UNIVERSIDAD PRIVADA-DE-TACNA

INGENIERIA DE SISTEMAS

TITULO:

INFORME DE LABORATORIO Nº 03

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Orlando Antonio Acosta Ortiz	(2015052775)
Orestes Ramirez Ticona	(2015053236)
Nilson Laura Atencio	(2015053846)
Roberto Zegarra Reyes	(2010036175)
Richard Cruz Escalante	(2013047247)

Índice

1. INFORMACIÓN GENERAL

1.1. Objetivos:

- Aplicar y Desarrollar Consultas con Pivot y Grouping Sets

1.2. Equipos, materiales, programas y recursos utilizados:

- Microsoft SQL Server 2016, 2017 o superior
- Base de datos TSQL
- Computadora
- Tener una cuenta en Github para subir los cambios

2. MARCO TEORICO

2.1. Base de datos TSQL:

- SQL (Structured Query Language), Lenguaje Estructurado de Consulta es el lenguaje utilizado para definir, controlar y acceder a los datos almacenados en una base de datos relacional. Como ejemplos de sistemas gestores de bases de datos que utilizan SQL podemos citar DB2, SQL Server, Oracle, MySql, Sybase, PostgreSQL o Access. El SQL es un lenguaje universal que se emplea en cualquier sistema gestor de bases de datos relacional. Tiene un estándar definido, a partir del cual cada sistema gestor ha desarrollado su versión propia. En SQL Server la versión de SQL que se utiliza se llama TRANSACT-SQL. EL SQL en principio es un lenguaje orientado únicamente a la definición y al acceso a los datos por lo que no se puede considerar como un lenguaje de programación como tal ya que no incluye funcionalidades como son estructuras condicionales, bucles, formateo de la salida, etc. (aunque veremos que esto está evolucionando). Se puede ejecutar directamente en modo interactivo, pero también se suele emplear embebido en programas escritos en lenguajes de programación convencionales. En estos programas se mezclan las instrucciones del propio lenguaje (denominado anfitrión) con llamadas a procedimientos de acceso a la base de datos que utilizan el SQL como lenguaje de acceso. Como por ejemplo en Visual Basic, Java, C, PHP .NET, etc.

2.2. Las instrucciones SQL se clasifican según su propósito en tres grupos:

- El DDL (Data Description Language) Lenguaje de Descripción de Datos..
- El DCL (Data Control Language) Lenguaje de Control de Datos.
- El DML (Data Manipulation Language) Lenguaje de Manipulación de Datos.

2.3. Consultas con Pivot :

- Las operaciones con Pivot nos permitirá convertir los resultados de una consulta que se presentan en filas y mostrarlos en columnas.
- Pivot utiliza las funciones de agregado para presentar los datos en columnas.
- El DML (Data Manipulation Language) Lenguaje de Manipulación de Datos.

2.4. Grouping Sets:

- GROUP BY GROUPING SETS es una poderosa extensión de la cláusula GROUP BY que permite computar múltiples cláusulas de grupo en una sola declaración. El conjunto de grupos es un conjunto de columnas de dimensión.GRUPO POR CONJUNTOS DE GRUPO es equivalente a la UNIONde dos o más operaciones de GRUPO POR en el mismo conjunto de resultados:

- GROUP BY GROUPING SETS((a))es equivalente a la operación de conjunto de agrupación única .GROUP BY a
- GROUP BY GROUPING SETS((a),(b))es equivalente a .GROUP BY a UNION ALL GROUP BY b

3. PROCEDIMIENTO

1. Escribiendo consultas con el operador PIVOT

   a) Paso 1: Escribir una sentencia SELECT para recuperar el numero de clientes para un grupo especifico de clientes.
   - Abrir el SQL Server Management Studio y conectar a la basa de datos (local) usando Windows.
   - Usar la base de datos TSQL
   - Ejecutar el siguiente codigo para crear una vista

```
CREATE VIEW Sales.CustGroups AS
SELECT
    custid,
    CHOOSE(custid % 3 + 1, N'A', N'B', N'C') AS custgroup,
    country
FROM Sales.Customers;

GO
```

   - En el panel de consulta, escribir la siguiente consulta.

```
SELECT
custid,
custgroup,
country
FROM Sales.CustGroups;
```

   - Luego modificamos el codigo, aplicando el operador PIVOT.

```
SELECT
country,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

   b) Paso 2: Especifique el elemento de agrupacion para el operador PIVOT.
   - Escribir la siguiente consulta para poder modificar la vista creada anteriormente, añadiendo 2 columnas adicionales.

```
ALTER VIEW Sales.CustGroups AS
SELECT
    custid,
    CHOOSE(custid % 3 + 1, N'A', N'B', N'C') AS custgroup,
    country,
    city,
    contactname
FROM Sales.Customers;

GO
```

   - Escribir la siguiente consulta.

```
SELECT
country,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

- Modificar la consulta para incluir columnas adicionales desde la vista.

```
SELECT
country,
city,
contactname,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

c) Paso 3: Use una expresion de tabla común (CTE) para especificar el elemnto de agrupacion para el operador PIVOT.
- Escribir la siguiente consulta y ejecutar.

```
WITH PivotCustGroups AS
(
SELECT
custid,
country,
custgroup
FROM Sales.CustGroups
)
SELECT
country,
p.A,
p.B,
p.C
FROM PivotCustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

d) Paso 4: Escribe una instruccion SELECT para recuperar el monto total de ventas para cada cliente y categoria de producto.
- Escribir la siguiente consulta.

```
WITH SalesByCategory AS
(
SELECT
o.custid,
d.qty * d.unitprice AS salesvalue,
c.categoryname
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON o.orderid = d.orderid
INNER JOIN Production.Products AS p ON p.productid = d.productid
INNER JOIN Production.Categories AS c ON c.categoryid = p.categoryid
WHERE o.orderdate >= '20080101' AND o.orderdate < '20090101'
)
SELECT
custid,
p.Beverages,
p.Condiments,
p.Confections,
p.[Dairy Products],
p.[Grains/Cereals],
p.[Meat/Poultry],
p.Produce,
p.Seafood
FROM SalesByCategory
PIVOT (SUM(salesvalue) FOR categoryname
IN (Beverages, Condiments, Confections, [Dairy Products], [Grains/Cereals],
[Meat/Poultry], Produce, Seafood)) AS p;
```

2. Escribiendo consultas con el operador UNPIVOT

a) Paso 1: Crear y consultar la vista Sale.PivotCustGroups.
- Escribir la siguiente consulta para crear una vista llamada Sales.PivotCustGroups.

```
CREATE VIEW Sales.PivotCustGroups AS
WITH PivotCustGroups AS
(
    SELECT
        custid,
        country,
        custgroup
    FROM Sales.CustGroups
)
SELECT
    country,
    p.A,
    p.B,
    p.C
FROM PivotCustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;

GO
```

- Despues escribrir la siguiente consulta.

```
SELECT
country, A, B, C
FROM Sales.PivotCustGroups;
```

b) Paso 2: Escriba una instruccion SELECT para recuperar una fila para cada pais y grupo de cliente.
- En el panel de consulta escribir la siguiente consulta.

```
SELECT
custgroup,
country,
numberofcustomers
FROM Sales.PivotCustGroups
UNPIVOT (numberofcustomers FOR custgroup IN (A, B, C)) AS p;
```

c) Paso 3: Eliminar las vistas creadas.

```
DROP VIEW Sales.CustGroups;
DROP VIEW Sales.PivotCustGroups;
```

83 %

Messages

Commands completed successfully.

3. Escribiendo consultas con las clausulas GROUPING SETS, CUBE, and ROLLUP.

a) Paso 1: Escriba una instruccion SELECT que use LA SUBCLAUSULA GROUPING SETS para devolver el número de Clientes para diferentes conjuntos de agrupación.
- Escribir la siguiente consulta y ejecutar.

```
SELECT
country,
city,
COUNT(custid) AS noofcustomers
FROM Sales.Customers
GROUP BY
GROUPING SETS
(
(country, city),
(country),
(city),
()
);
```

b) Paso 2: Escriba una instruccion SELECT que use la subclausula CUBE para recuperar Grouping sets basados en valores de ventas anuales, mensuales y diarios.
- Escribir la siguiente consulta.

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
CUBE (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

c) Paso 3: Escriba la misma instruccion SELECT usando la subclausula ROLLUP.
- Escribir la siguiente consulta.

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

d) Paso 4: Analizar el valor total de ventas por año y mes.
- Escribir la siguiente consulta y ejecutar.

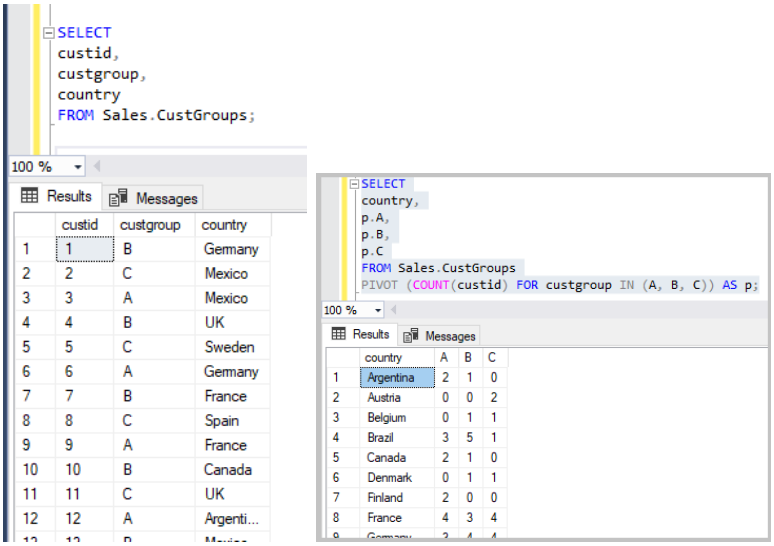
```
SELECT
GROUPING_ID(YEAR(orderdate), MONTH(orderdate)) as groupid,
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate))
ORDER BY groupid, orderyear, ordermonth;
```

4. ANALISIS E INTERPRETACION DE RESULTADOS

A lo largo del desarrollo de este laboratorio, se realizaron difenrentes consultas con PIVOT y Grouping Sets .
El resultado de las consultas mencionadas se mostraran a continuacion:

1. Escribiendo consultas con el operador PIVOT

   a) Paso 1: Escribir una sentencia SELECT para recuperar el numero de clientes para un grupo especifico de clientes.
   - Comparamos el resultado de la primera consulta con la que se modifico aplicando el operador PIVOT.



```
SELECT
custid,
custgroup,
country
FROM Sales.CustGroups;
```

Results | Messages

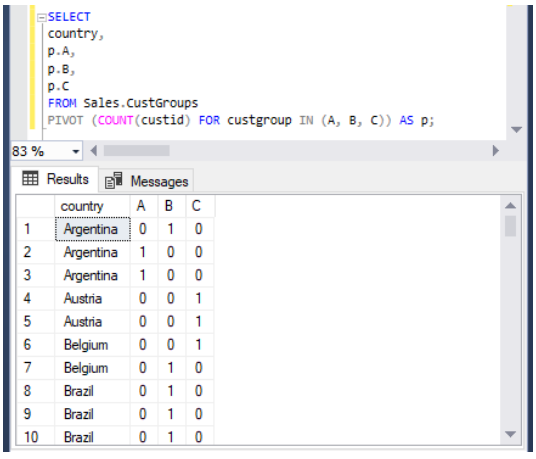
	custid	custgroup	country
1	1	B	Germany
2	2	C	Mexico
3	3	A	Mexico
4	4	B	UK
5	5	C	Sweden
6	6	A	Germany
7	7	B	France
8	8	C	Spain
9	9	A	France
10	10	B	Canada
11	11	C	UK
12	12	A	Argenti...

```
SELECT
country,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

Results | Messages

	country	A	B	C
1	Argentina	2	1	0
2	Austria	0	0	2
3	Belgium	0	1	1
4	Brazil	3	5	1
5	Canada	2	1	0
6	Denmark	0	1	1
7	Finland	2	0	0
8	France	4	3	4

   b) Paso 2: Especifique el elemento de agrupacion para el operador PIVOT.
   - Ejecutamos la siguiente consulta, y vemos que su resultado se parece al del paso 1, sin embargo no es lo mismo. Màs filas se devolvieron despues de modificar la vista.



```
SELECT
country,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

Results | Messages

	country	A	B	C
1	Argentina	0	1	0
2	Argentina	1	0	0
3	Argentina	1	0	0
4	Austria	0	0	1
5	Austria	0	0	1
6	Belgium	0	0	1
7	Belgium	0	1	0
8	Brazil	0	1	0
9	Brazil	0	1	0
10	Brazil	0	1	0

   - Modificar la consulta para incluir columnas adicionales desde la vista y ejecutar; notamos que es el mismo resultado que la consulta anterior. El operador PIVOT asume que todas las

columnas excepto la de agregacion y elementos de extension son parte de la agrupacion de columnas.

```
SELECT
country,
city,
contactname,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

83 %

Results | Messages

	country	city	contactname	A	B	C
1	Argentina	Buenos Aires	Gaffney, Lawrie	0	1	0
2	Argentina	Buenos Aires	Ray, Mike	1	0	0
3	Argentina	Buenos Aires	Tiano, Mike	1	0	0
4	Austria	Graz	Kane, John	0	0	1
5	Austria	Salzburg	Meston, Tosh	0	0	1
6	Belgium	Bruxelles	Mace, Donald	0	0	1
7	Belgium	Charleroi	Gulbis, Katrin	0	1	0
8	Brazil	Campinas	Cheng, Yao-Qiang	0	1	0
9	Brazil	Resende	Li, Yan	0	1	0
10	Brazil	Rio de Jan...	Cohen, Shy	0	1	0

c) Paso 3: Use una expresion de tabla común (CTE) para especificar el elemento de agrupacion para el operador PIVOT.
- Escribir la siguiente consulta y ejecutar. Se puede observar que el resultado es el mismo que el del paso1; en este paso, la CTE da 3 posibles columnas al operador PIVOT. Y en el paso1, la vista tambien provee 3 columnas a operador PIVOT.

```
WITH PivotCustGroups AS
(
SELECT
custid,
country,
custgroup
FROM Sales.CustGroups
)
SELECT
country,
p.A,
p.B,
p.C
FROM PivotCustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

83 %

Results | Messages

	country	A	B	C
1	Argentina	2	1	0
2	Austria	0	0	2
3	Belgium	0	1	1
4	Brazil	3	5	1
5	Canada	2	1	0
6	Denmark	0	1	1
7	Finland	2	0	0
8	France	4	3	4

Cuando usamos el operador PIVOT, no puede especificar directamente el elemento de agrupación porque SQL Server asume automáticamente que todas las columnas deben usarse como elementos de agrupación, con la excepción de los elementos de expansión y agregación. Con un CTE, puede especificar las columnas exactas y, por lo tanto, controlar el uso de las columnas para la agrupación.

d) Paso 4: Escribe una instruccion SELECT para recuperar el monto total de ventas para cada cliente y categoria de producto.
- Escribir la siguiente consulta y ejecutar.

```
WITH SalesByCategory AS
(
SELECT
o.custid,
d.qty * d.unitprice AS salesvalue,
c.categoryname
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON o.orderid = d.orderid
INNER JOIN Production.Products AS p ON p.productid = d.productid
INNER JOIN Production.Categories AS c ON c.categoryid = p.categoryid
WHERE o.orderdate >= '20080101' AND o.orderdate < '20090101'
)
SELECT
custid,
p.Beverages,
p.Condiments,
p.Confections,
p.[Dairy Products],
p.[Grains/Cereals],
p.[Meat/Poultry],
p.Produce,
p.Seafood
FROM SalesByCategory
PIVOT (SUM(salesvalue) FOR categoryname
IN (Beverages, Condiments, Confections, [Dairy Products], [Grains/Cereals],
[Meat/Poultry], Produce, Seafood)) AS p;
```

Results | Messages

	custid	Beverages	Condiments	Confections	Dairy Products	Grains/Cereals	Meat/Poultry	Produce	Seafood
1	1	NULL	426.00	NULL	1255.00	NULL	NULL	91.20	530.00
2	2	NULL	NULL	64.40	390.00	NULL	NULL	NULL	60.00
3	3	380.00	NULL	NULL	NULL	280.00	NULL	NULL	NULL
4	4	282.00	NULL	4440.00	812.50	NULL	NULL	NULL	304.00
5	5	850.50	300.00	2202.55	NULL	NULL	1237.90	1368.00	2151.60
6	6	NULL	114.00	283.00	714.00	NULL	NULL	424.00	625.00
7	7	NULL	NULL	NULL	437.50	292.50	NULL	NULL	NULL
8	8	NULL	NULL	NULL	NULL	280.00	NULL	NULL	NULL
9	9	533.00	1750.00	1515.10	556.80	665.00	624.00	705.00	837.00
10	10	1706.50	1290.10	4518.30	992.50	684.00	234.00	1872.00	930.00
11	11	1380.00	NULL	NULL	220.00	441.00	NULL	120.00	270.00
12	12	1037.00	NULL	NULL	25.00	NULL	NULL	364.80	150.00

2. Escribiendo consultas con el operador UNPIVOT

a) Paso 1: Crear y consultar la vista Sale.PivotCustGroups.

- Ejecutar la siguiente consulta.

```
SELECT
country, A, B, C
FROM Sales.PivotCustGroups;
```

33 %

Results | Messages

	country	A	B	C
1	Argentina	2	1	0
2	Austria	0	0	2
3	Belgium	0	1	1
4	Brazil	3	5	1
5	Canada	2	1	0
6	Denmark	0	1	1

b) Paso 2: Escriba una instruccion SELECT para recuperar una fila para cada pais y grupo de cliente.
- Escribir la siguiente consulta y ejecutar.

```
SELECT
custgroup,
country,
numberofcustomers
FROM Sales.PivotCustGroups
UNPIVOT (numberofcustomers FOR custgroup IN (A, B, C)) AS p;
```

83 %

Results | Messages

	custgroup	country	numberofcustomers
1	A	Argentina	2
2	B	Argentina	1
3	C	Argentina	0
4	A	Austria	0
5	B	Austria	0
6	C	Austria	2
7	A	Belgium	0
8	B	Belgium	1
9	C	Belgium	1
10	A	Brazil	3

3. Escribiendo consultas con las clausulas GROUPING SETS, CUBE, and ROLLUP.

a) Paso 1: Escriba una instruccion SELECT que use LA SUBCLAUSULA GROUPING SETS para devolver el número de Clientes para diferentes conjuntos de agrupación.
- Escribir la siguiente consulta y ejecutar.

```
SELECT
country,
city,
COUNT(custid) AS noofcustomers
FROM Sales.Customers
GROUP BY
GROUPING SETS
(
(country, city),
(country),
(city),
()
);
```

83 %

Results | Messages

	country	city	noofcustomers
1	Germany	Aachen	1
2	NULL	Aachen	1
3	USA	Albuquerque	1
4	NULL	Albuquerque	1
5	USA	Anchorage	1
6	NULL	Anchorage	1
7	Denmark	Århus	1

b) Paso 2: Escriba una instruccion SELECT que use la subclausula CUBE para recuperar Grouping sets basados en valores de ventas anuales, mensuales y diarios.
- Escribir la siguiente consulta y ejecutar.

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
CUBE (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

83 %

Results | Messages

	orderyear	ordermonth	orderday	salesvalue
1	2007	1	1	6931.60
2	2008	1	1	1738.00
3	NULL	1	1	8669.60
4	2007	4	1	851.20
5	2008	4	1	11549.89

c) Paso 3: Escriba la misma instruccion SELECT usando la cubclausula ROLLUP.
- Ejecutalos la siguiente consulta, observamos el resultado. ¿Cuál es la diferencia entre las subcláusulas ROLLUP y CUBE de la cláusula GROUP BY? Al igual que la subcláusula CUBE, la subcláusula ROLLUP proporciona una forma abreviada de definir conjuntos de agrupación múltiples. Sin embargo, a diferencia de CUBE, ROLLUP no produce todos los conjuntos de agrupación posibles que pueden definirse en función de los miembros de entrada, sino que produce un subconjunto de ellos. ROLLUP asume una jerarquía entre los miembros de entrada y produce todos los conjuntos de agrupación que tienen sentido, teniendo en cuenta la jerarquía. En otras palabras, mientras CUBE (a, b, c) produce los ocho conjuntos de agrupación posibles de los tres miembros de entrada, ROLLUP (a, b, c) produce solo cuatro conjuntos de agrupación, asumiendo la jerarquía a¿b¿c. ROLLUP (a, b, c) es el equivalente a especificar CONJUNTOS DE AGRUPACIÓN ((a, b, c), (a, b), (a), ()). ¿Cuál es la subcláusula más apropiada para usar en este ejemplo? Desde año, mes y día forman un Jerarquía, la cláusula ROLLUP es más adecuada. Probablemente no hay mucho interés en mostrar se acumula durante un mes independientemente del año, pero al revés es interesante.

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

83 %

Results | Messages

	orderyear	ordermonth	orderday	salesvalue
1	2006	7	4	440.00
2	2006	7	5	1863.40
3	2006	7	8	2206.66
4	2006	7	9	3597.90
5	2006	7	10	1444.80

d) Paso 4: Analizar el valor total de ventas por año y mes.
- Escribir la siguiente consulta y ejecutar.

```
SELECT
GROUPING_ID(YEAR(orderdate), MONTH(orderdate)) as groupid,
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate))
ORDER BY groupid, orderyear, ordermonth;
```

83 %

Results | Messages

	groupid	orderyear	ordermonth	salesvalue
1	0	2006	7	27861.90
2	0	2006	8	25485.28
3	0	2006	9	26381.40
4	0	2006	10	37515.73
5	0	2006	11	45600.05

5. CONCLUSIONES

- Durante el desarrolo del laboratorio se puso en practica todo sobre el uso del T-SQL para realizar el cambio de filas a columnas. Una de estas opciones fue el PIVOT que importante y simple que nos permite, en lugar de crear un codigo complejo, un código con referencias cruzadas a partir de cualquier tabla.

- Con PIVOT podemos cambiar la orientación de la tabla hasta visualizar los datos de forma que le sea más útil y fácil el análisis. Cambiar la orientación permite examinar las secciones cruzadas seleccionadas de datos.

6. REFERENCIAS

- https://support.office.com/es-es/article/acerca-de-adventure-works-y-la-base-de-datos-de-negocio-de-muestra-00a88101-ef11-4a8d-8904-b9747f53c961
- https://www.manualsqlserver.com/?p=507
- https://grimpidev.wordpress.com/2008/11/29/lo-nuevo-de-sql-server-2008-grouping-sets/
- https://searchdatacenter.techtarget.com/es/definicion/SQL-Server