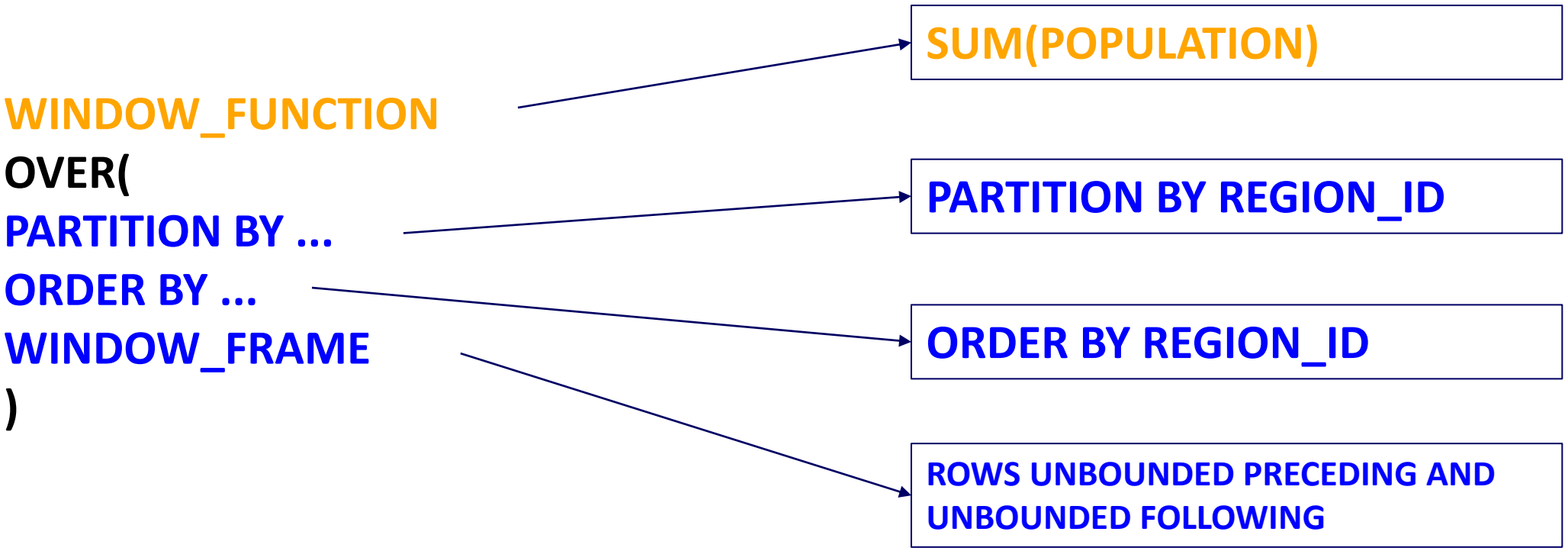
Introduction to Analytical Functions

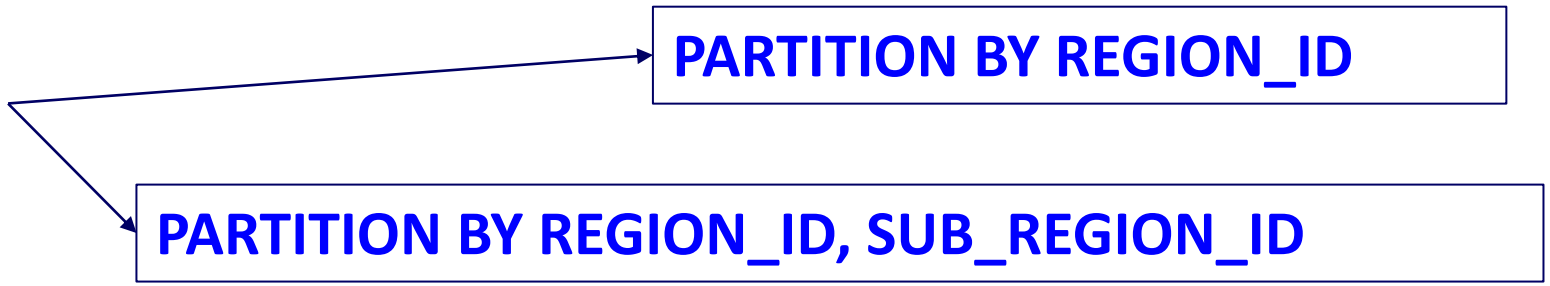
- Analytical Functions use the OVER() clause and are quite different to standard SQL
- Analytical Functions are also known as “Window Functions”
- Analytical Functions compute their result based on a “window” of one or more rows
- Unlike aggregation operations such as Group By, they do not collapse rows

Syntax


WINDOW_FUNCTION
OVER(
PARTITION BY ...
ORDER BY ...
WINDOW_FRAME
)
SUM(POPULATION)
PARTITION BY REGION_ID
ORDER BY REGION_ID
ROWS UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

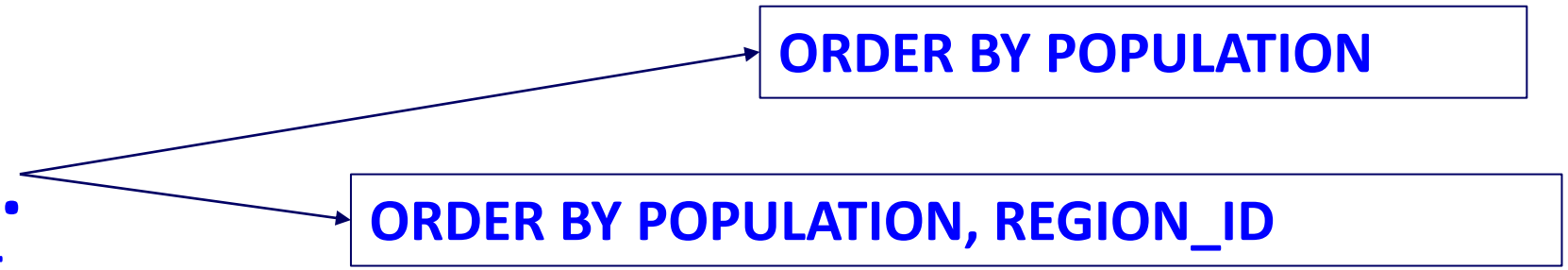
Partition By

```
WINDOW_FUNCTION
OVER(
PARTITION BY ...
ORDER BY ...
WINDOW_FRAME
)
```

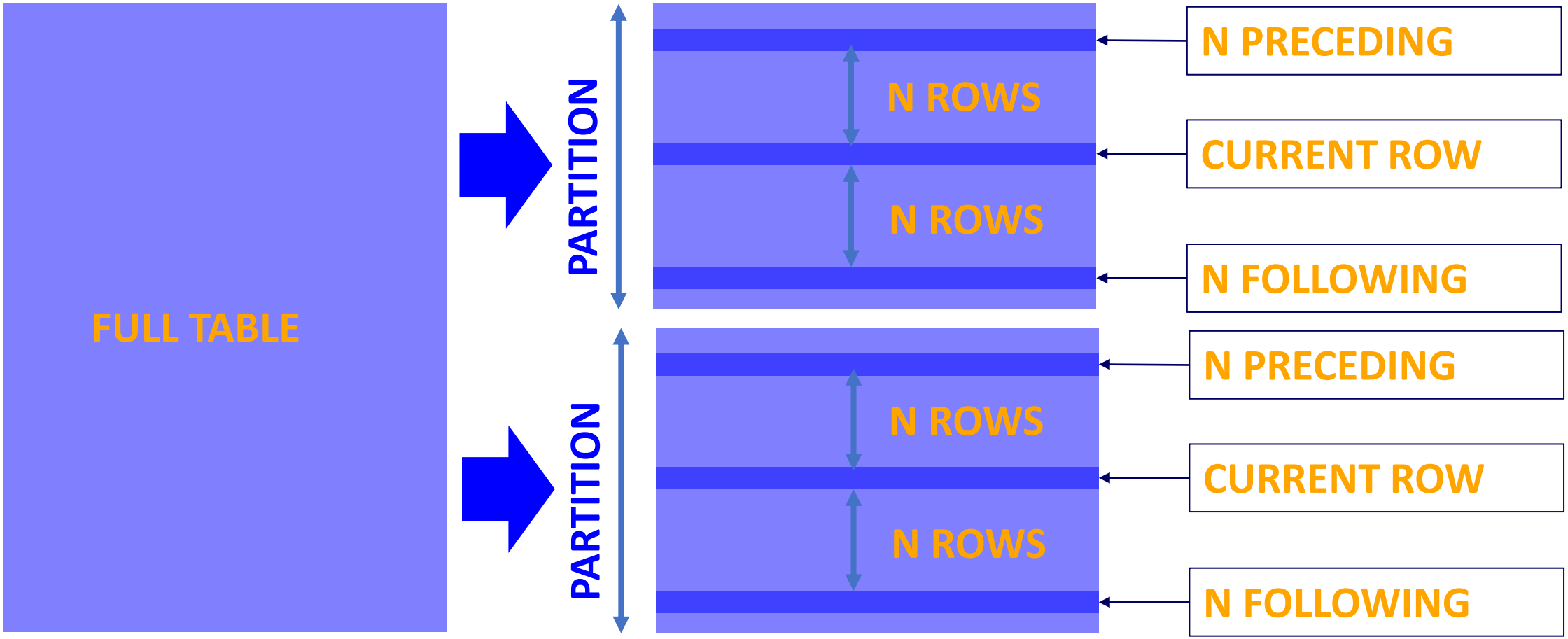


Order By

```
WINDOW_FUNCTION
OVER(
PARTITION BY ...
ORDER BY ...
WINDOW_FRAME
)
```



Window Frame

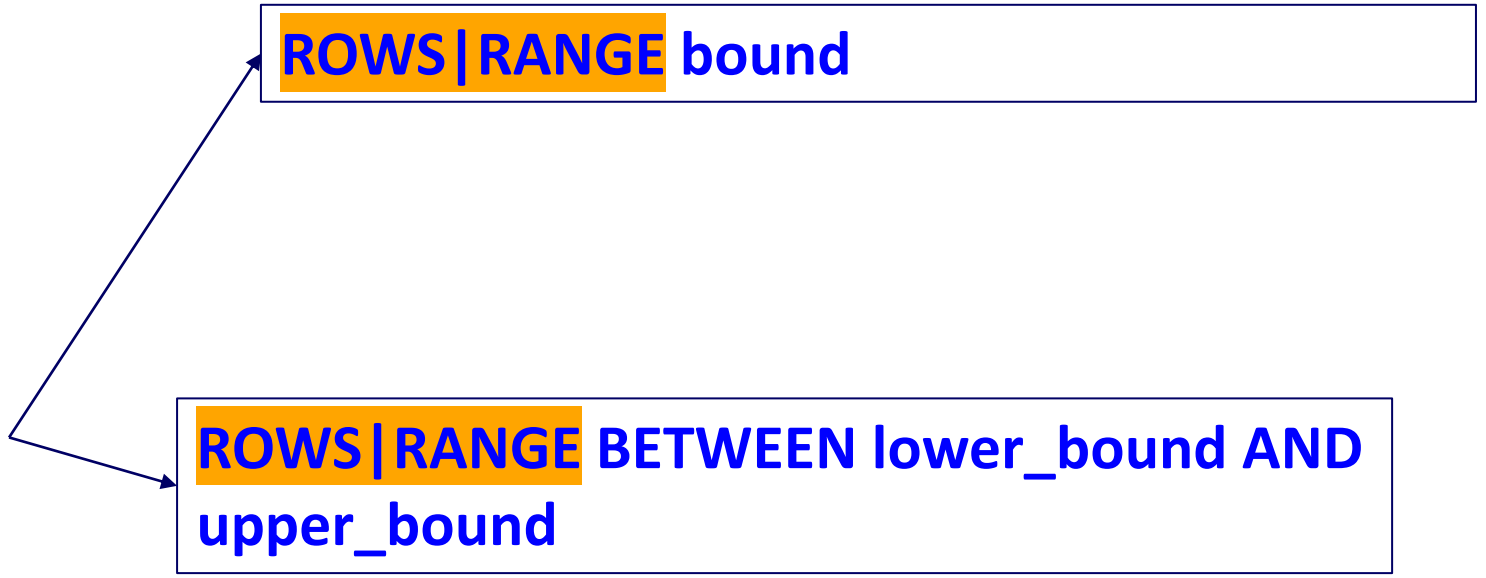


Window Frame

BOUND	DEFINITION
UNBOUNDED PRECEDING	UNBOUNDED PRECEDING AND CURRENT ROW
N PRECEDING	N PRECEDING AND CURRENT ROW
CURRENT ROW	CURRENT ROW
N FOLLOWING	CURRENT ROW AND N FOLLOWING
UNBOUNNDED FOLLOWING	CURRENT ROW AND UNBOUNDED FOLLOWING

Syntax

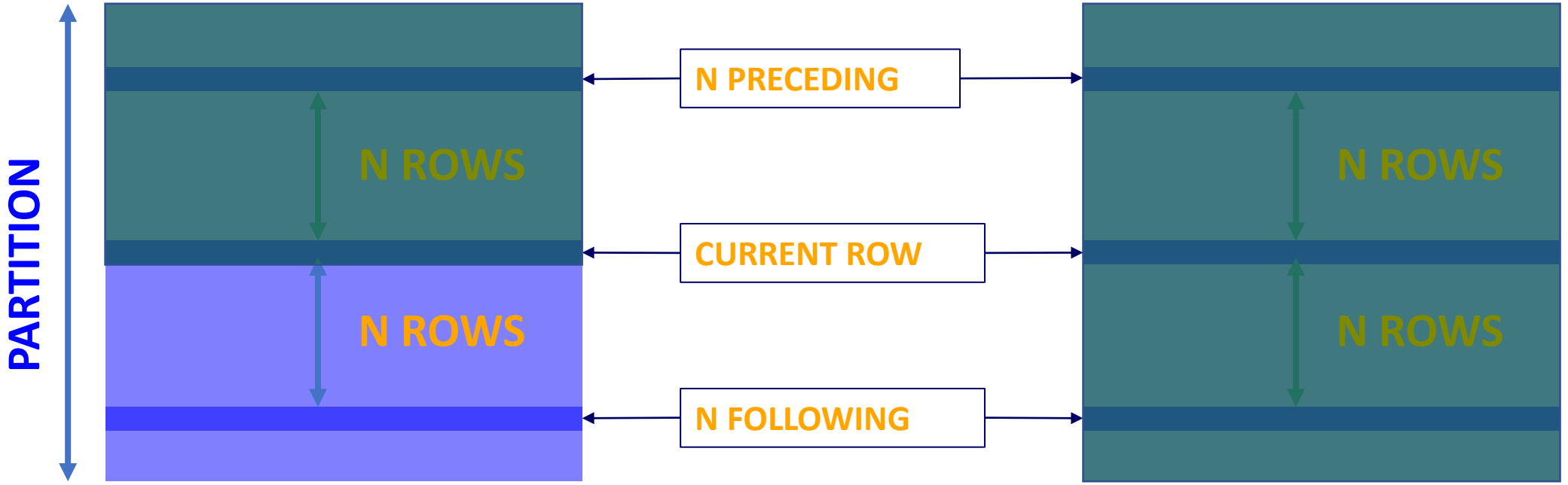
```
WINDOW_FUNCTION
OVER(
PARTITION BY ...
ORDER BY ...
WINDOW_FRAME
)
```



Window Frame - Default

If ORDER BY is specified, then the frame is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

IF ORDER BY is NOT specified, the frame is ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.



Window Frame - RANGE

```
SUM(POPULATION)
OVER(
PARTITION BY REGION_ID
ORDER BY SUB_REGION_ID
RANGE 10 PRECEDING)
```

PARTITION FOR REGION_ID = 20

NAME	REGION_ID	SUB_REGION_ID	POPULATION
Country A	20	10	1,000,000
Country B	20	10	2,000,000
Country C	20	20	500,000
Country D	20	30	55,000,000
Country E	20	40	10,000
Country F	20	40	20,000,000
Country G	20	50	1,000,000,000

500,000+55,000,000
=
55,500,000

```
SUM(POPULATION)
OVER(
PARTITION BY REGION_ID
ORDER BY SUB_REGION ID
RANGE BETWEEN UNBOUNDED PRECEDING AND
10 FOLLOWING)
```

NAME	REGION_ID	SUB_REGION_ID	POPULATION
Country A	20	10	1,000,000
Country B	20	10	2,000,000
Country C	20	20	500,000
Country D	20	30	55,000,000
Country E	20	40	10,000
Country F	20	40	20,000,000
Country G	20	50	1,000,000,000

1,000,000+2,000,000+500,000+55,000,000+10,000+20,000,000
=
78,510,000

Window Frame - ROWS

```
SUM(POPULATION)
OVER(
PARTITION BY REGION_ID
ORDER BY SUB_REGION ID
ROWS UNBOUNDED PRECEDING)
```

PARTITION FOR SUB_REGION_ID = 20

NAME	REGION_ID	SUB_REGION_ID	POPULATION
Country A	20	10	1,000,000
Country B	20	10	2,000,000
Country C	20	20	500,000
Country D	20	30	55,000,000
Country E	20	40	10,000
Country F	20	40	20,000,000
Country G	20	50	1,000,000,000

1,000,000 + 2,000,000 + 500,000 + 55,000,000
=
58,500,000

```
SUM(POPULATION)
OVER(
PARTITION BY REGION_ID
ORDER BY SUB_REGION ID
ROWS BETWEEN UNBOUNDED PRECEDING AND 1
FOLLOWING)
```

NAME	REGION_ID	SUB_REGION_ID	POPULATION
Country A	20	10	1,000,000
Country B	20	10	2,000,000
Country C	20	20	500,000
Country D	20	30	55,000,000
Country E	20	40	10,000
Country F	20	40	20,000,000
Country G	20	50	1,000,000,000

1,000,000 + 2,000,000 + 500,000 + 55,000,000+10,000
=
58,510,000

Types of Window Functions

AGGREGATE
SUM()
AVG()
MAX()
MIN()
COUNT()

RANKING
ROW_NUMBER()
RANK()
DENSE_RANK()

DISTRIBUTION
PERCENT_RANK()
CUME_DIST()

ANALYTICAL
LEAD()
LAG()
NTILE()
NTH_VALUE()

Ranking Functions

ROW_NUMBER() OVER(order_by_clause)

RANK() OVER(order_by_clause)

DENSE_RANK() OVER(order_by_clause)

order_by_clause
These ranking functions must be used with an order by clause in the OVER clause in **Oracle SQL**

Order of Operations

FROM
JOIN
WHERE
GROUP BY
HAVING
WINDOW
SELECT
ORDER BY
LIMIT / FETCH / TOP

Distribution Functions

(RANK – 1)/(Total Number of Rows in partition – 1)

PERCENT_RANK() OVER(order_by_clause)

CUME_DIST() OVER(order_by_clause)

(Number of rows <= Current rows value)/(Total Number of Rows in partition)

order_by_clause
These ranking functions must be used with an order by clause in the OVER clause in **Oracle SQL**

Analytical Functions – LAG()

offset
Optional. This is the offset from the current row

order_by_clause
LAG() must be used with an order by clause in the OVER clause in Oracle SQL

```
LAG(expression, offset [,default]) OVER(order_by_clause)
```

expression
This can be a field, it can also contain a built-in function but not another analytical function

default
default is the value to be returned if offset goes beyond the scope of the partition. It defaults to NULL.

Analytical Functions – LEAD()

offset
Optional. This is the offset from the current row

order_by_clause
LAG() must be used with an order by clause in the OVER clause in Oracle SQL

LEAD(expression, offset [,default]) OVER(order_by_clause)

expression
This can be a field, it can also contain a built-in function but not another analytical function

default
default is the value to be returned if offset goes beyond the scope of the partition. It defaults to NULL.

Analytical Functions – NTILE()

order_by_clause
NTILE() must be used with an order by clause in the OVER clause in Oracle SQL

NTILE(number) OVER(order_by_clause)

number
This is the number of groups

Analytical Functions – NTH_VALUE()