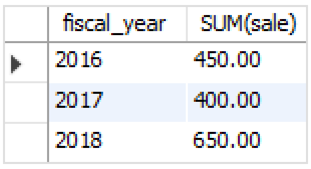
Window Functions:

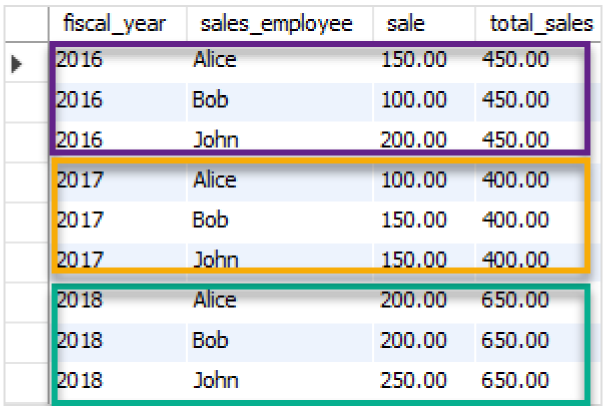
MySQL window functions perform calculations across a set of related table rows (a "window") without collapsing them into a single result row using GROUP BY.

Normal SUM():



fiscal_year	SUM(sale)
2016	450.00
2017	400.00
2018	650.00

Window Function SUM():



fiscal_year	sales_employee	sale	total_sales
2016	Alice	150.00	450.00
2016	Bob	100.00	450.00
2016	John	200.00	450.00
2017	Alice	100.00	400.00
2017	Bob	150.00	400.00
2017	John	150.00	400.00
2018	Alice	200.00	650.00
2018	Bob	200.00	650.00
2018	John	250.00	650.00

Window functions are categorized into two main types:

Aggregate Functions: Standard aggregate functions (SUM, AVG, COUNT, MIN, MAX) can be used as window functions.

Analytical (or Ranking and Value) Functions: Specialized functions for ranking, numbering, and accessing adjacent rows (ROW_NUMBER, RANK, DENSE_RANK, LEAD, LAG, etc

Syntax:

```
WINDOW_FUNCTION(expression) OVER (
   [PARTITION BY partition_expression, ...]
   [ORDER BY order_expression [ASC | DESC], ...]
)
```

RANK():

Ranking Within Partitions (RANK(), DENSE_RANK())

```
SELECT
    employee_name,
    department,
    salary,
    RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS department_rank_with_gaps,
    DENSE_RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS department_rank_no_gaps
FROM
    employee_salaries;
```

Accessing Previous/Next Rows (LAG(), LEAD():

LAG(): Previous Rows

Syntax:

```
LAG(expression,offset, default_value) OVER ( PARTITION BY partition_expression
ORDER BY order_expresion ASC|DESC )
```

expression

This is the column or expression from which you want to retrieve the next value.

offset

The offset specifies the number of rows to look ahead. If you skip it, it defaults to 1, which is the immediate row.

default_value

This is the default value if there is no next row. For example, the last row in the result set (or in a partition) will not have the next row.

If you don't specify the default_value, it'll default to NULL.

```
CREATE TABLE sales(
    sales_employee VARCHAR(50) NOT NULL,
    fiscal_year INT NOT NULL,
    sale DECIMAL(14,2) NOT NULL,
    PRIMARY KEY(sales_employee,fiscal_year)
);

-------------------------------------------------------------------------------------

INSERT INTO sales(sales_employee,fiscal_year,sale)

VALUES('Bob',2016,100),

      ('Bob',2017,150),

      ('Bob',2018,200),

      ('Alice',2016,150),

      ('Alice',2017,100),

      ('Alice',2018,200),

       ('John',2016,200),

      ('John',2017,150),

      ('John',2018,250);


SELECT * FROM sales;

-----------------------------------------------------------------------

SELECT
  sales_employee,
  fiscal_year,
  sale,
  LAG(sale, 1 , 0) OVER (
    PARTITION BY sales_employee
    ORDER BY fiscal_year
  ) 'previous year sale'
FROM
  sales;

+----------------+-------------+--------+--------------------+
| sales_employee | fiscal_year | sale   | previous year sale |
+----------------+-------------+--------+--------------------+
| Alice          |        2016 | 150.00 |               0.00 |
| Alice          |        2017 | 100.00 |             150.00 |
| Alice          |        2018 | 200.00 |             100.00 |
| Bob            |        2016 | 100.00 |               0.00 |
| Bob            |        2017 | 150.00 |             100.00 |
| Bob            |        2018 | 200.00 |             150.00 |
| John           |        2016 | 200.00 |               0.00 |
| John           |        2017 | 150.00 |             200.00 |
| John           |        2018 | 250.00 |             150.00 |
+----------------+-------------+--------+--------------------+
9 rows in set (0.00 sec)
```

```
SELECT
  sales_employee,
  fiscal_year,
  sale,
  LAG(sale, 1, 0) OVER (
    PARTITION BY sales_employee
    ORDER BY fiscal_year
  ) AS previous_year_sale,
  sale - LAG(sale, 1, 0) OVER (
    PARTITION BY sales_employee
    ORDER BY fiscal_year
  ) AS vs_previous_year
FROM
  sales;
```

```
+----------------+-------------+--------+--------------------+------------------+
| sales_employee | fiscal_year | sale   | previous_year_sale | vs_previous_year |
+----------------+-------------+--------+--------------------+------------------+
| Alice          |        2016 | 150.00 |               0.00 |           150.00 |
| Alice          |        2017 | 100.00 |             150.00 |           -50.00 |
| Alice          |        2018 | 200.00 |             100.00 |           100.00 |
| Bob            |        2016 | 100.00 |               0.00 |           100.00 |
| Bob            |        2017 | 150.00 |             100.00 |            50.00 |
| Bob            |        2018 | 200.00 |             150.00 |            50.00 |
| John           |        2016 | 200.00 |               0.00 |           200.00 |
| John           |        2017 | 150.00 |             200.00 |           -50.00 |
| John           |        2018 | 250.00 |             150.00 |           100.00 |
+----------------+-------------+--------+--------------------+------------------+
9 rows in set (0.00 sec)
```