Execute OVER and PARTITION BY Clause in SQL Queries

SQL PARTITION BY Clause:

The SQL PARTITION BY clause is a subclause of the OVER clause, commonly used in conjunction with window functions such as AVG(), MAX(), and RANK(). It is employed in scenarios where window functions operate on window frames—sets of rows that can vary for each record in the query result.

Syntax:

```
SELECT
   <column>,
   <window function> OVER(PARTITION BY <column> [ORDER BY <column>])
FROM table;
```

The PARTITION BY clause divides the result set into partitions based on specified columns, and the window function is then applied within each partition. This allows for more granular analysis within distinct groups of data.

1. Basic SELECT Query:

The basic SELECT query retrieves specific columns from the City table where a certain condition is met (e.g., CountryCode is 'AUS' or 'NZL'). This is a standard query for data retrieval.

2. GROUP BY with WITH ROLLUP:

The query demonstrates the use of the GROUP BY clause with ROLLUP, providing subtotals and a grand total for city populations. This is helpful for summarizing data and generating reports with hierarchical totals.

3. Query Without Labels (using NULLs):

Similar to the previous query, but without explicit labels. NULL values represent subtotal and grand total rows. This showcases an alternative presentation style.

Key Concepts:

GROUP BY Clause:

- Used to group rows based on specific criteria, allowing the application of aggregate functions to yield one result per group.

WITH ROLLUP:

- Extends the GROUP BY functionality by including extra rows that represent subtotals and a grand total.

Aggregation Functions (SUM, AVG):

- Functions like SUM() and AVG() are employed to perform calculations on groups of rows, providing aggregated results.

PARTITION BY Clause:

- Used within the OVER clause to define partitions for window functions, enabling analysis within distinct subsets of data.
-

Window Functions:

- Functions like AVG(), MAX(), etc., operate on window frames, and the PARTITION BY clause helps delineate these frames for specific analyses.

ORDER BY Clause:

- Optionally used within the PARTITION BY clause to establish the order of records within the window frame. Some window functions, like LEAD() and LAG(), require an ORDER BY clause.

Window Frame Bounds:

- Specifies the limits of the window frame using expressions like UNBOUNDED PRECEDING, n PRECEDING, CURRENT ROW, n FOLLOWING, and UNBOUNDED FOLLOWING.

```
USE coding_challenge1;
-- Create another table for car_sales
CREATE TABLE IF NOT EXISTS car_sales (
    sale_id INT PRIMARY KEY,
    car_make VARCHAR(50),
    car_model VARCHAR(50),
    sale_price DECIMAL(10, 2),
    sale_date DATE
);

-- Insert data into the car_sales table
INSERT INTO car_sales VALUES
(1, 'Ford', 'Mondeo', 20000, '2023-01-15'),
(2, 'Renault', 'Fuego', 18000, '2023-01-20'),
(3, 'Citroen', 'Cactus', 22000, '2023-02-05'),
(4, 'Ford', 'Falcon', 9500, '2023-02-10'),
(5, 'Ford', 'Galaxy', 13000, '2023-03-01'),
(6, 'Renault', 'Megane', 15000, '2023-03-05'),
(7, 'Citroen', 'Picasso', 25000, '2023-03-10');
```

```
-- Query 1: Average price across all car sales and average price by car type
SELECT
   car_make,
   car_model,
   sale_price,
   AVG(sale_price) OVER() AS "overall_average_price",
   AVG(sale_price) OVER (PARTITION BY car_make) AS "car_make_average_price"
FROM car_sales;
```

car_make	car_model	sale_price	overall_average_price	car_make_average_price
Citroen	Cactus	22000.00	17500.000000	23500.000000
Citroen	Picasso	25000.00	17500.000000	23500.000000
Ford	Mondeo	20000.00	17500.000000	14166.666667
Ford	Falcon	9500.00	17500.000000	14166.666667
Ford	Galaxy	13000.00	17500.000000	14166.666667
Renault	Fuego	18000.00	17500.000000	16500.000000
Renault	Megane	15000.00	17500.000000	16500.000000

```
-- Query 2: Average price and top price per car make
SELECT
   car_make,
   AVG(sale_price) AS average_price,
   MAX(sale_price) AS top_price
FROM car_sales
GROUP BY car_make;
```

car_make	average_price	top_price
Ford	14166.666667	20000.00
Renault	16500.000000	18000.00
Citroen	23500.000000	25000.00

```
-- Query 3: Average make price using window functions
SELECT
   car_make,
   car_model,
   sale_price,
   AVG(sale_price) OVER (PARTITION BY car_make) AS average_make
FROM car_sales;
```

	car_make	average_price	top_price
▶	Ford	14166.666667	20000.00
	Renault	16500.000000	18000.00
	Citroen	23500.000000	25000.00

```
-- Query 4: Monthly variation in car sales prices
WITH year_month_data AS (
  SELECT DISTINCT
      DATE_FORMAT(sale_date, '%Y') AS year,
      DATE_FORMAT(sale_date, '%m') AS month,
      AVG(sale_price) AS average_price
   FROM car_sales
   GROUP BY 1, 2
 )
 SELECT  year,
       month,
       average_price,
       LAG(average_price) OVER (ORDER BY year, month) AS
avg_price_previous_month,
```

average_price - LAG(average_price) OVER (ORDER BY year, month) AS avg_price_delta

FROM year_month_data;

year	month	average_price	avg_price_previous_month	avg_price_delta
2023	01	19000.000000	NULL	NULL
2023	02	15750.000000	19000.000000	-3250.000000
2023	03	17666.666667	15750.000000	1916.666667

Totals and Subtotals

Subtotals and Total Aggregations:

Subtotals are generated by the ROLLUP operator, which creates aggregated results for subsets of the grouping columns.

The grand total represents the overall sum of the specified column (Population in this case).

The GROUP BY clause, in conjunction with the ROLLUP operator, allows for efficient calculation of subtotals and grand totals in a single query.

ROLLUP Operator:

The ROLLUP operator is used in the GROUP BY clause to generate subtotals and a grand total for a result set.

It is used to create multiple grouping sets in a single query, allowing you to aggregate data at different levels of granularity.

Concepts in the Query:

IF(GROUPING(CountryCode), 'All Countries', CountryCode):

The GROUPING function is used to check if a column is part of the grouping set.

If GROUPING(CountryCode) evaluates to 1, it means that the row represents a subtotal or grand total, and 'All Countries' is displayed; otherwise, the actual CountryCode is displayed.

SUM(Population):

Aggregates the population values for each grouping set.

-- Create a new database

CREATE DATABASE IF NOT EXISTS city_population_Subtotals;

USE city_population_Subtotals;

```
-- Create a table for City
CREATE TABLE IF NOT EXISTS City (
    CountryCode CHAR(3),
    District VARCHAR(255),
    Name VARCHAR(255),
    Population INT
);

-- Insert values into the City table
INSERT INTO City (CountryCode, District, Name, Population) VALUES
    ('AUS', 'New South Wales', 'Sydney', 3276207),
    ('AUS', 'Victoria', 'Melbourne', 2865329),
    ('AUS', 'Queensland', 'Brisbane', 1291117),
```

('AUS', 'West Australia', 'Perth', 1096829),

('AUS', 'South Australia', 'Adelaide', 978100),

('AUS', 'Capital Region', 'Canberra', 322723),

('AUS', 'Queensland', 'Gold Coast', 311932),

('AUS', 'New South Wales', 'Newcastle', 270324),

('AUS', 'New South Wales', 'Central Coast', 227657),

('AUS', 'New South Wales', 'Wollongong', 219761),

('AUS', 'Tasmania', 'Hobart', 126118),

('AUS', 'Victoria', 'Geelong', 125382),

('AUS', 'Queensland', 'Townsville', 109914),

('AUS', 'Queensland', 'Cairns', 92273),

('NZL', 'Auckland', 'Auckland', 381800),

('NZL', 'Canterbury', 'Christchurch', 324200),

('NZL', 'Auckland', 'Manukau', 281800),

('NZL', 'Auckland', 'North Shore', 187700),

('NZL', 'Auckland', 'Waitakere', 170600),

('NZL', 'Wellington', 'Wellington', 166700),

('NZL', 'Dunedin', 'Dunedin', 119600),

('NZL', 'Hamilton', 'Hamilton', 117100),

('NZL', 'Wellington', 'Lower Hutt', 98100);

```
-- Query to select all records from the City table
SELECT
    CountryCode,
    District,
    Name,
    Population
FROM City
WHERE CountryCode IN ('AUS', 'NZL');
```

CountryCode	District	Name	Population
AUS	New South Wales	Sydney	3276207
AUS	Victoria	Melbourne	2865329
AUS	Queensland	Brisbane	1291117
AUS	West Australia	Perth	1096829
AUS	South Australia	Adelaide	978100
AUS	Capital Region	Canberra	322723
AUS	Queensland	Gold Coast	311932
AUS	New South Wales	Newcastle	270324
AUS	New South Wales	Central Coast	227657
AUS	New South Wales	Wollongong	219761
AUS	Tasmania	Hobart	126118
AUS	Victoria	Geelong	125382
AUS	Queensland	Townsville	109914
AUS	Queensland	Cairns	92273
NZL	Auckland	Auckland	381800
NZL	Canterbury	Christchurch	324200
NZL	Auckland	Manukau	281800
NZL	Auckland	North Shore	187700
NZL	Auckland	Waitakere	170600
NZL	Wellington	Wellington	166700
NZL	Dunedin	Dunedin	119600

```
-- Query to retrieve city populations with subtotals and grand total
SELECT
    IF(GROUPING(CountryCode), 'All Countries', CountryCode) AS CountryCode,
    IF(GROUPING(District), 'All Districts', District) AS District,
    IF(GROUPING(Name), 'All Cities', Name) As CityName,
    SUM(Population) AS Population
FROM City
WHERE CountryCode IN ('AUS', 'NZL')
GROUP BY CountryCode, District, Name WITH ROLLUP;
```

CountryCode	District	CityName	Population
AUS	Victoria	All Cities	2990711
AUS	West Australia	Perth	1096829
AUS	West Australia	All Cities	1096829
AUS	All Districts	All Cities	11313666
NZL	Auckland	Auckland	381800
NZL	Auckland	Manukau	281800
NZL	Auckland	North Shore	187700
NZL	Auckland	Waitakere	170600
NZL	Auckland	All Cities	1021900
NZL	Canterbury	Christchurch	324200
NZL	Canterbury	All Cities	324200
NZL	Dunedin	Dunedin	119600
NZL	Dunedin	All Cities	119600
NZL	Hamilton	Hamilton	117100
NZL	Hamilton	All Cities	117100
NZL	Wellington	Lower Hutt	98100
NZL	Wellington	Wellington	166700
NZL	Wellington	All Cities	264800
NZL	All Districts	All Cities	1847600
All Countries	All Districts	All Cities	13161266

```
-- Query without labels (using NULLs) to get the same output
SELECT
   CountryCode,
   District,
   Name,
   SUM(Population) AS Population
FROM City
WHERE CountryCode IN ('AUS', 'NZL')
GROUP BY CountryCode, District, Name WITH ROLLUP;
```

CountryCode	District	Name	Population
AUS	New South Wales	Wollongong	219761
AUS	New South Wales	NULL	3993949
AUS	Queensland	Brisbane	1291117
AUS	Queensland	Cairns	92273
AUS	Queensland	Gold Coast	311932
AUS	Queensland	Townsville	109914
AUS	Queensland	NULL	1805236
AUS	South Australia	Adelaide	978100
AUS	South Australia	NULL	978100
AUS	Tasmania	Hobart	126118
AUS	Tasmania	NULL	126118
AUS	Victoria	Geelong	125382
AUS	Victoria	Melbourne	2865329
AUS	Victoria	NULL	2990711
AUS	West Australia	Perth	1096829
AUS	West Australia	NULL	1096829
AUS	NULL	NULL	11313666
NZL	Auckland	Auckland	381800
NZL	Auckland	Manukau	281800
NZL	Auckland	North Shore	187700
NZL	Auckland	Waitakere	170600
NZL	Auckland	NULL	1021900
NZL	Canterbury	Christchurch	324200
NZL	Canterbury	NULL	324200
NZL	Hamilton	Hamilton	117100
NZL	Hamilton	NULL	117100
NZL	Wellington	Lower Hutt	98100
NZL	Wellington	Wellington	166700
NZL	Wellington	NULL	264800
NZL	NULL	NULL	1847600
NULL	NULL	NULL	13161266