NAME: SHARATH CHANDRA
BATCH: DATA ENGINEERING
TOPIC: PARTITION BY CLAUSE

PARTITION BY CLAUSE : The Partition by clause in SQL is used with window functions to divide the result set into partitions to which the window function is applied. It allows you to perform calculations, aggregations, and rankings within each partition separately

Syntax: SELECT

<column>,

<window function=""> OVER(PARTITION BY <column> [ORDER BY <column>])

FROM table;

</column></column>

</window></column>

EXAMPLE FOR PARTITION BY CLAUSE :

—>First I have created a table named Flights and inserted some values into it.

```
mysql> CREATE TABLE Flights (
    ->     flight_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     aircraft_make VARCHAR(50),
    ->     aircraft_model VARCHAR(50),
    ->     flight_number VARCHAR(10),
    ->     scheduled_departure DATETIME,
    ->     real_departure DATETIME,
    ->     scheduled_arrival DATETIME,
    ->     num_of_passengers INT,
    ->     total_revenue DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Inserting the provided data into the Flights table
mysql> INSERT INTO Flights (aircraft_make, aircraft_model, flight_number, scheduled_departure, real_departure, scheduled_arrival, n
um_of_passengers, total_revenue)
    -> VALUES
    ->     ('Boeing', '757 300', 'FLP003', '2019-01-30 15:00:00', '2019-01-30 15:00:00', '2019-01-30 15:00:00', 260, 82630.10),
    ->     ('Boeing', '737 200', 'FLP003', '2019-02-01 15:00:00', '2019-02-01 15:10:00', '2019-02-01 15:55:00', 195, 58459.34),
    ->     ('Airbus', 'A500', 'FLP003', '2019-02-01 15:00:00', '2019-02-01 15:03:00', '2019-02-01 15:03:55', 312, 91570.87),
    ->     ('Airbus', 'A500', 'FLP001', '2019-10-28 05:00:00', '2019-10-28 05:04:00', '2019-10-28 05:55:00', 298, 87943.00),
    ->     ('Boeing', '737 200', 'FLP002', '2019-10-28 09:00:00', '2019-10-28 09:00:00', '2019-10-28 09:55:00', 178, 56342.45);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

—>Then I performed the Partition by clause by writing a query.

EXAMPLE-1: To Calculate Total Passengers and Total Revenue Partitioned by Flight Number and Aircraft Model.

```
mysql> SELECT DISTINCT
    ->        flight_number,
    ->        aircraft_model,
    ->     SUM(num_of_passengers) OVER (PARTITION BY flight_number, aircraft_model)
    ->                                                     AS total_passengers,
    ->     SUM(total_revenue) OVER (PARTITION BY flight_number, aircraft_model)
    ->                                                     AS total_revenue
    -> FROM flights
    -> ORDER BY flight_number, aircraft_model;
+---------------+----------------+------------------+---------------+
| flight_number | aircraft_model | total_passengers | total_revenue |
+---------------+----------------+------------------+---------------+
| FLP001        | A500           |              298 |      87943.00 |
| FLP002        | 737 200        |              178 |      56342.45 |
| FLP003        | 737 200        |              195 |      58459.34 |
| FLP003        | 757 300        |              260 |      82630.10 |
| FLP003        | A500           |              312 |      91570.87 |
+---------------+----------------+------------------+---------------+
5 rows in set (0.01 sec)
```

EXAMPLE-2: To calculate the Total Number of Passengers for Each Distinct Year and Month.

```
mysql> WITH year_month_data AS (
    ->   SELECT DISTINCT
    ->        EXTRACT(YEAR FROM scheduled_departure) AS year,
    ->        EXTRACT(MONTH FROM scheduled_departure) AS month,
    ->        SUM(num_of_passengers)
    ->               OVER (PARTITION BY EXTRACT(YEAR FROM scheduled_departure),
    ->                                  EXTRACT(MONTH FROM scheduled_departure)
    ->                    ) AS passengers
    ->    FROM  flights
    ->   ORDER BY 1, 2
    -> )
    -> SELECT  year,
    ->         month,
    ->      passengers,
    ->      LAG(passengers) OVER (ORDER BY year, month) passengers_previous_month,
    ->      passengers - LAG(passengers) OVER (ORDER BY year, month) AS passengers_delta
    -> FROM year_month_data;
+------+-------+------------+--------------------------+------------------+
| year | month | passengers | passengers_previous_month | passengers_delta |
+------+-------+------------+--------------------------+------------------+
| 2019 |     1 |        260 |                     NULL |             NULL |
| 2019 |     2 |        507 |                      260 |              247 |
| 2019 |    10 |        476 |                      507 |              -31 |
+------+-------+------------+--------------------------+------------------+
3 rows in set (0.00 sec)
```

EXAMPLE-3: To Calculate Average Monthly Delays for Flights between Paris and London.

```
mysql> WITH paris_london_delays AS (
    ->   SELECT DISTINCT
    ->        aircraft_model,
    ->        EXTRACT(YEAR FROM scheduled_departure) AS year,
    ->        EXTRACT(MONTH FROM scheduled_departure) AS month,
    ->        AVG(real_departure - scheduled_departure) AS month_delay
    ->    FROM  flights
    ->    GROUP BY 1, 2, 3
    -> )
    -> SELECT  DISTINCT
    ->      aircraft_model,
    ->      year,
    ->      month,
    ->      month_delay AS monthly_avg_delay,
    ->      AVG(month_delay) OVER (PARTITION BY aircraft_model, year) AS year_avg_delay,
    ->      AVG(month_delay) OVER (PARTITION BY year) AS year_avg_delay_all_models,
    ->      AVG(month_delay) OVER (PARTITION BY aircraft_model, year
    ->                             ORDER BY month
    ->                             ROWS BETWEEN 3 PRECEDING AND CURRENT ROW
    ->                           ) AS rolling_average_last_4_months
    ->
    -> FROM paris_london_delays
    -> ORDER BY 1,2,3;
+----------------+------+-------+-------------------+----------------+--------------------------+-------------------------------+
| aircraft_model | year | month | monthly_avg_delay | year_avg_delay | year_avg_delay_all_models | rolling_average_last_4_months |
+----------------+------+-------+-------------------+----------------+--------------------------+-------------------------------+
| 737 200        | 2019 |     2 |         1000.0000 |   500.00000000 |              340.00000000 |                1000.00000000 |
| 737 200        | 2019 |    10 |            0.0000 |   500.00000000 |              340.00000000 |                 500.00000000 |
| 757 300        | 2019 |     1 |            0.0000 |     0.00000000 |              340.00000000 |                   0.00000000 |
| A500           | 2019 |     2 |          300.0000 |   350.00000000 |              340.00000000 |                 300.00000000 |
| A500           | 2019 |    10 |          400.0000 |   350.00000000 |              340.00000000 |                 350.00000000 |
+----------------+------+-------+-------------------+----------------+--------------------------+-------------------------------+
```