**Tutorial Query Planning** (video relacionado: 5.7 Query planning. Contenido relacionado: Query planning in Amazon Redshift)

Trabajaremos sobre la misma  [base de datos](https://drive.google.com/file/d/1psT9IvEwCIJ-TjILeHFaAwgtMHBI2CpS/view?usp=sharing) que utilizamos para explicar las bases de datos columnares. En esta oportunidad el data Manager te ha pedido que le ayudes a analizar la escalabilidad de un requerimiento del equipo de ventas y que tan demandante podría ser

**El requerimiento es:**

Calcular el monto promedio (en miles) para cada categoría de decisión (F, D, C, A y X) en el contexto bancario según las siguientes reglas:

* Si el objetivo es verdadero, el nombre\_tipo\_contrato es 'Cash loans' y el monto del crédito es superior a 50 000, la categoría de decisión es 'F'.
* Si el destino es verdadero, el nombre\_tipo\_contrato es 'Revolving loans' y el monto del crédito es superior a 9.000, la categoría de decisión es 'D'.
* Si el objetivo es falso, el nombre\_tipo\_contrato es 'Revolving loans' y el monto del crédito es superior a 100 000, la categoría de decisión es 'C'.
* Si el destino es falso, el nombre\_tipo\_contrato es 'Cash loans' y el monto del crédito es superior a 120 000, la categoría de decisión es 'A'.
* Para cualquier otro caso que no cumpla con las condiciones anteriores, la categoría de decisión es 'X'.

**Deberás analizar el costo en tiempo y en cantidad de $ que involucraría este requerimiento.**

**Solución:**

1. Primero debemos generar la consulta para resolver el problema

```sql

SELECT x.decision as decision, AVG(x.cantidad)/1000 as Monto\_K

FROM

(SELECT target AS Moroso,

name\_contract\_type AS Contrato,

code\_gender AS Genero,

amt\_credit AS Cantidad,

CASE

WHEN (target = true AND name\_contract\_type = 'Cash loans' AND amt\_credit>50000) THEN 'F'

WHEN (target = true AND name\_contract\_type = 'Revolving loans' AND amt\_credit> 9000) THEN 'D'

WHEN (target = false AND name\_contract\_type = 'Revolving loans' AND amt\_credit > 100000) THEN 'C'

WHEN (target = false AND name\_contract\_type = 'Cash loans' AND amt\_credit > 120000) THEN 'A'

ELSE 'X'

END AS Decision

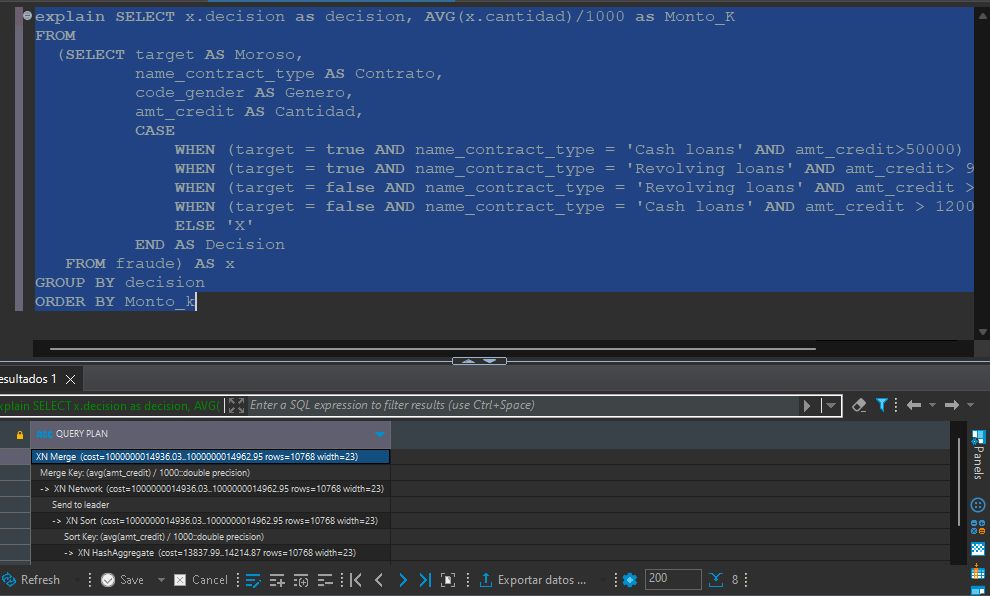
FROM fraude) AS x

GROUP BY decision

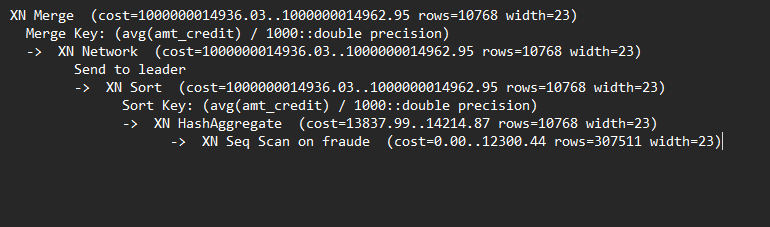
ORDER BY Monto\_k

```

1. Ahora podemos colocar la palabra Explain al inicio para analizar la escalabilidad de la consulta:



1. Ahora podemos seleccionar todo el texto que sale como resultado y lo podemos copiar a un bloc de notas:



O en su defecto ponerlo en un json:

XN Merge (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)

Merge Key: (avg(amt\_credit) / 1000::double precision)

-> XN Network (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)

Send to leader

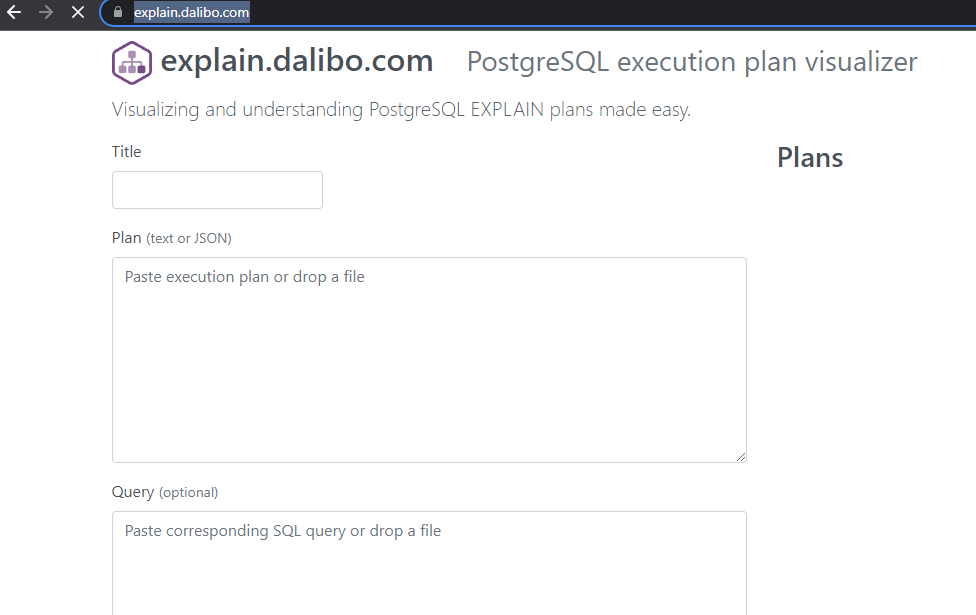
-> XN Sort (cost=1000000014936.03..1000000014962.95 rows=10768 width=23)

Sort Key: (avg(amt\_credit) / 1000::double precision)

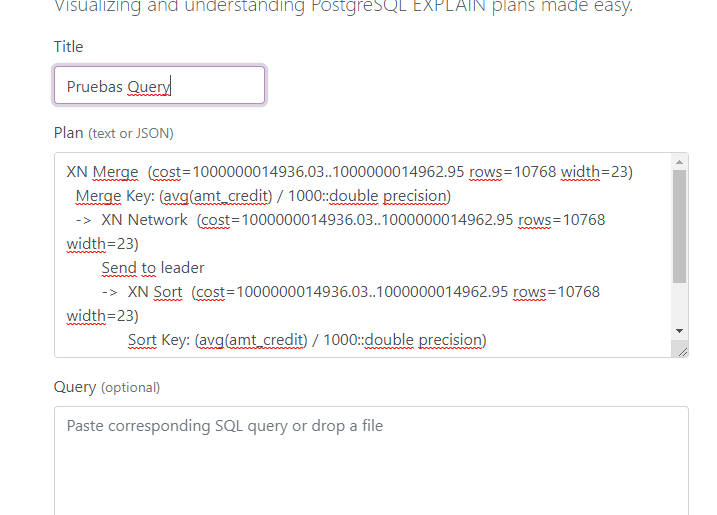
-> XN HashAggregate (cost=13837.99..14214.87 rows=10768 width=23)

-> XN Seq Scan on fraude (cost=0.00..12300.44 rows=307511 width=23)

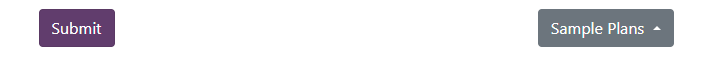
1. Luego podemos ir a la pagina explain.dalibo.com:



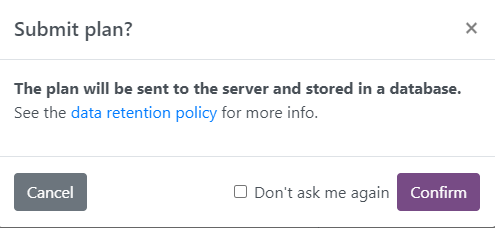
1. Podemos copiar todo el contenido del bloc de notas en la sección Plan. También podemos ponerle un nombre al plan que analizaremos e.g Prueba Query:



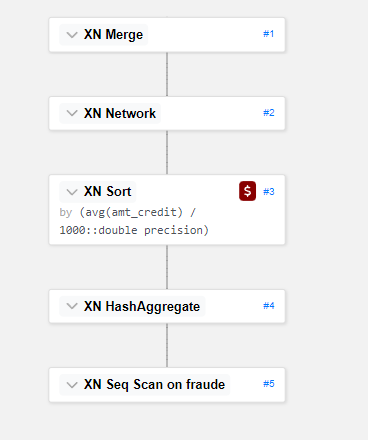
1. Bajamos y damos click en submit:



1. Luego presionamos confirm:

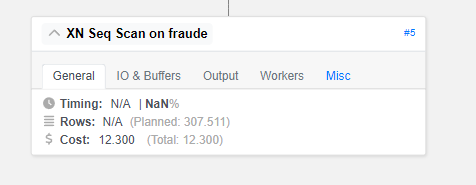


1. Obtendremos una salida como ésta:



1. El orden de ejecución es de abajo hacia arriba. Presionamos en XN Seq Scan on fraude.

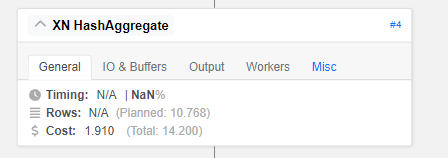
**Interpretación:** análisis secuencial de la tabla "fraude", lo que implica que la consulta está leyendo **todas las filas** de esta tabla. El costo estimado de este análisis se muestra 12300 unidades **(ahora bien ten presente este número porque si bien no es un costo real nos permite comparar qué operaciones son más demandantes)**. El número de filas estimado para este escaneo es 307511.



**ACLARACIÓN IMPORTANTE: El costo es acumulado en cada etapa no lo olvides (es decir cada sección no tiene un costo independiente)**

1. Luego podemos dar click en XN HashAggregate y veremos algo como esto

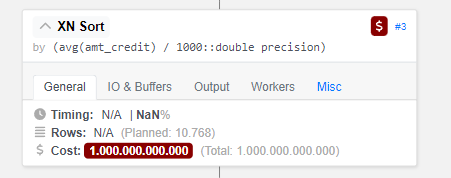
**Interpretación:** Después de escanear la tabla "fraude", los datos se pasan al paso HashAggregate. Este paso realiza operaciones de agregación con el GROUP BY. El costo estimado para este paso se muestra como 14200 **(apenas aumentó ligeramente)** y se espera producir 10.768 filas.



1. Ahora damos click en XN sort y veremos algo como esto:

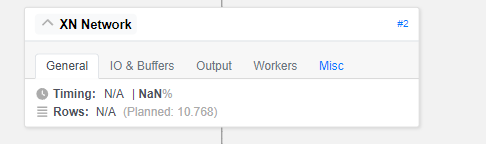
**Interpretación:** Siguiendo el paso HashAggregate, los datos se ordenan según la expresión "(avg(amt\_credit) / 1000::double precision)" del paso anterior Como puedes ver hay un color especial **(es el paso más**

**demandante)**. El costo estimado para esta operación aumenta hasta 1000000000000 y se realiza en las 10768 filas producidas por el paso anterior.:



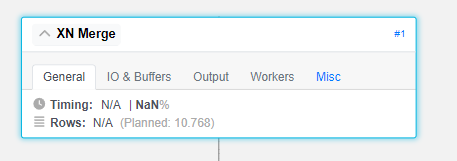
1. Ahora podemos analizar XN Network:

**Interpretación:** Luego, los datos ordenados se envían al nodo líder.En Redshift **(recuerdas el concepto de nodos líder y de cómputo, bueno acá es importante recordarlo)**, y este paso implica transferir los datos de los nodos de cálculo al nodo líder para su posterior procesamiento. El costo no se mueve del acumulado ~1000000000000



1. Finalmente analizamos XN Merge

**Interpretación:** Finalmente, los datos recibidos por el nodo líder se fusionana través de "(avg(amt\_credit) / 1000::double precision)".El costo no se mueve del acumulado ~1000000000000



**Anexo**

Lo que hicimos en Amazon Redshift también se puede realizar en PostgreSQL

**En postgre tenemos lo siguientes**

```sql

explain SELECT x.decision as decision, AVG(x.cantidad)/1000 as Monto\_K

FROM

(SELECT target AS Moroso,

name\_contract\_type AS Contrato,

code\_gender AS Genero,

amt\_credit AS Cantidad,

CASE

WHEN (target = true AND name\_contract\_type = 'Cash loans' AND amt\_credit>50000) THEN 'F'

WHEN (target = true AND name\_contract\_type = 'Revolving loans' AND amt\_credit> 9000) THEN 'D'

WHEN (target = false AND name\_contract\_type = 'Revolving loans' AND amt\_credit > 100000) THEN 'C'

WHEN (target = false AND name\_contract\_type = 'Cash loans' AND amt\_credit > 120000) THEN 'A'

ELSE 'X'

END AS Decision

FROM fraude) AS x

GROUP BY decision

ORDER BY Monto\_k

```

Seguro notarás que los resultados no son exactamente lo mismo que en Amazon Redshift, bueno tiene sentido ya que es un sistema análogo a OLTP y no OLAP:

"Sort (cost=12110.55..12139.87 rows=11728 width=40)"

" Sort Key: ((avg(fraude.amt\_credit) / '1000'::double precision))"

" -> Finalize HashAggregate (cost=10907.39..11317.87 rows=11728 width=40)"

" Group Key: (CASE WHEN (fraude.target AND ((fraude.name\_contract\_type)::text = 'Cash loans'::text) AND (fraude.amt\_credit > '50000'::double precision)) THEN 'F'::text WHEN (fraude.target AND ((fraude.name\_contract\_type)::text = 'Revolving loans'::text) AND (fraude.amt\_credit > '9000'::double precision)) THEN 'D'::text WHEN ((NOT fraude.target) AND ((fraude.name\_contract\_type)::text = 'Revolving loans'::text) AND (fraude.amt\_credit > '100000'::double precision)) THEN 'C'::text WHEN ((NOT fraude.target) AND ((fraude.name\_contract\_type)::text = 'Cash loans'::text) AND (fraude.amt\_credit > '120000'::double precision)) THEN 'A'::text ELSE 'X'::text END)"

" -> Gather (cost=9324.11..10848.75 rows=11728 width=64)"

" Workers Planned: 1"

" -> Partial HashAggregate (cost=8324.11..8675.95 rows=11728 width=64)"

" Group Key: CASE WHEN (fraude.target AND ((fraude.name\_contract\_type)::text = 'Cash loans'::text) AND (fraude.amt\_credit > '50000'::double precision)) THEN 'F'::text WHEN (fraude.target AND ((fraude.name\_contract\_type)::text = 'Revolving loans'::text) AND (fraude.amt\_credit > '9000'::double precision)) THEN 'D'::text WHEN ((NOT fraude.target) AND ((fraude.name\_contract\_type)::text = 'Revolving loans'::text) AND (fraude.amt\_credit > '100000'::double precision)) THEN 'C'::text WHEN ((NOT fraude.target) AND ((fraude.name\_contract\_type)::text = 'Cash loans'::text) AND (fraude.amt\_credit > '120000'::double precision)) THEN 'A'::text ELSE 'X'::text END"

" -> Parallel Seq Scan on fraude (cost=0.00..7419.67 rows=180889 width=40)"

