Trabajo 4 - Bases de Datos 2

Integrantes:

- Sebastián Arango Urrea
- Kevin Leonardo Arias Orrego
- Tomás Gutiérrez Orrego

Punto 1:

- ❖ 100 datos en s, 1.000 datos en p y 99.999 datos en sp, se eliminó: DELETE FROM sp WHERE sn = 'S2' AND pn = 'P3'.

Explain plan consulta 1:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1214664205

--------------------------------------------------------------------------------
------

| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)| Time
    |

--------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |    99 |  2376 |   158   (0)| 00:0
0:01 |

|*  1 |  FILTER               |             |       |       |            |
    |

|   2 |   TABLE ACCESS FULL   | S           |   100 |  2400 |     2   (0)| 00:0
0:01 |

|*  3 |   FILTER              |             |       |       |            |
    |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |    INDEX FAST FULL SCAN| SYS_C008429 |     2 |    10 |     2   (0)| 00:0
0:01 |

|*  5 |    INDEX UNIQUE SCAN  | SYS_C008430 |     1 |     9 |     1   (0)| 00:0
0:01 |

--------------------------------------------------------------------------------
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	158	0:00:01
	Filter		
Tabla p: 1000	Table Access Full (s)	2	0:00:01
	Filter		
Tabla sp: 99999	Index Fast Full Scan	2	0:00:01
	Index Uniqune Scan	1	0:00:01

Análisis:
Orden de las operaciones:
Index Unique Scan
Index Fast Full Scan
Table Access Full (s)
Filter
Filter
Select Statement

Costo total bajo (158): El costo de la consulta es relativamente bajo, lo que indica que las operaciones son eficientes y manejan cantidades moderadas de datos.
Index Unique Scan: Esta operación tiene un costo bajo (**1**), lo que sugiere que se está accediendo de manera efectiva a registros únicos en la tabla.
Index Fast Full Scan: Tiene un costo de **2**, lo que indica que se está escaneando rápidamente todos los valores en un índice. Esto es positivo para el rendimiento, ya que evita escaneos completos de la tabla.
Table Access Full (s): El escaneo completo de la tabla s también tiene un costo bajo (**2**), lo que sugiere que la tabla puede ser pequeña o que se está filtrando adecuadamente.

TKPROF consulta 1:

```
SELECT *
FROM s
WHERE NOT EXISTS(SELECT *
FROM p
WHERE NOT EXISTS (SELECT *
FROM sp
WHERE sp.pn = p.pn
AND
sp.sn = s.sn
)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0         38         14           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.19       0.33          0     103168          0          99
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.19       0.34          0     103206         14          99
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	1.042,626
p = 1000	i / j	12,375
sp = 99999	k / (f+g)	0,000

Análisis:

Un valor de **1.042,626** implica una posible ineficiencia en el acceso a los datos, indicando que se podrían optimizar las consultas o el uso de índices para reducir la cantidad de LIOs necesarias para procesar filas.
Un valor de **12,375** indica que por cada fetch, se están retornando aproximadamente **12.38 filas**, lo que sugiere que el sistema está funcionando eficientemente en la recuperación de datos. Este aspecto es favorable en términos de rendimiento, ya que reduce el número total de fetches necesarios para obtener un conjunto de datos.
La ausencia de lecturas de disco es un indicativo positivo, sugiriendo que los datos requeridos se están manejando eficientemente en la memoria caché, lo que minimiza la latencia asociada con las lecturas de disco.

Explain plan consulta 2:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 413911273

--------------------------------------------------------------------------------
------

| Id  | Operation             | Name        | Rows  | Bytes | Cost (%CPU)| Time
    |

--------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |   100 |  2400 |     8   (0)| 00:0
0:01 |

|*  1 |  FILTER               |             |       |       |            |
    |

|   2 |   TABLE ACCESS FULL   | S           |   100 |  2400 |     2   (0)| 00:0
0:01 |

|   3 |   SORT AGGREGATE      |             |     1 |     4 |            |
    |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |    INDEX RANGE SCAN   | SYS_C008430 |  1000 |  4000 |     6   (0)| 00:0
0:01 |

|   5 |   SORT AGGREGATE      |             |     1 |       |            |
    |

|   6 |    INDEX FAST FULL SCAN| SYS_C008429 |  1000 |       |     3   (0)| 00:0
0:01 |

--------------------------------------------------------------------------------
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	8	0:00:01
	Filter		
Tabla p: 1000	Table Access Full (s)	2	0:00:01
	Sort Aggregate		
Tabla sp: 99999	Index Fast Full Scan	6	0:00:01
	Sort Aggregate		
	Index Uniqune Scan	3	0:00:01

Análisis:
Orden de las operaciones:
Index Unique Scan
Index Fast Full Scan
Table Access Full (s)
Sort Aggregate
Sort Aggregate
Filter
Select Statement

Costo total muy bajo (8): El costo de la consulta es extremadamente bajo, lo que indica que las operaciones son altamente eficientes y que se manejan cantidades mínimas de datos.
Index Unique Scan: Tiene un costo bajo (**3**), lo que sugiere que se está accediendo eficientemente a registros únicos.
Index Fast Full Scan: Esta operación tiene un costo de **6**, lo que indica que se está escaneando rápidamente todos los valores en un índice. Es un método efectivo para acceder a un conjunto de datos sin realizar un escaneo completo de la tabla.
Table Access Full (s): El escaneo completo de la tabla s tiene un costo muy bajo (**2**), lo que es aceptable dado que la tabla puede ser pequeña o contener pocos registros relevantes para la consulta.
Sort Aggregate: Se realizan dos operaciones de agregación y ordenamiento. El costo bajo sugiere que no se están procesando grandes volúmenes de datos en estas etapas.

TKPROF consulta 2:

```
SELECT *
FROM s
WHERE (SELECT COUNT(*)
FROM sp
WHERE s.sn = sp.sn) = (SELECT COUNT(*) FROM p)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0         76          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.03       0.03          0        590          0          99
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.03       0.04          0        666          0          99
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	6,727
p = 1000	i / j	12,375
sp = 99999	k / (f+g)	0,000

Análisis:

Este valor indica que, en promedio, hay **6.73 LIOs** (Lecturas Lógicas de Entrada/Salida) por cada fila procesada. Un valor tan alto sugiere que el sistema está realizando un número excesivo de lecturas lógicas en comparación con el número de filas que realmente se están procesando.

Explain plan consulta 3:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 613308256

--------------------------------------------------------------------------------
------------

| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)
| Time     |

--------------------------------------------------------------------------------
------------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |    31 |    62  (13)
| 00:00:01 |

|   1 |  NESTED LOOPS                |             |     1 |    31 |    62  (13)
| 00:00:01 |

|   2 |   NESTED LOOPS               |             |     1 |    31 |    62  (13)
| 00:00:01 |

|   3 |    VIEW                      | VW_NSO_1    |     1 |     7 |    60  (12)
| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |     FILTER                   |             |       |       |
|          |

|   5 |      HASH GROUP BY           |             |     1 |     4 |    60  (12)
| 00:00:01 |

|   6 |       TABLE ACCESS FULL      | SP          | 99999 |   390K|    54   (2)
| 00:00:01 |

|   7 |      SORT AGGREGATE          |             |     1 |       |

|   8 |       INDEX FAST FULL SCAN   | SYS_C008429 |  1000 |       |     3   (0)
| 00:00:01 |

|*  9 |    INDEX UNIQUE SCAN         | SYS_C008426 |     1 |       |     0   (0)
| 00:00:01 |

|  10 |   TABLE ACCESS BY INDEX ROWID| S           |     1 |    24 |     1   (0)
| 00:00:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	62	0:00:01
	Nested Loops	62	
	Nested Loops	62	0:00:01
	View	60	
Tabla p: 1000	Filter		
	Hash Group By	60	0:00:01
	Table Access Full (s)	54	0:00:01
	Sort Aggregate		
Tabla sp: 99999	Index Fast Full Scan	3	0:00:01
	Index Uniqune Scan	0	0:00:01
	Table Access By Index Rowid (s)	1	0:00:01

Análisis:
Orden de las operaciones:
Index Unique Scan
Index Fast Full Scan
Table Access By Index Rowid (s)
Table Access Full (s)
Sort Aggregate
Hash Group By
Filter
View
Nested Loops
Nested Loops
Select Statement

Costo total bajo (62): El costo de la consulta es relativamente bajo, lo que indica que las operaciones son eficientes y se manejan volúmenes de datos razonables.
Index Unique Scan: Esta operación tiene un costo muy bajo (**0**), lo que indica que se está utilizando de manera efectiva para acceder a registros únicos.
Index Fast Full Scan: Tiene un costo bajo (**3**), lo que significa que se escanean rápidamente todos los valores en un índice, lo que es favorable para el rendimiento.
Table Access By Index Rowid (s): El acceso a la tabla mediante el Rowid también tiene un costo muy bajo (**1**), lo que muestra que la consulta está optimizada para acceder a filas específicas.

Table Access Full (s): El escaneo completo de la tabla s tiene un costo moderado (**54**), lo que sugiere que podría haber una oportunidad para optimizar el acceso a esta tabla.
Sort Aggregate y Hash Group By: Ambas operaciones tienen costos bajos, lo que indica que no se están manejando grandes volúmenes de datos en estas etapas, lo que es positivo para el rendimiento general.
Nested Loops: Estas operaciones de bucle anidado pueden ser costosas si se manejan grandes conjuntos de datos, aunque en este caso, el costo total de la consulta sigue siendo bajo.

TKPROF consulta 3:

```
SELECT *
FROM s
WHERE sn IN (SELECT sn
FROM sp
GROUP BY sn
HAVING COUNT(*) = (SELECT COUNT(*)
FROM p)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.02       0.01          0         24          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.00       0.00          0        206          0          99
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.02       0.02          0        230          1          99
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	2,333
p = 1000	i / j	12,375
sp = 99999	k / (f+g)	0,000

Análisis:

Este valor indica que hay aproximadamente **2.33 LIOs** (Lecturas Lógicas de Entrada/Salida) por cada fila procesada. Aunque este valor es significativamente mejor que el anterior (6.727), aún sugiere que hay un número considerable de LIOs en comparación con las filas procesadas.

Explain plan consulta 4:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3039656237

--------------------------------------------------------------------------------
------

| Id  | Operation              | Name        | Rows  | Bytes | Cost (%CPU)| Time
    |

--------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |             |  1000 | 28000 |    63  (13)| 00:0
0:01 |

|*  1 |  FILTER                |             |       |       |            |
    |

|   2 |   HASH GROUP BY        |             |  1000 | 28000 |    63  (13)| 00:0
0:01 |

|*  3 |    HASH JOIN           |             | 99999 |  2734K|    57   (4)| 00:0
0:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |     TABLE ACCESS FULL  | S           |   100 |  2400 |     2   (0)| 00:0
0:01 |

|   5 |     TABLE ACCESS FULL  | SP          | 99999 |   390K|    54   (2)| 00:0
0:01 |

|   6 |   SORT AGGREGATE       |             |     1 |       |            |
    |

|   7 |    INDEX FAST FULL SCAN| SYS_C008429 |  1000 |       |     3   (0)| 00:0
0:01 |
--------------------------------------------------------------------------------
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	63	0:00:01
	Filter		
Tabla p: 1000	Hash Group By	63	0:00:01
	Hash Join	57	0:00:01
	Table Acces Full (s)	2	0:00:01
Tabla sp: 99999	Table Acces Full (sp)	54	0:00:01
	Sort Aggregate		
	Index Fast Full Scan	3	0:00:01

Análisis:
Orden de las operaciones:
Index Fast Full Scan
Table Access Full (s)
Table Access Full (sp)
Sort Aggregate
Hash Join
Hash Group By
Filter
Select Statement

Costo total bajo (63): El costo de la consulta es bajo, lo que indica que las operaciones son eficientes y que se están manejando volúmenes de datos razonables.
Index Fast Full Scan: Esta operación tiene un costo bajo (**3**), lo que sugiere que se está escaneando rápidamente todos los valores en un índice. Esto es una buena práctica para mejorar el rendimiento de la consulta.
Table Access Full (s) y **Table Access Full (sp)**: Ambos escaneos completos tienen costos bajos (**2** y **54**, respectivamente), lo que indica que no son excesivos y que las tablas involucradas no son demasiado grandes.
Sort Aggregate: Aunque esta operación puede ser costosa, en este caso no se observa un costo significativo, lo que sugiere que no se están procesando grandes volúmenes de datos.
Hash Join: Esta operación tiene un costo moderado (**57**), indicando que se están combinando conjuntos de datos, pero aún en un rango manejable.
Hash Group By: Agrupa los resultados después de las operaciones anteriores y tiene un costo que coincide con el total de la consulta, lo que sugiere que el agrupamiento es una operación clave.

TKPROF consulta 4:

```
SELECT s.sn, s.snombre, s.ciudad
FROM s
JOIN sp ON s.sn = sp.sn
GROUP BY s.sn, s.snombre, s.ciudad
HAVING COUNT(sp.pn) = (SELECT COUNT(p.pn) FROM p)
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.02	0.03	0	6	1	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	8	0.21	0.21	0	206	0	99
total	10	0.23	0.24	0	212	1	99

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	2,152
p = 1000	i / j	12,375
sp = 99999	k / (f+g)	0,000

Análisis:

Este valor indica que hay aproximadamente **2.15 LIOs** (Lecturas Lógicas de Entrada/Salida) por cada fila procesada. Este es un valor aún más favorable que los anteriores (2.333), lo que sugiere que el sistema está mejorando en la eficiencia de acceso a datos.

- 1.000 datos en s, 1.000 datos en p y 999.998 datos en sp, se eliminaron:
  DELETE FROM sp WHERE sn = 'S2' AND pn = 'P3'.
  DELETE FROM sp WHERE sn = 'S5' AND pn = 'P4'.

Explain plan consulta 1:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 4233794398

--------------------------------------------------------------------------------
------

| Id  | Operation              | Name        | Rows  | Bytes | Cost (%CPU)| Time
    |

--------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |             |   968 | 30008 |  1603   (1)| 00:0
0:01 |

|*  1 |  FILTER                |             |       |       |            |
    |

|   2 |   TABLE ACCESS FULL    | S           |  1000 | 31000 |     3   (0)| 00:0
0:01 |

|*  3 |   FILTER               |             |       |       |            |
    |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |    INDEX FAST FULL SCAN| SYS_C008447 |  1000 |  7000 |     2   (0)| 00:0
0:01 |

|*  5 |    INDEX UNIQUE SCAN   | SYS_C008448 |     1 |    14 |     1   (0)| 00:0
0:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	1603	0:00:01
	Filter		
Tabla p: 1000	Table Access Full (s)	3	0:00:01
	Filter		
Tabla sp: 999998	Index Fast Full Scan	2	0:00:01
	Index Uniqune Scan	1	0:00:01

Análisis:
Orden de las operaciones:
Index Unique Scan
Index Fast Full Scan
Table Access Full (s)
Filter
Filter
Select Statement

Costo total moderado (1603): El costo de la consulta es moderado, lo que sugiere que, aunque las operaciones son eficientes, puede haber ciertas áreas de mejora.
Index Unique Scan: Esta operación tiene un costo muy bajo (**1**) y es eficiente para acceder a un único registro en una tabla, lo que indica que se está utilizando un índice de manera efectiva.
Index Fast Full Scan: También tiene un costo bajo (**2**), lo que significa que se está escaneando rápidamente todos los valores en un índice. Esto es beneficioso, pero si se utiliza para grandes volúmenes de datos, podría tener un impacto en el costo total.
Table Access Full (s): El escaneo completo de la tabla s tiene un costo bajo (**3**), lo que es aceptable si la tabla no es grande.

TKPROF consulta 1:

```
SELECT *
FROM s
WHERE NOT EXISTS(SELECT *
FROM p
WHERE NOT EXISTS (SELECT *
FROM sp
WHERE sp.pn = p.pn
AND
sp.sn = s.sn
)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.02       0.01          0         40          4           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      2.56       2.92       4492    1063699          0         998
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      2.58       2.93       4492    1063739          4         998
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	1.065,875
p = 1000	i / j	14,676
sp = 999998	k / (f+g)	0,004

Análisis:

El valor del **LIOs** sobre filas procesadas es extremadamente alto y sugiere que el sistema está realizando un número desproporcionado de lecturas lógicas en comparación con las filas que realmente se procesan.
Este valor indica que por cada fetch se están retornando aproximadamente **14.68 filas**. Este es un ratio excelente y demuestra que el sistema está maximizando la recuperación de filas en cada operación de fetch, lo que reduce el número total de fetches necesarios.
Un valor de **0,004** indica que hay muy pocas lecturas de disco en relación con las LIOs. Aunque este valor es bajo y sugiere una buena eficiencia en la gestión de datos, no es completamente cero, lo que podría indicar que hay algunas áreas donde se podría trabajar para mejorar aún más el acceso a datos y reducir las lecturas de disco.

Explain plan consulta 2:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3241009236

--------------------------------------------------------------------------------
---
| Id  | Operation           | Name        | Rows  | Bytes | Cost (%CPU)| Time
  |

--------------------------------------------------------------------------------
---


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |             |  1000 | 31000 |    23   (0)| 00:00:0
1 |

|*  1 |  FILTER             |             |       |       |            |
  |

|   2 |   TABLE ACCESS FULL | S           |  1000 | 31000 |     3   (0)| 00:00:0
1 |

|   3 |   SORT AGGREGATE    |             |     1 |     7 |            |
  |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |    INDEX RANGE SCAN | SYS_C008448 |  8112 | 56784 |    20   (0)| 00:00:0
1 |

|   5 |   SORT AGGREGATE    |             |     1 |       |            |
  |

|   6 |    TABLE ACCESS FULL| P           |  1000 |       |     3   (0)| 00:00:0
1 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	23	0:00:01
	Filter		
Tabla p: 1000	Table Access Full (s)	3	0:00:01
	Sort Aggregate		
Tabla sp: 999998	Index Range Scan	20	0:00:01
	Sort Aggregate		
	Table Access Full (p)	3	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Index Range Scan
Sort Aggregate
Sort Aggregate
Filter
Select Statement

Costo total bajo (23): El costo de la consulta es extremadamente bajo, lo que indica que las operaciones se están llevando a cabo de manera muy eficiente y se manejan pocos datos.
Table Access Full (s): El escaneo completo de la tabla s tiene un costo bajo (**3**), lo que es aceptable, indicando que la tabla probablemente no es muy grande.
Index Range Scan: Esta operación tiene un costo de **20**, lo que sugiere que el uso del índice es eficiente para acceder a un rango específico de datos, mejorando así el rendimiento de la consulta.
Sort Aggregate: Se realizan dos operaciones de agregación y ordenamiento. Aunque estas pueden ser costosas, el bajo costo indica que no se están manejando grandes volúmenes de datos en estas operaciones.

TKPROF consulta 2:

```
SELECT *
FROM s
WHERE (SELECT COUNT(*)
FROM sp
WHERE s.sn = sp.sn) = (SELECT COUNT(*) FROM p)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0         77          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.17       0.16          3       6837          0         998
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.17       0.16          3       6914          0         998
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	6,928
p = 1000	i / j	14,676
sp = 999998	k / (f+g)	0,0004

Análisis:

Aunque es una mejora en comparación con el primer valor de más de mil, sigue siendo relativamente alto, lo que sugiere que el sistema aún está realizando un número significativo de lecturas lógicas en relación con las filas procesadas.
Un valor de **0,0004** indica que hay muy pocas lecturas de disco en relación con las LIOs. Esto significa que la mayoría de las operaciones de lectura están siendo gestionadas en memoria, lo que es ideal para el rendimiento.

Explain plan consulta 3:

```
| Id  | Operation             | Name     | Rows  | Bytes | Cost (%CPU)| Time
 |

-------------------------------------------------------------------------------
--


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |          |  1000 | 38000 |   622  (11)| 00:00:01
 |

|*  1 |  HASH JOIN SEMI       |          |  1000 | 38000 |   622  (11)| 00:00:01
 |

|   2 |   TABLE ACCESS FULL   | S        |  1000 | 31000 |     3   (0)| 00:00:01
 |

|   3 |   VIEW                | VW_NSO_1 |   811K|  5545K|   614  (10)| 00:00:01
 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------

|*  4 |    FILTER             |          |       |       |            |
 |

|   5 |     HASH GROUP BY     |          |   811K|  5545K|   614  (10)| 00:00:01
 |

|   6 |      TABLE ACCESS FULL| SP       |   811K|  5545K|   563   (2)| 00:00:01
 |

|   7 |     SORT AGGREGATE    |          |     1 |       |            |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
 |

|   8 |      TABLE ACCESS FULL| P        |  1000 |       |     3   (0)| 00:00:01
 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	622	0:00:01
	Hash Join Semi	622	0:00:01
	Table Access Full (s)	3	0:00:01
Tabla p: 1000	View	614	0:00:01
	Filter		
	Hash Group By	614	0:00:01
Tabla sp: 999998	Table Access Full (sp)	563	0:00:01
	Sort Aggregate		
	Table Access Full (p)	3	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Table Access Full (sp)
View
Hash Group By
Filter
Hash Join Semi
Select Statement

Costo total bajo (622): El costo total de la consulta es relativamente bajo, lo que sugiere que se están manejando eficientemente los recursos y la cantidad de datos.
Múltiples Table Access Full: Se realizan escaneos completos de las tablas s, sp, y p. El escaneo completo de sp tiene un costo considerable (**563**), lo que sugiere que podría beneficiarse de índices para optimizar el acceso.
View: La vista tiene un costo notable (**614**), lo que indica que puede estar manejando un conjunto considerable de datos o que contiene cálculos complejos.
Hash Group By: Esta operación tiene un costo equivalente al de la vista, lo que sugiere que el agrupamiento de datos puede ser significativo en el total de la consulta.
Filter: La operación de filtro no tiene un costo visible, lo que indica que se está aplicando de manera eficiente, posiblemente limitando los resultados después de las agrupaciones.
Hash Join Semi: Esta operación tiene un costo alto, lo que indica que está combinando conjuntos de datos importantes. Es una operación crítica, ya que filtra las filas de la tabla principal según las coincidencias en la segunda tabla.

TKPROF consulta 3:

```
SELECT *
FROM s
WHERE sn IN (SELECT sn
FROM sp
GROUP BY sn
HAVING COUNT(*) = (SELECT COUNT(*)
FROM p)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0         24          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.11       0.10          0       2059          0         998
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.12       0.12          0       2083          1         998
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	2,088
p = 1000	i / j	14,676
sp = 999998	k / (f+g)	0,000

Análisis:

Este es un avance positivo y sugiere que el sistema está realizando un número más razonable de lecturas lógicas en comparación con las filas procesadas.

Explain plan consulta 4:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2511177261

--------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |   811K|    29M|   622  (11)| 00:00:01 |
|*  1 |  FILTER              |      |       |       |            |          |
|   2 |   HASH GROUP BY      |      |   811K|    29M|   622  (11)| 00:00:01 |
|*  3 |    HASH JOIN         |      |   811K|    29M|   572   (3)| 00:00:01 |
|   4 |     TABLE ACCESS FULL| S    |  1000 | 31000 |     3   (0)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| SP   |   811K|  5545K|   563   (2)| 00:00:01 |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   6 |   SORT AGGREGATE     |      |     1 |       |            |          |
|   7 |    TABLE ACCESS FULL | P    |  1000 |       |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	622	0:00:01
	Filter		
Tabla p: 1000	Hash Group By	622	0:00:01
	Hash Join	572	0:00:01
	Table Acces Full (s)	3	0:00:01
Tabla sp: 999998	Table Acces Full (sp)	563	0:00:01
	Sort Aggregate		
	Table Access Full (p)	3	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Table Access Full (sp)
Sort Aggregate
Hash Join
Hash Group By
Filter
Select Statement

Costo total bajo (622): El costo total de la consulta es bajo, lo que indica que las operaciones realizadas son eficientes y manejan una cantidad razonable de datos.
Múltiples Table Access Full: Se realizan escaneos completos de las tablas s, sp, y p. El costo del escaneo de sp es notablemente más alto (**563**), lo que sugiere que podría ser una buena oportunidad para optimizar su acceso mediante índices.
Sort Aggregate: Esta operación no tiene un costo visible, lo que sugiere que no requiere procesamiento adicional significativo, posiblemente porque no se están manejando grandes volúmenes de datos en esta etapa.
Hash Join: Esta operación se utiliza para combinar conjuntos de datos, y aunque su costo es moderado (**572**), esto puede indicar que está combinando conjuntos de datos razonablemente grandes.
Hash Group By: Agrupa los resultados antes de aplicar los filtros. El costo coincide con el total de la consulta, lo que sugiere que el agrupamiento es una operación clave.

TKPROF consulta 4:

```
SELECT s.sn, s.snombre, s.ciudad
FROM s
JOIN sp ON s.sn = sp.sn
GROUP BY s.sn, s.snombre, s.ciudad
HAVING COUNT(sp.pn) = (SELECT COUNT(p.pn) FROM p)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          6          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.40       0.42          0       2059          0         998
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.40       0.43          0       2065          1         998
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	2,070
p = 1000	i / j	14,676
sp = 999998	k / (f+g)	0,000

Análisis:

Este es un resultado positivo y muestra una ligera mejora en comparación con el valor anterior de 2.09 LIOs.

- 100.000 datos en s, 100 datos en p y 10.000.000 datos en sp:

Explain plan consulta 1:

```
| Id  | Operation              | Name        | Rows  | Bytes | Cost (%CPU)| Time
     |

--------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |             |   117K|  3562K|   782K  (1)| 00:0
0:31 |

|*  1 |  FILTER                |             |       |       |            |
     |

|   2 |   TABLE ACCESS FULL    | S           |   117K|  3563K|   145   (2)| 00:0
0:01 |

|*  3 |   FILTER               |             |       |       |            |
     |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|   4 |    INDEX FAST FULL SCAN| SYS_C008510 |   100 |   700 |     2   (0)| 00:0
0:01 |

|*  5 |    INDEX UNIQUE SCAN   | SYS_C008511 |     1 |    14 |     1   (0)| 00:0
0:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	782K	0:00:31
	Filter		
Tabla p: 100	Table Access Full (s)	145	0:00:01
	Filter		
Tabla sp: 10000000	Index Fast Full Scan	2	0:00:01
	Index Unique Scan	1	0:00:01

Análisis:
Orden de las operaciones:
Index Unique Scan
Index Fast Full Scan
Table Access Full (s)
Filter
Filter
Select Statement

Costo total alto (782K): El costo total de la consulta es considerablemente alto, lo que indica que se están manejando grandes volúmenes de datos o que se están realizando operaciones costosas.
Index Unique Scan: Esta operación tiene un costo muy bajo (**1**) y es eficiente para acceder a un único registro en una tabla, lo que sugiere que se está utilizando un índice de manera efectiva.
Index Fast Full Scan: Esta operación también tiene un costo muy bajo (**2**) y se utiliza para escanear rápidamente todos los valores en un índice. Aunque es eficiente, si se está utilizando para grandes volúmenes de datos, el costo total puede incrementarse.
Table Access Full (s): El escaneo completo de la tabla s tiene un costo moderado (**145**), lo que es aceptable, pero puede ser ineficiente si la tabla es grande.
Filtros: Se aplican dos filtros, pero no tienen un costo visible asociado. Sin embargo, estos son esenciales para reducir el conjunto de resultados y optimizar la consulta.

TKPROF consulta 1:

```
SELECT *
FROM s
WHERE NOT EXISTS(SELECT *
FROM p
WHERE NOT EXISTS (SELECT *
FROM sp
WHERE sp.pn = p.pn
AND
sp.sn = s.sn
)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0        174          4           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668     25.79      32.22        369   10929739          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670     25.81      32.24        369   10929913          4      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	10,930
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,00003

Análisis:

Para ser la lectura con mayor volumen de datos el valor de LIOs sobre filas es bastante aceptable aunque con gran posibilidad de mejora.
Este valor de **14,997** indica que por cada fetch se están retornando aproximadamente **14.99 filas**. Este es un resultado positivo y refleja una buena eficiencia en la recuperación de datos, lo que ayuda a minimizar el número total de fetches necesarios.
Un valor de **0,00003** indica que hay muy pocas lecturas de disco en relación con las LIOs. Esto sugiere que la mayoría de las lecturas se están gestionando en memoria, lo cual es ideal.

Explain plan consulta 2:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2559858421

--------------------------------------------------------------------------------
---
| Id  | Operation           | Name        | Rows  | Bytes | Cost (%CPU)| Time
  |

--------------------------------------------------------------------------------
---


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |             |   117K|  3563K|   278   (1)| 00:00:0
1 |

|*  1 |  FILTER             |             |       |       |            |
  |

|   2 |   TABLE ACCESS FULL | S           |   117K|  3563K|   145   (2)| 00:00:0
1 |

|   3 |   SORT AGGREGATE    |             |     1 |     7 |            |
  |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------

|*  4 |    INDEX RANGE SCAN | SYS_C008511 | 96676 |   660K|   133   (0)| 00:00:0
1 |

|   5 |   SORT AGGREGATE    |             |     1 |       |            |
  |

|   6 |    TABLE ACCESS FULL| P           |   100 |       |     2   (0)| 00:00:0
1 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	278	0:00:01
	Filter		
Tabla p: 100	Table Access Full (s)	145	0:00:01
	Sort Aggregate		
	Index Range Scan	133	0:00:01
Tabla sp: 10000000	Sort Aggregate		
	Table Access Full (p)	2	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Index Range Scan
Sort Aggregate
Sort Aggregate
Filter
Select Statement

Costo total bajo (278): El costo de la consulta es relativamente bajo, lo que sugiere que se están manejando eficientemente los recursos y la cantidad de datos.
Table Access Full (s): El escaneo completo de la tabla s tiene un costo moderado (**145**), lo que puede ser aceptable si la tabla no es grande.
Index Range Scan: Esta operación tiene un costo bajo (**133**), lo que indica que el uso del índice está mejorando la eficiencia de la consulta al permitir el acceso a un rango específico de datos.
Sort Aggregate: Se realizan dos operaciones de agregación y ordenamiento. Aunque estas pueden ser costosas, en este caso el costo es manejable, lo que sugiere que no se están manejando grandes volúmenes de datos en las operaciones de agregación.
Filter: La operación de filtro se aplica después de las agregaciones, y aunque no tiene un costo visible, es una operación clave para limitar los resultados finales.

TKPROF consulta 2:

```
SELECT *
FROM s
WHERE (SELECT COUNT(*)
FROM sp
WHERE s.sn = sp.sn) = (SELECT COUNT(*) FROM p)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0        160          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668      2.74       2.79          0     246501          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670      2.74       2.80          0     246661          0      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	2,467
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,000

Análisis:

Aunque este número es mejor que el anterior de **10.93**, sigue siendo relativamente alto, lo que sugiere que el sistema está realizando más lecturas lógicas de las deseadas en relación con las filas procesadas.
Un valor de **0,000** indica que no hay lecturas de disco en relación con las LIOs, lo que es óptimo. Esto significa que todas las operaciones de lectura están siendo gestionadas en memoria, eliminando la latencia asociada a las lecturas de disco.

Explain plan consulta 3:

```
| Id  | Operation            | Name     | Rows  | Bytes |TempSpc| Cost (%CPU)|
Time     |

-------------------------------------------------------------------------------
----------


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |          |   117K|  4368K|       | 15909   (6)|
00:00:01 |

|*  1 |  HASH JOIN SEMI      |          |   117K|  4368K|  4944K| 15909   (6)|
00:00:01 |

|   2 |   TABLE ACCESS FULL  | S        |   117K|  3563K|       |   145   (2)|
00:00:01 |

|   3 |   VIEW               | VW_NSO_1 |  9667K|    64M|       |  6704  (13)|
00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------

|*  4 |    FILTER            |          |       |       |       |            |
         |

|   5 |     HASH GROUP BY    |          |  9667K|    64M|       |  6704  (13)|
00:00:01 |

|   6 |      TABLE ACCESS FULL| SP       |  9667K|    64M|       |  6009   (2)|
00:00:01 |

|   7 |     SORT AGGREGATE   |          |     1 |       |       |            |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
         |

|   8 |      TABLE ACCESS FULL| P        |   100 |       |       |     2   (0)|
00:00:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	15909	0:00:01
	Hash Join Semi	15909	0:00:01
	Table Access Full (s)	145	0:00:01
	View	6704	0:00:01
Tabla p: 100	Filter		
	Hash Group By	6704	0:00:01
	Table Access Ful (sp)	6009	0:00:01
	Sort Aggregate		
Tabla sp: 10000000	Table Access Full (p)	2	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Table Access Full (sp)
View
Hash Group By
Filter
Hash Join Semi
Select Statement

Costo total alto (15909): El costo total de la consulta es elevado, lo que sugiere que se está procesando una cantidad significativa de datos o que hay varias operaciones costosas involucradas.
Múltiples Table Access Full: Se realizan escaneos completos de las tablas s, sp, y p. El escaneo completo de sp es particularmente alto (**6009**), lo que indica que puede haber oportunidades de optimización mediante el uso de índices.
View: La vista tiene un costo moderado (**6704**), lo que sugiere que puede estar manejando un conjunto considerable de datos o complejidad en su definición.
Hash Group By: Esta operación agrupa los datos y tiene un costo que coincide con la vista, lo que indica que se están agrupando grandes volúmenes de datos.
Filter: La operación de filtro no tiene un costo asociado visible, lo que indica que se puede estar aplicando de manera eficiente, posiblemente limitando los resultados después de las agrupaciones.

Hash Join Semi: Esta operación se utiliza para devolver filas de la tabla principal que cumplen con la condición de unión. Su alto costo indica que está trabajando con conjuntos de datos grandes.

TKPROF consulta 3:

```
SELECT *
FROM s
WHERE sn IN (SELECT sn
FROM sp
GROUP BY sn
HAVING COUNT(*) = (SELECT COUNT(*)
FROM p)
)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.01          0         24          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668      1.06       1.20      21773      44015          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670      1.07       1.21      21773      44039          1      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	0,440
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,494

Análisis:

Este es un resultado muy positivo y sugiere que el sistema está realizando un número eficiente de lecturas lógicas en relación con las filas procesadas.
Un valor de **0,494** indica que casi la mitad de las LIOs corresponden a lecturas de disco, lo que puede ser motivo de preocupación. Aunque las LIOs son bajas, una relación tan alta puede sugerir que hay acceso frecuente a disco que podría estar afectando el rendimiento.

Explain plan consulta 4:

```
| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time
    |

-------------------------------------------------------------------------------
-----


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |  9667K|   350M|       | 15909   (6)| 00:00
:01 |

|*  1 |  FILTER              |      |       |       |       |            |
    |

|   2 |   HASH GROUP BY      |      |  9667K|   350M|       | 15909   (6)| 00:00
:01 |

|*  3 |    HASH JOIN         |      |  9667K|   350M|  4944K| 15214   (2)| 00:00
:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------

|   4 |     TABLE ACCESS FULL| S    |   117K|  3563K|       |   145   (2)| 00:00
:01 |

|   5 |     TABLE ACCESS FULL| SP   |  9667K|    64M|       |  6009   (2)| 00:00
:01 |

|   6 |   SORT AGGREGATE     |      |     1 |       |       |            |
    |

|   7 |    TABLE ACCESS FULL | P    |   100 |       |       |     2   (0)| 00:00

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	15909	0:00:01
	Filter		
	Hash Group By	15909	0:00:01
Tabla p: 100	Hash Join	15214	0:00:01
	Table Access Full (s)	145	0:00:01
	Table Access Full (sp)	6009	0:00:01
	Sort Aggregate	0	0:00:01
Tabla sp: 1000000	Table Access Full (p)	2	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (p)
Table Access Full (s)
Table Access Full (sp)
Sort Aggregate
Hash Join
Hash Group By
Filter
Select Statement

Costo total alto (15909): El costo total de la consulta es elevado, lo que indica que la operación implica un procesamiento intensivo de datos, lo que puede afectar el rendimiento.
Múltiples Table Access Full: Se realizan escaneos completos de las tablas s, sp, y p, con costos moderados. La tabla sp tiene un costo notablemente más alto (**6009**), lo que sugiere que podría ser una oportunidad para optimizar su acceso mediante índices.
Sort Aggregate: Esta operación tiene un costo de **0**, lo que indica que no se requieren operaciones de ordenamiento o que se pueden realizar sin coste adicional. Esto es positivo, ya que sugiere que no hay grandes volúmenes de datos a ordenar.
Hash Join: Este es uno de los pasos más costosos, con un costo de **15214**, lo que sugiere que está combinando conjuntos de datos grandes. Si es posible, considera optimizar esta operación para mejorar el rendimiento.
Hash Group By: Agrupa los resultados obtenidos antes de aplicar el filtro. Su costo coincide con el de la consulta, lo que indica que el agrupamiento es significativo en el total.
Filter: La operación de filtro se realiza después de las agrupaciones y uniones, lo que sugiere que se están limitando los resultados finales antes de generar el conjunto de resultados.

TKPROF consulta 4:

```
SELECT s.sn, s.snombre, s.ciudad
FROM s
JOIN sp ON s.sn = sp.sn
GROUP BY s.sn, s.snombre, s.ciudad
HAVING COUNT(sp.pn) = (SELECT COUNT(p.pn) FROM p)
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	6	1	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	6668	7.27	7.55	21773	44015	0	100000
total	6670	7.27	7.56	21773	44021	1	100000

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	0,440
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,494

Análisis:

Los valores son idénticos a la consulta anterior, lo que significa que aunque hace un buen trabajo con respecto al acceso de las filas el problema radica en las lecturas en disco que debe hacer, que es un poco alto.

Punto 2:

- 100 datos en s, 1.000 datos en p y 99.999 datos en sp, se eliminó: DELETE FROM sp WHERE sn = 'S2' AND pn = 'P3'.

Explain plan consulta 1:

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 3797811149

------------------------------------------------------------------------------
-
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time
|

------------------------------------------------------------------------------
-


PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |   100 |  2400 |   315   (1)| 00:00:01
|

|   1 |  SORT AGGREGATE   |             |     1 |     4 |            |
|

|*  2 |   INDEX RANGE SCAN| SYS_C008430 |  1000 |  4000 |     6   (0)| 00:00:01
|

|   3 |  TABLE ACCESS FULL| S           |   100 |  2400 |     2   (0)| 00:00:01
|
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	315	0:00:01
Tabla p: 1000	Sort Aggregate		
	Index Range Scan	6	0:00:01
Tabla sp: 99999	Table Access Full (s)	2	0:00:01

Análisis:
Orden de las operaciones:
Index Range Scan
Table Access Full (s)
Sort Aggregate
Select Statement

Costo total bajo (315): El costo total de la consulta es bajo, lo que indica que las operaciones realizadas son eficientes y manejan una cantidad moderada de datos.
Index Range Scan: La utilización de un índice para realizar un escaneo de rango tiene un costo muy bajo (**6**). Esto es positivo, ya que mejora el tiempo de búsqueda y reduce la necesidad de escaneos completos.
Table Access Full (s): Hay un escaneo completo de la tabla s, pero el costo es muy bajo (**2**). Esto sugiere que la tabla no es grande o que el escaneo completo es apropiado en este contexto.
Sort Aggregate: Esta operación agrega y ordena los datos antes de generar el resultado final. Aunque es una operación que puede ser costosa, su costo es manejable en este caso.

TKPROF consulta 1:

```
SELECT s.*, (SELECT COUNT(*)
FROM sp
WHERE sp.sn = s.sn) AS cuantos
FROM s
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.00       0.01          0        580          0         100
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.00       0.01          0        580          0         100
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	5,800
p = 1000	i / j	12,500
sp = 99999	k / (f+g)	0,000

Análisis:

Aunque el valor de lecturas lógicas de las deseadas en relación con las filas procesadas.es bajo **5,8,** para la cantidad de datos podría ser mejor.
Este valor de **12,500** indica que por cada fetch se están retornando aproximadamente **12.5 filas**. Es un ratio razonable y refleja una buena eficiencia en la recuperación de datos.
Un valor de **0,000** sugiere que no hay lecturas de disco en relación con las LIOs, lo que es óptimo. Esto significa que todas las operaciones de lectura se están manejando en memoria, eliminando la latencia asociada a las lecturas de disco.

Explain plan consulta 2:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 3205455276

-------------------------------------------------------------------------------
| Id  | Operation              | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |      |   101 |  4428 |   120   (9)| 00:00:01 |
|   1 |  HASH UNIQUE           |      |   101 |  4428 |   120   (9)| 00:00:01 |
|   2 |   UNION-ALL            |      |       |       |            |          |
|*  3 |    HASH JOIN           |      |   100 |  4400 |    62  (12)| 00:00:01 |
|   4 |     TABLE ACCESS FULL  | S    |   100 |  2400 |     2   (0)| 00:00:01 |
|   5 |     VIEW               |      |   100 |  2000 |    60  (12)| 00:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   6 |      HASH GROUP BY     |      |   100 |   400 |    60  (12)| 00:00:01 |
|   7 |       TABLE ACCESS FULL| SP   | 99999 |   390K|    54   (2)| 00:00:01 |
|*  8 |    HASH JOIN ANTI      |      |     1 |    28 |    57   (4)| 00:00:01 |
|   9 |     TABLE ACCESS FULL  | S    |   100 |  2400 |     2   (0)| 00:00:01 |
|  10 |     TABLE ACCESS FULL  | SP   | 99999 |   390K|    54   (2)| 00:00:01 |
-------------------------------------------------------------------------------
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	120	0:00:01
	Hash Unique	120	0:00:01
	Union-All		
	Hash Join	62	0:00:01

Tabla p: 1000	Table Access Full (s)	2	0:00:01
	View	60	0:00:01
	Hash Group By	60	0:00:01
	Table Access Full (sp)	54	0:00:01
Tabla sp: 99999	Hash Join Anti	57	0:00:01
	Table Access Full (s)	2	0:00:01
	Table Access Full (sp)	54	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (s)
Table Access Full (sp)
Hash Join Anti
Table Access Full (s)
Table Access Full (sp)
Hash Join
Hash Group By
View
Union-All
Hash Unique
Select Statement

Costo total bajo (120): El costo total de la consulta es bajo, lo que indica que el procesamiento de datos se maneja de manera eficiente, posiblemente debido a un volumen pequeño de datos.
Múltiples Table Access Full: Se realizan escaneos completos de las tablas s y sp dos veces, lo que puede ser innecesario. Si las tablas son pequeñas, esto puede ser aceptable, pero en caso contrario, se recomienda implementar índices para optimizar el acceso.
Hash Join Anti: Esta operación se utiliza para excluir filas en una combinación. El costo es moderado (57), pero el uso de índices podría ayudar a reducir este costo en caso de que las tablas sean grandes.
Hash Group By: Esta operación se utiliza para agrupar resultados, y su costo (60) es bajo, lo que sugiere que no hay grandes volúmenes de datos a agrupar.
Union-All y Hash Unique: La operación **Union-All** combina resultados sin eliminar duplicados, seguida por un **Hash Unique**, que asegura que no haya duplicados en el resultado final. Dado el bajo costo, esto se maneja eficientemente.

TKPROF consulta 2:

```
SELECT *
FROM s
NATURAL JOIN (SELECT sn, COUNT(*) AS cuantos
FROM sp
GROUP BY sn)
UNION
SELECT s.*, 0 AS cuantos
FROM s WHERE sn NOT IN (SELECT sn FROM sp)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.04       0.04          0        395          0         100
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.04       0.04          0        395          0         100
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	3,950
p = 1000	i / j	12,500
sp = 99999	k / (f+g)	0,000

Análisis:

Aunque ha mejorado en comparación con el valor anterior, aún sugiere que el sistema realiza un número significativo de lecturas lógicas en relación con las filas procesadas.

Explain plan consulta 3:

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2599889890

--------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |   100 |  4400 |    62  (12)| 00:00:01 |
|*  1 |  HASH JOIN OUTER     |      |   100 |  4400 |    62  (12)| 00:00:01 |
|   2 |   TABLE ACCESS FULL  | S    |   100 |  2400 |     2   (0)| 00:00:01 |
|   3 |   VIEW               |      |   100 |  2000 |    60  (12)| 00:00:01 |
|   4 |    HASH GROUP BY     |      |   100 |   400 |    60  (12)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| SP   | 99999 |   390K|    54   (2)| 00:00:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100	Select Statement	62	0:00:01
	Hash Join Outer	62	0:00:01
Tabla p: 1000	Table Access Full (s)	2	0:00:01
	View	60	0:00:01
Tabla sp: 99999	Hash Group By	60	0:00:01
	Table Access Full (sp)	54	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (sp)
Hash Group By
View
Table Access Full (s)
Hash Join Outer
Select Statement

Costo total bajo (62): El costo total de la consulta es muy bajo, lo que sugiere que la operación está manejando una cantidad pequeña de datos o que se están utilizando recursos de manera eficiente.
Hash Join Outer: Esta operación implica una combinación externa de los resultados, y aunque es una de las más costosas en este plan, el costo total sigue siendo bajo, lo que indica que el volumen de datos involucrados es pequeño.
Table Access Full (sp) y **Table Access Full (s)**: Ambas tablas se escanean completamente, pero los costos son bajos. Esto puede indicar que las tablas son pequeñas o que no hay índices necesarios en este caso.
Hash Group By: Agrupa los datos obtenidos de sp, y el costo es también bajo. Esto indica que no se están agrupando grandes volúmenes de datos.
View: La vista es una representación de los resultados del **Hash Group By** y tiene un costo bajo, lo que sugiere que es una operación simple.

TKPROF consulta 3:

```
SELECT sn, snombre, ciudad, NVL(sp_cuantos.cuantos, 0) AS cuantos
FROM s
NATURAL LEFT OUTER JOIN (
  SELECT sn, COUNT(*) AS cuantos
  FROM sp
  GROUP BY sn
) sp_cuantos
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        8      0.00       0.00          0        199          0         100
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       10      0.00       0.00          0        199          0         100
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100	(f+g) / h	1,990
p = 1000	i / j	12,500
sp = 99999	k / (f+g)	0,000

Análisis:

Este resultado es notablemente más bajo que los anteriores, lo que sugiere una mejora en la eficiencia del sistema.

- 1.000 datos en s, 1.000 datos en p y 999.998 datos en sp, se eliminaron:
  DELETE FROM sp WHERE sn = 'S2' AND pn = 'P3'.
  DELETE FROM sp WHERE sn = 'S5' AND pn = 'P4'.

Explain plan consulta 1:

```
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time
|

-------------------------------------------------------------------------------
-


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |  1000 | 31000 |   783   (0)| 00:00:01
|

|   1 |  SORT AGGREGATE   |             |     1 |     7 |            |
|

|*  2 |   INDEX RANGE SCAN| SYS_C008448 |  8112 | 56784 |    20   (0)| 00:00:01
|

|   3 |  TABLE ACCESS FULL| S           |  1000 | 31000 |     3   (0)| 00:00:01
|
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	783	0:00:01
Tabla p: 1000	Sort Aggregate		
	Index Range Scan	20	0:00:01
Tabla sp: 999998	Table Access Full (s)	3	0:00:01

Análisis:
Orden de las operaciones:
Index Range Scan
Table Access Full (s)
Sort Aggregate
Select Statement

Costo total moderado (783): El costo de la consulta es relativamente bajo, lo que indica que la cantidad de datos o los recursos necesarios son moderados.
Index Range Scan: Se utiliza un índice para escanear un rango de valores, lo cual es eficiente y tiene un costo muy bajo (**20**). Esta es una operación positiva, ya que reduce el tiempo de búsqueda.
Table Access Full (s): Hay un escaneo completo de la tabla s, pero el costo es bajo (**3**), lo que sugiere que la tabla no es muy grande o que el escaneo completo es apropiado en este caso.
Sort Aggregate: Esta operación agrega y ordena los datos antes de generar el resultado final, lo cual es una operación costosa en comparación con las otras. Puede estar relacionada con funciones de agregación como SUM, AVG, etc.

TKPROF consulta 1:

```
SELECT s.*, (SELECT COUNT(*)
FROM sp
WHERE sp.sn = s.sn) AS cuantos
FROM s
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.00          0          4          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.13       0.13          0       6825          0        1000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.15       0.13          0       6829          0        1000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	6,829
p = 1000	i / j	14,706
sp = 999998	k / (f+g)	0,000

Análisis:

El valor **6,829** está entre lo recomendado y no es tan alto a pesar de la cantidad de datos.
Este valor de **14,706** indica que por cada fetch se retornan aproximadamente **14.71 filas**. Este es un ratio positivo que muestra una buena eficiencia en la recuperación de datos.
Un valor de **0,000** sugiere que no hay lecturas de disco en relación con las LIOs, lo que es ideal. Esto implica que todas las operaciones de lectura se están gestionando en memoria, eliminando la latencia asociada a las lecturas de disco.

Explain plan consulta 2:

```
| Id  | Operation             | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time
     |

-------------------------------------------------------------------------------
------


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |      |   812K|    38M|       | 11210   (2)| 00:0
0:01 |

|   1 |  HASH UNIQUE          |      |   812K|    38M|       | 11210   (2)| 00:0
0:01 |

|   2 |   UNION-ALL           |      |       |       |       |            |
     |

|   3 |    HASH GROUP BY      |      |   811K|    38M|    43M| 10637   (2)| 00:0
0:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|*  4 |     HASH JOIN         |      |   811K|    38M|       |   572   (3)| 00:0
0:01 |

|   5 |      TABLE ACCESS FULL| S    |  1000 | 43000 |       |     3   (0)| 00:0
0:01 |

|   6 |      TABLE ACCESS FULL| SP   |   811K|  5545K|       |   563   (2)| 00:0
0:01 |

|*  7 |    HASH JOIN ANTI     |      |  1000 | 38000 |       |   572   (3)| 00:0

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
0:01 |

|   8 |     TABLE ACCESS FULL | S    |  1000 | 31000 |       |     3   (0)| 00:0
0:01 |

|   9 |     TABLE ACCESS FULL | SP   |   811K|  5545K|       |   563   (2)| 00:0
0:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	11210	0:00:01
	Hash Unique	11210	0:00:01
	Union-All		
	Hash Group By	10637	0:00:01
Tabla p: 1000	Hash Join Anti	572	0:00:01
	Table Access Full (s)	3	0:00:01
	Table Access Full (sp)	563	0:00:01
	Hash Join Anti	572	0:00:01
Tabla sp: 999998	Table Access Full (s)	3	0:00:01
	Table Access Full (sp)	563	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (s)
Table Access Full (sp)
Hash Join Anti
Table Access Full (s)
Table Access Full (sp)
Hash Join Anti
Hash Group By
Union-All
Hash Unique
Select Statement

Costo total moderado (11210): El costo total de la consulta es significativo pero no extremadamente alto, sugiriendo que la consulta procesa un volumen considerable de datos o ejecuta varias operaciones costosas.
Uso de Hash Join Anti: Hay dos operaciones de **Hash Join Anti**, lo que sugiere que la consulta está excluyendo filas en dos conjuntos diferentes. Estas operaciones pueden ser costosas, especialmente cuando se combinan con escaneos completos de las tablas s y sp.
Múltiples Table Access Full: Ambas tablas (s y sp) son escaneadas completamente dos veces, lo que puede ser ineficiente si las tablas son grandes. Si no se están utilizando índices, podría ser una oportunidad importante para optimización.
Hash Group By: Agrupa los resultados antes de aplicar la operación **Union-All**, que combina los resultados de diferentes subconjuntos sin eliminar duplicados.

Hash Unique: Finalmente, se aplica una operación de **Hash Unique** para eliminar duplicados, lo cual es costoso pero necesario debido al uso de **Union-All**.

TKPROF consulta 2:

```
SELECT *
FROM s
NATURAL JOIN (SELECT sn, COUNT(*) AS cuantos
FROM sp
GROUP BY sn)
UNION
SELECT s.*, 0 AS cuantos
FROM s WHERE sn NOT IN (SELECT sn FROM sp)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0         24          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.41       0.42          0       4100          0        1000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.42       0.43          0       4124          1        1000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	4,125
p = 1000	i / j	14,706
sp = 999998	k / (f+g)	0,000

Análisis:

Esta consulta ha mejorado en comparación con el valor anterior (6.83) de la otra consulta.

Explain plan consulta 3:

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 2599889890

------------------------------------------------------------------------------
| Id  | Operation            | Name | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |   811K|    39M|   622  (11)| 00:00:01 |
|*  1 |  HASH JOIN OUTER     |      |   811K|    39M|   622  (11)| 00:00:01 |
|   2 |   TABLE ACCESS FULL  | S    |  1000 | 31000 |     3   (0)| 00:00:01 |
|   3 |   VIEW               |      |   811K|    15M|   614  (10)| 00:00:01 |
|   4 |    HASH GROUP BY     |      |   811K|  5545K|   614  (10)| 00:00:01 |
|   5 |     TABLE ACCESS FULL| SP   |   811K|  5545K|   563   (2)| 00:00:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 1000	Select Statement	622	0:00:01
	Hash Join Outer	622	0:00:01
Tabla p: 1000	Table Access Full (s)	3	0:00:01
	View	614	0:00:01
Tabla sp: 999998	Hash Group By	614	0:00:01
	Table Access Full (sp)	563	0:00:01

Análisis:
Orden de las operaciones:
Table Access Full (sp)
Hash Group By
View
Table Access Full (s)
Hash Join Outer
Select Statement

Costo moderado (622): El costo total de la consulta es relativamente bajo en comparación con otros casos, lo que sugiere que la cantidad de datos o los recursos necesarios para procesarlos no es excesiva.
Hash Join Outer: Esta operación de combinación externa es una de las más costosas en este plan. Si se optimiza esta operación (por ejemplo, utilizando índices adecuados), podría mejorar el rendimiento general.
Table Access Full (sp) y **Table Access Full (s)**: Ambas tablas se escanean completamente. El escaneo de la tabla sp es más costoso que el de s, lo que podría indicar que sp es más grande o menos optimizada. El uso de índices en ambas tablas podría reducir estos costos.
Hash Group By: Esta operación agrupa los datos antes de ser utilizados en la combinación. El costo moderado de esta operación sugiere que no se están agrupando enormes cantidades de datos, pero sigue siendo una oportunidad de optimización.

TKPROF consulta 3:

```
SELECT sn, snombre, ciudad, NVL(sp_cuantos.cuantos, 0) AS cuantos
FROM s
NATURAL LEFT OUTER JOIN (
  SELECT sn, COUNT(*) AS cuantos
  FROM sp
  GROUP BY sn
) sp_cuantos
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.00          0         12          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch       68      0.09       0.09          0       2052          0        1000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total       70      0.10       0.10          0       2064          0        1000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 1000	(f+g) / h	2,064
p = 1000	i / j	14,706
sp = 999998	k / (f+g)	0,000

Análisis:

Este es un resultado positivo, ya que sugiere una mejora significativa en la eficiencia en comparación con valores anteriores.

- ❖ 100.000 datos en s, 100 datos en p y 10.000.000 datos en sp:

Explain plan consulta 1:

```
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time
|

--------------------------------------------------------------------------------
-


PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |             |   117K|  3563K|    11M  (1)| 00:07:32
|

|   1 |  SORT AGGREGATE   |             |     1 |     7 |            |
|

|*  2 |   INDEX RANGE SCAN| SYS_C008511 | 96676 |   660K|   133   (0)| 00:00:01
|

|   3 |  TABLE ACCESS FULL| S           |   117K|  3563K|   145   (2)| 00:00:01
|
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	11M	0:07:32
Tabla p: 100	Sort Aggregate		
	Index Range Scan	133	0:00:01
Tabla sp: 10000000	Table Access Full (s)	145	0:00:01

Análisis:

Orden de las operaciones:

Index Range Scan
Table Access Full (s)
Sort Aggregate
Select Statement

Costo total alto (11M): El costo total de la consulta es considerablemente alto, lo que indica que la operación es intensiva en recursos.
Sort Aggregate: Esta operación implica que se están realizando agregaciones que requieren un ordenamiento de los datos, lo cual puede ser costoso dependiendo del volumen de los datos. Esto podría estar relacionado con funciones de agregación como SUM, COUNT, etc.
Index Range Scan: Se está utilizando un índice para escanear un rango de valores. Esta es una operación eficiente si el índice está bien optimizado para la consulta. El costo asociado es bajo (**133**), lo cual es una señal positiva.
Table Access Full (s): Hay un escaneo completo de la tabla s, que aunque tiene un costo bajo (**145**), puede ser un área de optimización si la tabla es grande. Considera la posibilidad de utilizar un índice para evitar el escaneo completo.

TKPROF consulta 1:

```
SELECT s.*, (SELECT COUNT(*)
FROM sp
WHERE sp.sn = s.sn) AS cuantos
FROM s
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          4          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668      3.00       2.89          0     246498          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670      3.00       2.89          0     246502          0      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	2,465
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,000

Análisis:

El valor **2,465** es un resultado bastante positivo considerando la magnitud de los datos.
Este valor se mantiene en **14,997**, lo que significa que por cada fetch se retornan aproximadamente **15 filas**. Este es un ratio excelente que muestra una buena eficiencia en la recuperación de datos.
Un valor de **0,000** indica que no hay lecturas de disco en relación con las LIOs, lo que es óptimo. Esto significa que todas las operaciones de lectura se están gestionando en memoria, eliminando la latencia asociada a las lecturas de disco.

Explain plan consulta 2:

```
| Id  | Operation              | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Ti
me     |

-------------------------------------------------------------------------------
--------


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |      |  9785K|   474M|       |   147K  (2)| 00
:00:06 |

|   1 |  HASH UNIQUE           |      |  9785K|   474M|       |   147K  (2)| 00
:00:06 |

|   2 |   UNION-ALL            |      |       |       |       |            |
       |

|   3 |    MERGE JOIN          |      |  9667K|   470M|       |  7860  (11)| 00
:00:01 |

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------

|   4 |     SORT JOIN          |      |  9667K|   184M|       |  6704  (13)| 00
:00:01 |

|   5 |      VIEW              |      |  9667K|   184M|       |  6704  (13)| 00
:00:01 |

|   6 |       HASH GROUP BY    |      |  9667K|    64M|       |  6704  (13)| 00
:00:01 |

|   7 |        TABLE ACCESS FULL| SP   |  9667K|    64M|       |  6009   (2)| 00

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
:00:01 |

|*  8 |     SORT JOIN          |      |   117K|  3563K|     9M|  1156   (1)| 00
:00:01 |

|   9 |      TABLE ACCESS FULL | S    |   117K|  3563K|       |   145   (2)| 00
:00:01 |

|* 10 |    HASH JOIN ANTI      |      |   117K|  4368K|  4944K| 15214   (2)| 00
:00:01 |


PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------------
|  11 |     TABLE ACCESS FULL  | S    |   117K|  3563K|       |   145   (2)| 00
:00:01 |

|  12 |     TABLE ACCESS FULL  | SP   |  9667K|    64M|       |  6009   (2)| 00
:00:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	147K	0:00:06
	Hash Unique	147K	0:00:06
	Union-All		
	Merge Join	7860	0:00:01
	Sort Join	6704	0:00:01
	View	6704	0:00:01
Tabla p: 100	Hash Group By	6704	0:00:01
	Table Access Full (sp)	6009	0:00:01
	Sort Join	1156	0:00:01
	Table Access Full (s)	145	0:00:01
	Hash Join Anti	15214	0:00:01
Tabla sp: 10000000	Table Access Full (s)	145	0:00:01
	Table Access Full (sp)	6009	0:00:01

Análisis:
El orden de las operaciones es:
Table Access Full (sp)
Hash Group By
View
Sort Join
Table Access Full (s)
Sort Join
Merge Join
Table Access Full (s)
Table Access Full (sp)
Hash Join Anti
Union-All
Hash Unique
Select Statement

El costo general de la consulta es considerablemente alto, lo que indica que la consulta involucra un procesamiento intensivo de datos. Esto podría deberse a la

combinación de múltiples uniones, escaneos completos de tablas grandes y la eliminación de duplicados.
El uso de **Union-All** implica que los resultados de varias subconsultas o conjuntos de datos se están combinando, y el **Hash Unique** al final está eliminando duplicados, lo que añade un costo significativo a la consulta.
Se están realizando múltiples **Full Table Scans** sobre las tablas s y sp, lo que sugiere que no hay índices que se estén utilizando de manera efectiva. Si estas tablas son grandes, esto puede ser un factor importante en el alto costo.
La presencia de un **Hash Join Anti** indica que se está excluyendo un conjunto de datos de otro. Esto también es una operación costosa, ya que requiere construir un hash de los datos y luego excluir las coincidencias.
Las operaciones de **Merge Join** son eficientes cuando ambas tablas están ordenadas, pero las dos operaciones de **Sort Join** agregan costo porque implican ordenar los datos antes de la unión. Si se pudiera utilizar un índice, estas operaciones podrían evitarse.

TKPROF consulta 2:

```
SELECT *
FROM s
NATURAL JOIN (SELECT sn, COUNT(*) AS cuantos
FROM sp
GROUP BY sn)
UNION
SELECT s.*, 0 AS cuantos
FROM s WHERE sn NOT IN (SELECT sn FROM sp)
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.01          0         24          1           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668      1.27       1.37      22092      44950          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670      1.28       1.39      22092      44974          1      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	0,450
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,491

Análisis:

Este valor indica que hay aproximadamente **0.45 LIOs** por cada fila procesada. Este es un resultado excepcionalmente bueno, sugiriendo una alta eficiencia en las lecturas lógicas en relación con las filas procesadas.

Un valor de **0,491** indica que hay lecturas de disco que representan aproximadamente el **49.1%** de las LIOs. Este número es extremadamente alto, y sugiere que una parte significativa de las lecturas está ocurriendo desde disco.

Explain plan consulta 3:

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 2599889890

------------------------------------------------------------------------------
-----

| Id  | Operation            | Name | Rows  | Bytes |TempSpc| Cost (%CPU)| Time
    |

------------------------------------------------------------------------------
-----


PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |  9667K|   470M|       | 21851   (5)| 00:00
:01 |

|*  1 |  HASH JOIN OUTER     |      |  9667K|   470M|  4944K| 21851   (5)| 00:00
:01 |

|   2 |   TABLE ACCESS FULL  | S    |   117K|  3563K|       |   145   (2)| 00:00
:01 |

|   3 |   VIEW               |      |  9667K|   184M|       |  6704  (13)| 00:00
:01 |

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------

|   4 |    HASH GROUP BY     |      |  9667K|    64M|       |  6704  (13)| 00:00
:01 |

|   5 |     TABLE ACCESS FULL| SP   |  9667K|    64M|       |  6009   (2)| 00:00
:01 |
```

Volumen de Datos	Operación	Costo	Tiempo
Tabla s: 100000	Select Statement	21851	0:00:01
	Hash Join Outer	21851	0:00:01
Tabla p: 100	Table Access Full (s)	145	0:00:01
	View	6704	0:00:01
	Hash Group By	6704	0:00:01
Tabla sp: 10000000	Table Access Full (sp)	6009	0:00:01

Análisis:
El orden de las operaciones está dado por:
Table Access Full (sp)
Hash Group By
View (creación de la vista con datos agrupados)
Table Access Full (s)
Hash Join Outer (combina s con los resultados de la vista)
Select Statement (resultado final)

El **Hash Join Outer** es el principal contribuyente al costo total. Si la consulta es lenta, es posible que la combinación entre las tablas s y sp sea una parte crítica para optimizar.
Se están realizando **Full Table Scans** tanto en la tabla s como en la tabla sp, lo que podría sugerir que no se están utilizando índices eficientemente. Si estas tablas son grandes, considerar la creación de índices apropiados podría reducir el costo significativamente.

TKPROF consulta 3:

```
SELECT sn, snombre, ciudad, NVL(sp_cuantos.cuantos, 0) AS cuantos
FROM s
NATURAL LEFT OUTER JOIN (
  SELECT sn, COUNT(*) AS cuantos
  FROM sp
  GROUP BY sn
) sp_cuantos
```

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.01       0.00          0         12          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch     6668      0.92       1.21      21773      44012          0      100000
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total     6670      0.93       1.21      21773      44024          0      100000
```

Datos en las tablas	Tasa	Valor Encontrado
s = 100000	(f+g) / h	0,440
p = 100	i / j	14,997
sp = 10000000	k / (f+g)	0,494

Análisis:

Esta consulta es por muy poco mejor a la anterior en cuanto a las lecturas lógicas en relación con las filas procesadas, pero es también un poco peor en cuanto a las lecturas del disco.