Día 5: SQL Avanzado II: Window Functions (Enfoque Profesional)

Checklist	☑
# Día	5
Estado	En progreso
Fecha	@27 de diciembre de 2025
Notas Técnicas	**1. Discrepancia entre Sintaxis y Ejecución Lógica:** Es fundamental recordar que SQL tiene una **Jerarquía de Escritura** (que inicia con `SELECT`) y un **Flujo Lógico de Ejecución** (que inicia con `FROM/JOIN`). Las **Window Functions** se calculan específicamente en el paso del `SELECT`, lo que significa que el motor ya ha filtrado las filas mediante el `WHERE` antes de realizar los cálculos de la ventana. **2. Gestión de Empates en Funciones de Ranking:** La elección de la función de ranking depende de la necesidad del análisis: • `ROW_NUMBER()`: Asigna un consecutivo único sin importar empates. • `RANK()`: Deja huecos en la secuencia tras un empate (ej. de 1, 1 salta a 3) para reflejar la posición física real • `DENSE_RANK()`: Mantiene una numeración compacta y continua (ej. de 1, 1 pasa a 2), ideal para clasificar niveles de importancia o valores distintos. **3. Preservación de la Granularidad con PARTITION BY:**

	A diferencia de la sentencia `GROUP BY`, que "aplasta" o colapsa los registros para mostrar solo el resumen del grupo, la cláusula `PARTITION BY` crea ventanas de cálculo sin perder los datos individuales de cada fila. Esto permite visualizar, por ejemplo, el detalle de habitaciones de una propiedad junto al conteo total de su categoría en una misma tabla. **4. Detección de Outliers mediante AVG() OVER():** El uso de funciones analíticas como el promedio por partición facilita la identificación inmediata de valores fuera de mercado. Al comparar el valor individual contra el `promedio_por_tipo`, se pueden detectar propiedades que están significativamente por encima (como en la colonia San Ángel) o por debajo (como en Santa Marta Acatitla) de la media de su categoría. **5. Patrón de Diseño CTE + Window Function para Filtrado:** Existe una limitación técnica donde no se puede usar una Window Function directamente en una sentencia `WHERE` debido al orden de ejecución de SQL. Para solucionar esto, se debe emplear una **CTE (Common Table Expression)** que "materialice" el cálculo en una tabla temporal (Fase 1), permitiendo que la consulta principal (Fase 2) pueda filtrar esos resultados como si fueran columnas físicas permanentes.
Tiempo Invertido	6 horas

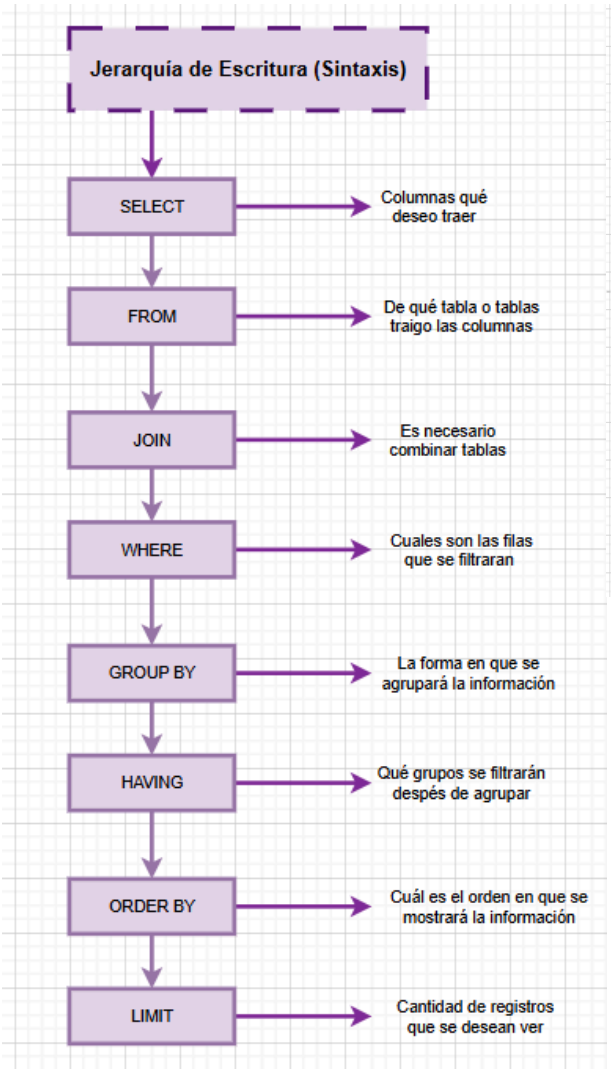
Día 5 – SQL Avanzado II: Window Functions

🎯 Objetivo del día

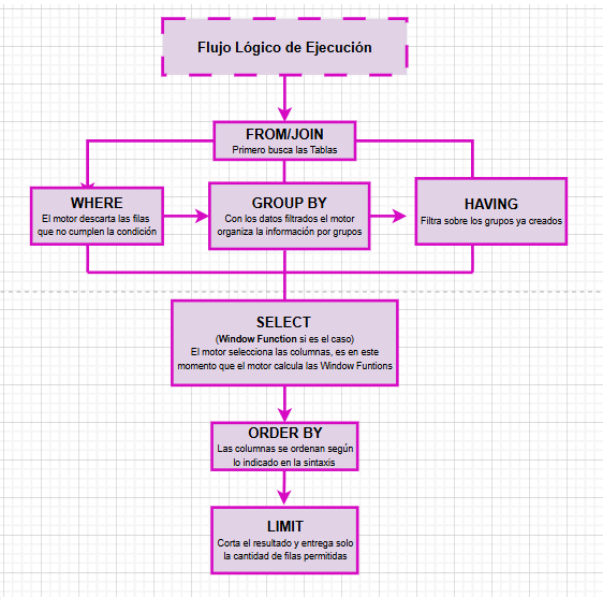
Aprender y aplicar **Window Functions** en PostgreSQL para resolver problemas analíticos reales, entendiendo claramente la diferencia entre **agregaciones tradicionales** y **funciones de ventana**.

🧱 BLOQUE 1 – Conceptos base:

Antes de empezar es necesario tener claro la Jerarquía de Escitura y la Jerarquía de Lectura, pues en SQL la sintaxis es diferente de como es el orden de ejecución de las acciones:



Flujo de escritura SQL



Flujo de la ejecución de acciones

Temas de estudio:

Window Function: Son un grupo de **funciones especiales**(Cálculos), que se aplican sobre un **grupo de filas relacionadas**, por medio de la cláusula **OVER()**, esta es una función muy parecida al GROUP BY, solo que esta no elimina la información de los registros individuales, permtiendo tener una visión general pero a la vez individual.

En este siguiente código, se busca tener un conteo de la cantidad de Tipos de Propiedades que hay en cada Colina, teniendo el detalle de cuantas Habitaciones tiene cada registro.

```
SELECT tipo, colonia, habitaciones,
   COUNT(*)
      OVER(
         PARTITION BY colonia, tipo
      ) as suma_de_tipo_por_colonia
FROM app.datos;
```

En el resultado podemos ver como ejemplo la colonia Ajusco tiene 3 Departamentos donde uno tiene 3 habitaciones, otro 2 habitaciones y el último 3 habitaciones, en cambio que la colina Álvaro Obregón tiene 21 Departamentos, donde se puede observar que unos tienen 2 habitaciones y otros 3.

	tipo text	colonia text	habitaciones integer	suma_de_tipo_por_colonia bigint
1	Casa de Condominio	Ajusco	4	1
2	Departamento	Ajusco	3	3
3	Departamento	Ajusco	2	3
4	Departamento	Ajusco	3	3
5	Galpón/Depósito/Alma...	Ajusco	0	1
6	Tienda/Salón	Ajusco	0	1
7	Casa	Álvaro Obregón	2	9
8	Casa	Álvaro Obregón	1	9
9	Casa	Álvaro Obregón	1	9
10	Casa	Álvaro Obregón	2	9
11	Casa	Álvaro Obregón	2	9
12	Casa	Álvaro Obregón	3	9
13	Casa	Álvaro Obregón	2	9
14	Casa	Álvaro Obregón	1	9
15	Casa	Álvaro Obregón	3	9
16	Casa de Vecindad	Álvaro Obregón	2	2
17	Casa de Vecindad	Álvaro Obregón	1	2
18	Departamento	Álvaro Obregón	2	21
19	Departamento	Álvaro Obregón	3	21
20	Departamento	Álvaro Obregón	2	21

Ventana de Datos: Una ventana de datos es la capacidad de SQL de realizar cálculos complejos sobre grupos específicos de filas sin "aplastar" o agrupar la tabla original, manteniendo el detalle de cada registro.

🧰 BLOQUE 2 – Primeras Window Functions (hands-on):

1️⃣ ROW_NUMBER()

En la siguiente consulta se busca tener el Ranking de la propiedad más cara a la más barata, eliminando los valores que sean Null en la columna Valor.

Para poder tener este Ranking se usa **window funtion** con la agregación de `ROW_NUMBER()` .

```
SELECT tipo, colonia,valor,
ROW_NUMBER() OVER(
   ORDER BY valor DESC) AS ranking
FROM app.datos
WHERE valor IS NOT NULL;
```

El resultado de esta consulta, será una tabla con las columnas Tipo, Colonia, Valor y Ranking, donde esta última se forma con base a la columna de Valor.

2️⃣ RANK() vs DENSE_RANK()

Esta consulta se busca comparar y entender las diferencia entre las funciones de ventana `RANK()` y `DENSE_RANK()` .

RANK() enumera los valores iguales con empate pero al momento de continuar con el siguiente valor diferente salta dejando por fuera el número que seguia en el concecutivo, pero el **DENSE_RANK()** aunque deja en empate los valores iguales, al momento de continuar con el próximo dato diferente, continua su enumeración sin saltar puestos.

Función	Transición de Empate	Comportamiento
`RANK()`	Del **1** salta al **3**	Deja huecos basados en la cantidad de duplicados.
`DENSE_RANK()`	Del **1** pasa al **2**	No salta ningún puesto, mantiene el conteo compacto.

```
SELECT
  colonia,
  valor,
RANK()OVER (ORDER BY valor DESC)AS rank,
DENSE_RANK()OVER (ORDER BY valor DESC)AS dense_rank
FROM app.datos
WHERE valor IS NOT NULL;
```

Al realizar la consulta se puede observar como en las filas 22,23 y 24 el RANK paso de 22,22 a 24 y como el DENSE_RANK paso de 22,22 a 23 sin saltar el número que venía en la secuencia, ya que el dato de la columna valor de las filas 22 y 23 era el mismo.

	colonia text	valor numeric	rank bigint	dense_rank bigint
13	Centro Histórico	2695490.0	13	13
14	Santa Fe	2583689.5	14	14
15	Roma	2378250.0	15	15
16	Santa Fe	2370760.0	16	16
17	Santa Fe	2275000.0	17	17
18	Centro Histórico	2195879.0	18	18
19	Santa Fe	2149000.0	19	19
20	Centro Histórico	2143925.0	20	20
21	La Condesa	2119845.0	21	21
22	Santa Fe	2100000.0	22	22
23	Santa Fe	2100000.0	22	22
24	Villa de Cortés	2016000.0	24	23
25	Centro Histórico	1960000.0	25	24

3️⃣ PARTITION BY

Se crea el Ranking, agrupando los datos por colinas, para esta consulta se usa la Window Function `PARTITION BY` y no la sentencia `GROUP BY`.

```
SELECT
  colonia,
  tipo,
  valor,
RANK()OVER (
PARTITION BY colonia
ORDER BY valor DESC
  )AS ranking_por_colonia
FROM app.datos;
```

En esta consulta tenemos como resultado el Rankig de las propiedades, agrupadas por Colinas y vemos los valores ordenados de forma descendente, del más caro al más barato por cada subgrupo, se puede observar que el grupo de las propiedades que están en la **Colina Ajusco** la propiedad **más cara** tiene un valor de **42.000** y la **más barata** un valor de **3.325**, pero en el grupo de propiedades de la **Colina Atlampa** la **más cara** es de **28.000** y la **más barata** de **2.975**.

	colonia text	tipo text	valor numeric	ranking_por_colonia bigint
1	Ajusco	Galpón/Depósito/Alm...	42000.0	1
2	Ajusco	Departamento	28000.0	2
3	Ajusco	Casa de Condominio	26250.0	3
4	Ajusco	Departamento	15750.0	4
5	Ajusco	Departamento	7700.0	5
6	Ajusco	Tienda/Salón	3325.0	6
7	Álvaro Obregón	Casa	5250.0	1

Colina Ajusco

	colonia text	tipo text	valor numeric	ranking_por_colonia bigint
141	Atlalilco	Conjunto Comercial/S...	2030.0	10
142	Atlampa	Galpón/Depósito/Alm...	28000.0	1
143	Atlampa	Casa Comercial	8750.0	2
144	Atlampa	Casa	8050.0	3
145	Atlampa	Casa	6300.0	4
146	Atlampa	Departamento	5775.0	5
147	Atlampa	Departamento	5250.0	6
148	Atlampa	Departamento	4830.0	7
149	Atlampa	Departamento	4550.0	8
150	Atlampa	Departamento	4375.0	9
151	Atlampa	Casa	3500.0	10
152	Atlampa	Departamento	3500.0	10
153	Atlampa	Departamento	3500.0	10
154	Atlampa	Departamento	3150.0	13
155	Atlampa	Departamento	2975.0	14

Colina Atlampa

Window Functions	
ROW_NUMBER()	Es la función de Ranking más "estricta", ya que no permite empates aunque el precio sea el mismo, simplemente asigna el número de fila que va en el consecutivo (1, 2, 3...)
RANKIG()	Se usa para saber la posición real, deja huecos basados en la cantidad de duplicados (ej. "Esta casa es la número 24 de toda la Colina")
DENSE_RANKING()	Se usa para saber el nivel de importancia, no se salta ningún puesto, mantiene el conteo compacto (ej. "Esta casa tiene el tercer precio más alto de la colonia")

🧱 BLOQUE 3 – Window Functions analíticas:

4️⃣ AVG() OVER()

La combinación de `AVG() OVER()`, permite tener el promedio grupal de la consulta, pero sin perder los datos individuales de cada registro, al agrupar los datos por el Tipo de propiedad usando `PARTITION BY`, se puede comparar que tan cerca o lejos esta el valor individual de cada promiedad con el promedio de la categoría de esa propiedad.

```
SELECT
  tipo,
  colonia,
  valor,
ROUND(
    AVG(valor)OVER(
```

```
        PARTITION BY tipo), 2
)AS promedio_por_tipo
FROM app.datos;
```

En el resultado de esta consulta se puede observar, que el Valor Promedio para las propiedades del **Tipo Casa es de 31.453,18**, pero al observar los valores individuales de cada fila, se aprecia que la propiedad **Casa de la Colina San Ángel** posee un valor de **87.500**, esto significa que está **por encia** del promedio de las propiedades de su categoría, en cambio que la propiedad de **Santa Marta Acatitla** está muy **por debajo**, ya que su valor es de **1.750**.

	tipo text	colonia text	valor numeric	promedio_por_tipo numeric
1	Casa	San Ángel	87500.0	31453.18
2	Casa	Santa Marta Acatitla	1750.0	31453.18

5️⃣ Comparar contra promedio general (ideal con CTE)

Usar la combinación **CTE + Window Function**, permite crear una tabla temporal, que luego puede ser usada para comparar los datos que están permanentes en la Base de Datos, ya que si no se usa la CTE, no es posible hacer la sentencia `WHERE valor> promedio_general`, pues según el flujo de ejecución de SQL, Primero filtraría las filas con el WHERE y depués calcularía el Promedio con la Window Function, pero, usando la funcionalidad del CTE se crea una tabla “real” de forma temporaria que permite ser usada para la comparación, pues la información del promedio está en esta tabla temporario, es como hacer un JOIN con una tabla que no existe realmente.

```
WITH base AS (
SELECT
*,
ROUND(
   AVG(valor)OVER (), 2
```

```
)AS promedio_general
FROM app.datos
)
SELECT*
FROM base
WHERE valor> promedio_general;
```

Fases de la consulta:

- **Fase 1 (CTE)**: El motor ejecuta la consulta interna, calcula el promedio para cada fila y "materializa" lógicamente este resultado.
- **Fase 2 (Consulta Principal)**: Ahora, para la consulta de afuera, `promedio_general` ya no es una función de ventana que se está calculando, sino una **columna física** dentro de la tabla temporal `base`.
- **El Filtro**: Al llegar al `WHERE` de la consulta principal, el valor ya está disponible y el motor puede filtrar sin problemas.

	tipo text	colonia text	habitaciones integer	area numeric	valor numeric	promedio_general numeric
1	Casa de Condominio	Santa Fe	5	750	77000.0	36883.98
2	Conjunto Comercial/Sala	Centro Histórico	0	695	122500.0	36883.98
3	Departamento	Condesa	4	243	45500.0	36883.98
4	Edificio Completo	Roma	0	536	98000.0	36883.98
5	Conjunto Comercial/Sala	Centro Histórico	0	1306	411390.0	36883.98
6	Conjunto Comercial/Sala	Centro Histórico	0	1170	491596.0	36883.98
7	Casa de Condominio	Santa Fe	5	1600	87500.0	36883.98

- [x] ~~Estudio del tema~~
- [x] ~~Ejecución de scripts~~
- [x] ~~Evidencia guardada~~
- [x] ~~Notas técnicas escritas~~