Repaso SQL

SQL para Data Engineer

- El lenguaje de consulta estructurado (SQL) es un tipo de lenguaje de consulta que permite a los ingenieros de datos comunicarse con una base de datos.
- Un ingeniero de datos usará SQL para crear una consulta para ver, transformar o cargar los datos específicos que desea en el DW.
- En un data warehouse, los ingenieros de datos pueden escribir consultas para obtener datos de las tablas.

Lenguaje de Definición de Datos (DDL)

Es un lenguaje de programación para definir estructuras de datos, proporcionado por los sistemas gestores de bases de datos. En inglés, Data Definition Language, de ahí sus siglas DDL.

Lenguaje de Definición de Datos (DDL)

Para definir las estructura disponemos de tres sentencias:

CREATE: se usa para crear una base de datos, tabla, vistas, etc.

ALTER: se utiliza para modificar la estructura, por ejemplo añadir o borrar columnas de una tabla.

DROP: con esta sentencia, podemos eliminar los objetos de la estructura, por ejemplo una tabla o una base de datos.

Lenguaje de Manipulación de Datos (DML)

También es un lenguaje proporcionado por los sistemas gestores de bases de datos. En inglés, Data Manipulation Language (DML).

Utilizando instrucciones de SQL, permite a los ingenieros introducir datos para posteriormente realizar tareas de consultas o modificación de los datos que contienen las Bases de Datos.

Lenguaje de Manipulación de Datos (DML)

SELECT: esta sentencia se utiliza para realizar consultas sobre los datos.

INSERT: con esta instrucción podemos insertar los valores en una base de datos o DW.

UPDATE: sirve para modificar los valores de uno o varios registros.

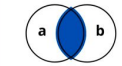
DELETE: se utiliza para eliminar las filas de una tabla.

Cheat Sheet

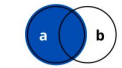
Commands / Clauses

SELECT	Select data from database
FROM	Specify table we're pulling from
WHERE	Filter query to match a condition
AS	Rename column or table with alias
JOIN	Combine rows from 2 or more tables
AND	Combine query conditions. All must be met
OR	Combine query conditions. One must be met
LIMIT	Limit rows returned. See also FETCH & TOP
IN	Specify multiple values when using WHERE
CASE	Return value on a specified condition
IS NULL	Return only rows with a NULL value
LIKE	Search for patterns in column
COMMIT	Write transaction to database
ROLLBACK	Undo a transaction block
ALTER TABLE	Add/Remove columns from table
UPDATE	Update table data
CREATE	Create TABLE, DATABASE, INDEX or VIEW
DELETE	Delete rows from table
INSERT	Add single row to table
DROP	Delete TABLE, DATABASE, or INDEX
GROUP BY	Group data into logical sets
ORDER BY	Set order of result. Use DESC to reverse order
HAVING	Same as WHERE but filters groups
COUNT	Count number of rows
SUM	Return sum of column
AVG	Return average of column
MIN	Return min value of column
MAX	Return max value of column

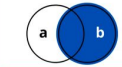
Joins



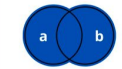
a INNER JOIN b



a LEFT JOIN b



a RIGHT JOIN b



a FULL OUTER JOIN b

Data Definition Language

CREATE

```
CREATE DATABASE MyDatabase;

CREATE TABLE MyTable (
    id int,
    name varchar(10));

CREATE INDEX IndexName
ON TableName(col1);
```

ALTER

```
ALTER TABLE MyTable
DROP COLUMN col5;

ALTER TABLE MyTable
ADD col5 int;
```

DROP

```
DROP DATABASE MyDatabase;
DROP TABLE MyTable;
```

Data Manipulation Language

UPDATE

```
UPDATE MyTable
SET col1 = 56
WHERE col2 = 'something';
```

INSERT

```
INSERT INTO MyTable (col1, col2)
VALUES ('value1', 'value2');
```

DELETE

```
DELETE FROM MyTable
WHERE col1 = 'something';
```

SELECT

```
SELECT col1, col2
FROM MyTable;
```

Order Of Execution

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY
7. LIMIT

Examples

Select all columns with filter applied

```
SELECT * FROM tbl
WHERE col > 5;
```

Select first 10 rows for two columns

```
SELECT col1, col2
FROM tbl LIMIT 10;
```

Select all columns with multiple filters

```
SELECT * FROM tbl
WHERE col1 > 5 OR col2 < 2;
```

Select all rows from col1 & col2 ordering by col1

```
SELECT col1, col2
FROM tbl ORDER BY 1;
```

Return count of rows in table

```
SELECT COUNT(*)
FROM tbl;
```

Return sum of col1

```
SELECT SUM(col1)
FROM tbl;
```

Return max value for col1

```
SELECT MAX(col1)
FROM tbl;
```

Compute summary stats by grouping col2

```
SELECT AVG(col1) FROM tbl
GROUP BY col2;
```

Combine data from 2 tables using left join

```
SELECT * FROM tbl1 AS t1 LEFT JOIN
tbl2 AS t2 ON t2.col1 = t1.col1;
```

Aggregate and filter result

```
SELECT col1,
       COUNT(*) AS total
FROM tbl
GROUP BY col1
HAVING COUNT(*) > 10;
```

Implementation of CASE statement

```
SELECT col1,
CASE
    WHEN col1 > 10 THEN 'more than 10'
    WHEN col1 < 10 THEN 'less than 10'
    ELSE '10'
END AS NewColumnName
FROM tbl;
```

Sentencias SQL

Select distinct

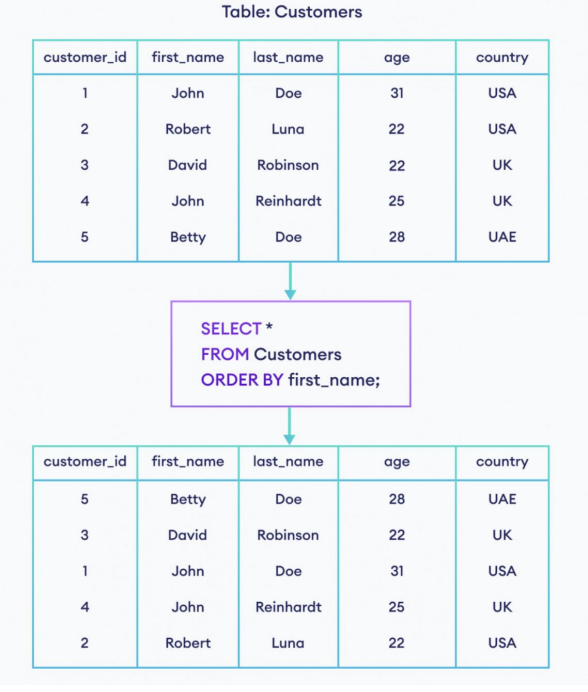
Es utilizado para que la consulta me devuelva valores únicos (no repetidos)

```
SELECT DISTINCT country
FROM Customers;
```

Order By

Es utilizado para ordenar los resultados de manera ascendente o descendente

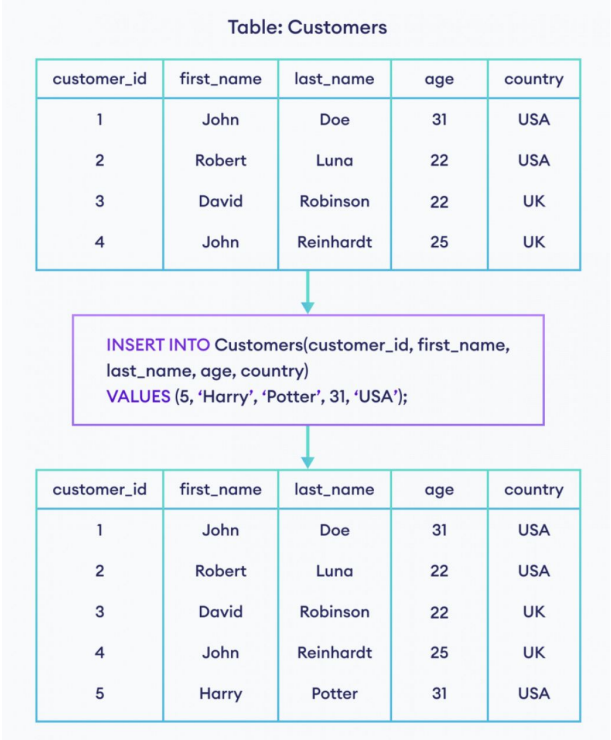
```sql
SELECT *
FROM Customers
ORDER BY first_name;
```



Insert Into

Es utilizado para insertar nuevos registros en una tabla

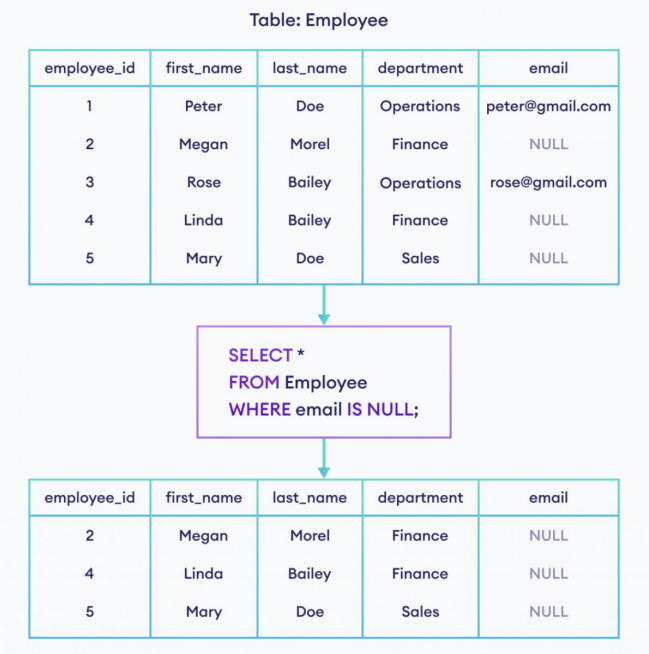
```
INSERT INTO Customers(customer_id, first_name, last_name, age, country)
VALUES
(5, 'Harry', 'Potter', 31, 'USA');
```



Valores Null - Coalesce

Es utilizado para seleccionar registros si un campo específico es nulo.

```sql
SELECT *
FROM Employee
WHERE email IS NULL;
```



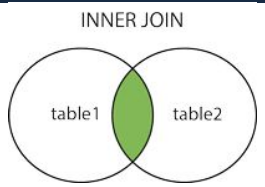
Valores Null - Coalesce

Es utilizado para evitar traer valores null y reemplazarlo por un valor de otro campo.

```sql
SELECT COALESCE(col1, col2, col3) AS non_null_value
FROM table_name;
```

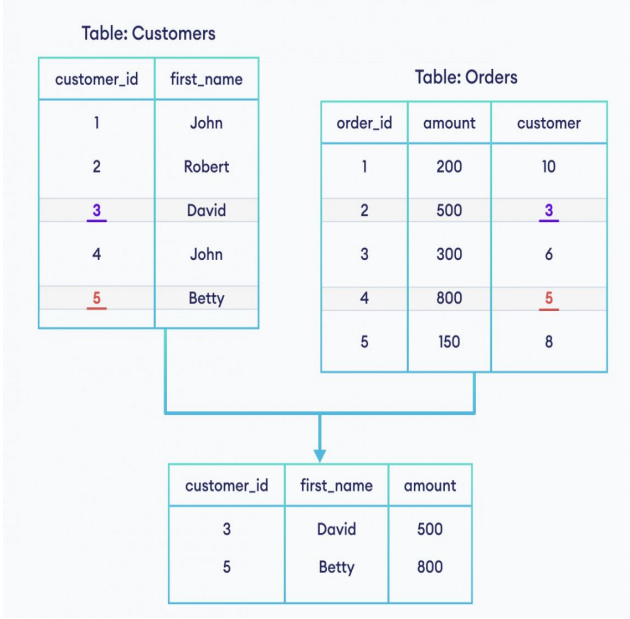
```sql
SELECT *
FROM table_name
WHERE COALESCE(col1, col2, col3) IS NOT NULL;
```

Inner Join

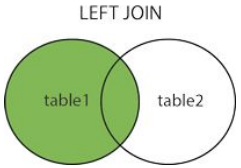


Es utilizado para seleccionar registros que tengan match entre dos (o más) tablas

```
SELECT Customers.customer_id, Customers.first_name, Orders.amount
FROM Customers
INNER JOIN Orders
ON Customers.customer_id = Orders.customer;
```



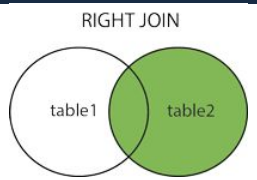
Left Join



Es utilizado para seleccionar registros de la tabla izquierda y los que hagan match con la tabla derecha.

```sql
SELECT Customers.customer_id, Customers.first_name, Orders.amount
FROM Customers
LEFT JOIN Orders
ON Customers.customer_id = Orders.customer;
```

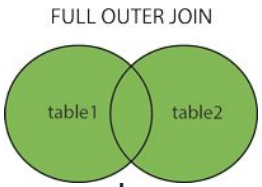
Right Join



Es utilizado para seleccionar registros de la tabla derecha y los que hagan match con la tabla izquierda.

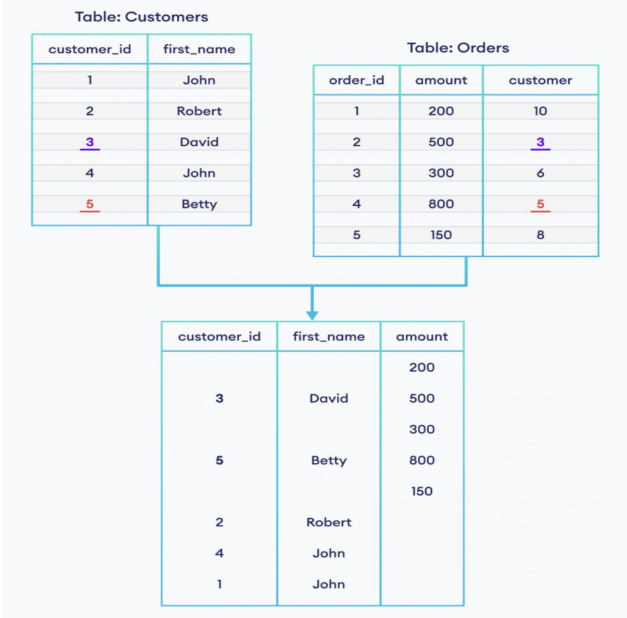
```sql
SELECT Customers.customer_id, Customers.first_name, Orders.amount
FROM Customers
RIGHT JOIN Orders
ON Customers.customer_id = Orders.customer;
```

Full Outer Join



Es utilizado para seleccionar registros que hacen match entre las dos tablas y deja los registros restantes de las dos tablas.

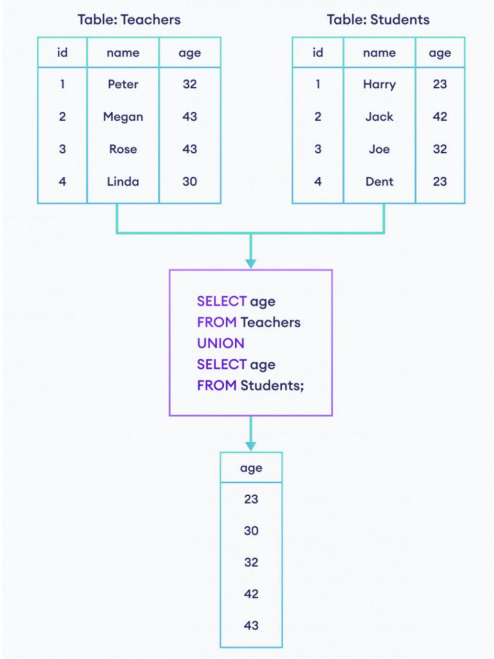
```
SELECT Customers.customer_id, Customers.first_name, Orders.amount
FROM Customers
FULL OUTER JOIN Orders
ON Customers.customer_id = Orders.customer;
```



Union

Es utilizado para combinar los resultados de dos o más sentencias select

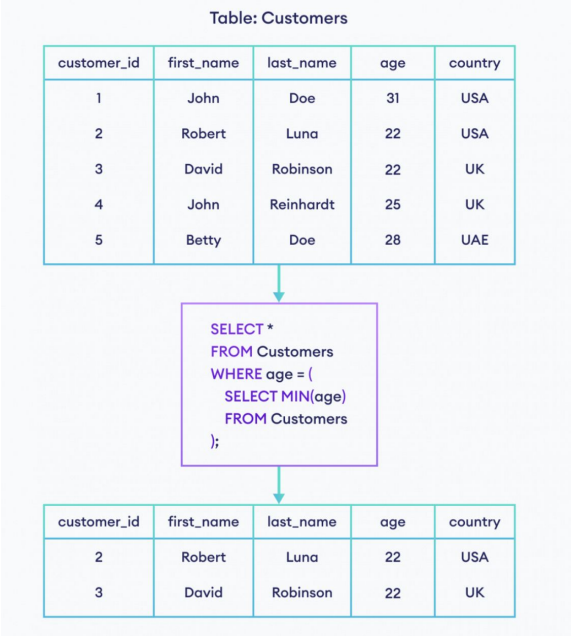
```
SELECT age
FROM Teachers
UNION
SELECT age
FROM Students;
```



Subquery

Es posible ejecutar una sentencia SQL dentro de otra query, llamada subquery

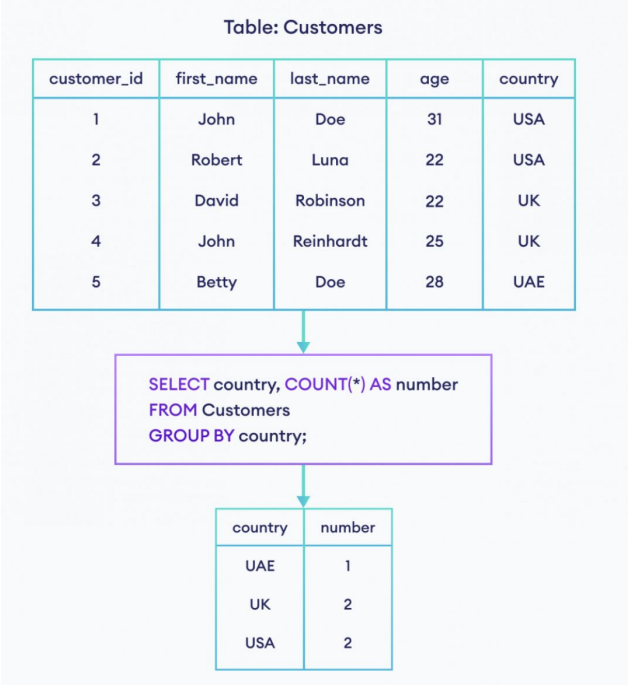
```sql
SELECT *
FROM Customers
WHERE age = (
  SELECT MIN(age)
  FROM Customers
);
```



Group by

Es utilizado para agrupar filas que tienen los mismos valores en filas de una operación matemática (SUM, MAX, COUNT, AVG, etc.)

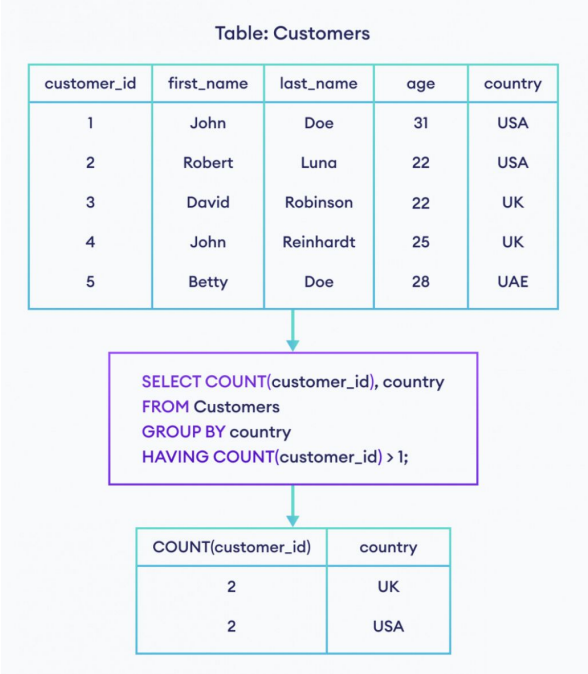
```
SELECT country, COUNT(*) AS number
FROM Customers
GROUP BY country;
```



Having

Se utiliza "having", seguido de la condición de búsqueda, para seleccionar ciertas filas retornadas por la cláusula "group by".

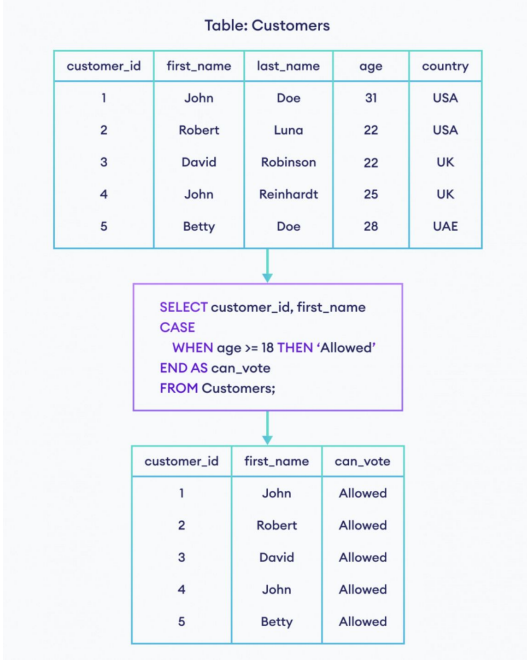
```
SELECT COUNT(customer_id), country
FROM Customers
GROUP BY country
HAVING COUNT(customer_id) > 1;
```



Case

La expresión CASE pasa por las condiciones y devuelve un valor cuando se cumple la primera condición. Una vez que una condición es verdadera, dejará de leer y devolverá el resultado. Si ninguna condición es verdadera, devuelve el valor de la cláusula ELSE.

```sql
SELECT customer_id, first_name,
CASE
  WHEN age >= 18 THEN 'Allowed'
END AS can_vote
FROM Customers;
```



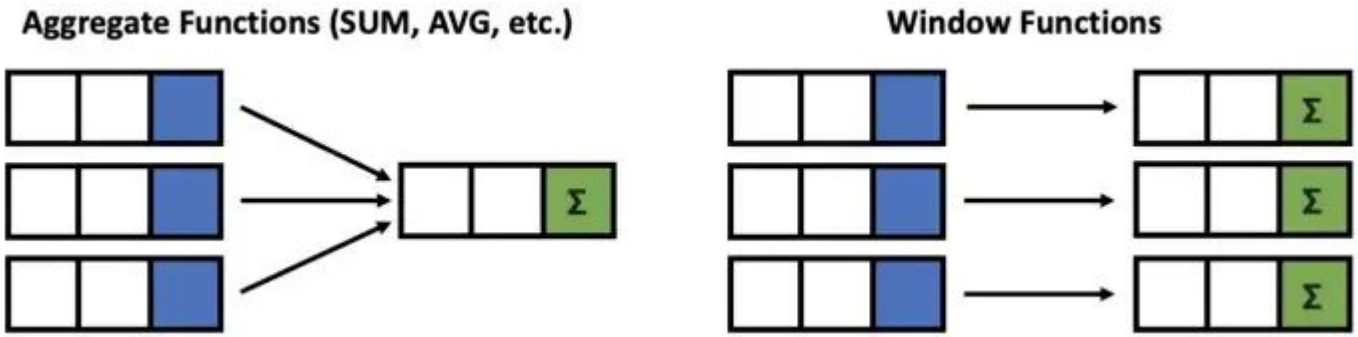
Repaso SQL

Windows functions

Windows function

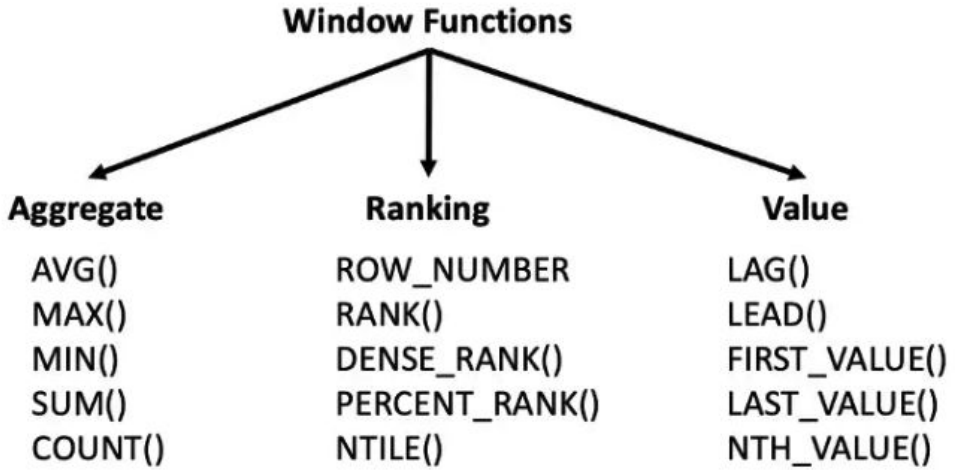
Son similares a la agregación realizada en la cláusula GROUP BY.

Las filas no se agrupan en una sola fila, cada fila conserva su identidad separada.



Windows function

Existen tres diferentes tipos de windows functions.



Windows function vs Group by

JOB_TITLE	SALARY
ANALYST	3100
ANALYST	2900
ANALYST	3250
SALES	1700
SALES	2500
SALES	4100
SALES	1600
SALES	2200
ENGINEER	3500
ENGINEER	3100
ENGINEER	4100

GROUP BY

JOB_TITLE	AVG_SALARY
ANALYST	3083.33333
ENGINEER	3566.66667
SALES	2420

Window Function

JOB_TITLE	SALARY	AVG_SALARY
ANALYST	3100	3083.333333
ANALYST	2900	3083.333333
ANALYST	3250	3083.333333
SALES	1700	2420
SALES	2500	2420
SALES	4100	2420
SALES	1600	2420
SALES	2200	2420
ENGINEER	3500	3566.666667
ENGINEER	3100	3566.666667
ENGINEER	4100	3566.666667

Funciones de Agregación

Varias funciones agregadas como SUM(), COUNT(), AVERAGE(), MAX(), MIN() aplicadas sobre una ventana particular (conjunto de filas).

Aggregate

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

Funciones de Agregación

AVG devuelve el promedio (media aritmética) de los valores de expresión de entrada. La función AVG funciona con valores numéricos e ignora los valores NULL

```
SELECT
        first_name,
        grade_level,
        test_score,
        AVG(test_score) OVER (PARTITION BY grade_level) AS average_for_grade
FROM
        class_test;
```

first_name	grade_level	test_score	average_for_grade
Frank	9	76	81.33333333333333
Humphrey	9	90	81.33333333333333
Iris	9	79	81.33333333333333
Sammy	9	85	81.33333333333333
Peter	9	80	81.33333333333333
Jojo	9	78	81.33333333333333
Brunhilda	12	92	89
Franco	12	94	89
Thomas	12	66	89
Gary	12	100	89
Charles	12	93	89
Jesse	12	89	89
Roseanna	11	94	73

Funciones de Agregación

SUM devuelve la suma de los valores de expresión de entrada.

```
select dealer_id, emp_name, sales,
sum(sales) over(partition by dealer_id) as
`sum`
from q1_sales;
```

dealer_id	emp_name	sales	sum
1	Ferris Brown	19745	57427
1	Noel Meyer	19745	57427
1	Raphael Hull	8227	57427
1	Jack Salazar	9710	57427
2	Beverly Lang	16233	41774
2	Kameko French	16233	41774
2	Haviva Montoya	9308	41774
3	Ursa George	15427	37104
3	Abel Kim	12369	37104
3	May Stout	9308	37104

Funciones de Ranking

Estas funciones permiten generar un ranking para clasificar el orden de un grupo de registros.

Ranking

ROW_NUMBER
RANK()
DENSE_RANK()
PERCENT_RANK()
NTILE()

Funciones de Ranking

Rank: devuelve la posición de cualquier fila dentro de la partición.

```
select salesid, qty,
rank() over (order by qty) as rnk
from winsales
order by 2,1;
```

```
salesid | qty | rnk
--------+-----+-----
10001 |  10 |  1
10006 |  10 |  1
30001 |  10 |  1
40005 |  10 |  1
30003 |  15 |  5
20001 |  20 |  6
20002 |  20 |  6
30004 |  20 |  6
10005 |  30 |  9
30007 |  30 |  9
40001 |  40 |  11
```

Funciones de Ranking

Row_number: determina el número ordinal de la fila actual dentro de un grupo de filas. Si la cláusula PARTITION BY opcional está presente, los números ordinales se restablecen para cada grupo de filas.

```
select salesid, sellerid, qty,
row_number() over
(partition by sellerid
 order by qty asc) as row
from winsales
order by 2,4;
```

salesid	sellerid	qty	row
10006	1	10	1
10001	1	10	2
10005	1	30	3
20001	2	20	1
20002	2	20	2
30001	3	10	1
30003	3	15	2
30004	3	20	3
30007	3	30	4
40005	4	10	1
40001	4	40	2

Funciones de Valor

Estas funciones permiten comparar valores de filas anteriores o siguientes dentro de la partición o el primer o último valor dentro de la partición.

Value

LAG()
LEAD()
FIRST_VALUE()
LAST_VALUE()
NTH_VALUE()

Funciones de Valor

LAG devuelve los valores de una fila en un desplazamiento determinado por encima (antes) de la fila actual en la partición.

```
select buyerid, saletime, qtysold,
lag(qtysold,1) over (order by buyerid, saletime) as prev_qtysold
from sales where buyerid = 3 order by buyerid, saletime;
```

buyerid	saletime	qtysold	prev_qtysold
3	2008-01-16 01:06:09	1	
3	2008-01-28 02:10:01	1	1
3	2008-03-12 10:39:53	1	1
3	2008-03-13 02:56:07	1	1
3	2008-03-29 08:21:39	2	1
3	2008-04-27 02:39:01	1	2
3	2008-08-16 07:04:37	2	1
3	2008-08-22 11:45:26	2	2
3	2008-09-12 09:11:25	1	2
3	2008-10-01 06:22:37	1	1
3	2008-10-20 01:55:51	2	1
3	2008-10-28 01:30:40	1	2

Funciones de Valor

LEAD devuelve los valores de una fila en un desplazamiento dado debajo (después) de la fila actual en la partición.

```
select eventid, commission, saletime,
lead(commission, 1) over (order by saletime) as next_comm
from sales where saletime between '2008-01-01 00:00:00' and '2008-01-02 12:59:59'
order by saletime;
```

eventid	commission	saletime	next_comm
6213	52.05	2008-01-01 01:00:19	106.20
7003	106.20	2008-01-01 02:30:52	103.20
8762	103.20	2008-01-01 03:50:02	70.80
1150	70.80	2008-01-01 06:06:57	50.55
1749	50.55	2008-01-01 07:05:02	125.40
8649	125.40	2008-01-01 07:26:20	35.10
2903	35.10	2008-01-01 09:41:06	259.50
6605	259.50	2008-01-01 12:50:55	628.80
6870	628.80	2008-01-01 12:59:34	74.10
6977	74.10	2008-01-02 01:11:16	13.50
4650	13.50	2008-01-02 01:40:59	26.55
4515	26.55	2008-01-02 01:52:35	22.80
5465	22.80	2008-01-02 02:28:01	45.60
5465	45.60	2008-01-02 02:28:02	53.10
7003	53.10	2008-01-02 02:31:12	70.35
4124	70.35	2008-01-02 03:12:50	36.15
1673	36.15	2008-01-02 03:15:00	1300.80

Repaso SQL