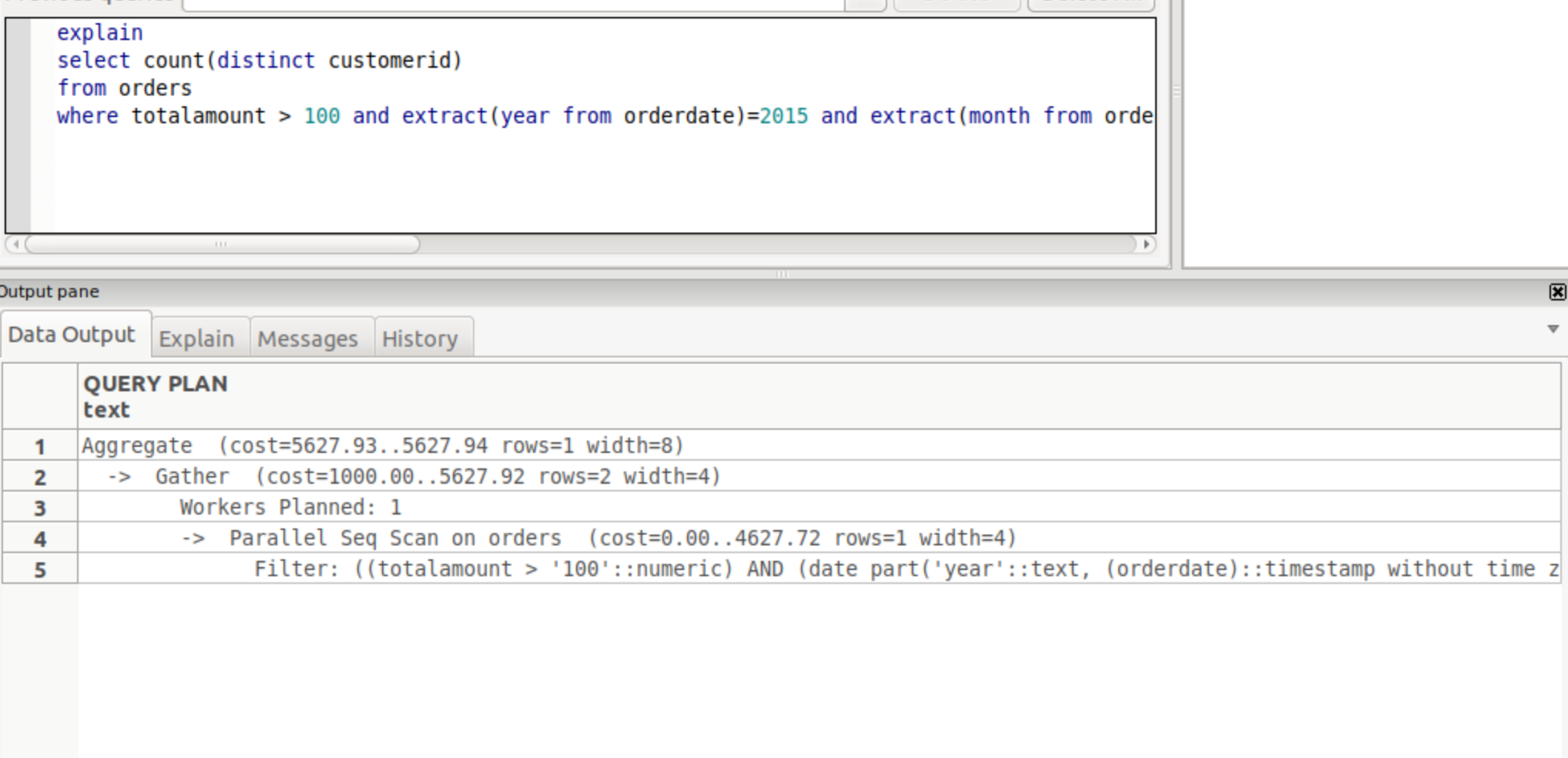
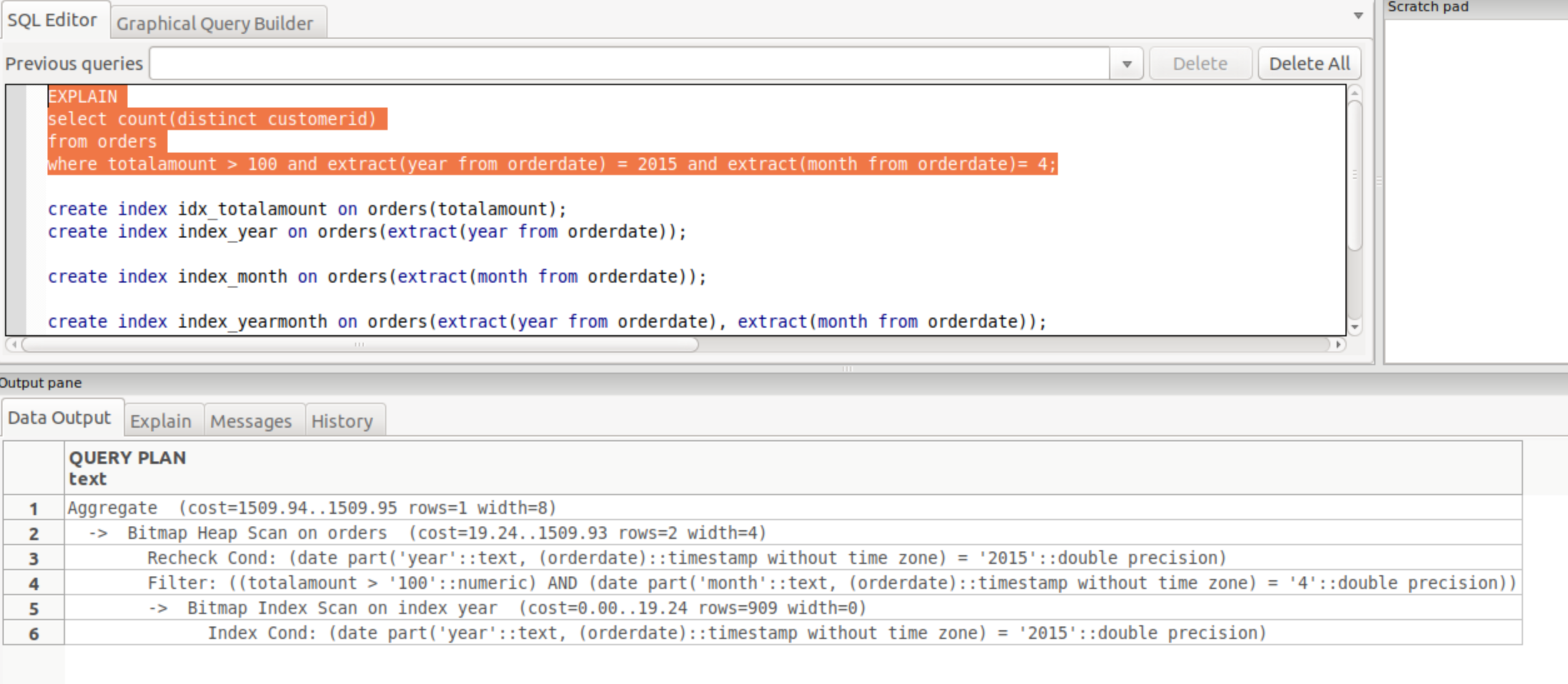
OPTIMIZACION

Se observa que la acción más costosa es el ‘seq scan’ pues este realiza un escaneo secuencial de la tabla de datos almacenada desde la primera fila hasta que la consulta se cumple.

Para mejorar el rendimiento creamos índices sobre esta tabla. Nuestra primera aproximación fue pensar en crear un índice para la selección de pedidos por el coste total, pero dado que más del 50% de los pedidos de la base de datos tenían un coste mayor de 100, el rendimiento no mejoraba mucho pues no descartábamos un gran número de filas, y con la creación de los siguientes índices (mucho más útiles) su mejora de rendimiento era imperceptible. Es por ello que lo descartamos.

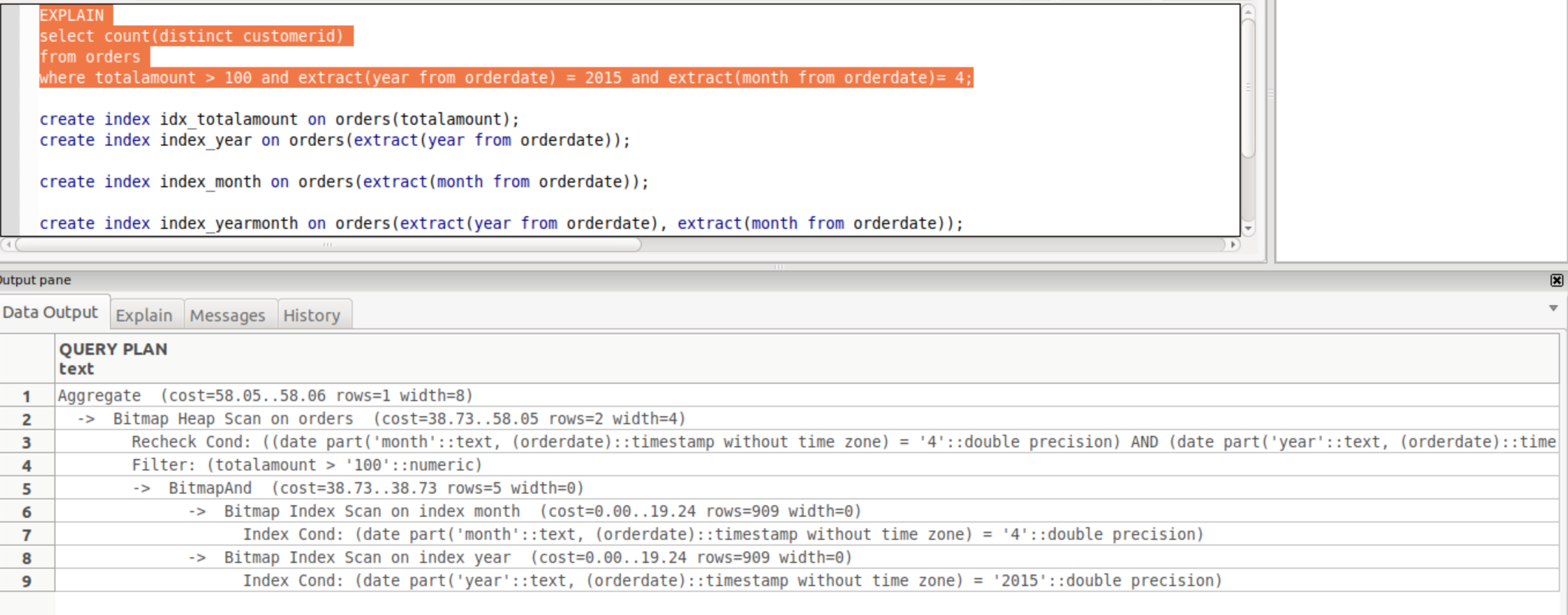
Lo siguiente que pensamos fue crear un índice para la operación de extracción del año en cuestión, esto es:

*create index index\_year on orders(extract(year from orderdate));*

**

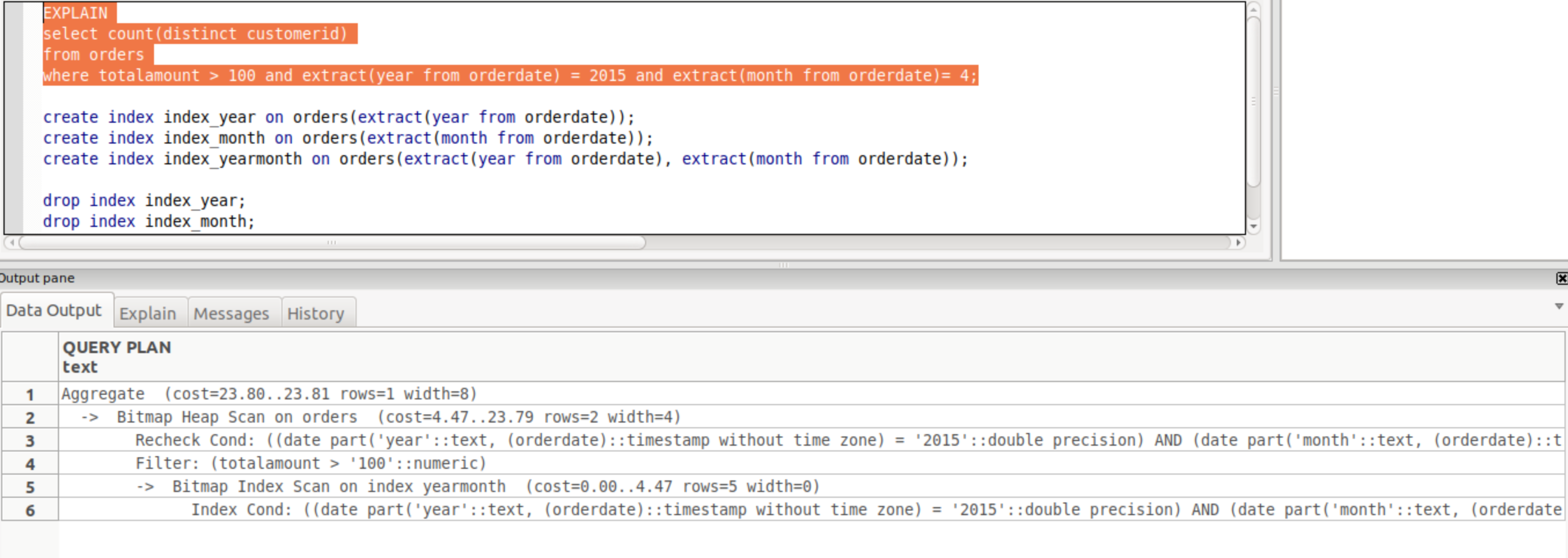
Al ejecutar de nuevo la query vimos una notable mejora en el rendimiento, pues el coste se reducía de 5627 a 1509. El cambio se observa en el cambio de ‘seq scan’ que se realiza en orders por un ‘bitmap heap scan’. Este se divide en dos pasos: la búsqueda de las filas que cumplan la condición en el índice creado y luego ‘traerlas’ del disco para mostrarlo. Traer las filas de una en una es más costoso que leerlas de manera secuencial, el motivo de que esto sea menos costoso es porque gracias al índice no tendremos que leer toda la tabla, pues el subconjunto de filas que tenemos que buscar es notablemente menor que el número total (aproximadamente un 12%).

Viendo esto decidimos probar a añadir otro índice en este caso para la extracción del mes, es decir:

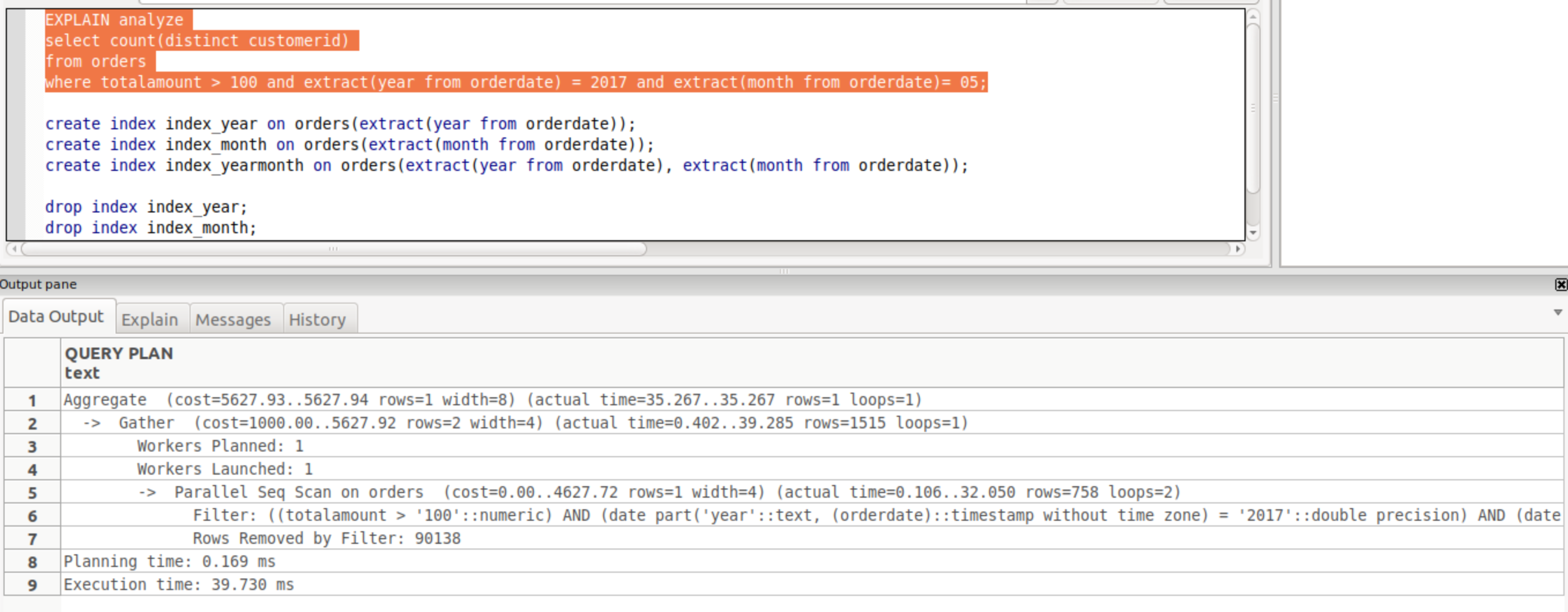
*create index index\_month on orders(extract(year from orderdate));*

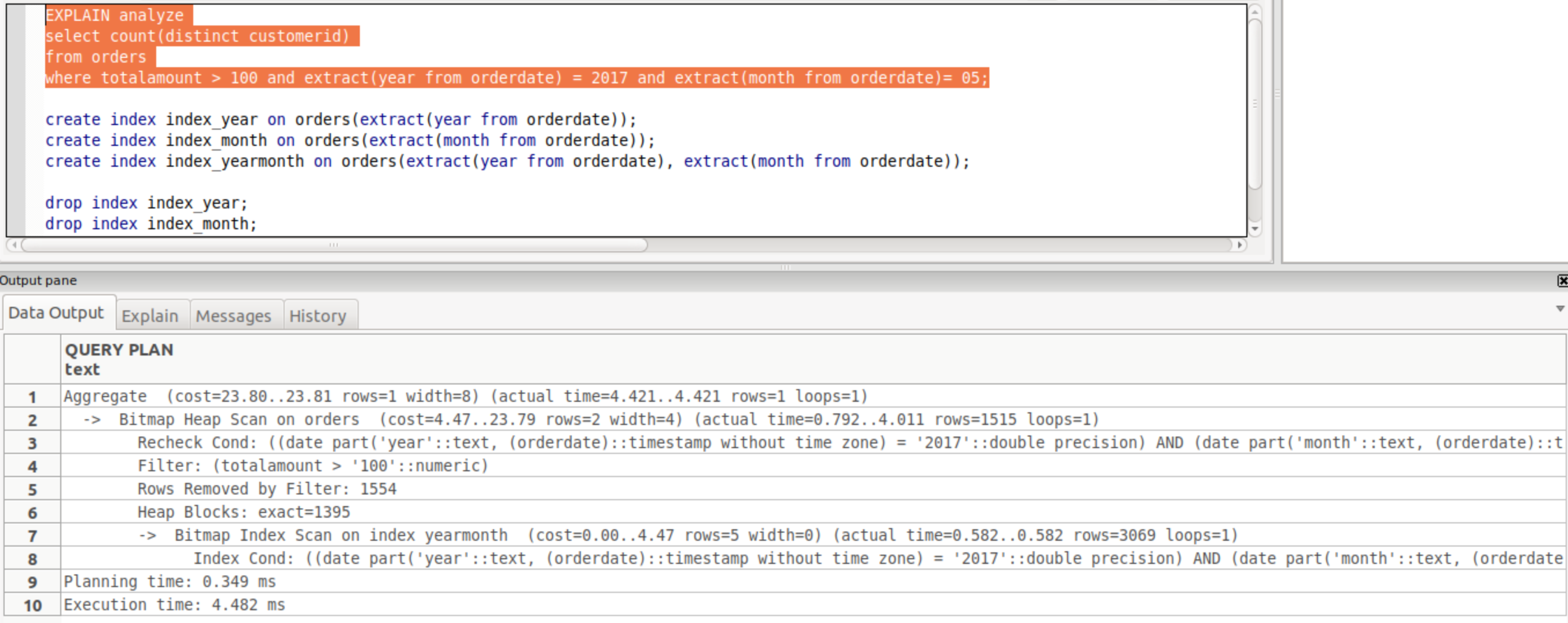
Observamos que el rendimiento de la consulta mejora de nuevo, pasando de un coste de 1509 a un coste total de 58. El razonamiento es el mismo que el anterior.

Viendo esto se nos ocurrió crear un índice más específico uniendo las dos situaciones anteriores en lugar de tenerlos por separados, es decir, un índice para la extracción de la fecha completa. De este modo nos lo imaginamos como un índice de un libro donde cada capítulo es una fecha, y en ellos están los pedidos que se realizaran ese día. Es claro que el número de filas que tendremos que ‘traer’ será menor, por lo que debería mejorar el rendimiento.

Así obtuvimos nuestro mejor resultado de rendimiento, pasando de 5627 a 23.80:

Haciendo uso de **EXPLAIN ANALYZE** podemos comprobar también una mejora en el tiempo de ejecución (a costa de tardar un poco más en tiempo de planificación pues ‘bitmap heap scan’ consta de dos pasos en su planificación) entre ejecutar la consulta sin índices, y con el último índice discutido (el de fecha completa):

**Sin índices:**

**Con índice:**