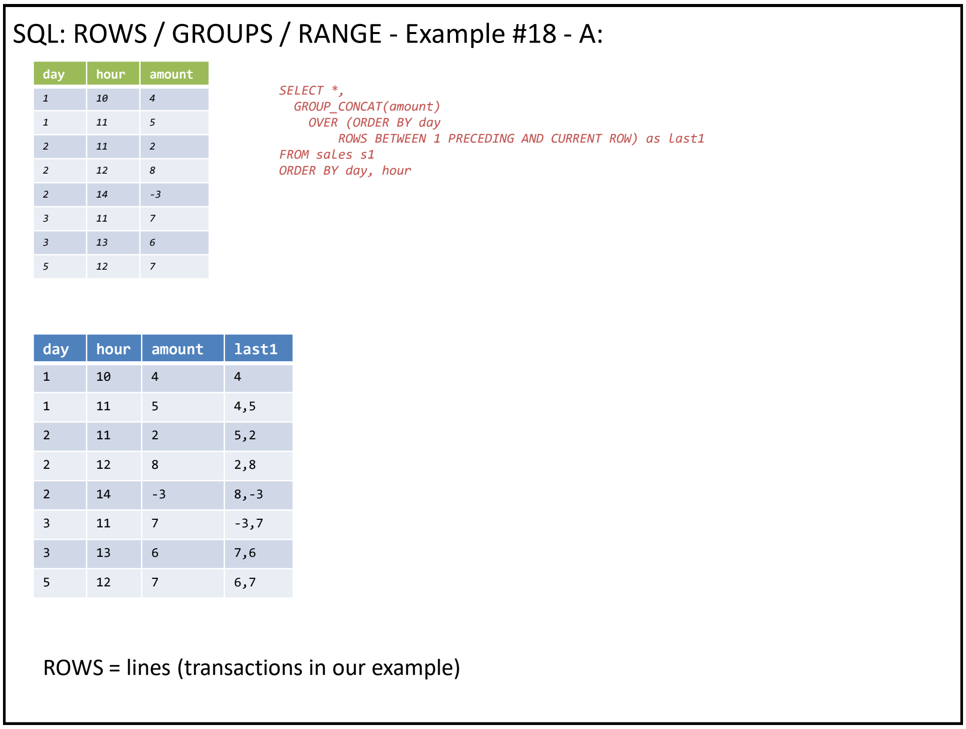
ROWS / GROUPS / RANGE



SQL: ROWS / GROUPS / RANGE - Example #18 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```



SQL: ROWS / GROUPS / RANGE - Example #18 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	last1
1	10	4	4
1	11	5	4,5
2	11	2	5,2
2	12	8	2,8
2	14	-3	8,-3
3	11	7	-3,7
3	13	6	7,6
5	12	7	6,7

ROWS = lines (transactions in our example)

SQL: ROWS / GROUPS / RANGE - Example #19 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        GROUPS BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```

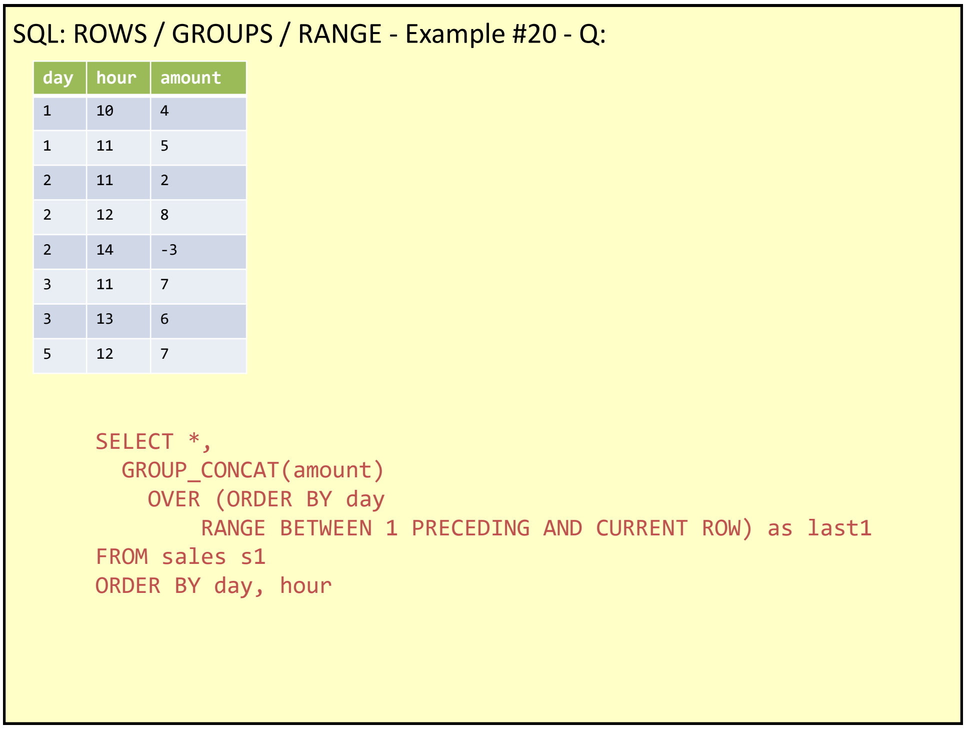
SQL: ROWS / GROUPS / RANGE - Example #19 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        GROUPS BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	last1
1	10	4	4,5
1	11	5	4,5
2	11	2	4,5,2,8,-3
2	12	8	4,5,2,8,-3
2	14	-3	4,5,2,8,-3
3	11	7	2,8,-3,7,6
3	13	6	2,8,-3,7,6
5	12	7	7,6,7

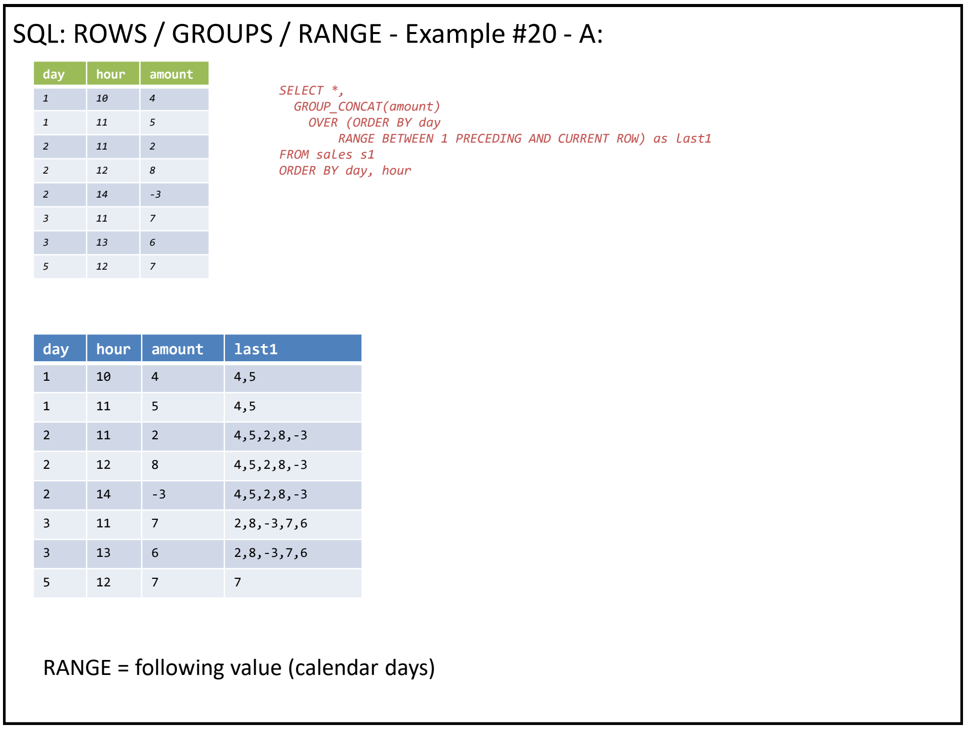
GROUPS = same vals (days in db)



SQL: ROWS / GROUPS / RANGE - Example #20 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        RANGE BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```



SQL: ROWS / GROUPS / RANGE - Example #20 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

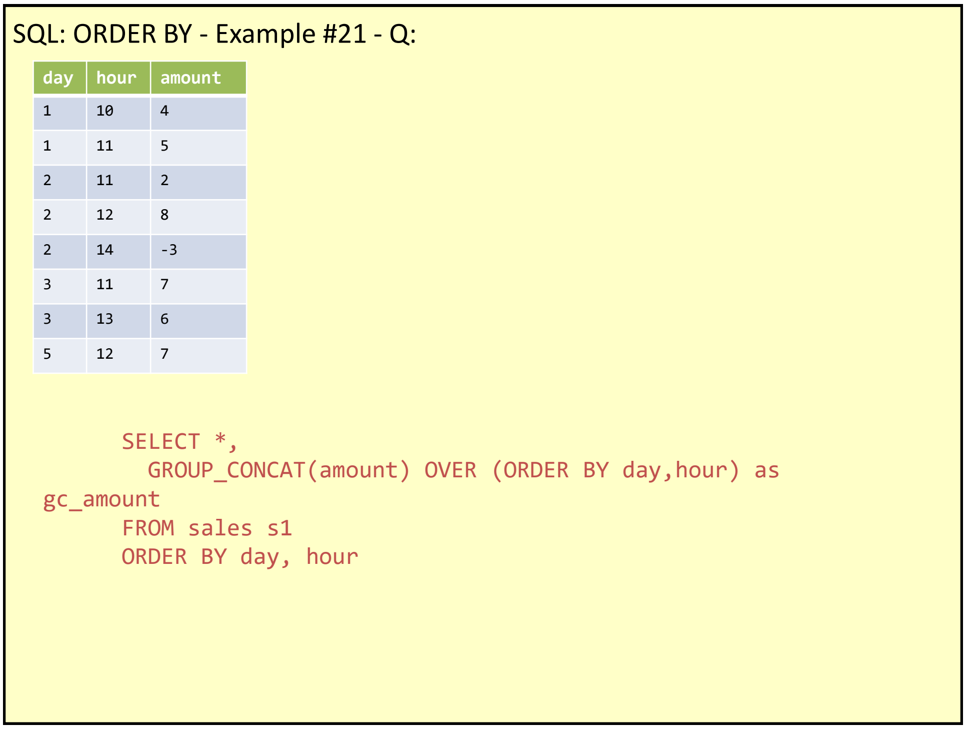
```
SELECT *,
  GROUP_CONCAT(amount)
    OVER (ORDER BY day
        RANGE BETWEEN 1 PRECEDING AND CURRENT ROW) as last1
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	last1
1	10	4	4,5
1	11	5	4,5
2	11	2	4,5,2,8,-3
2	12	8	4,5,2,8,-3
2	14	-3	4,5,2,8,-3
3	11	7	2,8,-3,7,6
3	13	6	2,8,-3,7,6
5	12	7	7

RANGE = following value (calendar days)

Learn SQL by Example (Chananel Perel 2023)

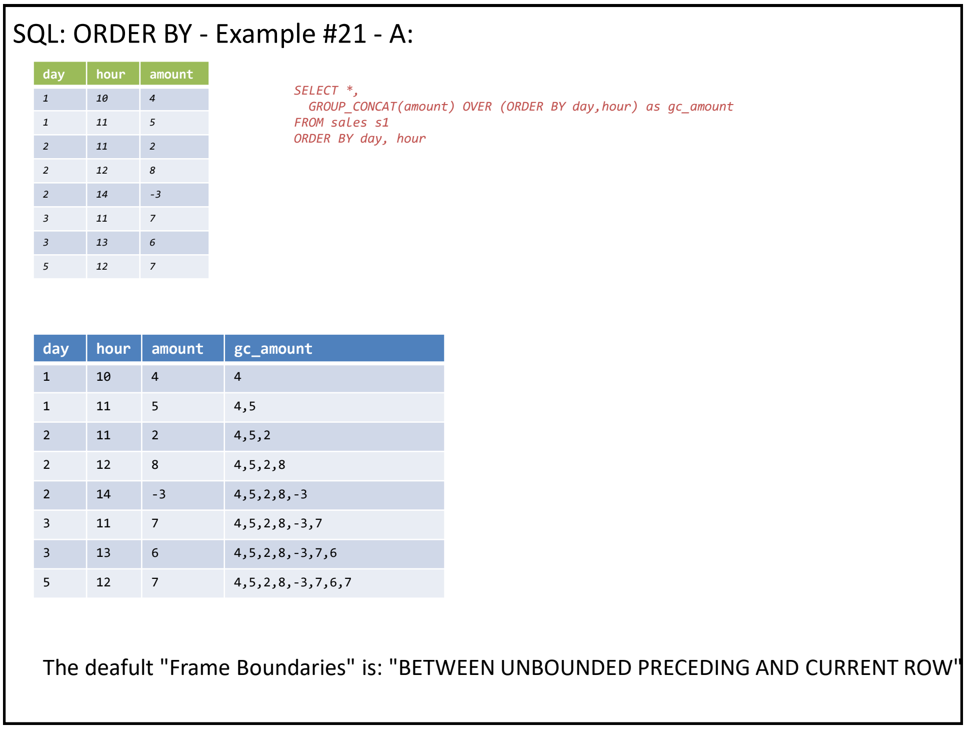
ORDER BY



SQL: ORDER BY - Example #21 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      SELECT *,
        GROUP_CONCAT(amount) OVER (ORDER BY day,hour) as
gc_amount
      FROM sales s1
      ORDER BY day, hour
```



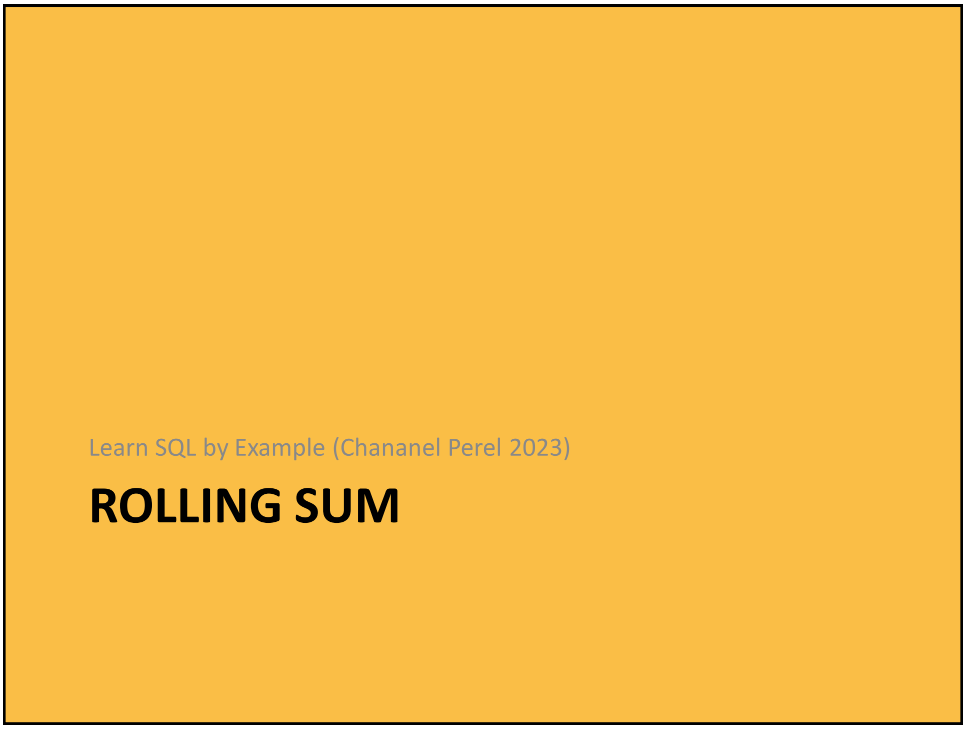
SQL: ORDER BY - Example #21 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
  GROUP_CONCAT(amount) OVER (ORDER BY day,hour) as gc_amount
FROM sales s1
ORDER BY day, hour
```

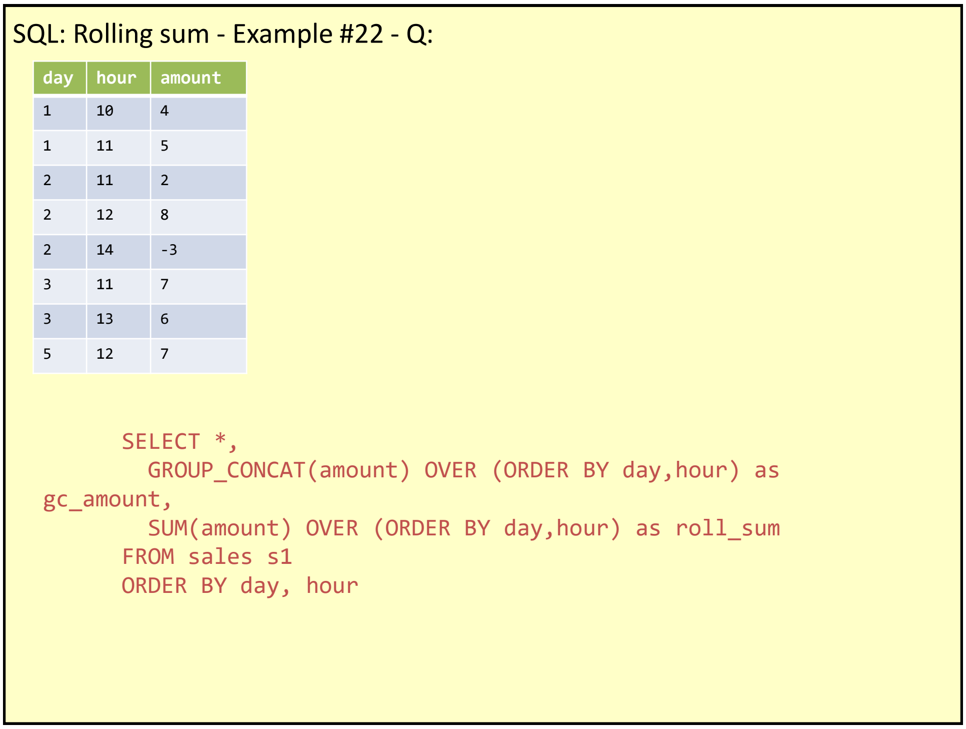
day	hour	amount	gc_amount
1	10	4	4
1	11	5	4,5
2	11	2	4,5,2
2	12	8	4,5,2,8
2	14	-3	4,5,2,8,-3
3	11	7	4,5,2,8,-3,7
3	13	6	4,5,2,8,-3,7,6
5	12	7	4,5,2,8,-3,7,6,7

The deafult "Frame Boundaries" is: "BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW"



Learn SQL by Example (Chananel Perel 2023)

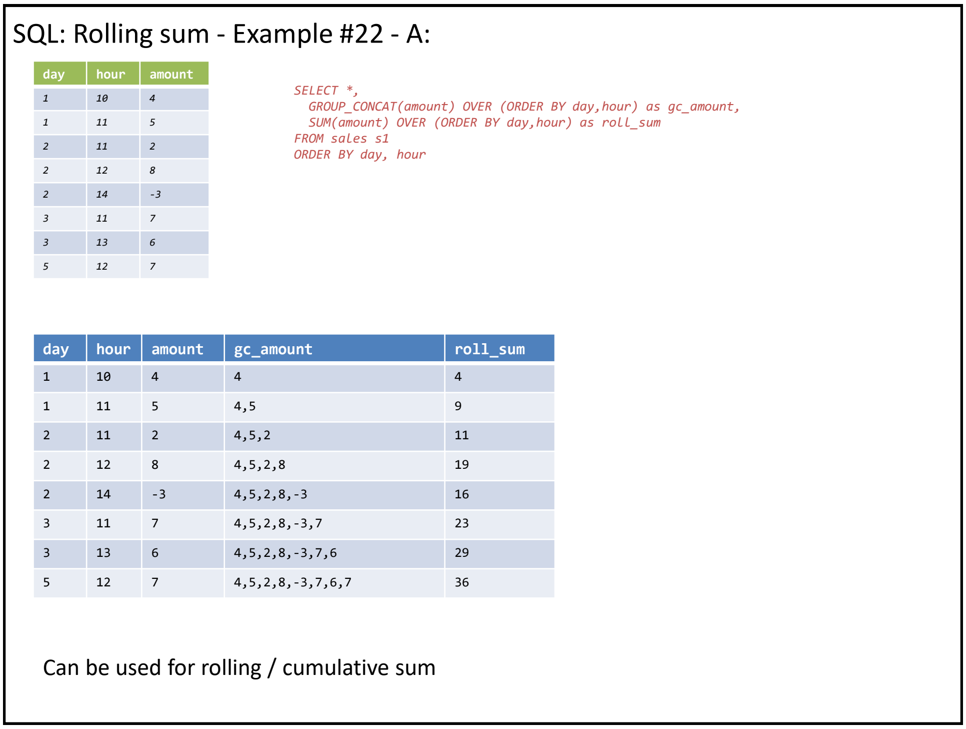
ROLLING SUM



SQL: Rolling sum - Example #22 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      SELECT *,
        GROUP_CONCAT(amount) OVER (ORDER BY day,hour) as
gc_amount,
        SUM(amount) OVER (ORDER BY day,hour) as roll_sum
      FROM sales s1
      ORDER BY day, hour
```



SQL: Rolling sum - Example #22 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

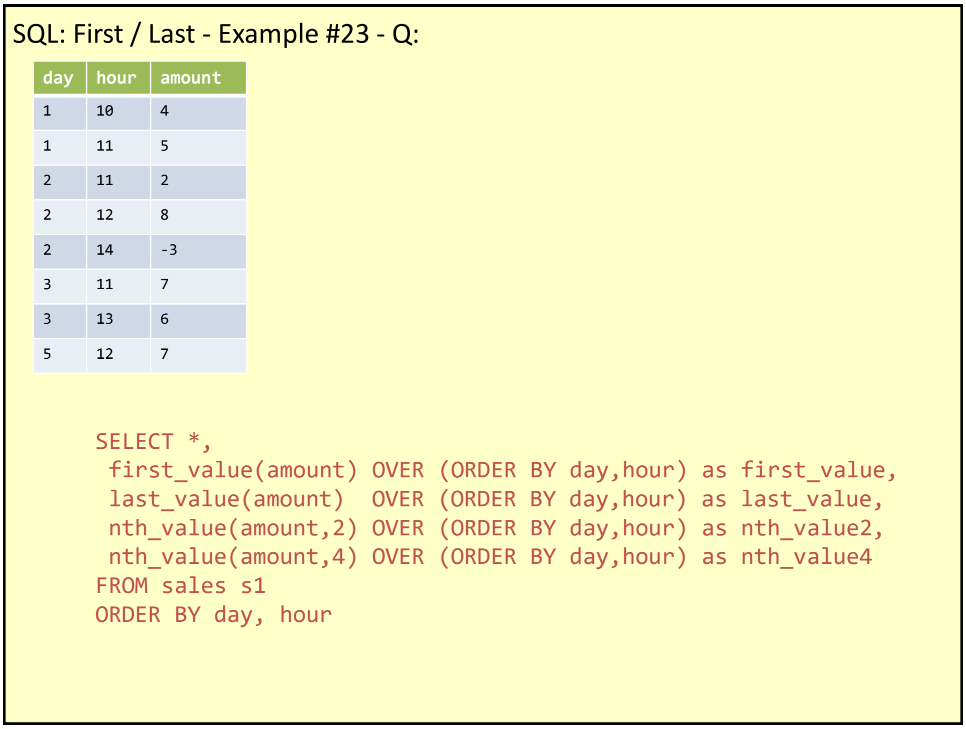
```
SELECT *,
  GROUP_CONCAT(amount) OVER (ORDER BY day,hour) as gc_amount,
  SUM(amount) OVER (ORDER BY day,hour) as roll_sum
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	gc_amount	roll_sum
1	10	4	4	4
1	11	5	4,5	9
2	11	2	4,5,2	11
2	12	8	4,5,2,8	19
2	14	-3	4,5,2,8,-3	16
3	11	7	4,5,2,8,-3,7	23
3	13	6	4,5,2,8,-3,7,6	29
5	12	7	4,5,2,8,-3,7,6,7	36

Can be used for rolling / cumulative sum

Learn SQL by Example (Chananel Perel 2023)

FIRST / LAST



SQL: First / Last - Example #23 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
 first_value(amount) OVER (ORDER BY day,hour) as first_value,
 last_value(amount)  OVER (ORDER BY day,hour) as last_value,
 nth_value(amount,2) OVER (ORDER BY day,hour) as nth_value2,
 nth_value(amount,4) OVER (ORDER BY day,hour) as nth_value4
FROM sales s1
ORDER BY day, hour
```

SQL: First / Last - Example #23 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
SELECT *,
 first_value(amount) OVER (ORDER BY day,hour) as first_value,
 last_value(amount)  OVER (ORDER BY day,hour) as last_value,
 nth_value(amount,2) OVER (ORDER BY day,hour) as nth_value2,
 nth_value(amount,4) OVER (ORDER BY day,hour) as nth_value4
FROM sales s1
ORDER BY day, hour
```

day	hour	amount	first_value	last_value	nth_value2	nth_value4
1	10	4	4	4	None	None
1	11	5	4	5	5	None
2	11	2	4	2	5	None
2	12	8	4	8	5	8
2	14	-3	4	-3	5	8
3	11	7	4	7	5	8
3	13	6	4	6	5	8
5	12	7	4	7	5	8

first / last / nth value

Learn SQL by Example (Chananel Perel 2023)

FIND ALL TRANSACTIONS THAT ARE MORE THAN *3 OF PREVIOUS ONE

SQL: Find all transactions that are more than *3 of previous one - Example #24 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (ORDER BY day,hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
ORDER BY day, hour
```

SQL: Find all transactions that are more than *3 of previous one - Example #24 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (ORDER BY day,hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
ORDER BY day, hour
```

day	hour	amount	lag_a
1	10	4	None
1	11	5	4
2	11	2	5
2	12	8	2
2	14	-3	8
3	11	7	-3
3	13	6	7
5	12	7	6

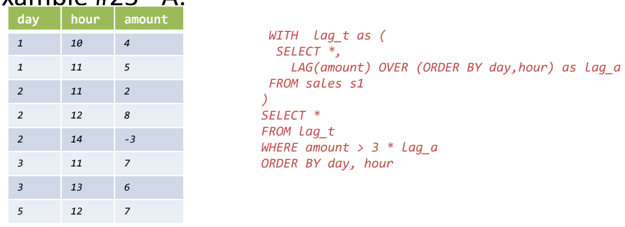
This we already saw

SQL: Find all transactions that are more than *3 of previous one - Example #25 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (ORDER BY day,hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
WHERE amount > 3 * lag_a
ORDER BY day, hour
```

SQL: Find all transactions that are more than *3 of previous one - Example #25 - A:

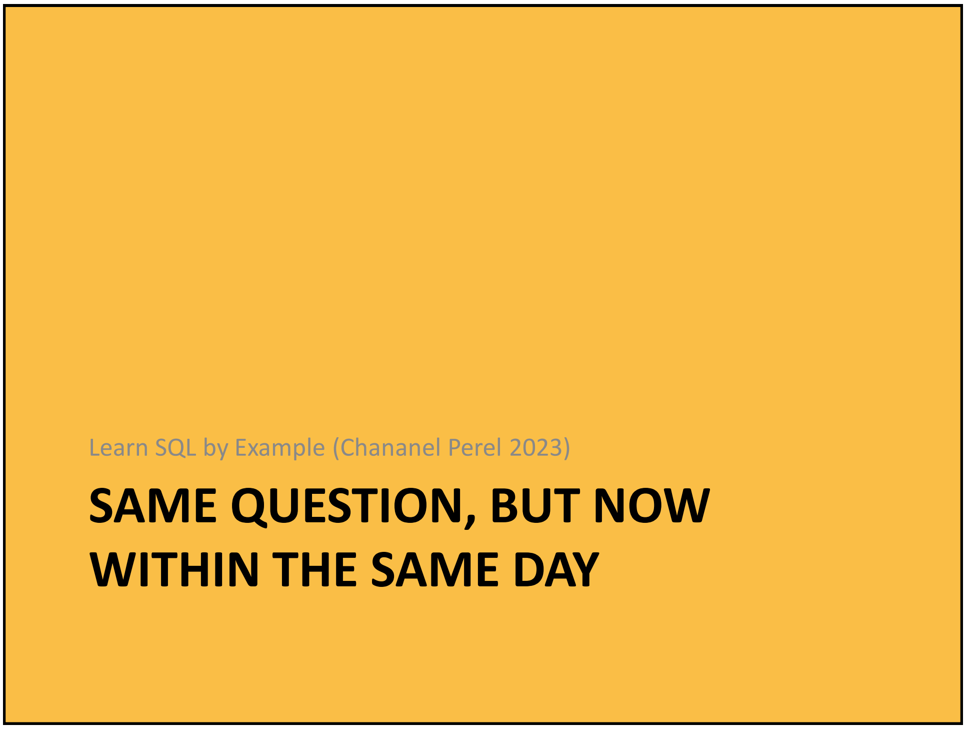


day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (ORDER BY day,hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
WHERE amount > 3 * lag_a
ORDER BY day, hour
```

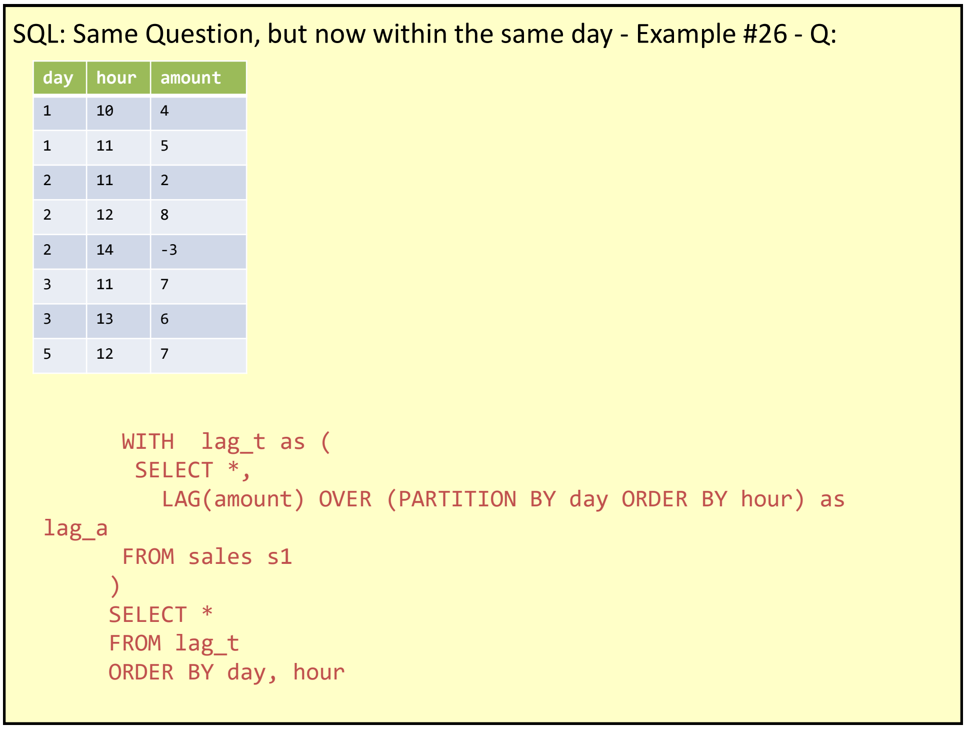
day	hour	amount	lag_a
2	12	8	2
3	11	7	-3

and now we can use WHERE as needed



Learn SQL by Example (Chananel Perel 2023)

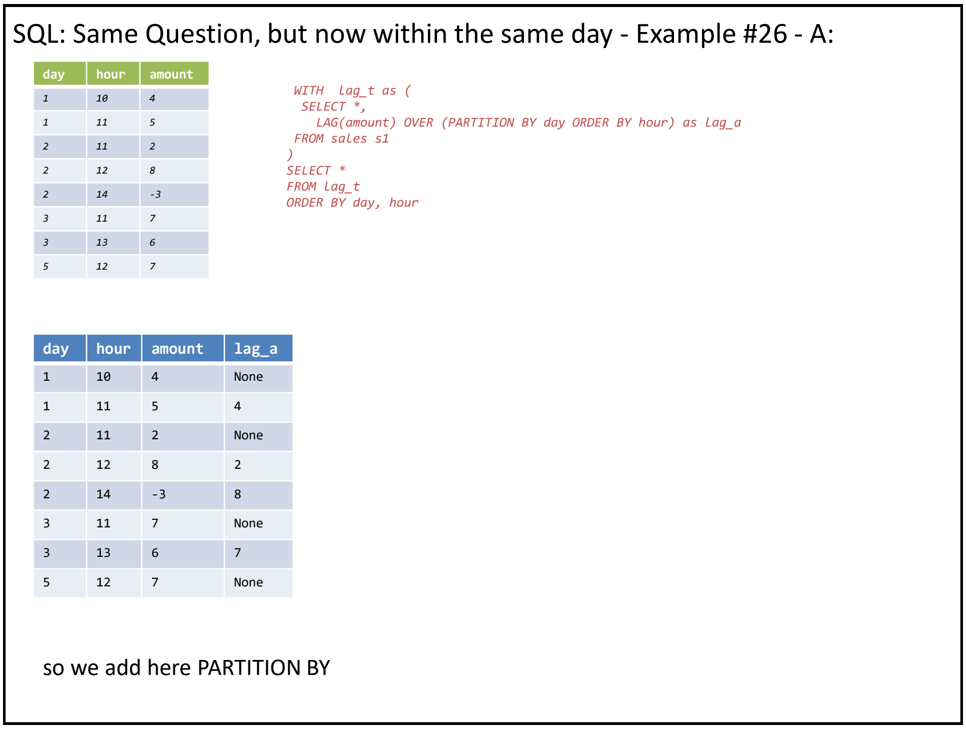
SAME QUESTION, BUT NOW WITHIN THE SAME DAY



SQL: Same Question, but now within the same day - Example #26 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      WITH  lag_t as (
       SELECT *,
         LAG(amount) OVER (PARTITION BY day ORDER BY hour) as
lag_a
      FROM sales s1
     )
     SELECT *
     FROM lag_t
     ORDER BY day, hour
```



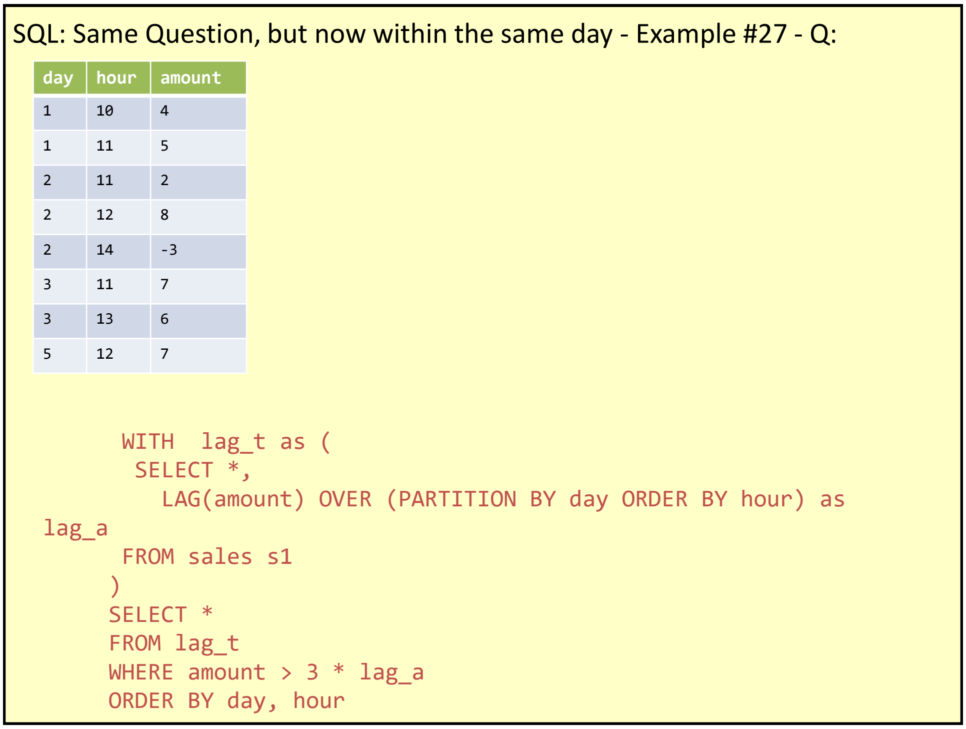
SQL: Same Question, but now within the same day - Example #26 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (PARTITION BY day ORDER BY hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
ORDER BY day, hour
```

day	hour	amount	lag_a
1	10	4	None
1	11	5	4
2	11	2	None
2	12	8	2
2	14	-3	8
3	11	7	None
3	13	6	7
5	12	7	None

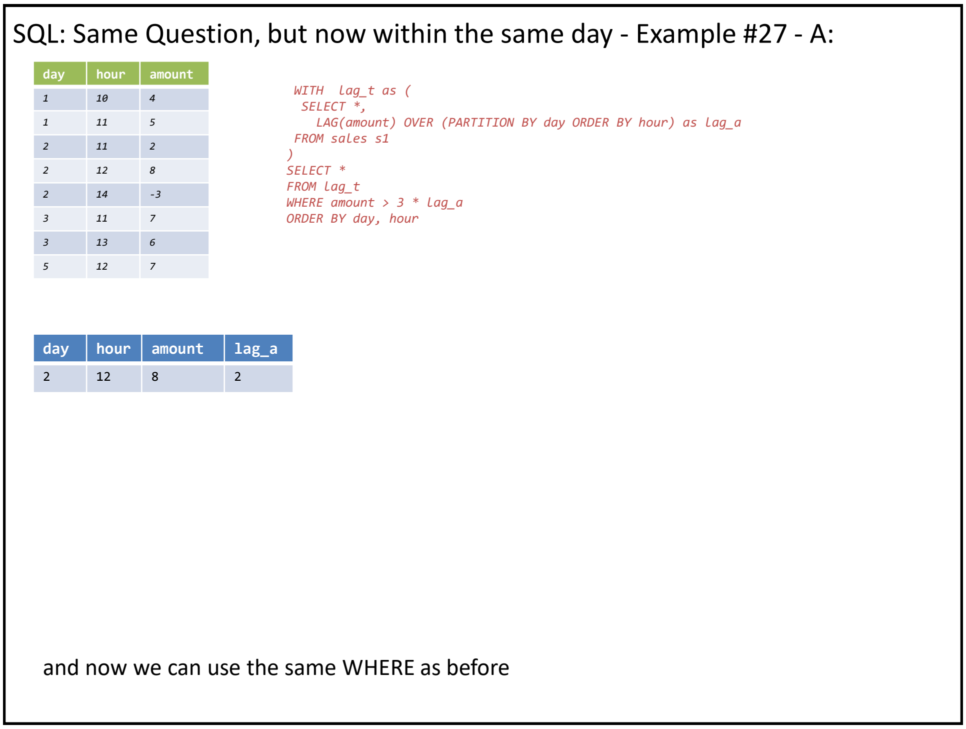
so we add here PARTITION BY



SQL: Same Question, but now within the same day - Example #27 - Q:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
      WITH  lag_t as (
       SELECT *,
         LAG(amount) OVER (PARTITION BY day ORDER BY hour) as
lag_a
      FROM sales s1
     )
     SELECT *
     FROM lag_t
     WHERE amount > 3 * lag_a
     ORDER BY day, hour
```



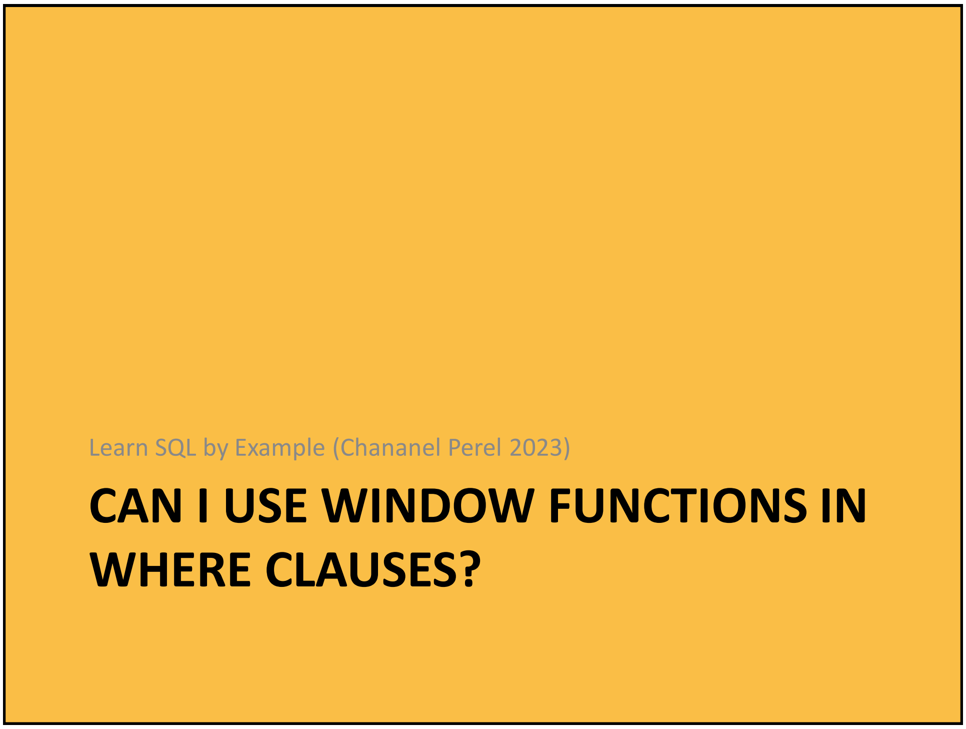
SQL: Same Question, but now within the same day - Example #27 - A:

day	hour	amount
1	10	4
1	11	5
2	11	2
2	12	8
2	14	-3
3	11	7
3	13	6
5	12	7

```
 WITH  lag_t as (
  SELECT *,
    LAG(amount) OVER (PARTITION BY day ORDER BY hour) as lag_a
 FROM sales s1
)
SELECT *
FROM lag_t
WHERE amount > 3 * lag_a
ORDER BY day, hour
```

day	hour	amount	lag_a
2	12	8	2

and now we can use the same WHERE as before



Learn SQL by Example (Chananel Perel 2023)

CAN I USE WINDOW FUNCTIONS IN WHERE CLAUSES?

Learn SQL by Example (Chananel Perel) 2024-05-01 15:55:35.111117

RECURSION

Learn SQL by Example (Chananel Perel 2023)

WITH RECURSIVE

SQL: WITH RECURSIVE - Example #28 - Q:

```
 WITH RECURSIVE cnt(x) AS (
     SELECT 1
     UNION ALL
     SELECT x+1 FROM cnt WHERE x < 9)
SELECT x FROM cnt
```

SQL: WITH RECURSIVE - Example #28 - A:

```
 WITH RECURSIVE cnt(x) AS (
     SELECT 1
     UNION ALL
     SELECT x+1 FROM cnt WHERE x < 9)
SELECT x FROM cnt
```

x
1
2
3
4
5
6
7
8

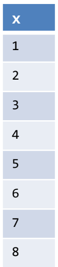
using WHERE, all integers between 1 and 9

SQL: WITH RECURSIVE - Example #29 - Q:

```
 WITH RECURSIVE cnt(x) AS (
     SELECT 1
     UNION ALL
     SELECT x+1 FROM cnt
     LIMIT 9)
SELECT x FROM cnt
```

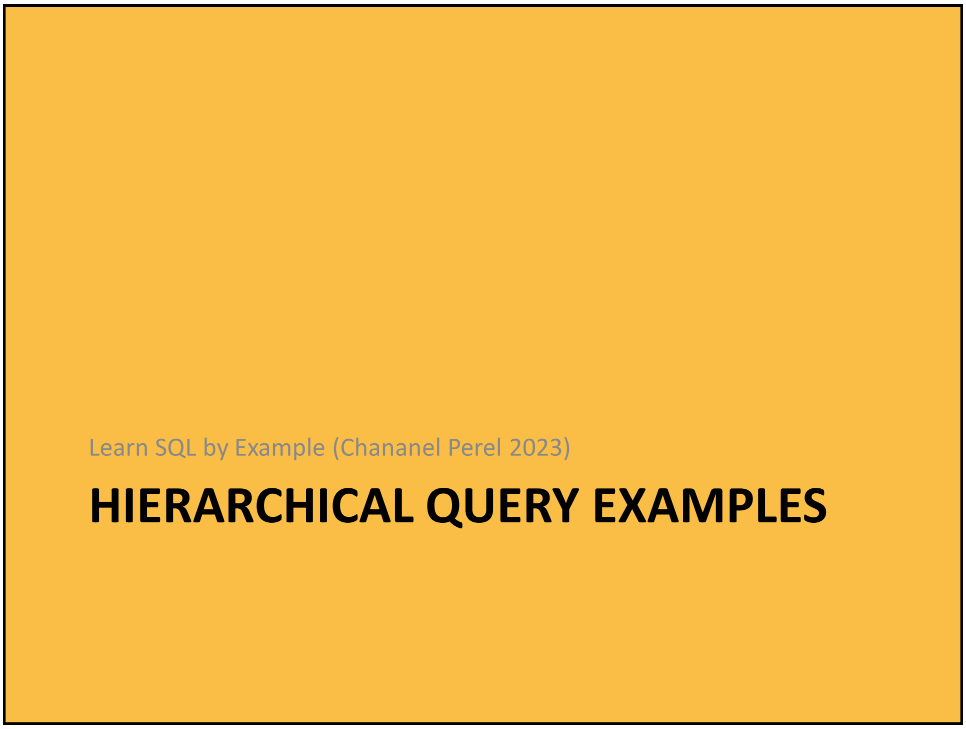
SQL: WITH RECURSIVE - Example #29 - A:

```
 WITH RECURSIVE cnt(x) AS (
     SELECT 1
     UNION ALL
     SELECT x+1 FROM cnt
     LIMIT 9)
SELECT x FROM cnt
```



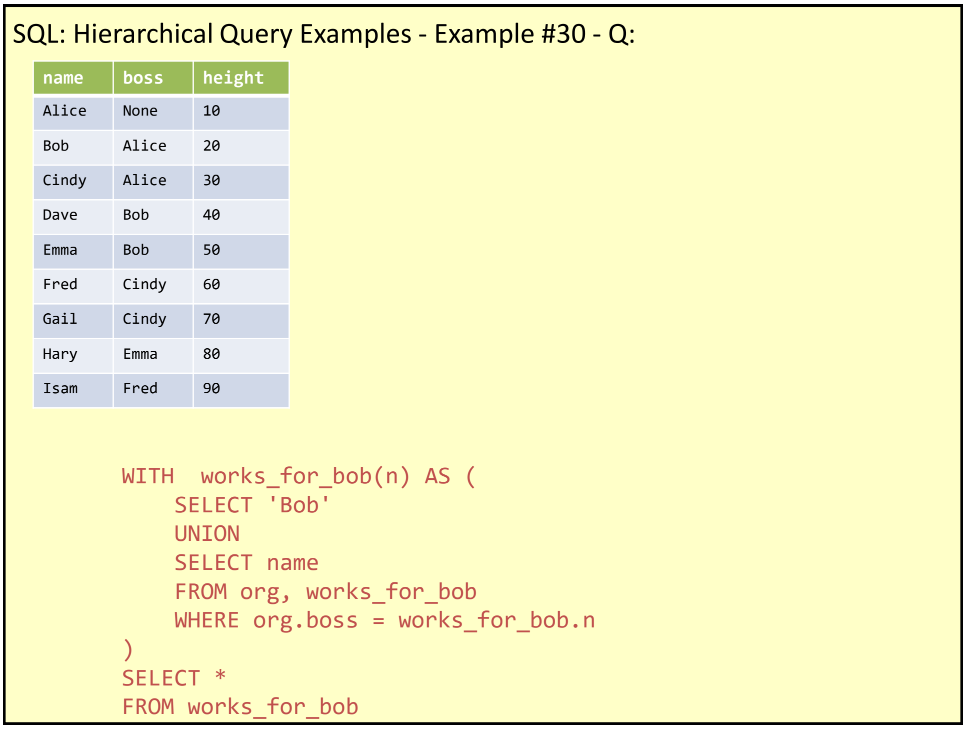
x
1
2
3
4
5
6
7
8

we could also use LIMIT



Learn SQL by Example (Chananel Perel 2023)

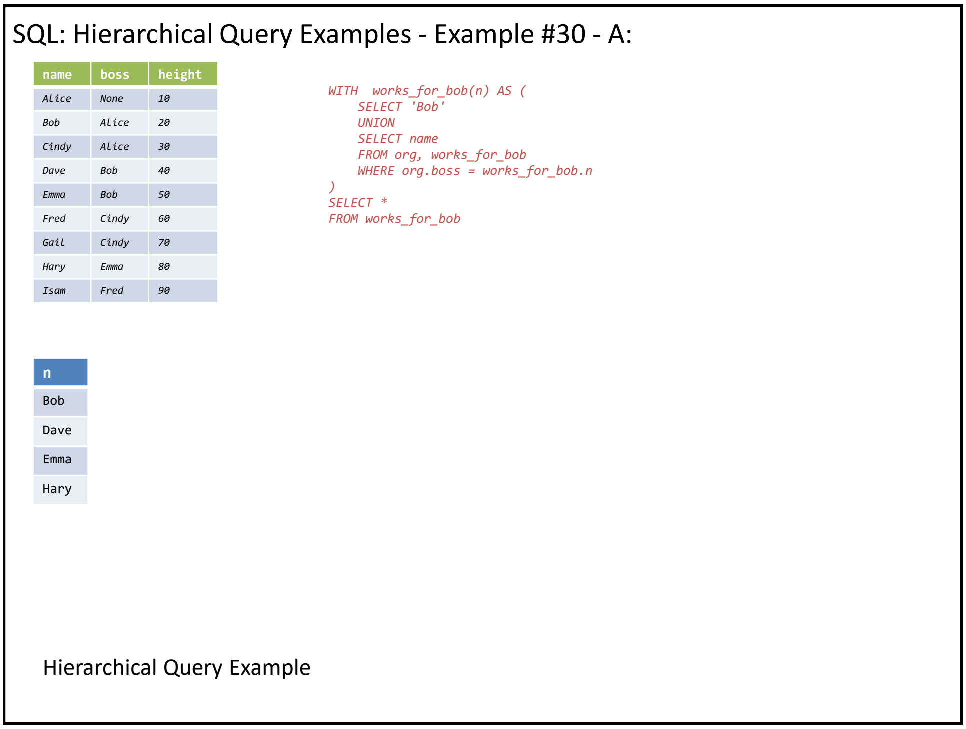
HIERARCHICAL QUERY EXAMPLES



SQL: Hierarchical Query Examples - Example #30 - Q:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  works_for_bob(n) AS (
    SELECT 'Bob'
    UNION
    SELECT name
    FROM org, works_for_bob
    WHERE org.boss = works_for_bob.n
)
SELECT *
FROM works_for_bob
```



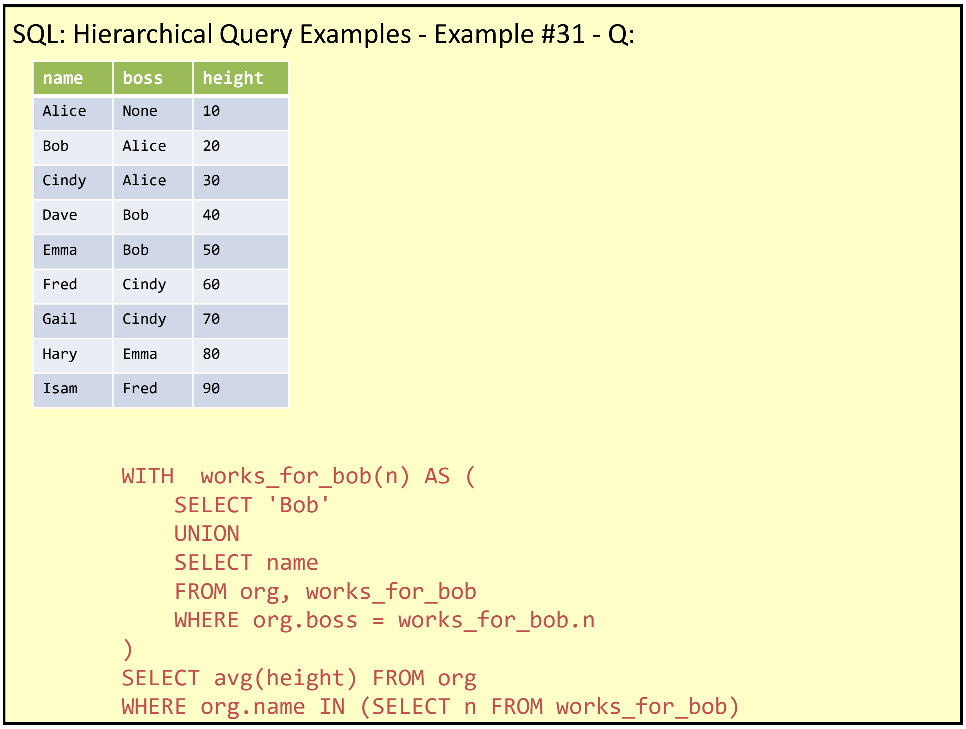
SQL: Hierarchical Query Examples - Example #30 - A:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  works_for_bob(n) AS (
    SELECT 'Bob'
    UNION
    SELECT name
    FROM org, works_for_bob
    WHERE org.boss = works_for_bob.n
)
SELECT *
FROM works_for_bob
```

n
Bob
Dave
Emma
Hary

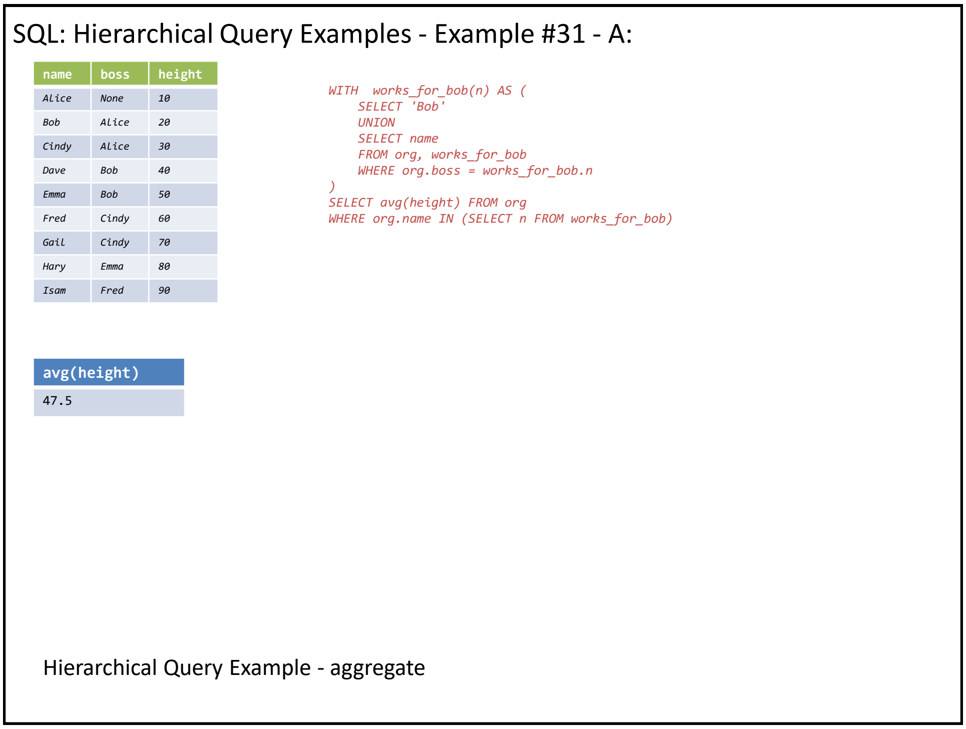
Hierarchical Query Example



SQL: Hierarchical Query Examples - Example #31 - Q:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  works_for_bob(n) AS (
    SELECT 'Bob'
    UNION
    SELECT name
    FROM org, works_for_bob
    WHERE org.boss = works_for_bob.n
)
SELECT avg(height) FROM org
WHERE org.name IN (SELECT n FROM works_for_bob)
```



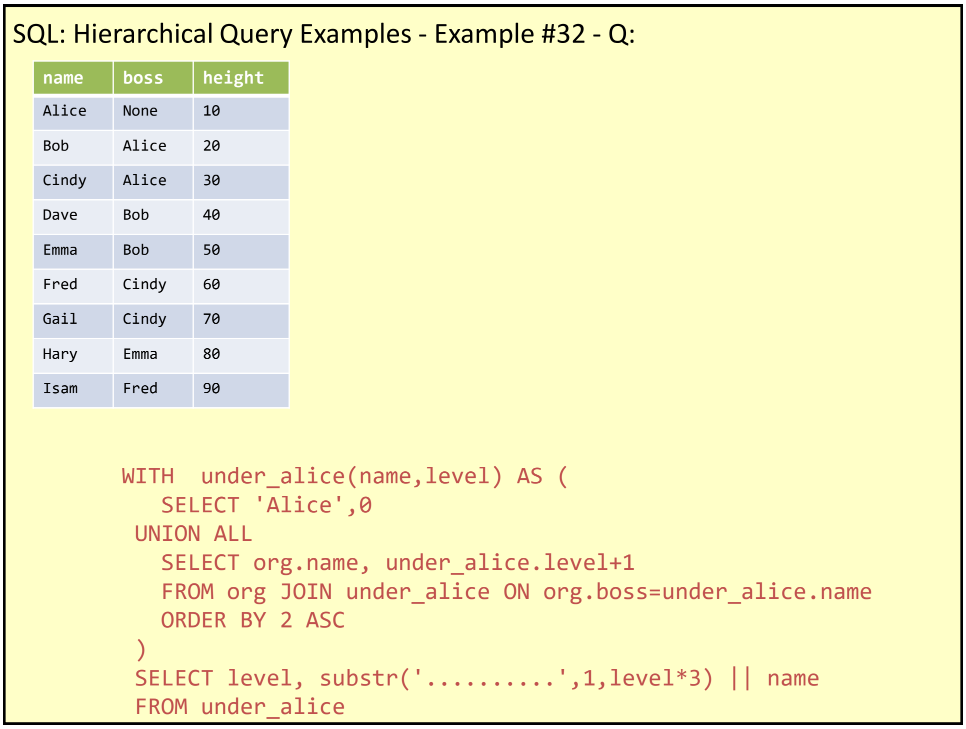
SQL: Hierarchical Query Examples - Example #31 - A:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  works_for_bob(n) AS (
    SELECT 'Bob'
    UNION
    SELECT name
    FROM org, works_for_bob
    WHERE org.boss = works_for_bob.n
)
SELECT avg(height) FROM org
WHERE org.name IN (SELECT n FROM works_for_bob)
```

avg(height)
47.5

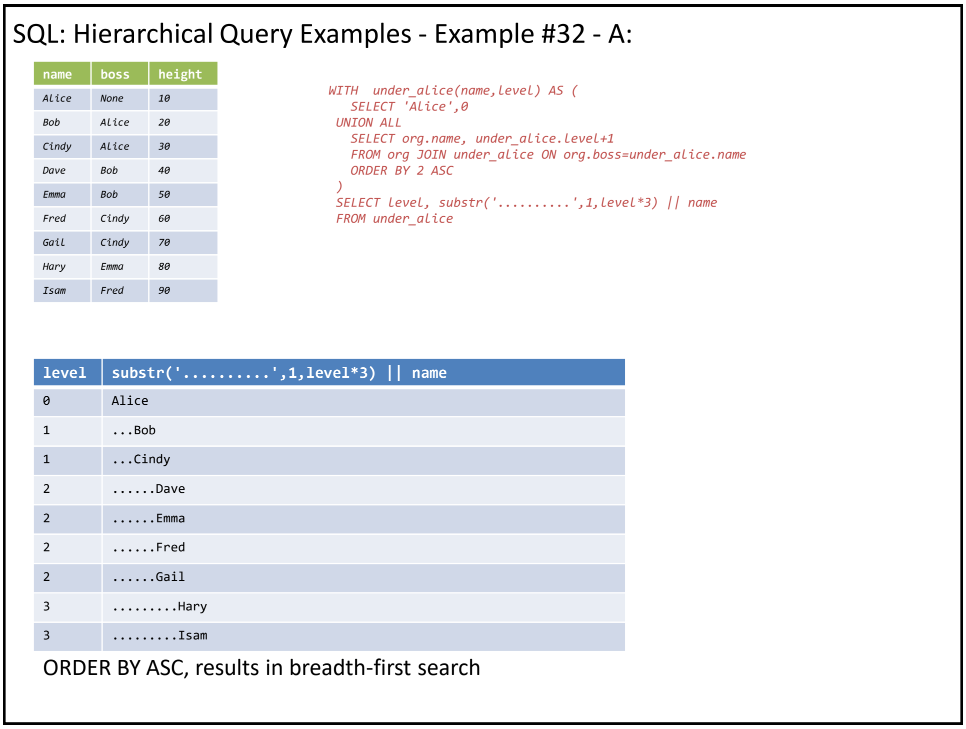
Hierarchical Query Example - aggregate



SQL: Hierarchical Query Examples - Example #32 - Q:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  under_alice(name,level) AS (
   SELECT 'Alice',0
 UNION ALL
   SELECT org.name, under_alice.level+1
   FROM org JOIN under_alice ON org.boss=under_alice.name
   ORDER BY 2 ASC
 )
 SELECT level, substr('..........',1,level*3) || name
 FROM under_alice
```



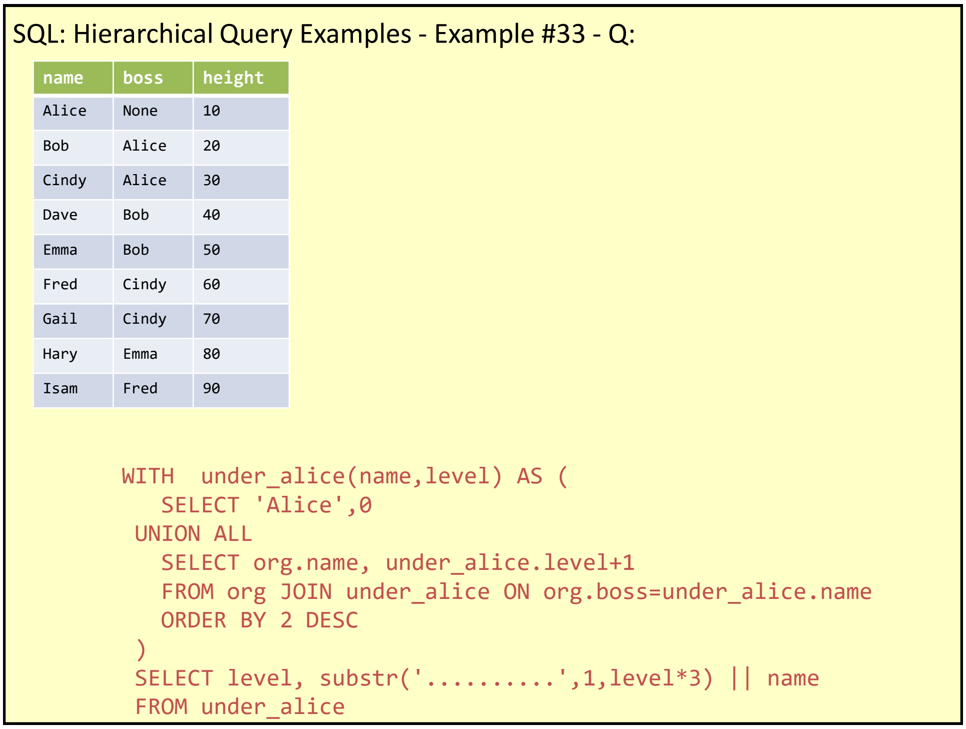
SQL: Hierarchical Query Examples - Example #32 - A:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  under_alice(name,level) AS (
   SELECT 'Alice',0
 UNION ALL
   SELECT org.name, under_alice.level+1
   FROM org JOIN under_alice ON org.boss=under_alice.name
   ORDER BY 2 ASC
 )
 SELECT level, substr('..........',1,level*3) || name
 FROM under_alice
```

level	substr('..........',1,level*3) || name
0	Alice
1	...Bob
1	...Cindy
2	......Dave
2	......Emma
2	......Fred
2	......Gail
3	.........Hary
3	.........Isam

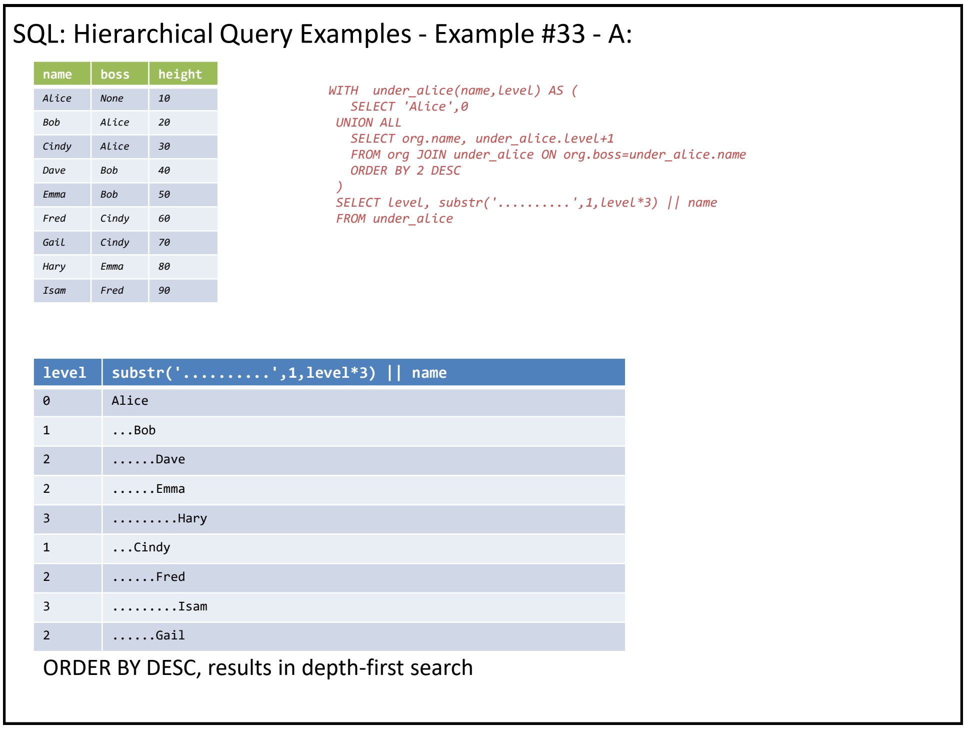
ORDER BY ASC, results in breadth-first search



SQL: Hierarchical Query Examples - Example #33 - Q:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  under_alice(name,level) AS (
   SELECT 'Alice',0
 UNION ALL
   SELECT org.name, under_alice.level+1
   FROM org JOIN under_alice ON org.boss=under_alice.name
   ORDER BY 2 DESC
 )
 SELECT level, substr('..........',1,level*3) || name
 FROM under_alice
```



SQL: Hierarchical Query Examples - Example #33 - A:

name	boss	height
Alice	None	10
Bob	Alice	20
Cindy	Alice	30
Dave	Bob	40
Emma	Bob	50
Fred	Cindy	60
Gail	Cindy	70
Hary	Emma	80
Isam	Fred	90

```
WITH  under_alice(name,level) AS (
   SELECT 'Alice',0
 UNION ALL
   SELECT org.name, under_alice.level+1
   FROM org JOIN under_alice ON org.boss=under_alice.name
   ORDER BY 2 DESC
 )
 SELECT level, substr('..........',1,level*3) || name
 FROM under_alice
```

level	substr('..........',1,level*3) || name
0	Alice
1	...Bob
2	......Dave
2	......Emma
3	.........Hary
1	...Cindy
2	......Fred
3	.........Isam
2	......Gail

ORDER BY DESC, results in depth-first search

Learn SQL by Example (Chananel Perel) 2024-05-01 15:55:35.238206

JSON

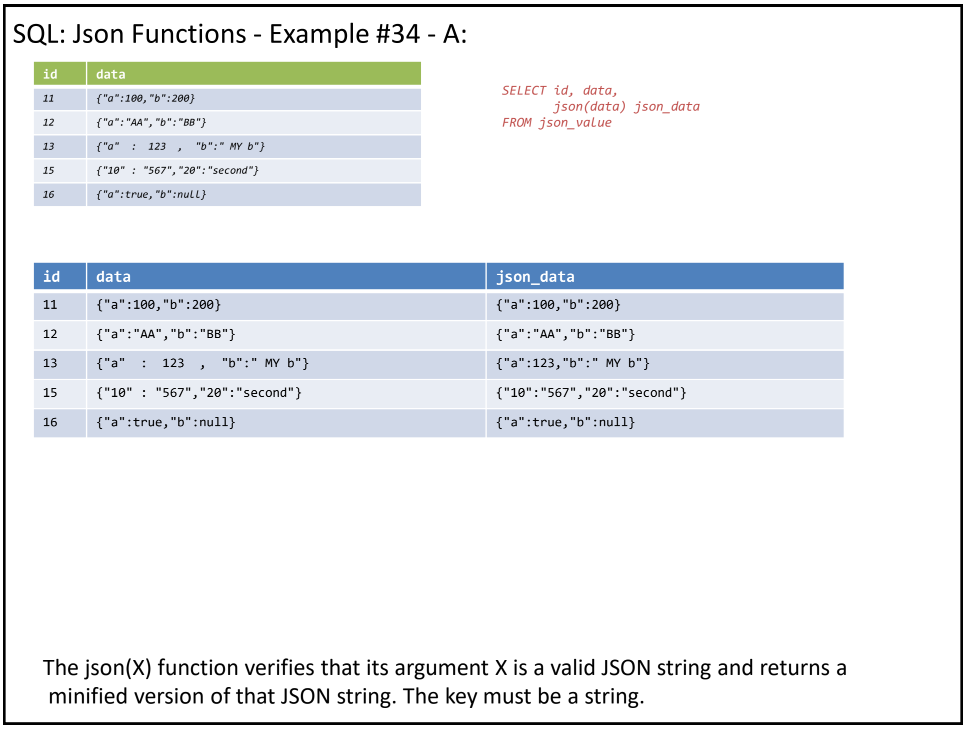
Learn SQL by Example (Chananel Perel 2023)

JSON FUNCTIONS

SQL: Json Functions - Example #34 - Q:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}

```
SELECT id, data,
       json(data) json_data
FROM json_value
```



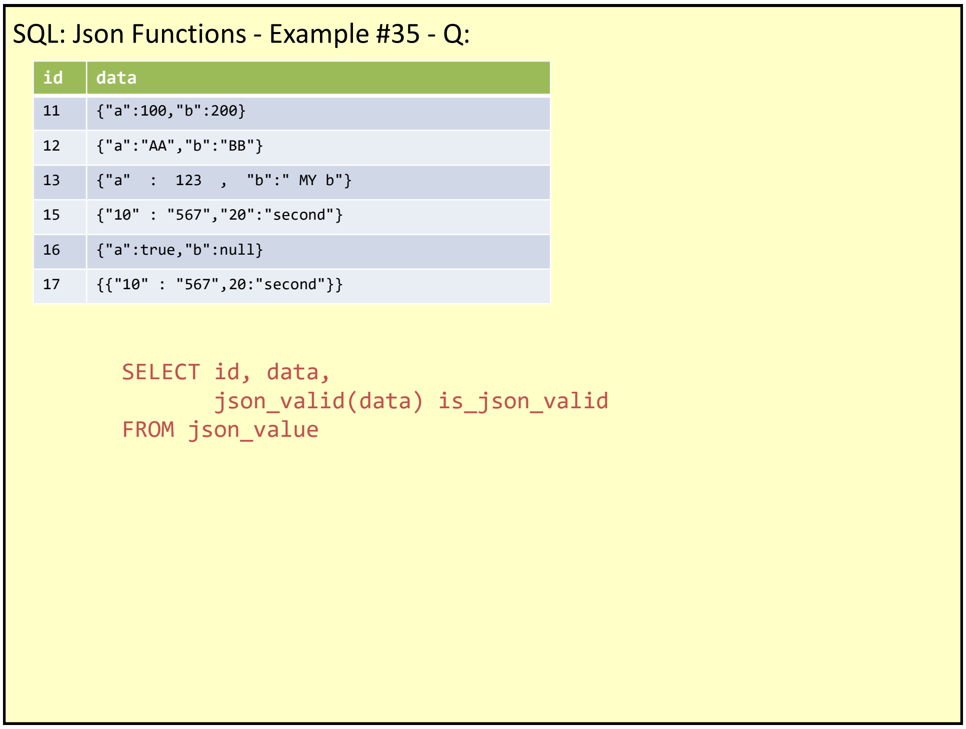
SQL: Json Functions - Example #34 - A:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}

```
SELECT id, data,
       json(data) json_data
FROM json_value
```

id	data	json_data
11	{"a":100,"b":200}	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}	{"a":123,"b":" MY b"}
15	{"10" : "567","20":"second"}	{"10":"567","20":"second"}
16	{"a":true,"b":null}	{"a":true,"b":null}

The json(X) function verifies that its argument X is a valid JSON string and returns a minified version of that JSON string. The key must be a string.



SQL: Json Functions - Example #35 - Q:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}
17	{{"10" : "567",20:"second"}}

```
SELECT id, data,
       json_valid(data) is_json_valid
FROM json_value
```

SQL: Json Functions - Example #35 - A:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}
17	{{"10" : "567",20:"second"}}

```
SELECT id, data,
       json_valid(data) is_json_valid
FROM json_value
```

id	data	is_json_valid
11	{"a":100,"b":200}	1
12	{"a":"AA","b":"BB"}	1
13	{"a" : 123 , "b":" MY b"}	1
15	{"10" : "567","20":"second"}	1
16	{"a":true,"b":null}	1
17	{{"10" : "567",20:"second"}}	0

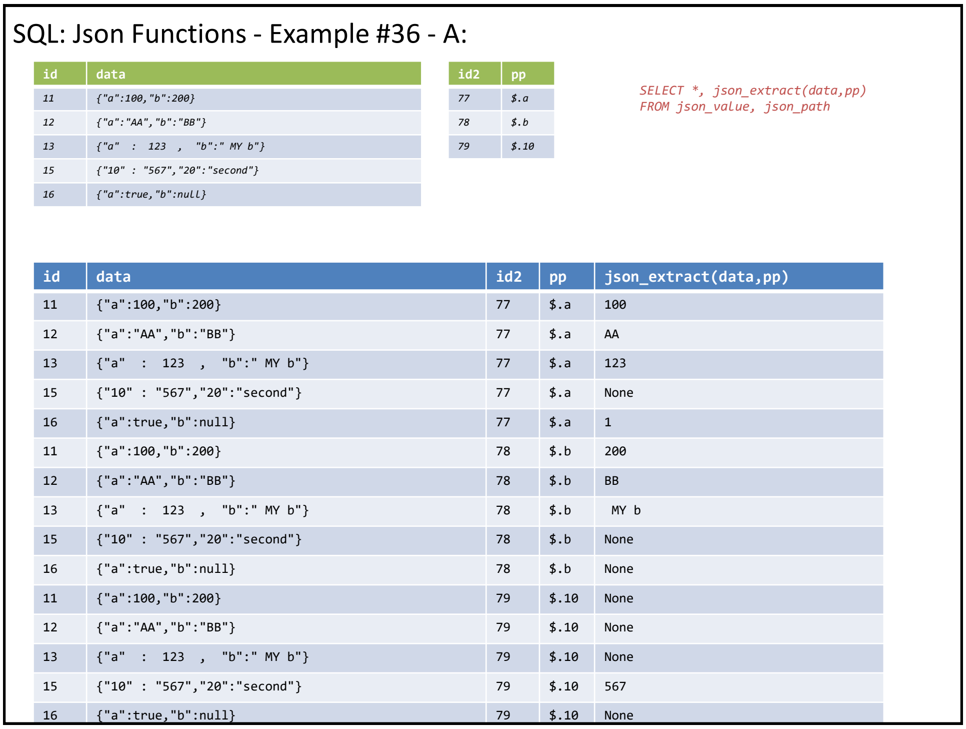
json_valid

SQL: Json Functions - Example #36 - Q:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}

id2	pp
77	$.a
78	$.b
79	$.10

```
SELECT *, json_extract(data,pp)
FROM json_value, json_path
```



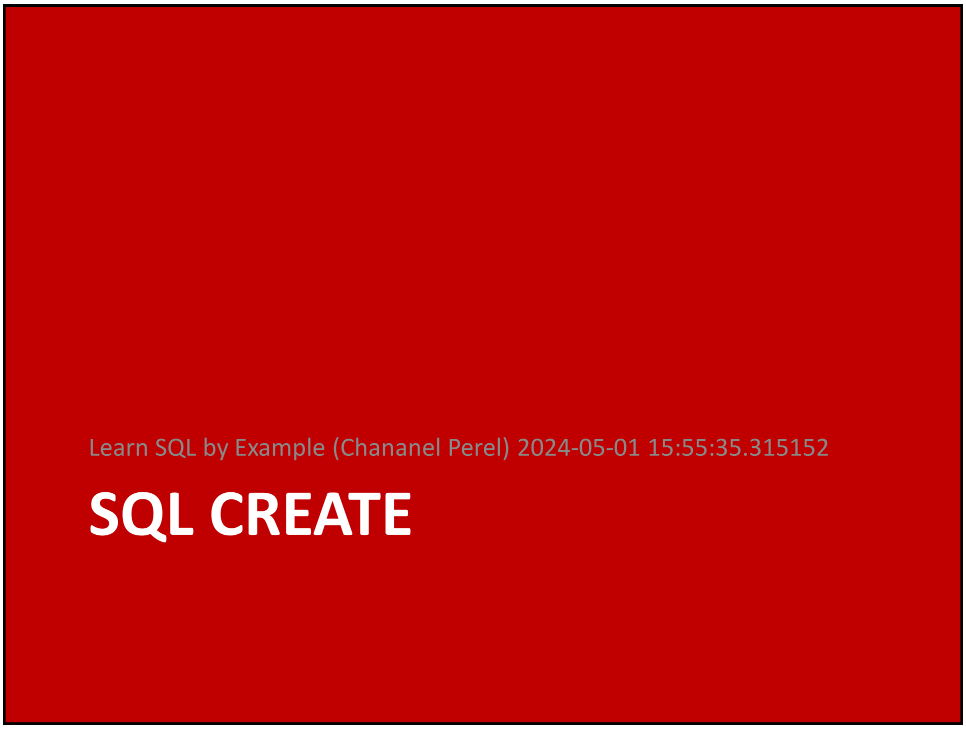
SQL: Json Functions - Example #36 - A:

id	data
11	{"a":100,"b":200}
12	{"a":"AA","b":"BB"}
13	{"a" : 123 , "b":" MY b"}
15	{"10" : "567","20":"second"}
16	{"a":true,"b":null}

id2	pp
77	$.a
78	$.b
79	$.10

```
SELECT *, json_extract(data,pp)
FROM json_value, json_path
```

id	data	id2	pp	json_extract(data,pp)
11	{"a":100,"b":200}	77	$.a	100
12	{"a":"AA","b":"BB"}	77	$.a	AA
13	{"a" : 123 , "b":" MY b"}	77	$.a	123
15	{"10" : "567","20":"second"}	77	$.a	None
16	{"a":true,"b":null}	77	$.a	1
11	{"a":100,"b":200}	78	$.b	200
12	{"a":"AA","b":"BB"}	78	$.b	BB
13	{"a" : 123 , "b":" MY b"}	78	$.b	MY b
15	{"10" : "567","20":"second"}	78	$.b	None
16	{"a":true,"b":null}	78	$.b	None
11	{"a":100,"b":200}	79	$.10	None
12	{"a":"AA","b":"BB"}	79	$.10	None
13	{"a" : 123 , "b":" MY b"}	79	$.10	None
15	{"10" : "567","20":"second"}	79	$.10	567
16	{"a":true,"b":null}	79	$.10	None



Learn SQL by Example (Chananel Perel) 2024-05-01 15:55:35.315152

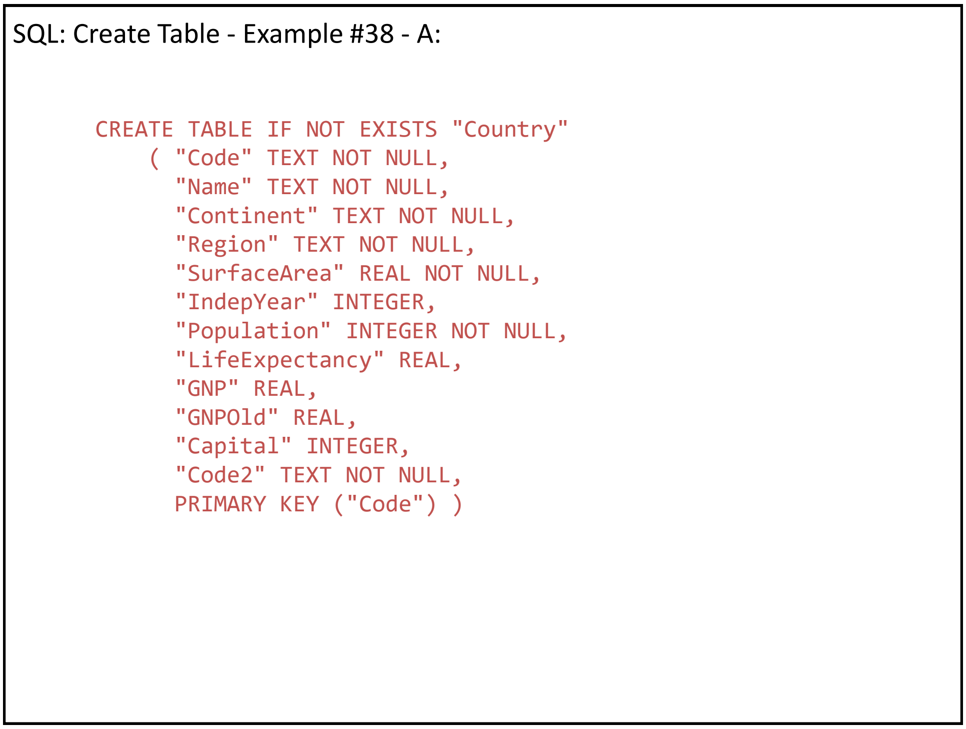
SQL CREATE

Learn SQL by Example (Chananel Perel 2023)

CREATE TABLE

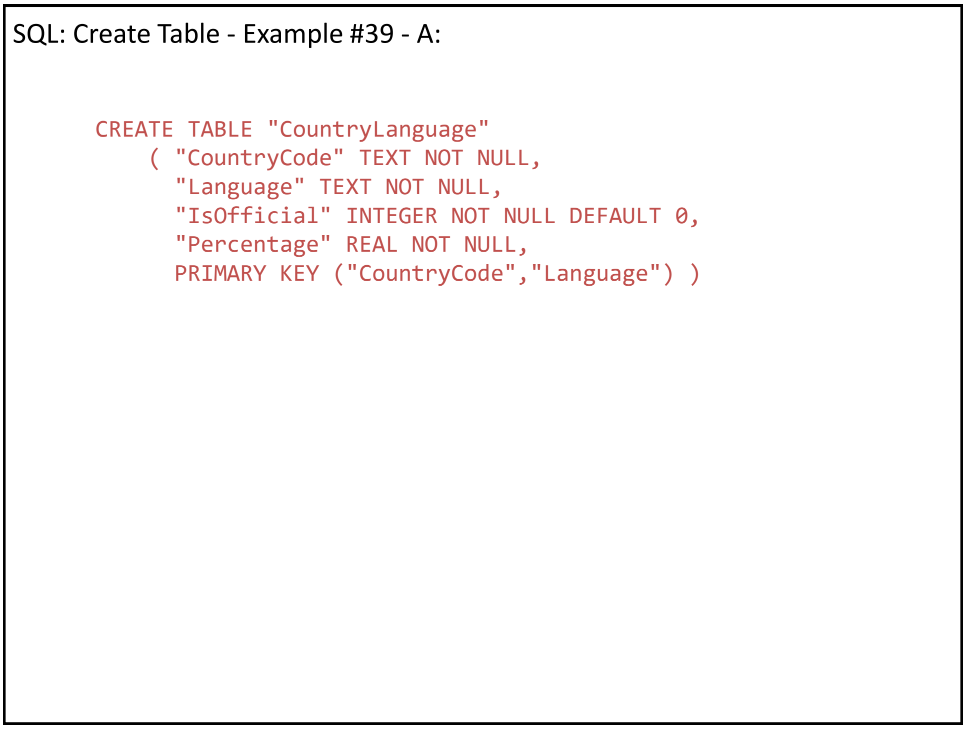
SQL: Create Table - Example #37 - A:

```
CREATE TABLE "City"
    ( "Id" INTEGER PRIMARY KEY NOT NULL, -- auto rowid
      "Name" TEXT NOT NULL DEFAULT '',
      "CountryCode" TEXT NOT NULL DEFAULT '',
      "District" TEXT DEFAULT '',
      "Population" INTEGER NOT NULL DEFAULT 0 )
```



SQL: Create Table - Example #38 - A:

```
CREATE TABLE IF NOT EXISTS "Country"
    ( "Code" TEXT NOT NULL,
      "Name" TEXT NOT NULL,
      "Continent" TEXT NOT NULL,
      "Region" TEXT NOT NULL,
      "SurfaceArea" REAL NOT NULL,
      "IndepYear" INTEGER,
      "Population" INTEGER NOT NULL,
      "LifeExpectancy" REAL,
      "GNP" REAL,
      "GNPOld" REAL,
      "Capital" INTEGER,
      "Code2" TEXT NOT NULL,
      PRIMARY KEY ("Code") )
```



SQL: Create Table - Example #39 - A:

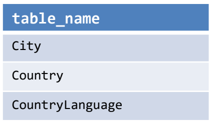
```
CREATE TABLE "CountryLanguage"
    ( "CountryCode" TEXT NOT NULL,
      "Language" TEXT NOT NULL,
      "IsOfficial" INTEGER NOT NULL DEFAULT 0,
      "Percentage" REAL NOT NULL,
      PRIMARY KEY ("CountryCode","Language") )
```

Learn SQL by Example (Chananel Perel 2023)

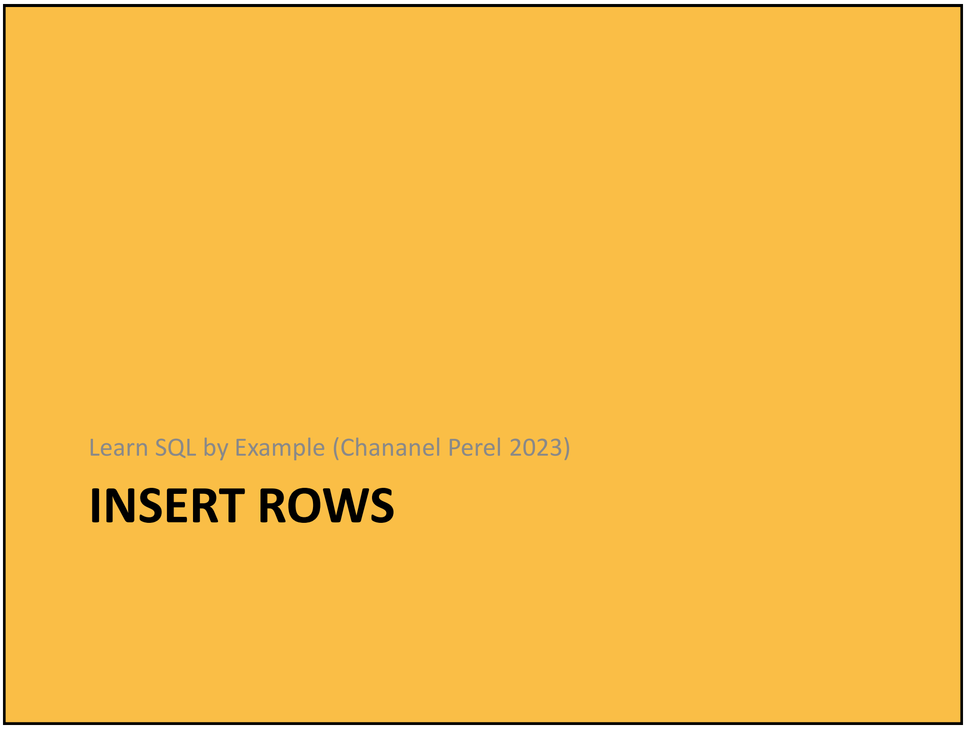
SHOW TABLES

SQL: Show Tables - Example #40 - A:

```
SELECT name table_name
  FROM sqlite_schema
  WHERE type ='table'
    AND name NOT LIKE 'sqlite_%'
```

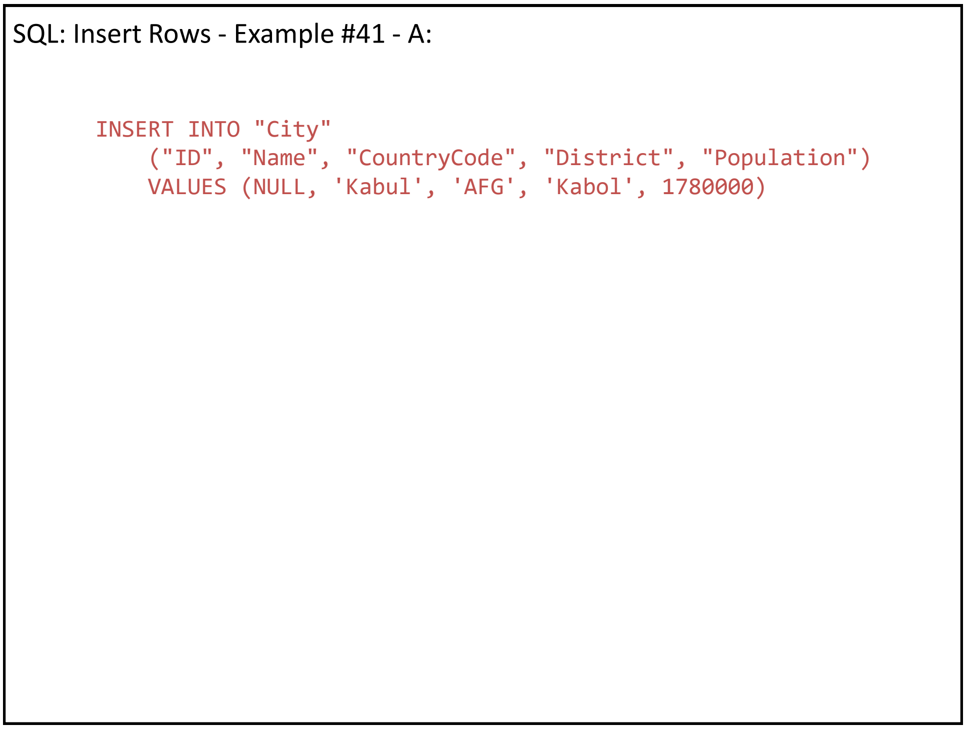


table_name
City
Country
CountryLanguage



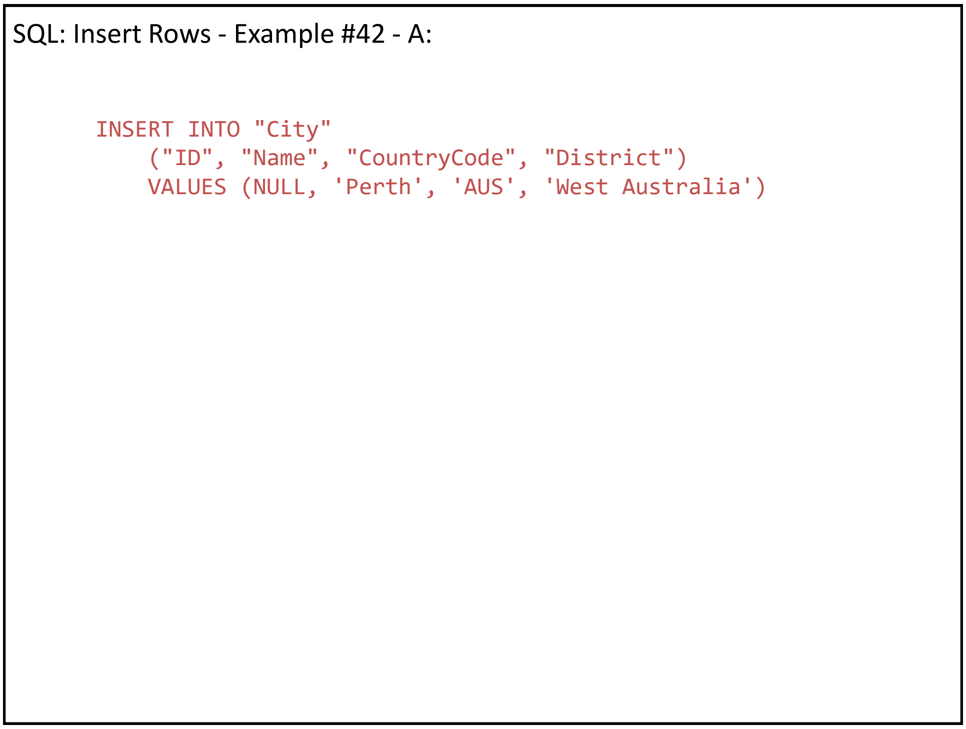
Learn SQL by Example (Chananel Perel 2023)

INSERT ROWS



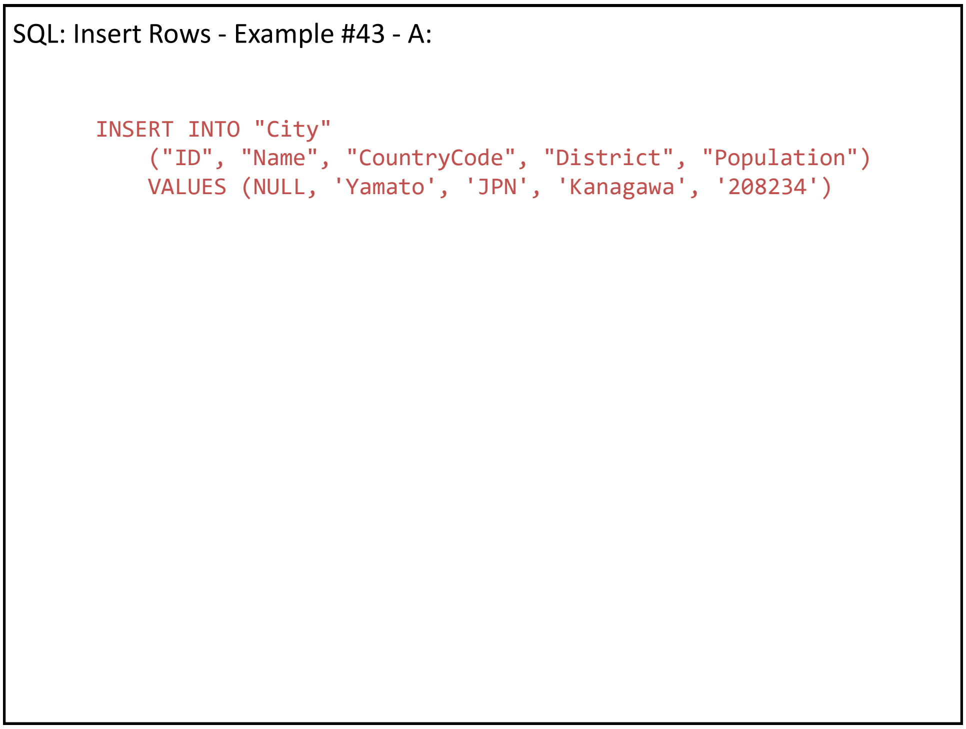
SQL: Insert Rows - Example #41 - A:

```
INSERT INTO "City"
    ("ID", "Name", "CountryCode", "District", "Population")
    VALUES (NULL, 'Kabul', 'AFG', 'Kabol', 1780000)
```



SQL: Insert Rows - Example #42 - A:

```
INSERT INTO "City"
    ("ID", "Name", "CountryCode", "District")
    VALUES (NULL, 'Perth', 'AUS', 'West Australia')
```



SQL: Insert Rows - Example #43 - A:

```
INSERT INTO "City"
    ("ID", "Name", "CountryCode", "District", "Population")
    VALUES (NULL, 'Yamato', 'JPN', 'Kanagawa', '208234')
```

SQL: Insert Rows - Example #44 - A:

```
SELECT * FROM City
```

Id	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Perth	AUS	West Australia	0
3	Yamato	JPN	Kanagawa	208234

Learn SQL by Example (Chananel Perel 2023)

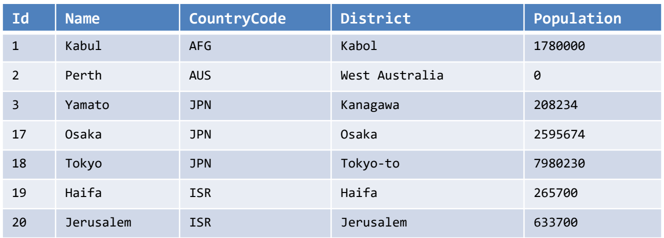
INSERT MANY ROWS

SQL: Insert Many Rows - Example #45 - A:

```
INSERT INTO "City"
    ("ID", "Name", "CountryCode", "District", "Population")
    VALUES (17, 'Osaka', 'JPN', 'Osaka', '2595674'),
           (NULL, 'Tokyo', 'JPN', 'Tokyo-to', '7980230'),
           (NULL, 'Haifa', 'ISR', 'Haifa', '265700'),
           (NULL, 'Jerusalem', 'ISR', 'Jerusalem', '633700')
```

SQL: Insert Many Rows - Example #46 - A:

```
SELECT * FROM City
```



Id	Name	CountryCode	District	Population
1	Kabul	AFG	Kabol	1780000
2	Perth	AUS	West Australia	0
3	Yamato	JPN	Kanagawa	208234
17	Osaka	JPN	Osaka	2595674
18	Tokyo	JPN	Tokyo-to	7980230
19	Haifa	ISR	Haifa	265700
20	Jerusalem	ISR	Jerusalem	633700

Learn SQL by Example (Chananel Perel 2023)

CTAS = CREATE TABLE AS

SQL: CTAS = Create Table As - Example #47 - A:

```
    CREATE TABLE city_millions AS
        SELECT Name city_name, Population/1000000.0
population_million
        FROM City
```

SQL: CTAS = Create Table As - Example #48 - A:

```
SELECT name table_name
  FROM sqlite_schema
  WHERE type ='table'
    AND name NOT LIKE 'sqlite_%'
```

table_name
City
Country
CountryLanguage
city_millions

SQL: CTAS = Create Table As - Example #49 - A:

```
SELECT * FROM city_millions
```

city_name	population_million
Kabul	1.78
Perth	0.0
Yamato	0.208234
Osaka	2.595674
Tokyo	7.98023
Haifa	0.2657
Jerusalem	0.6337

Learn SQL by Example (Chananel Perel 2023)

UPDATE ROWS

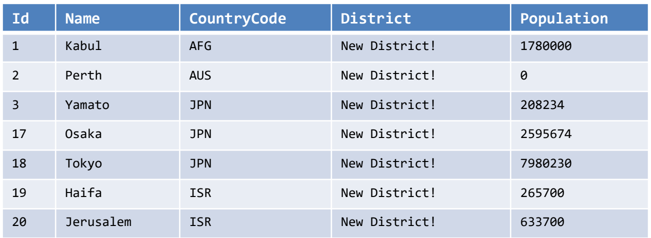
SQL: Update Rows - Example #50 - A:

```
UPDATE "City"
SET District = 'New District!'
```

This updates all rows, probably we do not want this..

SQL: Update Rows - Example #51 - A:

```sql
SELECT * FROM City
```



Id	Name	CountryCode	District	Population
1	Kabul	AFG	New District!	1780000
2	Perth	AUS	New District!	0
3	Yamato	JPN	New District!	208234
17	Osaka	JPN	New District!	2595674
18	Tokyo	JPN	New District!	7980230
19	Haifa	ISR	New District!	265700
20	Jerusalem	ISR	New District!	633700

Learn SQL by Example (Chananel Perel 2023)

UPDATE SOME ROWS

SQL: Update Some Rows - Example #52 - A:

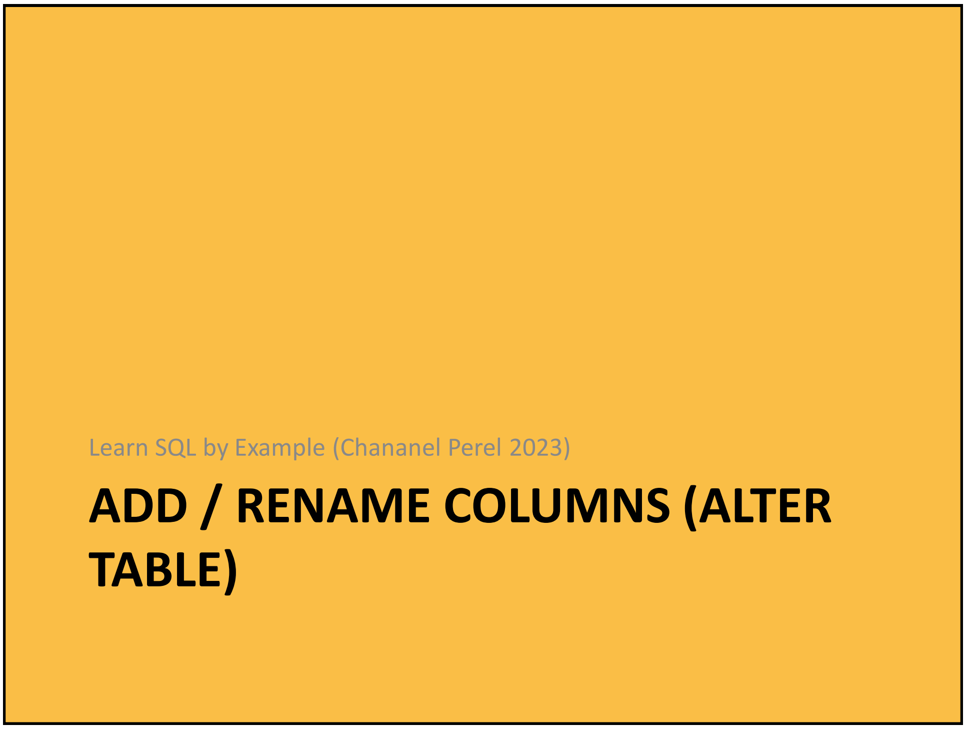
```
UPDATE "City"
SET  CountryCode = 'JPN_2',
     Population = Population * 4
WHERE CountryCode = 'JPN'
```

This is just for rows that filter is true (and we also use the old val to calc new one..

SQL: Update Some Rows - Example #53 - A:

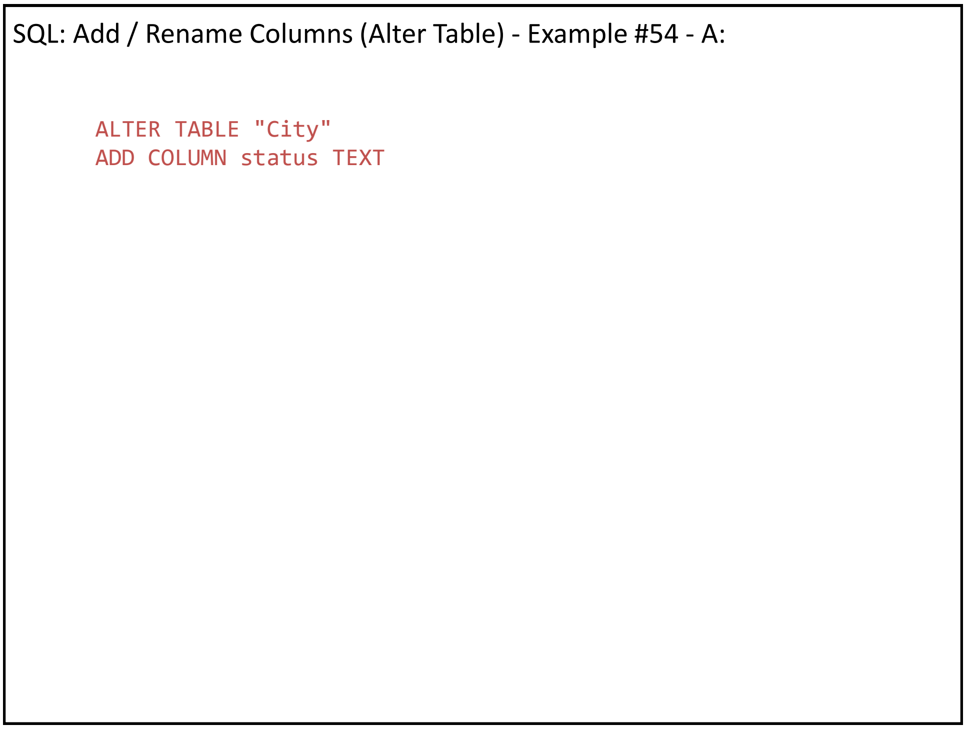
```
SELECT * FROM City
```

Id	Name	CountryCode	District	Population
1	Kabul	AFG	New District!	1780000
2	Perth	AUS	New District!	0
3	Yamato	JPN_2	New District!	832936
17	Osaka	JPN_2	New District!	10382696
18	Tokyo	JPN_2	New District!	31920920
19	Haifa	ISR	New District!	265700
20	Jerusalem	ISR	New District!	633700



Learn SQL by Example (Chananel Perel 2023)

ADD / RENAME COLUMNS (ALTER TABLE)



SQL: Add / Rename Columns (Alter Table) - Example #54 - A:

```
ALTER TABLE "City"
ADD COLUMN status TEXT
```

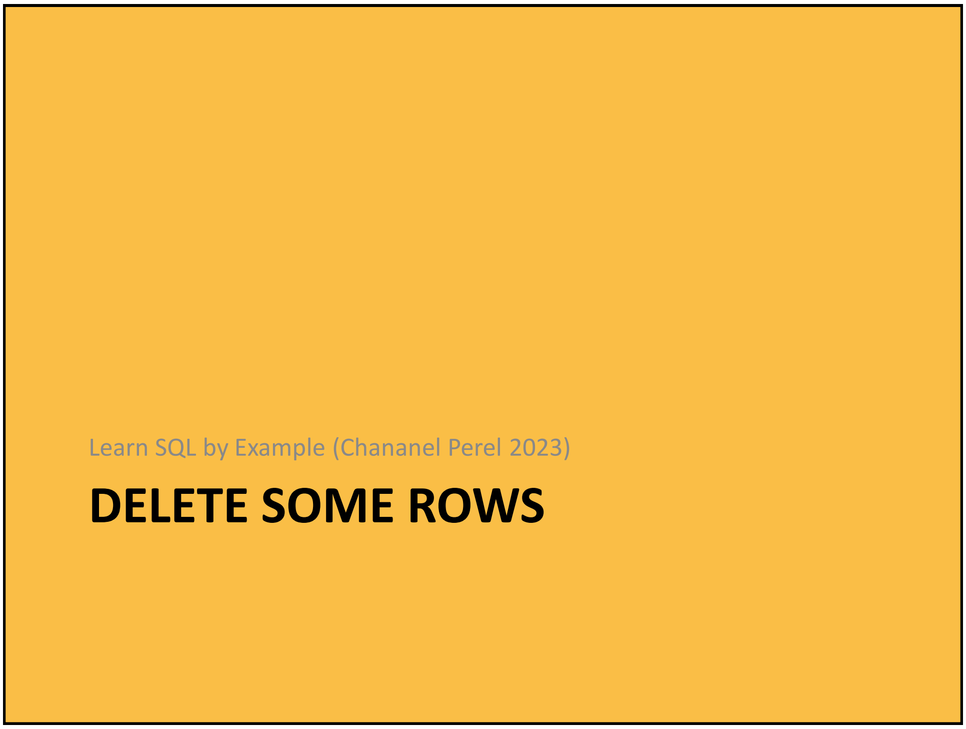
SQL: Add / Rename Columns (Alter Table) - Example #55 - A:

```
ALTER TABLE "City"
RENAME COLUMN District TO District_New_Name
```

SQL: Add / Rename Columns (Alter Table) - Example #56 - A:

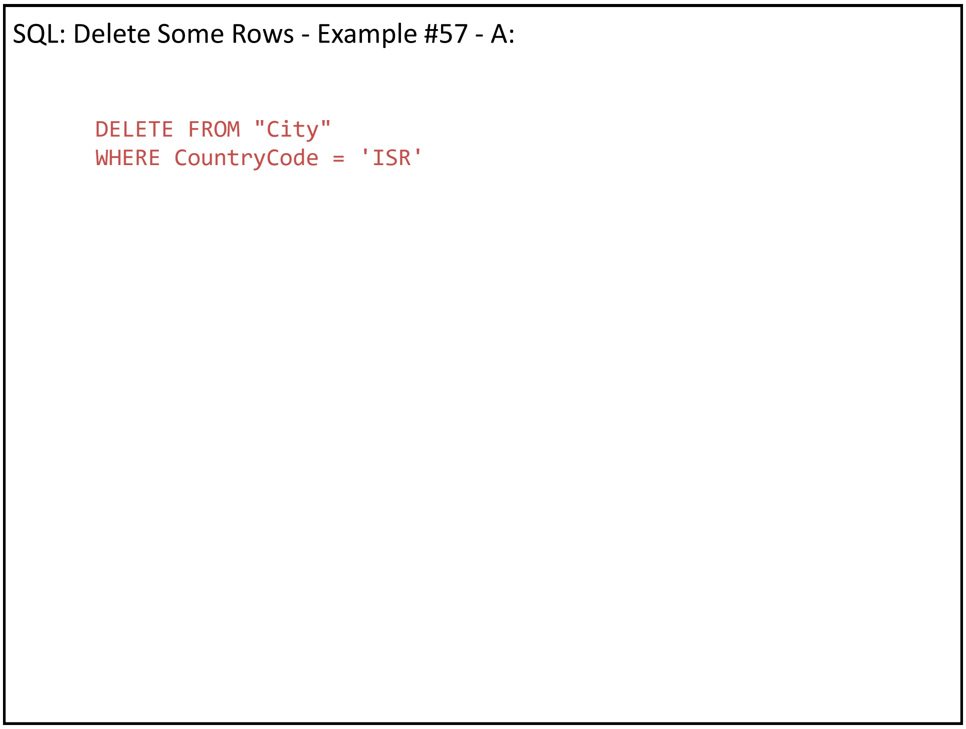
```
SELECT * FROM City
```

Id	Name	CountryCode	District_New_Name	Population	status
1	Kabul	AFG	New District!	1780000	None
2	Perth	AUS	New District!	0	None
3	Yamato	JPN_2	New District!	832936	None
17	Osaka	JPN_2	New District!	10382696	None
18	Tokyo	JPN_2	New District!	31920920	None
19	Haifa	ISR	New District!	265700	None
20	Jerusalem	ISR	New District!	633700	None



Learn SQL by Example (Chananel Perel 2023)

DELETE SOME ROWS



SQL: Delete Some Rows - Example #57 - A:

```
DELETE FROM "City"
WHERE CountryCode = 'ISR'
```

SQL: Delete Some Rows - Example #58 - A:

```
SELECT * FROM City
```

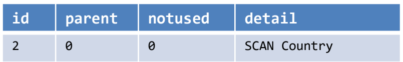
Id	Name	CountryCode	District_New_Name	Population	status
1	Kabul	AFG	New District!	1780000	None
2	Perth	AUS	New District!	0	None
3	Yamato	JPN_2	New District!	832936	None
17	Osaka	JPN_2	New District!	10382696	None
18	Tokyo	JPN_2	New District!	31920920	None

Learn SQL by Example (Chananel Perel 2023)

INDEX

SQL: INDEX - Example #59 - A:

```
EXPLAIN QUERY PLAN
SELECT * FROM Country WHERE Code2='IL'
```



id	parent	notused	detail
2	0	0	SCAN Country

SQL: INDEX - Example #60 - A:

```
CREATE INDEX new_contry_index
ON Country(Code2);
```

SQL: INDEX - Example #61 - A:

```
SELECT *
    FROM sqlite_schema
    WHERE type='index'
    ORDER BY tbl_name,type;
```

type	name	tbl_name	rootpage	sql
index	sqlite_autoindex_Country_1	Country	4	None
index	new_contry_index	Country	8	CREATE IN ON Co
index	sqlite_autoindex_CountryLanguage_1	CountryLanguage	6	None

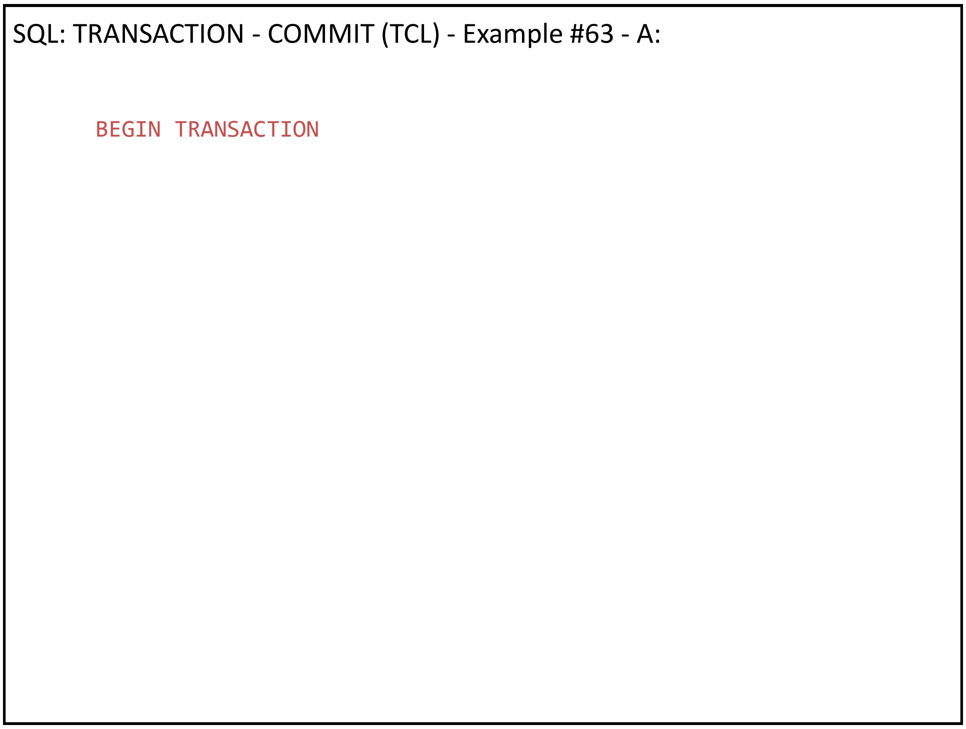
SQL: INDEX - Example #62 - A:

```
EXPLAIN QUERY PLAN
SELECT * FROM Country WHERE Code2='IL'
```

id	parent	notused	detail
3	0	0	SEARCH Country USING INDEX new_contry_index (Code2=?)

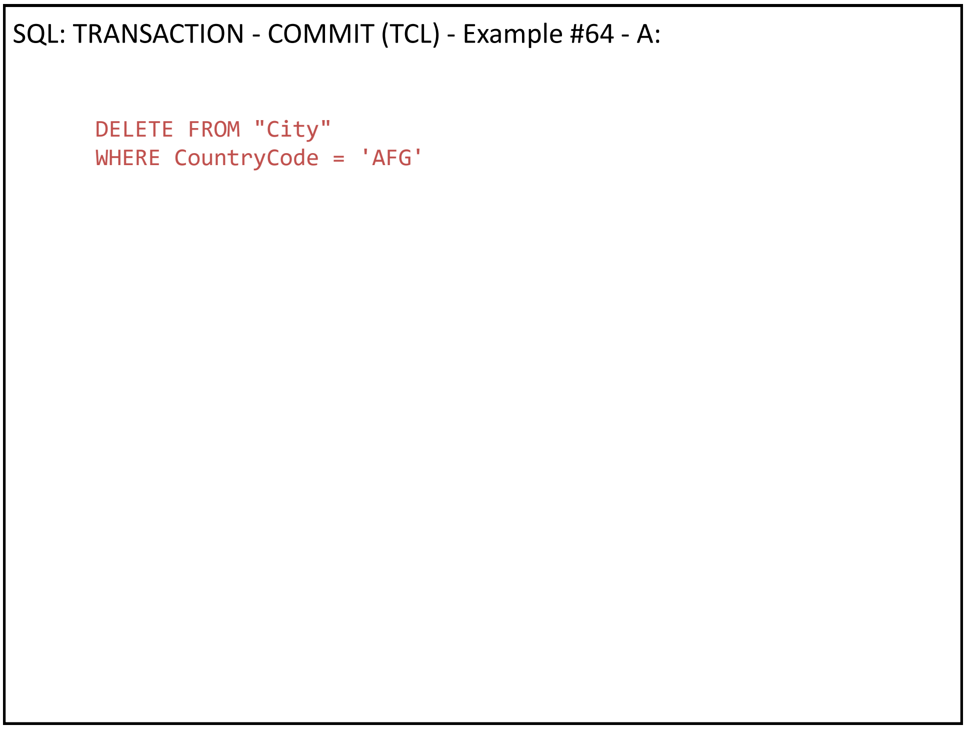
Learn SQL by Example (Chananel Perel) 2024-05-01 15:55:35.473652

TRANSACTION - COMMIT (TCL)



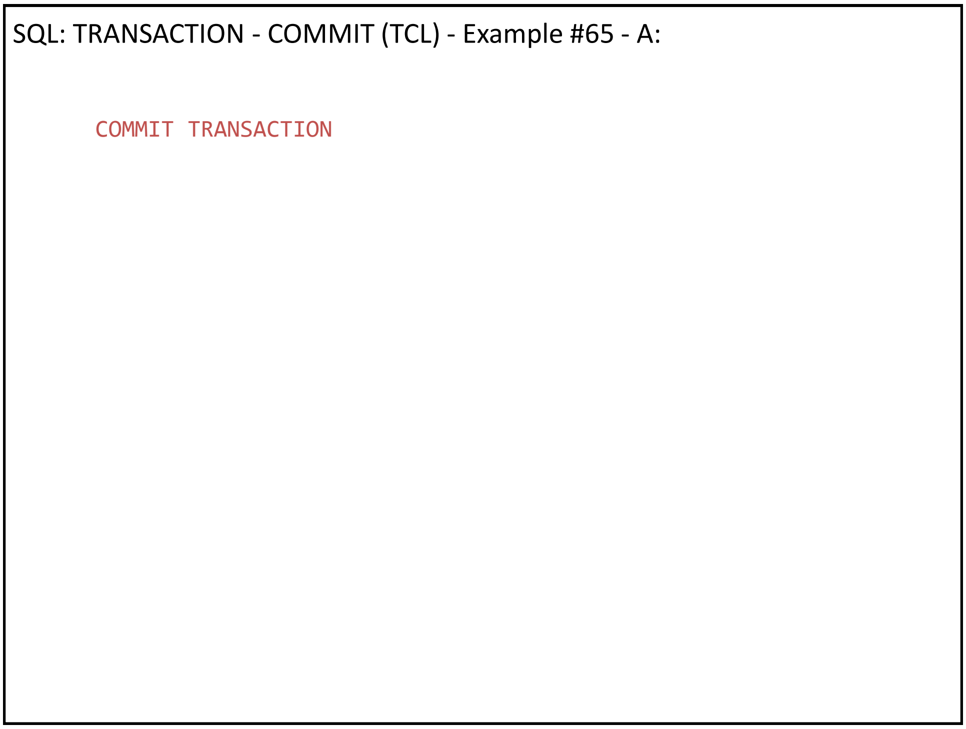
SQL: TRANSACTION - COMMIT (TCL) - Example #63 - A:

```
BEGIN TRANSACTION
```



SQL: TRANSACTION - COMMIT (TCL) - Example #64 - A:

```
DELETE FROM "City"
WHERE CountryCode = 'AFG'
```

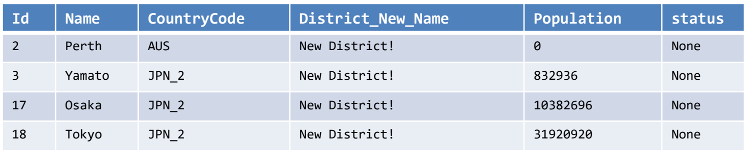


SQL: TRANSACTION - COMMIT (TCL) - Example #65 - A:

```
COMMIT TRANSACTION
```

SQL: TRANSACTION - COMMIT (TCL) - Example #66 - A:

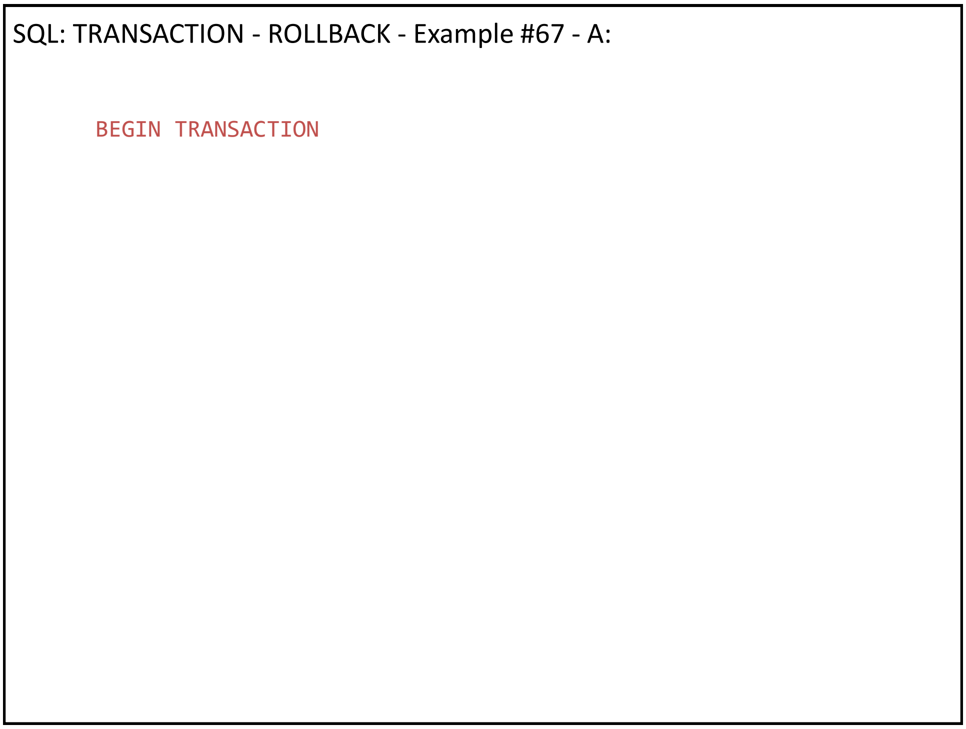
```
SELECT * FROM City
```



Id	Name	CountryCode	District_New_Name	Population	status
2	Perth	AUS	New District!	0	None
3	Yamato	JPN_2	New District!	832936	None
17	Osaka	JPN_2	New District!	10382696	None
18	Tokyo	JPN_2	New District!	31920920	None

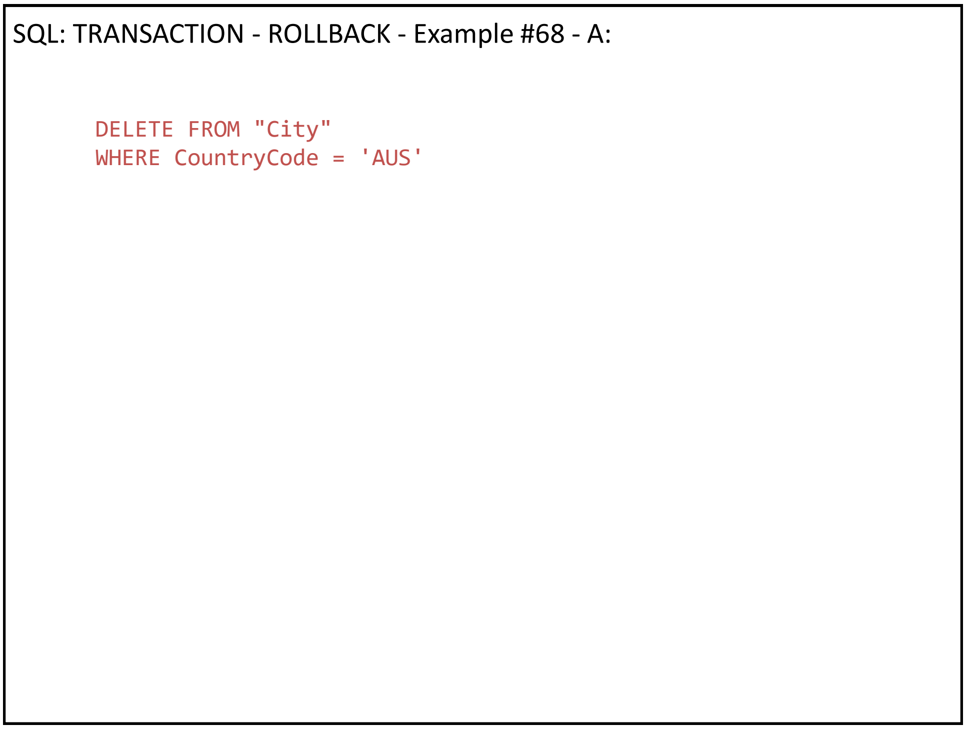
Learn SQL by Example (Chananel Perel 2023)

TRANSACTION - ROLLBACK



SQL: TRANSACTION - ROLLBACK - Example #67 - A:

```
BEGIN TRANSACTION
```



SQL: TRANSACTION - ROLLBACK - Example #68 - A:

```
DELETE FROM "City"
WHERE CountryCode = 'AUS'
```

SQL: TRANSACTION - ROLLBACK - Example #69 - A:

```
     ROLLBACK TRANSACTION
```

SQL: TRANSACTION - ROLLBACK - Example #70 - A:

```
SELECT * FROM City
```

Id	Name	CountryCode	District_New_Name	Population	status
2	Perth	AUS	New District!	0	None
3	Yamato	JPN_2	New District!	832936	None
17	Osaka	JPN_2	New District!	10382696	None
18	Tokyo	JPN_2	New District!	31920920	None