

SQL - DML

Data Manipulation Language

UTN - FRBA
Ing. en Sistemas de Información
Gestión de Datos

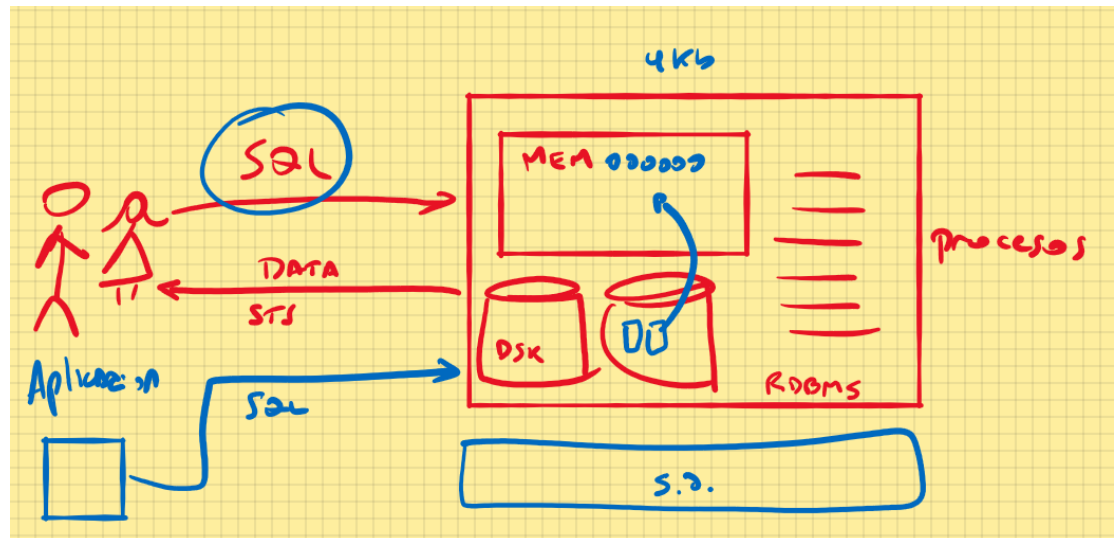
Prof.: Ing. Juan Zaffaroni

SQL

Sub Lenguajes

- DDL – Data Definition Language
- DML – Data Manipulation Language

- SELECT
- INSERT
- UPDATE
- DELETE



SQL – Operador SELECT

SELECT * | lista de columnas

FROM nom_tabla | lista de tablas

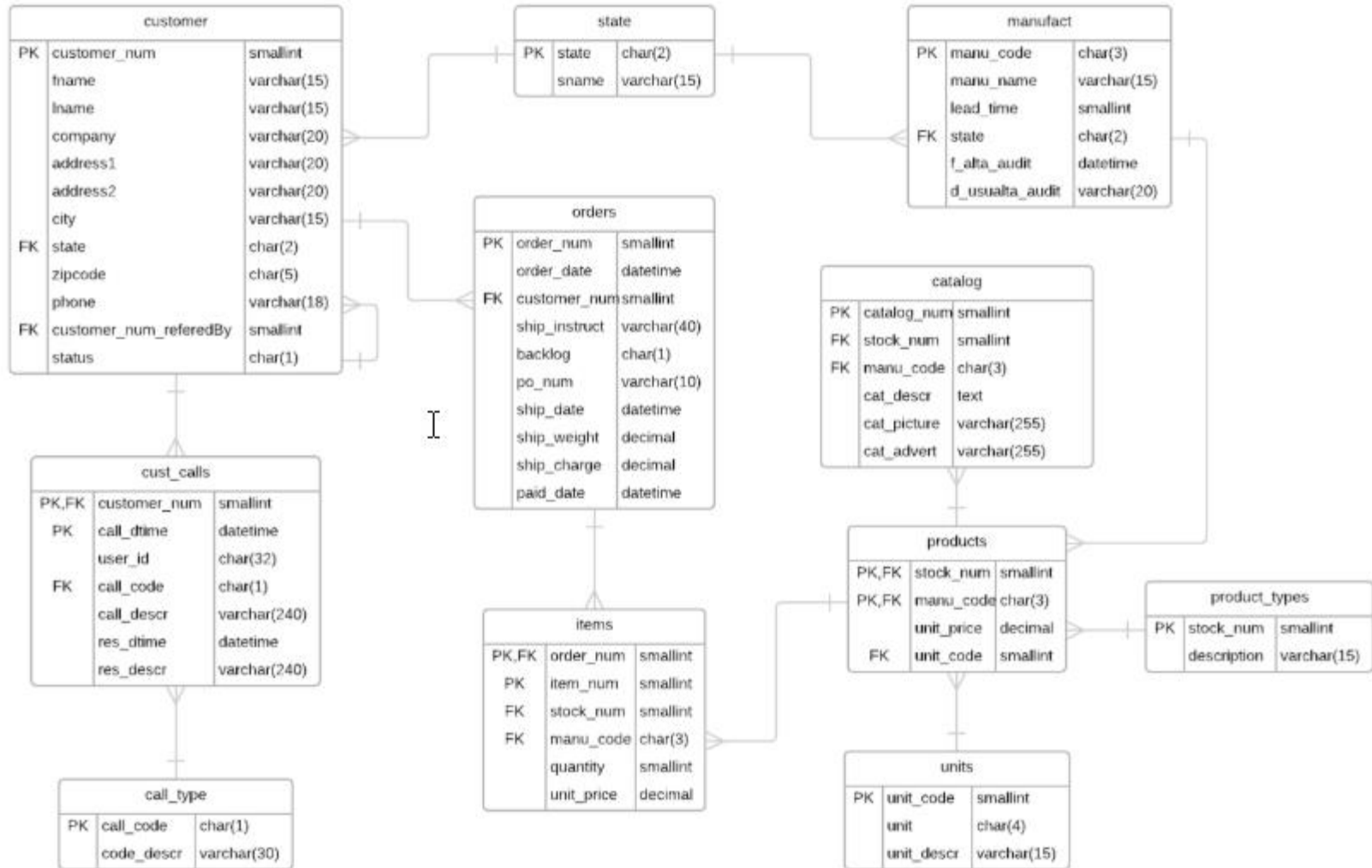
WHERE condiciones ó filtros

GROUP BY columnas clave de agrupamiento

HAVING condiciones sobre lo agrupado

ORDER BY columnas clave de ordenamiento

Base de Datos de Ejemplo



SQL – Operador SELECT

SELECT *

FROM customer

Consulta todas las columnas de la tabla customer.

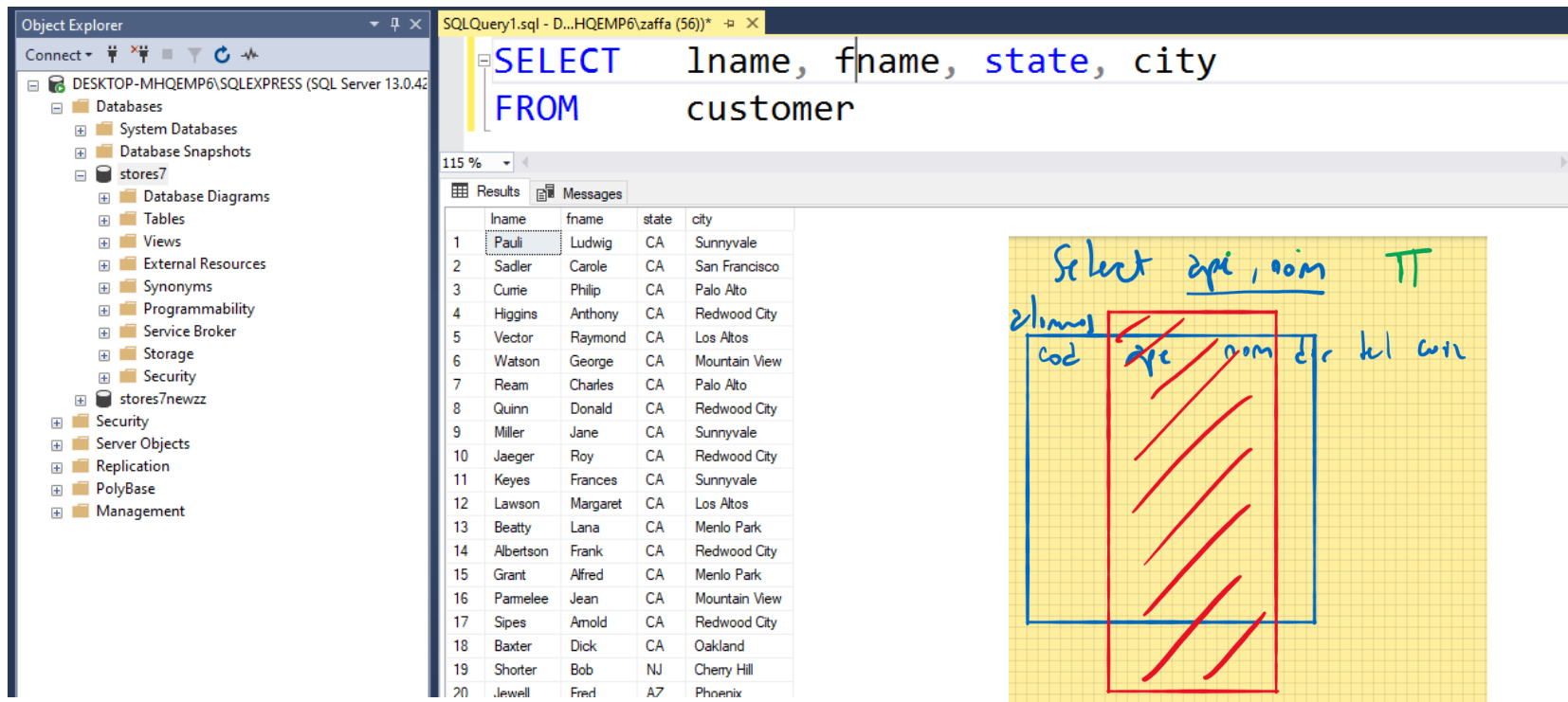
The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor displays the SQL statement: `SELECT * FROM customer;`. The Object Explorer on the left shows the database structure, with the 'customer' table selected under the 'stores7' database. The Results pane on the right displays the query output as a table with 19 rows and 11 columns. The status bar at the bottom indicates that the query was executed successfully, returning 28 rows.

	customer_num	fname	lname	company	address1	address2	city	state	zipcode	phone	customer_num_j
1	101	Ludwig	Pauli	All Sports Supplies	213 Erstwild Court		Sunnyvale	CA	94086	408-789-8075	NULL
2	102	Carole	Sadler	Sports Spot	785 Geary St		San Francisco	CA	94117	415-822-1289	101
3	103	Philip	Curie	Phil's Sports	654 Poplar	P. O. Box 3498	Palo Alto	CA	94303	415-328-4543	101
4	104	Anthony	Higgins	Play Ball!	East Shopping Cntr.	422 Bay Road	Redwood City	CA	94026	415-368-1100	103
5	105	Raymond	Vector	Los Altos Sports	1899 La Loma Drive		Los Altos	CA	94022	415-776-3249	103
6	106	George	Watson	Watson & Son	1143 Carver Place		Mountain View	CA	94063	415-389-8789	103
7	107	Charles	Ream	Athletic Supplies	41 Jordan Avenue		Palo Alto	CA	94304	415-356-9876	NULL
8	108	Donald	Quinn	Quinn's Sports	587 Alvarado		Redwood City	CA	94063	415-544-8729	107
9	109	Jane	Miller	Sport Stuff	Mayfair Mart	7345 Ross Blvd.	Sunnyvale	CA	94086	408-723-8789	107
10	110	Roy	Jaeger	AA Athletics	520 Topaz Way		Redwood City	CA	94062	415-743-3611	NULL
11	111	Frances	Keyes	Sports Center	3199 Sterling Court		Sunnyvale	CA	94085	408-277-7245	NULL
12	112	Margaret	Lawson	Runners & Others	234 Wyandotte Way		Los Altos	CA	94022	415-887-7235	111
13	113	Lana	Beatty	Sportstown	654 Oak Grove		Menlo Park	CA	94025	415-356-9982	111
14	114	Frank	Albertson	Sporting Place	947 Waverly Place		Redwood City	CA	94062	415-886-6677	111
15	115	Alfred	Grant	Gold Medal Sports	776 Gary Avenue		Menlo Park	CA	94025	415-356-1123	NULL
16	116	Jean	Pamelee	Olympic City	1104 Spinosa Drive		Mountain View	CA	94040	415-534-8822	115
17	117	Arnold	Sipes	Kids Komer	850 Lytton Court		Redwood City	CA	94063	415-245-4578	115
18	118	Dick	Baxter	Blue Ribbon Sports	5427 College		Oakland	CA	94609	415-655-0011	115
19	119	Bob	Shorter	The Triathletes Club	2405 Kings Highway		Cherry Hill	NJ	8002	609-663-6079	115

SQL – Operador SELECT

SELECT lname, fname, state, city
FROM customer

Consulta de distintas columnas pertenecientes a la tabla customer.



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'DESKTOP-MHQEMP6\SQLEXPRESS (SQL Server 13.0.42)'. The central pane shows a SQL query window with the following query:

```
SELECT lname, fname, state, city
FROM customer
```

Below the query window, the Results tab shows a table with 20 rows of data. The columns are lname, fname, state, and city. The first row is highlighted.

	lname	fname	state	city
1	Pauli	Ludwig	CA	Sunnyvale
2	Sadler	Carole	CA	San Francisco
3	Cumie	Philip	CA	Palo Alto
4	Higgins	Anthony	CA	Redwood City
5	Vector	Raymond	CA	Los Altos
6	Watson	George	CA	Mountain View
7	Ream	Charles	CA	Palo Alto
8	Quinn	Donald	CA	Redwood City
9	Miller	Jane	CA	Sunnyvale
10	Jaeger	Roy	CA	Redwood City
11	Keyes	Frances	CA	Sunnyvale
12	Lawson	Margaret	CA	Los Altos
13	Beatty	Lana	CA	Menlo Park
14	Albertson	Frank	CA	Redwood City
15	Grant	Alfred	CA	Menlo Park
16	Parmelee	Jean	CA	Mountain View
17	Sipes	Arnold	CA	Redwood City
18	Baxter	Dick	CA	Oakland
19	Shorter	Bob	NJ	Cherry Hill
20	Jewell	Fred	AZ	Phoenix

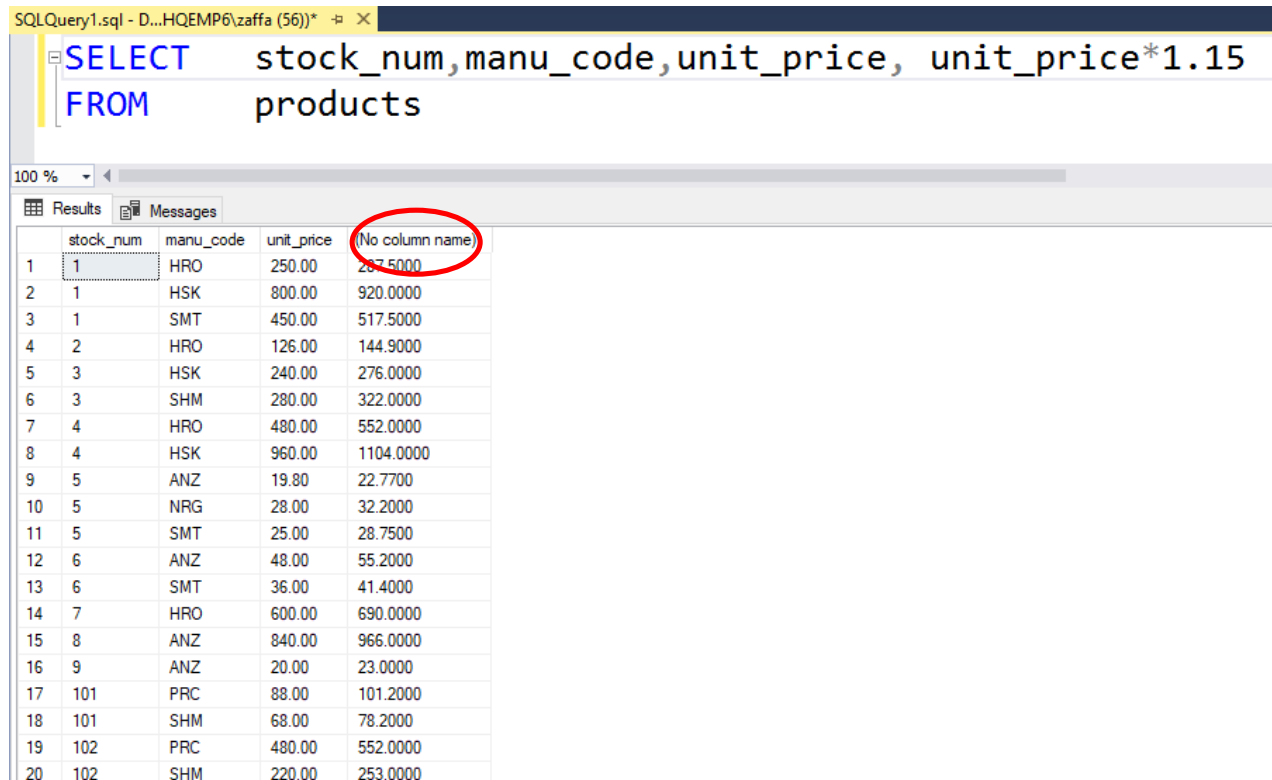
On the right side of the screenshot, there is a handwritten note on a yellow grid background. The note reads: 'Select lname, fname TT' and '21/11/2013'. Below this, there is a red box with diagonal lines and the text 'cod' and 'lname' written next to it.

SQL – Operador SELECT

SELECT stock_num,manu_code,unit_price, **unit_price*1.15**

FROM products

Consulta de distintas columnas con expresiones aritméticas.



The screenshot shows a SQL query window with the following text:

```
SELECT stock_num,manu_code,unit_price, unit_price*1.15
FROM products
```

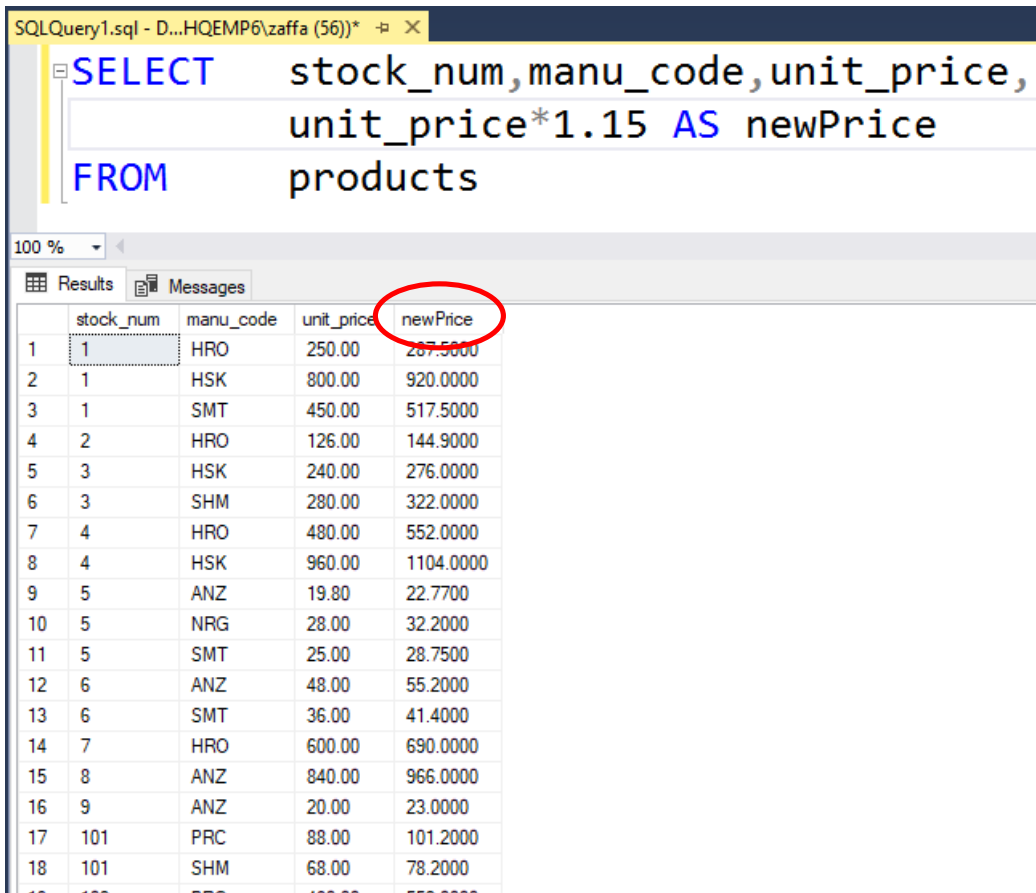
Below the query window, the 'Results' tab is active, displaying a table with 20 rows. The columns are 'stock_num', 'manu_code', 'unit_price', and an unnamed column for the calculated value. The first row is highlighted, and the calculated value '287.5000' is circled in red.

	stock_num	manu_code	unit_price	(No column name)
1	1	HRO	250.00	287.5000
2	1	HSK	800.00	920.0000
3	1	SMT	450.00	517.5000
4	2	HRO	126.00	144.9000
5	3	HSK	240.00	276.0000
6	3	SHM	280.00	322.0000
7	4	HRO	480.00	552.0000
8	4	HSK	960.00	1104.0000
9	5	ANZ	19.80	22.7700
10	5	NRG	28.00	32.2000
11	5	SMT	25.00	28.7500
12	6	ANZ	48.00	55.2000
13	6	SMT	36.00	41.4000
14	7	HRO	600.00	690.0000
15	8	ANZ	840.00	966.0000
16	9	ANZ	20.00	23.0000
17	101	PRC	88.00	101.2000
18	101	SHM	68.00	78.2000
19	102	PRC	480.00	552.0000
20	102	SHM	220.00	253.0000

SQL – Operador SELECT

SELECT stock_num,manu_code,unit_price, **unit_price*1.15 NewPrice**
FROM products

Alias de Columnas o
Etiquetas



SQLQuery1.sql - D:\HQEMP6\zaffa (56))

```
SELECT stock_num,manu_code,unit_price,  
unit_price*1.15 AS newPrice  
FROM products
```

100 %

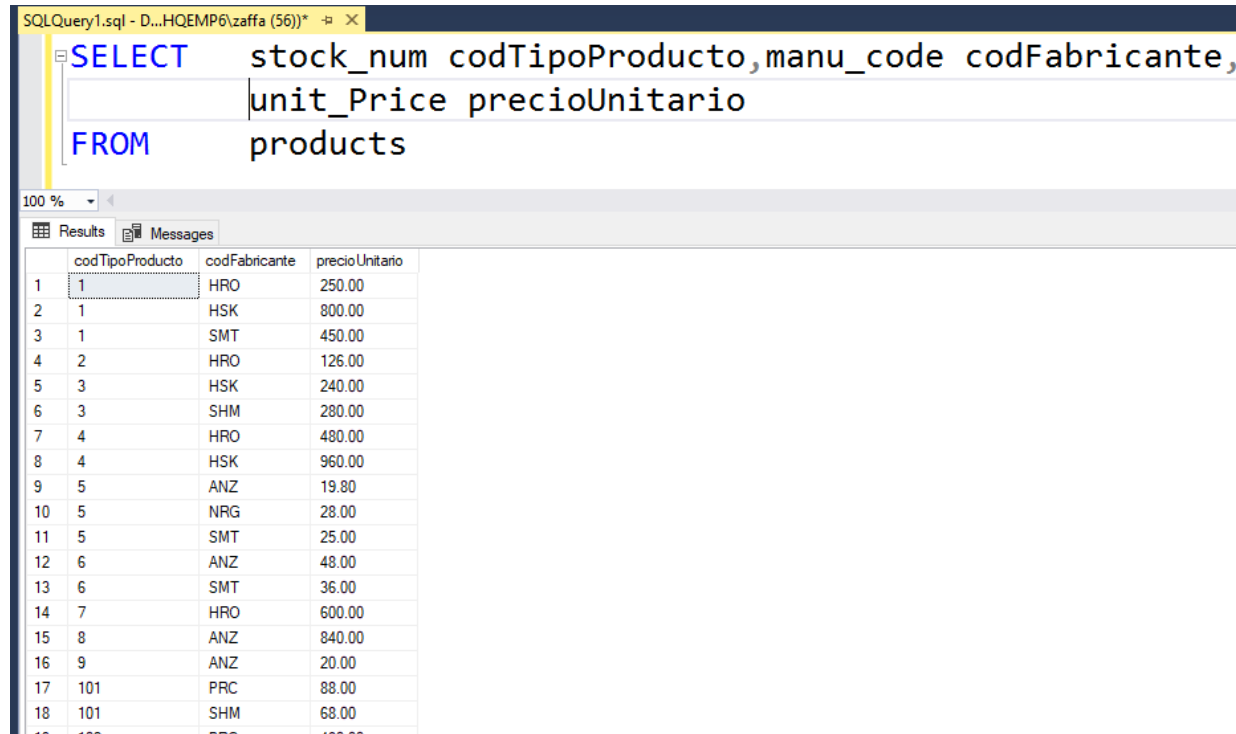
Results Messages

	stock_num	manu_code	unit_price	newPrice
1	1	HRO	250.00	277.5000
2	1	HSK	800.00	920.0000
3	1	SMT	450.00	517.5000
4	2	HRO	126.00	144.9000
5	3	HSK	240.00	276.0000
6	3	SHM	280.00	322.0000
7	4	HRO	480.00	552.0000
8	4	HSK	960.00	1104.0000
9	5	ANZ	19.80	22.7700
10	5	NRG	28.00	32.2000
11	5	SMT	25.00	28.7500
12	6	ANZ	48.00	55.2000
13	6	SMT	36.00	41.4000
14	7	HRO	600.00	690.0000
15	8	ANZ	840.00	966.0000
16	9	ANZ	20.00	23.0000
17	101	PRC	88.00	101.2000
18	101	SHM	68.00	78.2000
19	102	PRC	400.00	552.0000

SQL – Operador SELECT

SELECT stock_num codTipoProducto,manu_code codFabricante,
unit_Price precioUnitario
FROM products

Alias de Columnas o
Etiquetas



The screenshot shows a SQL query window with the following text:

```
SELECT stock_num codTipoProducto,manu_code codFabricante,  
unit_Price precioUnitario  
FROM products
```

Below the query, the 'Results' tab is active, displaying a table with 4 columns: 'codTipoProducto', 'codFabricante', and 'precioUnitario'. The first column is labeled '1' in the first row, which corresponds to the 'stock_num' column in the query. The table contains 18 rows of data.

	codTipoProducto	codFabricante	precioUnitario
1	1	HRO	250.00
2	1	HSK	800.00
3	1	SMT	450.00
4	2	HRO	126.00
5	3	HSK	240.00
6	3	SHM	280.00
7	4	HRO	480.00
8	4	HSK	960.00
9	5	ANZ	19.80
10	5	NRG	28.00
11	5	SMT	25.00
12	6	ANZ	48.00
13	6	SMT	36.00
14	7	HRO	600.00
15	8	ANZ	840.00
16	9	ANZ	20.00
17	101	PRC	88.00
18	101	SHM	68.00

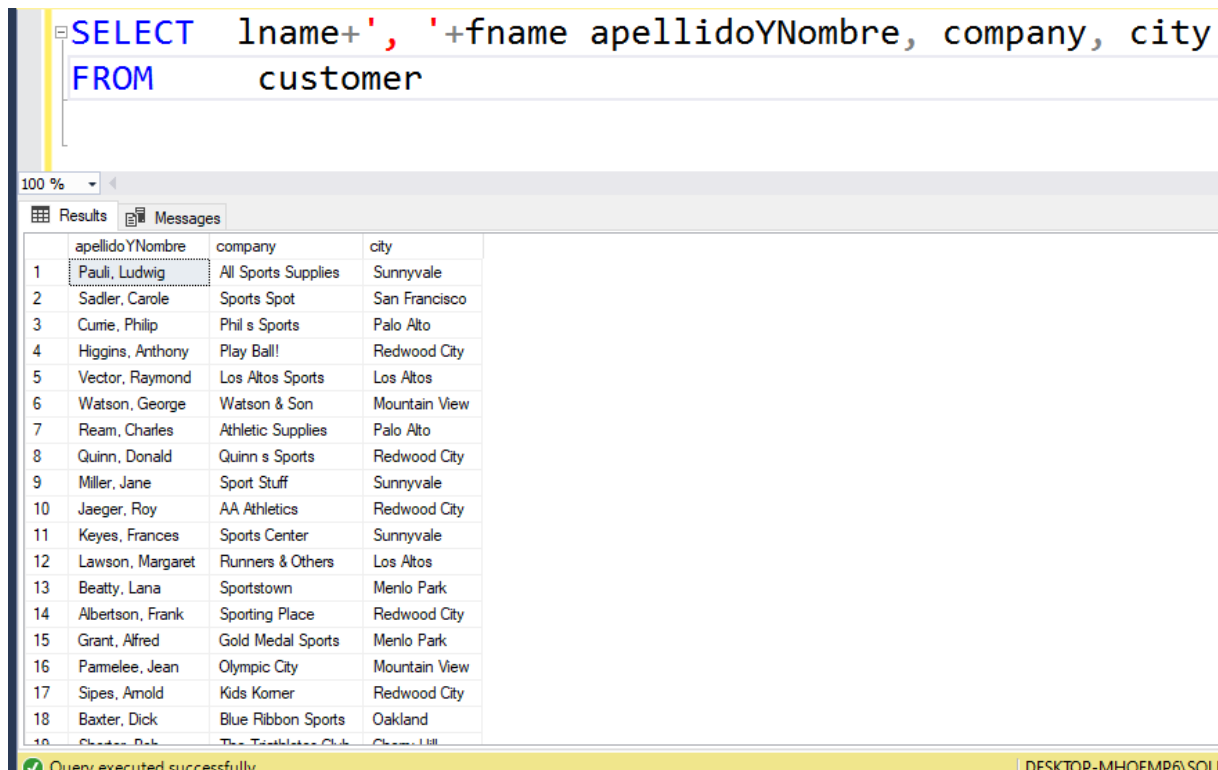
Se puede utilizar
también para mostrar
columnas con otros
nombres

SQL – Operador SELECT

SELECT lname + ', ' + fname apellidoYNombre, company, city
FROM customer

Concatenar columnas en una sola nueva columna de salida.

Además en el ejemplo le ponemos el alias apellidoYNombre



```
SELECT lname+', '+fname apellidoYNombre, company, city
FROM customer
```

	apellidoYNombre	company	city
1	Pauli, Ludwig	All Sports Supplies	Sunnyvale
2	Sadler, Carole	Sports Spot	San Francisco
3	Cumie, Philip	Phil's Sports	Palo Alto
4	Higgins, Anthony	Play Ball!	Redwood City
5	Vector, Raymond	Los Altos Sports	Los Altos
6	Watson, George	Watson & Son	Mountain View
7	Ream, Charles	Athletic Supplies	Palo Alto
8	Quinn, Donald	Quinn's Sports	Redwood City
9	Miller, Jane	Sport Stuff	Sunnyvale
10	Jaeger, Roy	AA Athletics	Redwood City
11	Keyes, Frances	Sports Center	Sunnyvale
12	Lawson, Margaret	Runners & Others	Los Altos
13	Beatty, Lana	Sportstown	Menlo Park
14	Albertson, Frank	Sporting Place	Redwood City
15	Grant, Alfred	Gold Medal Sports	Menlo Park
16	Parmelee, Jean	Olympic City	Mountain View
17	Sipes, Arnold	Kids Komer	Redwood City
18	Baxter, Dick	Blue Ribbon Sports	Oakland
19	Chase, Bob	The Tacklers Club	Chapel Hill

SQL – Operador SELECT

```
SELECT order_num, order_date, YEAR(order_date) anio,  
        MONTH(order_date) mes, DAY(order_date) dia,  
        USER_NAME()
```

```
FROM orders
```

```
SELECT order_num, order_date, YEAR(order_date) anio,  
       MONTH(order_date) mes, DAY(order_date) dia, USER_NAME()  
FROM orders
```

	order_num	order_date	anio	mes	dia	(No column name)
1	1001	2015-05-16 00:00:00.000	2015	5	16	dbo
2	1002	2015-05-17 00:00:00.000	2015	5	17	dbo
3	1003	2015-05-18 00:00:00.000	2015	5	18	dbo
4	1004	2015-05-18 00:00:00.000	2015	5	18	dbo
5	1005	2015-05-20 00:00:00.000	2015	5	20	dbo
6	1006	2015-05-26 00:00:00.000	2015	5	26	dbo
7	1007	2015-05-27 00:00:00.000	2015	5	27	dbo
8	1008	2015-06-03 00:00:00.000	2015	6	3	dbo
9	1009	2015-06-10 00:00:00.000	2015	6	10	dbo
10	1010	2015-06-13 00:00:00.000	2015	6	13	dbo
11	1011	2015-06-14 00:00:00.000	2015	6	14	dbo
12	1012	2015-06-14 00:00:00.000	2015	6	14	dbo
13	1013	2015-06-18 00:00:00.000	2015	6	18	dbo
14	1014	2015-06-21 00:00:00.000	2015	6	21	dbo
15	1015	2015-06-23 00:00:00.000	2015	6	23	dbo
16	1016	2015-06-25 00:00:00.000	2015	6	25	dbo
17	1017	2015-07-05 00:00:00.000	2015	7	5	dbo
18	1018	2015-07-06 00:00:00.000	2015	7	6	dbo
19	1019	2015-07-07 00:00:00.000	2015	7	7	dbo

Utilización de
funciones especiales.

ARITMETICAS
TRIGONOMETRICAS
FINANCIERAS
DE FECHA
DE STRINGS
Entre otras..

SQL – Operador SELECT

SELECT order_num, order_date, **YEAR**(order_date) anio,
MONTH(order_date) mes, **DAY**(order_date) dia,
USER_NAME()

FROM orders

WHERE condiciones

Condiciones

AND, OR, NOT

=

<> !=

>, >=

<, <=

[NOT] LIKE

[NOT] BETWEEN

[NOT] IN

IS [NOT] NULL

igualdad

distinto

mayor, mayor igual

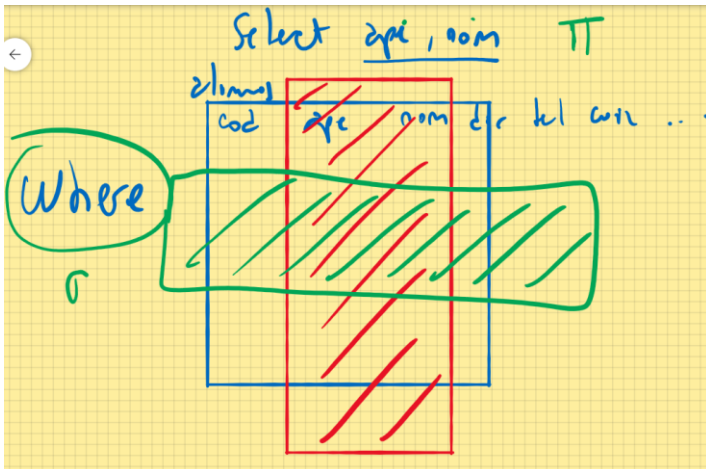
menor, menor igual

validar substrings

entre rango

en lista de valores

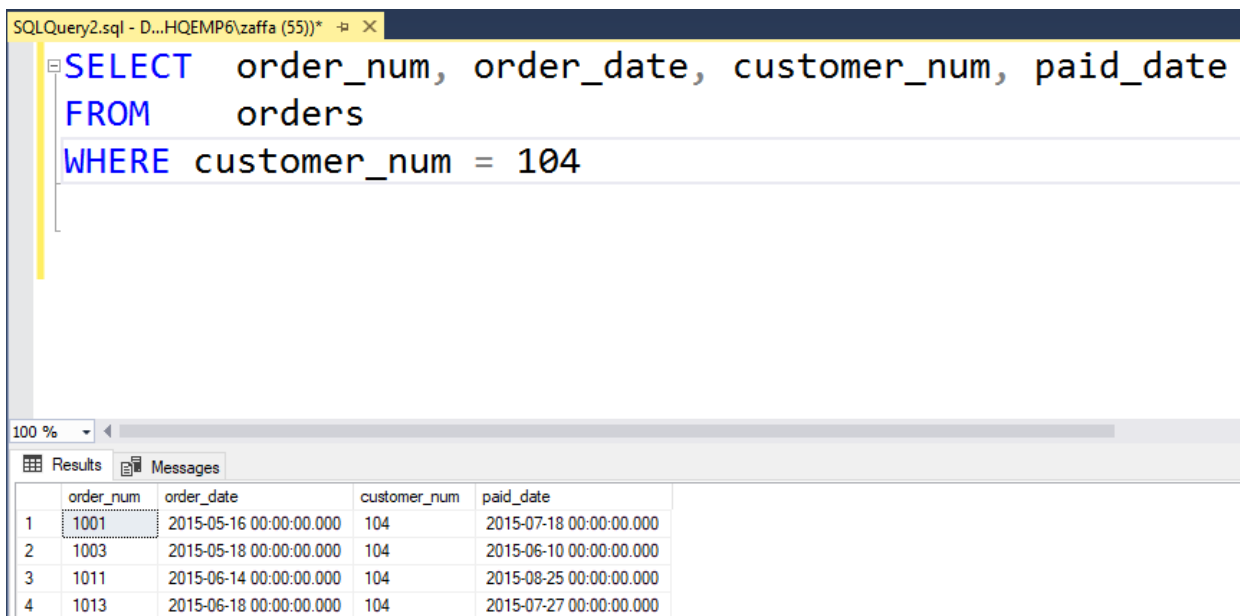
es o no es nulo



SQL – Operador SELECT

SELECT order_num, order_date, customer_num, paid_date
FROM orders
WHERE customer_num=104

Condiciones por igualdad.



The screenshot shows a SQL query editor window titled 'SQLQuery2.sql - D...HQEMP6\zaffa (55))'. The query is: `SELECT order_num, order_date, customer_num, paid_date FROM orders WHERE customer_num = 104`. Below the query, the 'Results' tab is active, displaying a table with 4 rows and 4 columns: order_num, order_date, customer_num, and paid_date. The first row is highlighted with a mouse cursor.

	order_num	order_date	customer_num	paid_date
1	1001	2015-05-16 00:00:00.000	104	2015-07-18 00:00:00.000
2	1003	2015-05-18 00:00:00.000	104	2015-06-10 00:00:00.000
3	1011	2015-06-14 00:00:00.000	104	2015-08-25 00:00:00.000
4	1013	2015-06-18 00:00:00.000	104	2015-07-27 00:00:00.000

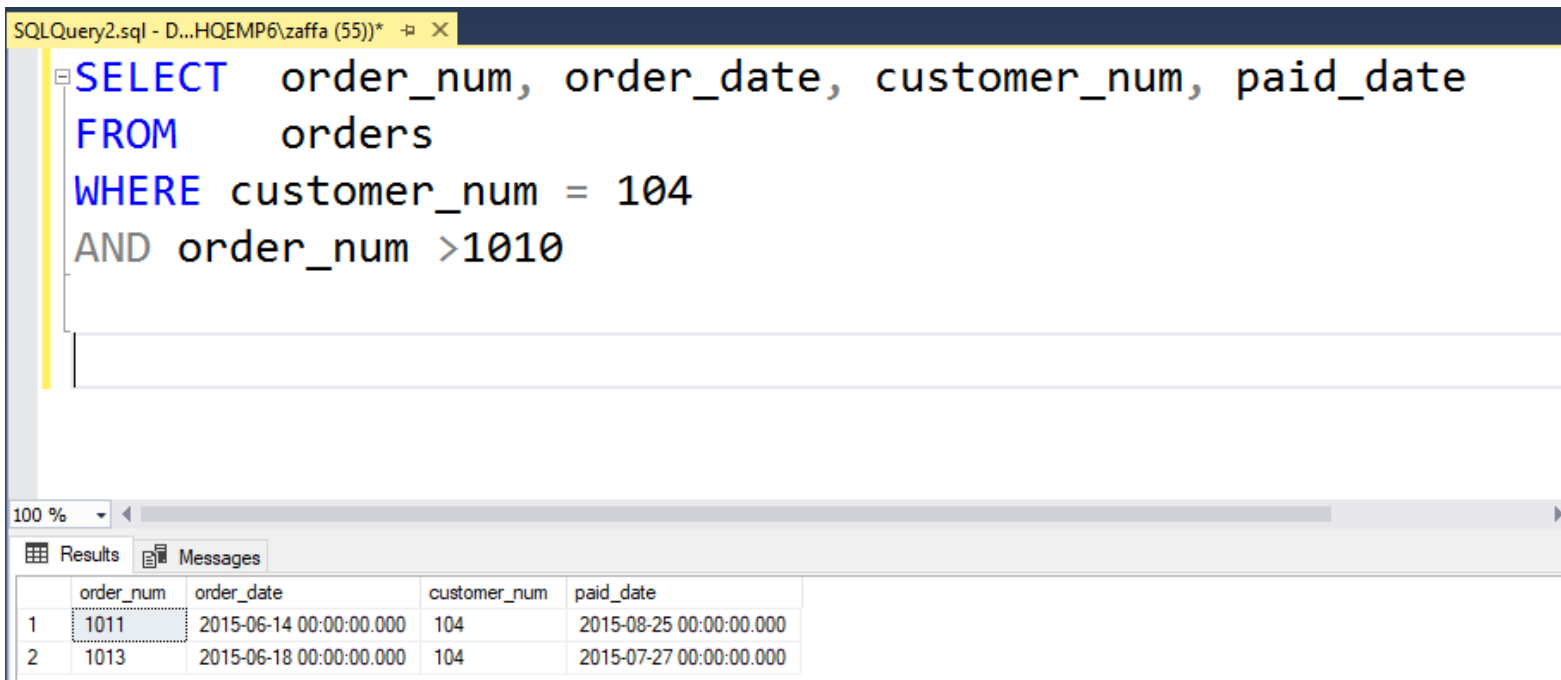
SQL – Operador SELECT

SELECT order_num, order_date, customer_num, paid_date

FROM orders

WHERE customer_num=104
AND order_num >1010

Varias Condiciones
con AND.



The screenshot shows a SQL query editor window titled "SQLQuery2.sql - D...HQEMP6\zaffa (55))". The query text is as follows:

```
SELECT order_num, order_date, customer_num, paid_date
FROM orders
WHERE customer_num = 104
AND order_num >1010
```

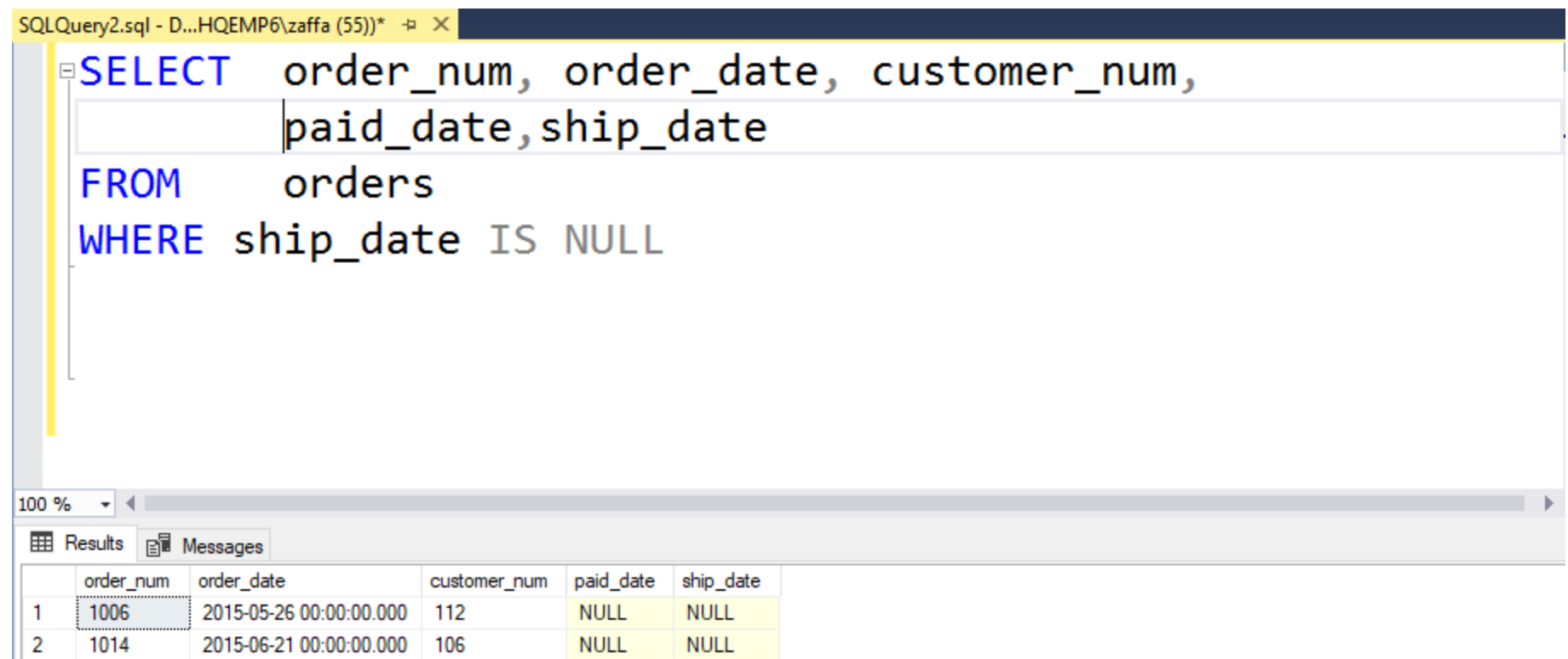
Below the query editor, the "Results" tab is active, displaying a table with the following data:

	order_num	order_date	customer_num	paid_date
1	1011	2015-06-14 00:00:00.000	104	2015-08-25 00:00:00.000
2	1013	2015-06-18 00:00:00.000	104	2015-07-27 00:00:00.000

SQL – Operador SELECT

SELECT order_num, order_date, customer_num, paid_date, ship_date
FROM orders
WHERE ship_date IS NULL

Condición por columnas con Nulos.



The screenshot shows a SQL query window with the following text:

```
SELECT order_num, order_date, customer_num,  
       paid_date, ship_date  
FROM   orders  
WHERE  ship_date IS NULL
```

Below the query window, the 'Results' tab is active, displaying a table with 6 columns: order_num, order_date, customer_num, paid_date, and ship_date. The table contains 2 rows of data.

	order_num	order_date	customer_num	paid_date	ship_date
1	1006	2015-05-26 00:00:00.000	112	NULL	NULL
2	1014	2015-06-21 00:00:00.000	106	NULL	NULL

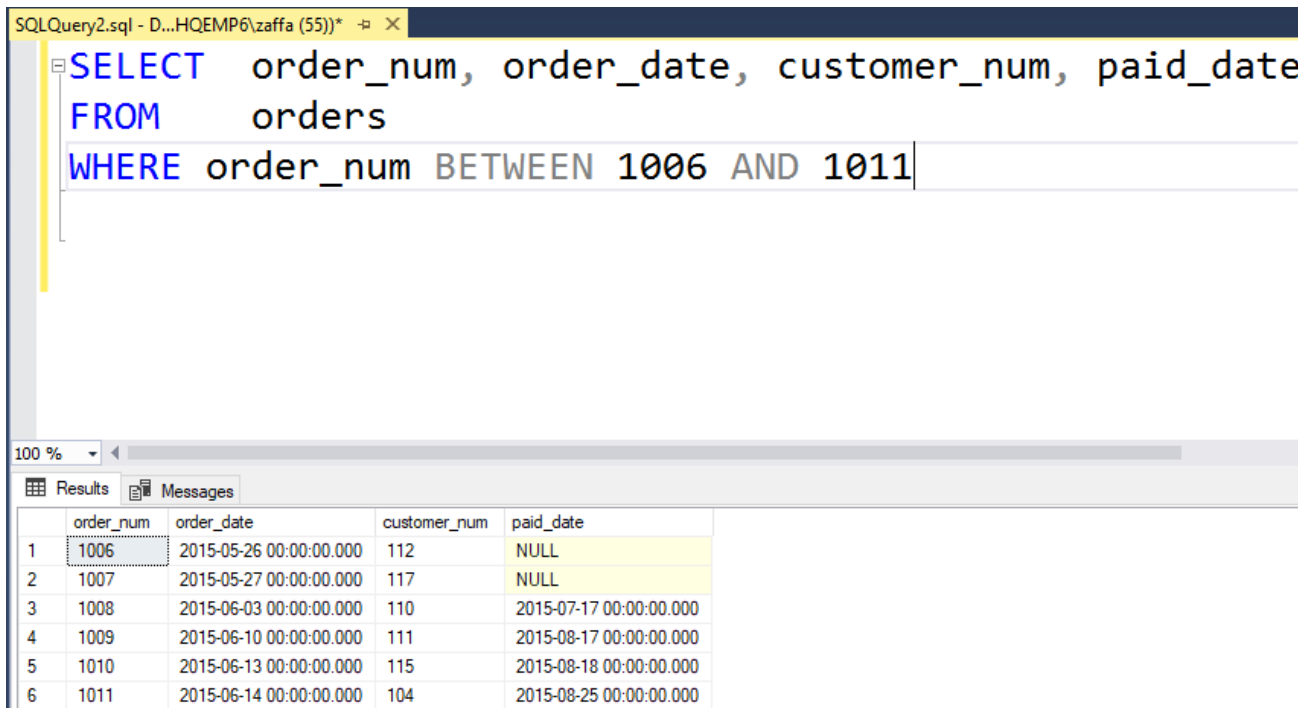
SQL – Operador SELECT

SELECT order_num, order_date, customer_num, paid_date

FROM orders

WHERE order_num BETWEEN 1004 AND 1020

Condición por un rango de valores para una columna.



```
SQLQuery2.sql - D...HQEMP6\zaffa (55))*  
SELECT order_num, order_date, customer_num, paid_date  
FROM orders  
WHERE order_num BETWEEN 1006 AND 1011
```

	order_num	order_date	customer_num	paid_date
1	1006	2015-05-26 00:00:00.000	112	NULL
2	1007	2015-05-27 00:00:00.000	117	NULL
3	1008	2015-06-03 00:00:00.000	110	2015-07-17 00:00:00.000
4	1009	2015-06-10 00:00:00.000	111	2015-08-17 00:00:00.000
5	1010	2015-06-13 00:00:00.000	115	2015-08-18 00:00:00.000
6	1011	2015-06-14 00:00:00.000	104	2015-08-25 00:00:00.000

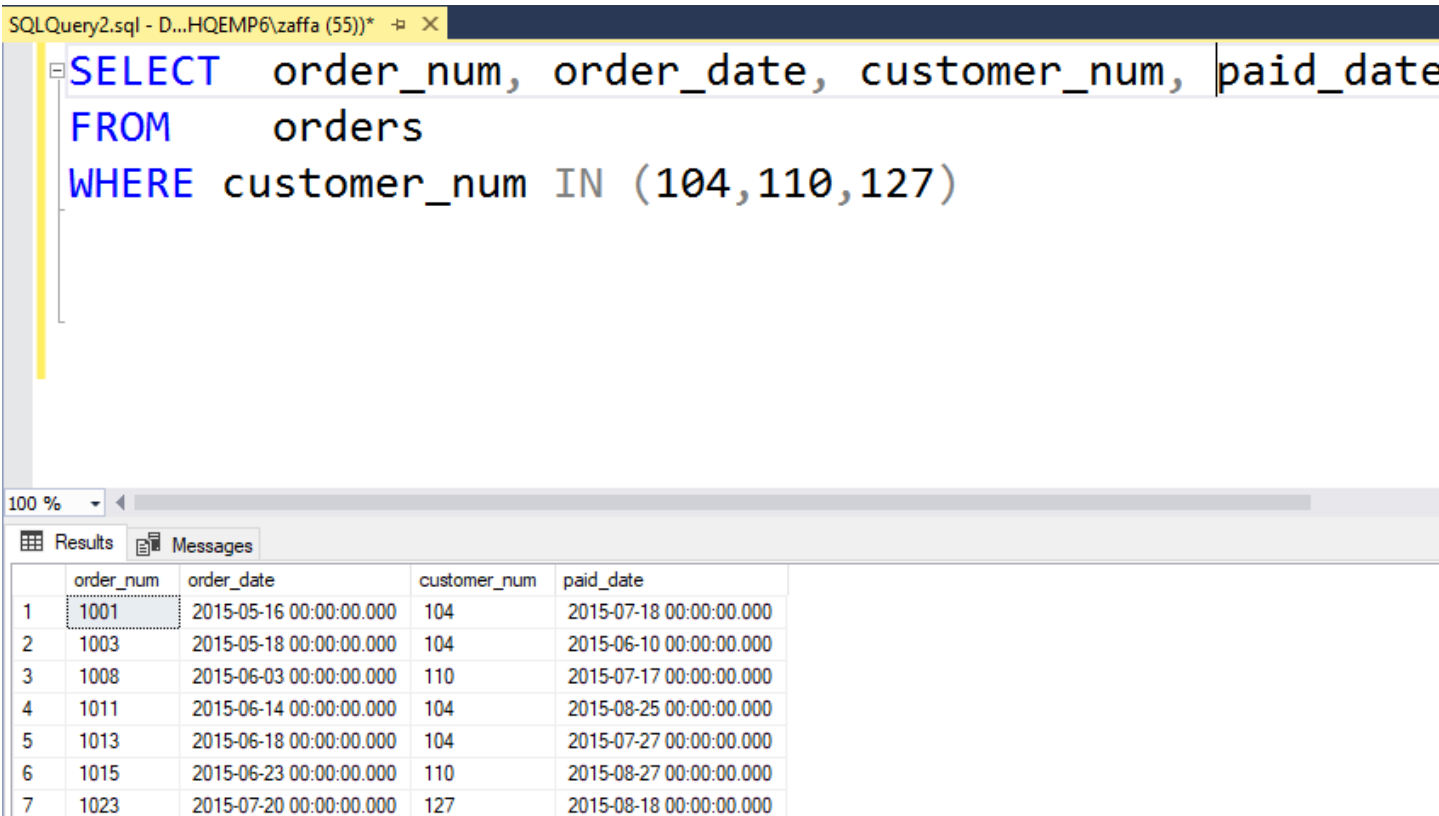
SQL – Operador SELECT

SELECT order_num, order_date, customer_num, paid_date

FROM orders

WHERE customer_num IN (104,110,127)

Condición de una columna por una lista de valores



The screenshot shows a SQL query window with the following text:

```
SELECT order_num, order_date, customer_num, paid_date
FROM orders
WHERE customer_num IN (104,110,127)
```

Below the query window, the 'Results' tab is active, displaying a table with 7 rows and 4 columns: order_num, order_date, customer_num, and paid_date. The first row is highlighted with a dashed border.

	order_num	order_date	customer_num	paid_date
1	1001	2015-05-16 00:00:00.000	104	2015-07-18 00:00:00.000
2	1003	2015-05-18 00:00:00.000	104	2015-06-10 00:00:00.000
3	1008	2015-06-03 00:00:00.000	110	2015-07-17 00:00:00.000
4	1011	2015-06-14 00:00:00.000	104	2015-08-25 00:00:00.000
5	1013	2015-06-18 00:00:00.000	104	2015-07-27 00:00:00.000
6	1015	2015-06-23 00:00:00.000	110	2015-08-27 00:00:00.000
7	1023	2015-07-20 00:00:00.000	127	2015-08-18 00:00:00.000

SQL – Operador SELECT

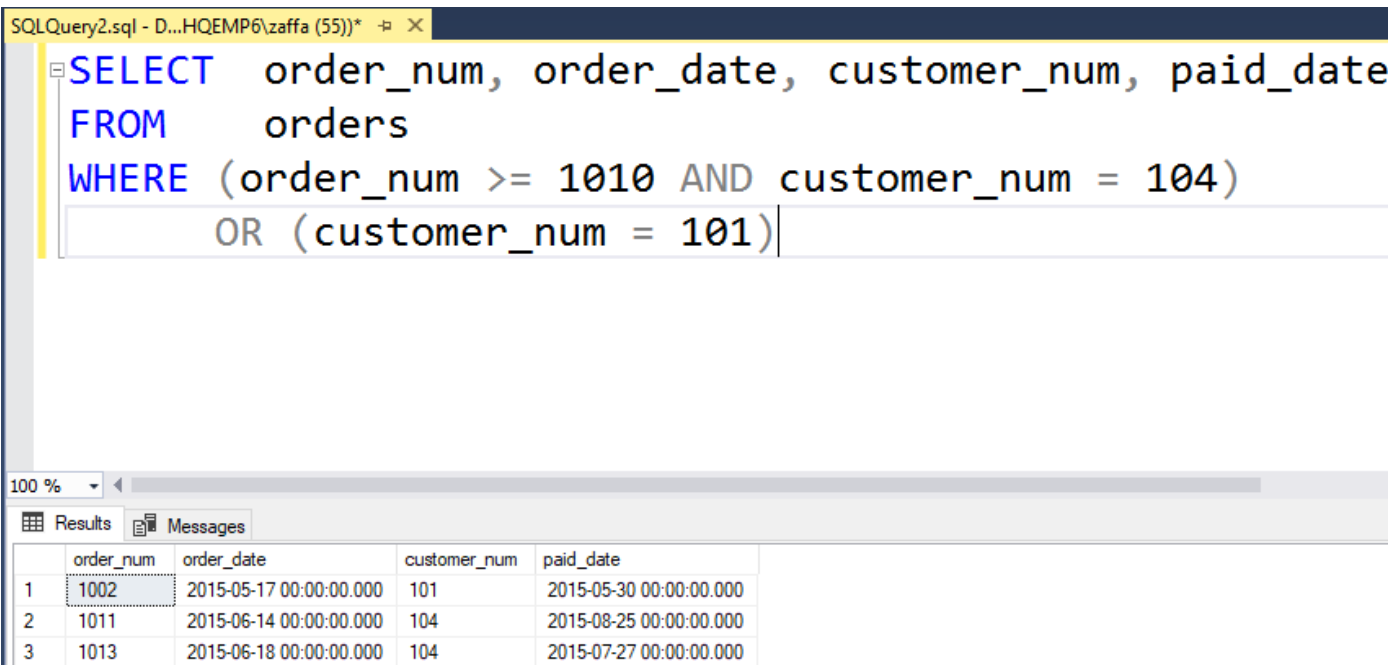
SELECT order_num, order_date, customer_num, paid_date

FROM orders

WHERE (order_num >= 1010 AND customer_num = 104)

OR (customer_num = 101)

Condiciones con AND
y OR.



```
SQLQuery2.sql - D...HQEMP6\zaffa (55))* X
```

```
SELECT order_num, order_date, customer_num, paid_date
FROM orders
WHERE (order_num >= 1010 AND customer_num = 104)
      OR (customer_num = 101)
```

	order_num	order_date	customer_num	paid_date
1	1002	2015-05-17 00:00:00.000	101	2015-05-30 00:00:00.000
2	1011	2015-06-14 00:00:00.000	104	2015-08-25 00:00:00.000
3	1013	2015-06-18 00:00:00.000	104	2015-07-27 00:00:00.000

SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE condiciones

Condiciones con operador LIKE

lname LIKE 'A%' apellidos que comiencen con 'A'.

lname LIKE '%th%' apellidos que contenga 'th' en cualquier parte.

lname LIKE 'A___' apellidos que comiencen con 'A' y tengan 4 letras

lname LIKE '[AE]%' apellidos que comiencen con 'A' ó 'E'.

lname LIKE '[A-E]%' apellidos que comiencen entre la 'A' y la 'E'.

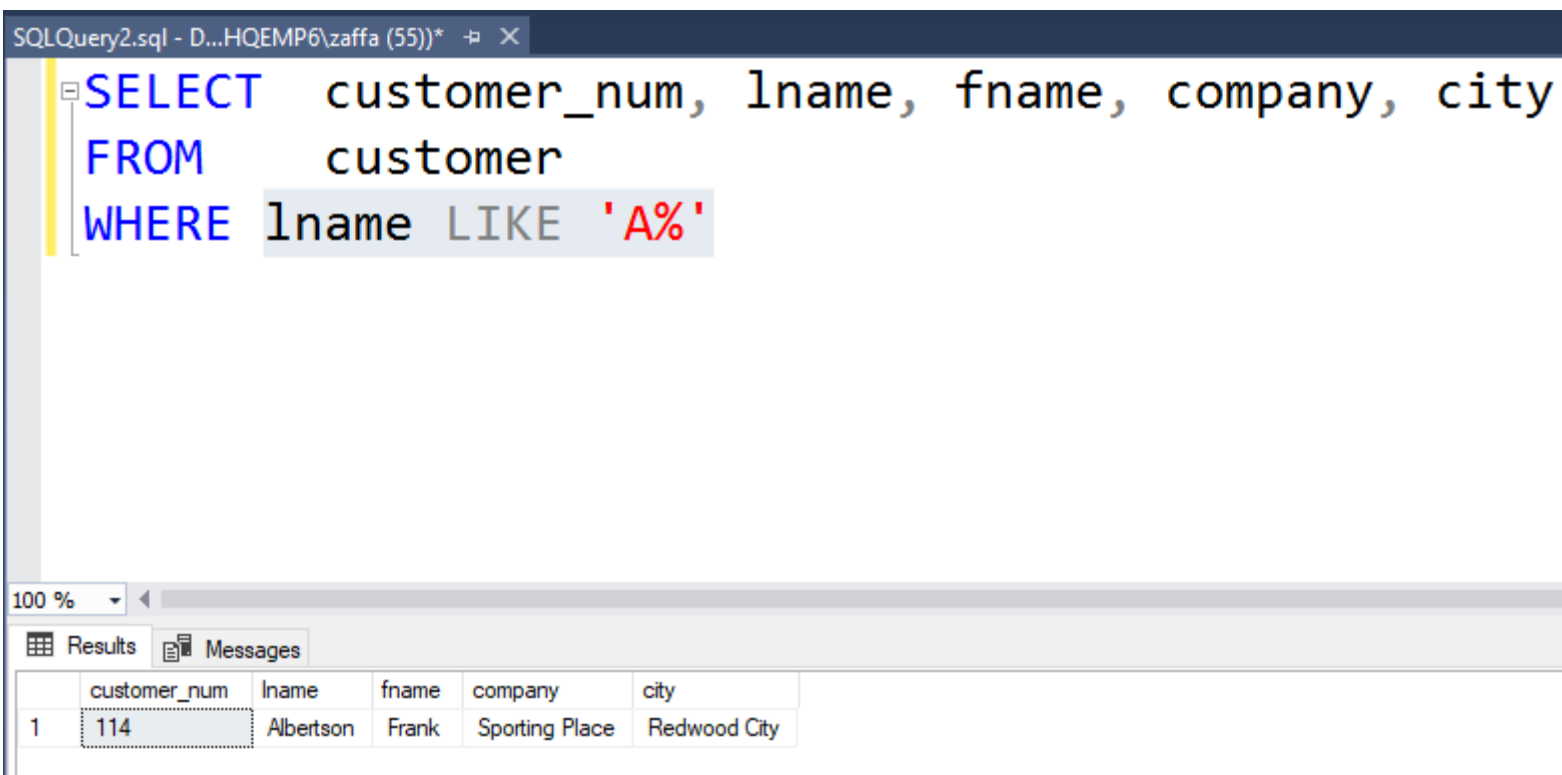
SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE 'A%'

% reemplaza a 0 o más caracteres.



The screenshot shows a SQL query editor window titled 'SQLQuery2.sql - D...HQEMP6\zaffa (55))' with the following SQL code:

```
SELECT customer_num, lname, fname, company, city
FROM customer
WHERE lname LIKE 'A%'
```

Below the editor is a results window with a 'Results' tab selected. It displays a single row of data:

	customer_num	lname	fname	company	city
1	114	Albertson	Frank	Sporting Place	Redwood City

SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE '%i%'

% reemplaza a 0 o más caracteres.

```
SQLQuery2.sql - D...HQEMP6\zaffa (55)) * X
SELECT customer_num, lname, fname, company, city
FROM customer
WHERE lname LIKE '%i%'
```

En este caso muestra todos los clientes cuyo apellido contenga una i, en cualquier lado.

100 %					
Results Messages					
	customer_num	lname	fname	company	city
1	101	Pauli	Ludwig	All Sports Supplies	Sunnyvale
2	103	Curie	Philip	Phil s Sports	Palo Alto
3	104	Higgins	Anthony	Play Ball!	Redwood City
4	108	Quinn	Donald	Quinn s Sports	Redwood City
5	109	Miller	Jane	Sport Stuff	Sunnyvale
6	117	Sipes	Arnold	Kids Komer	Redwood City
7	122	O Brian	Cathy	The Sporting Life	Princeton
8	126	Neelie	Eileen	Neelie s Discount Sp	Denver
9	127	Satifer	Kim	Big Blue Bike Shop	Blue Island

SQL – Operador SELECT

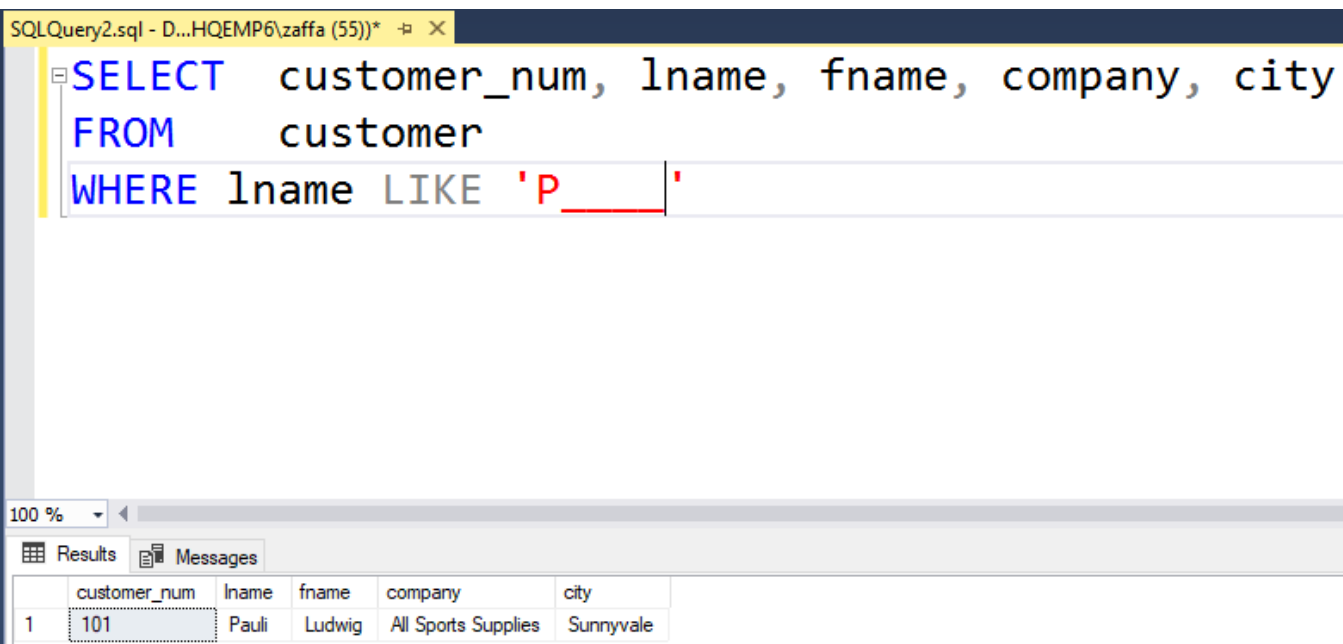
SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE 'P____'

Reemplaza a 1 sólo caracteres.

En este caso muestra todos los clientes cuyo apellido comience con P y tenga 5 letras.



The screenshot shows a SQL query editor window titled 'SQLQuery2.sql - D:\...HQUEMP6\zaffa (55))' with a query that filters customers by last name starting with 'P'. Below the query, the 'Results' tab is active, displaying a single row of data for customer 101.

```
SELECT customer_num, lname, fname, company, city
FROM customer
WHERE lname LIKE 'P____'
```

	customer_num	lname	fname	company	city
1	101	Pauli	Ludwig	All Sports Supplies	Sunnyvale

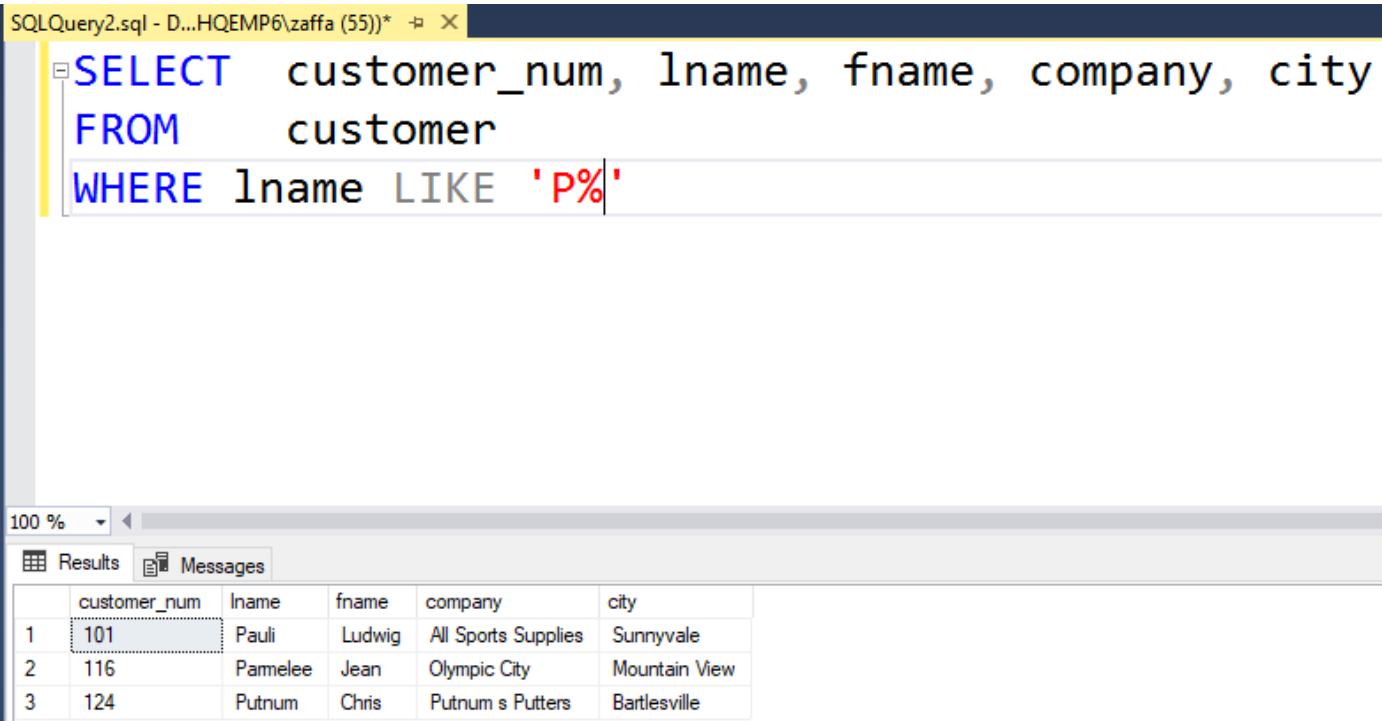
SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE 'P%'

Diferencia entre _ y %



```
SQLQuery2.sql - D...HQEMP6\zaffa (55))*  
SELECT customer_num, lname, fname, company, city  
FROM customer  
WHERE lname LIKE 'P%'
```

	customer_num	lname	fname	company	city
1	101	Pauli	Ludwig	All Sports Supplies	Sunnyvale
2	116	Pamelee	Jean	Olympic City	Mountain View
3	124	Putnum	Chris	Putnum s Putters	Bartlesville

En este caso muestra todos los clientes cuyo apellido comience con P, sin importar la cantidad de letras.

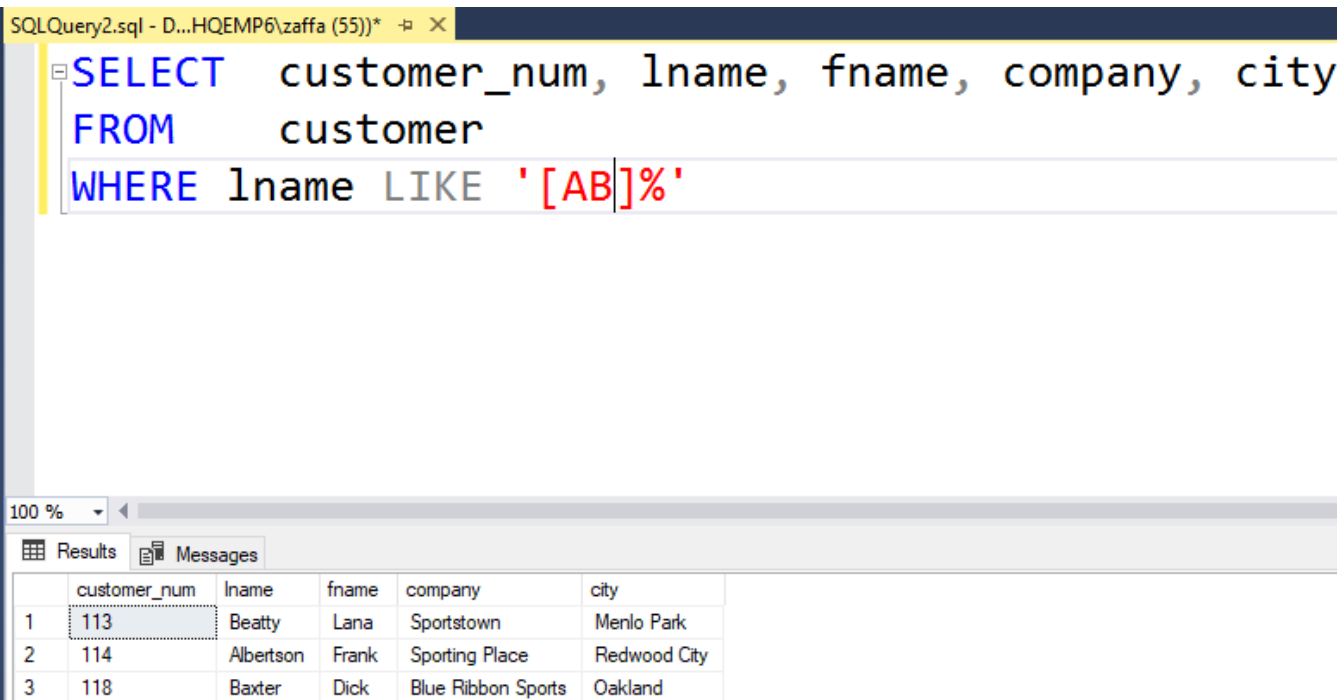
SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE '[AB]%'

[] Evalúa un sólo carácter.



SQLQuery2.sql - D:\HQEMP6\zaffa (55))*

```
SELECT customer_num, lname, fname, company, city
FROM customer
WHERE lname LIKE '[AB]%'
```

100 %

Results Messages

	customer_num	lname	fname	company	city
1	113	Beatty	Lana	Sportstown	Menlo Park
2	114	Albertson	Frank	Sporting Place	Redwood City
3	118	Baxter	Dick	Blue Ribbon Sports	Oakland

En este caso muestra todos los clientes cuyo apellido comience con A ó con B.

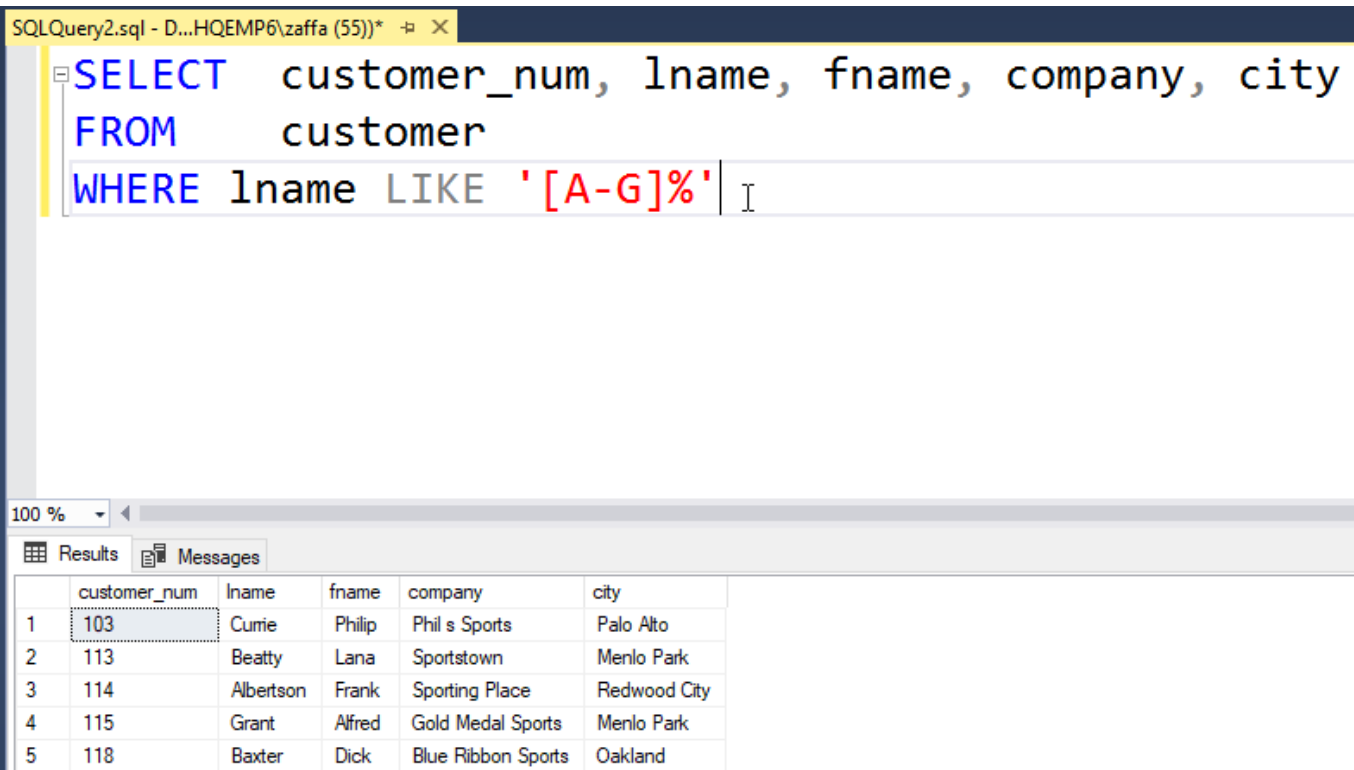
SQL – Operador SELECT

SELECT customer_num, lname, fname, company, city

FROM customer

WHERE lname LIKE '[A-G]%'

[] Evalúa un sólo carácter.



SQLQuery2.sql - D:\HQEMP6\zaffa (55))*

```
SELECT customer_num, lname, fname, company, city
FROM customer
WHERE lname LIKE '[A-G]%'
```

100 %

Results Messages

	customer_num	lname	fname	company	city
1	103	Curie	Philip	Phil's Sports	Palo Alto
2	113	Beatty	Lana	Sportstown	Menlo Park
3	114	Albertson	Frank	Sporting Place	Redwood City
4	115	Grant	Alfred	Gold Medal Sports	Menlo Park
5	118	Baxter	Dick	Blue Ribbon Sports	Oakland

En este caso muestra todos los clientes cuyo apellido comience en el rango de A hasta la G inclusive.

VAMOS A UN KAHOOT

Break



SQL – Operador SELECT

Cláusula ORDER BY

SELECT * | lista de columnas

FROM nom_tabla | lista de tablas

WHERE condiciones ó filtros

[GROUP BY columnas clave de agrupamiento

HAVING condiciones sobre lo agrupado]

ORDER BY columnas clave de ordenamiento

SQL – Operador SELECT

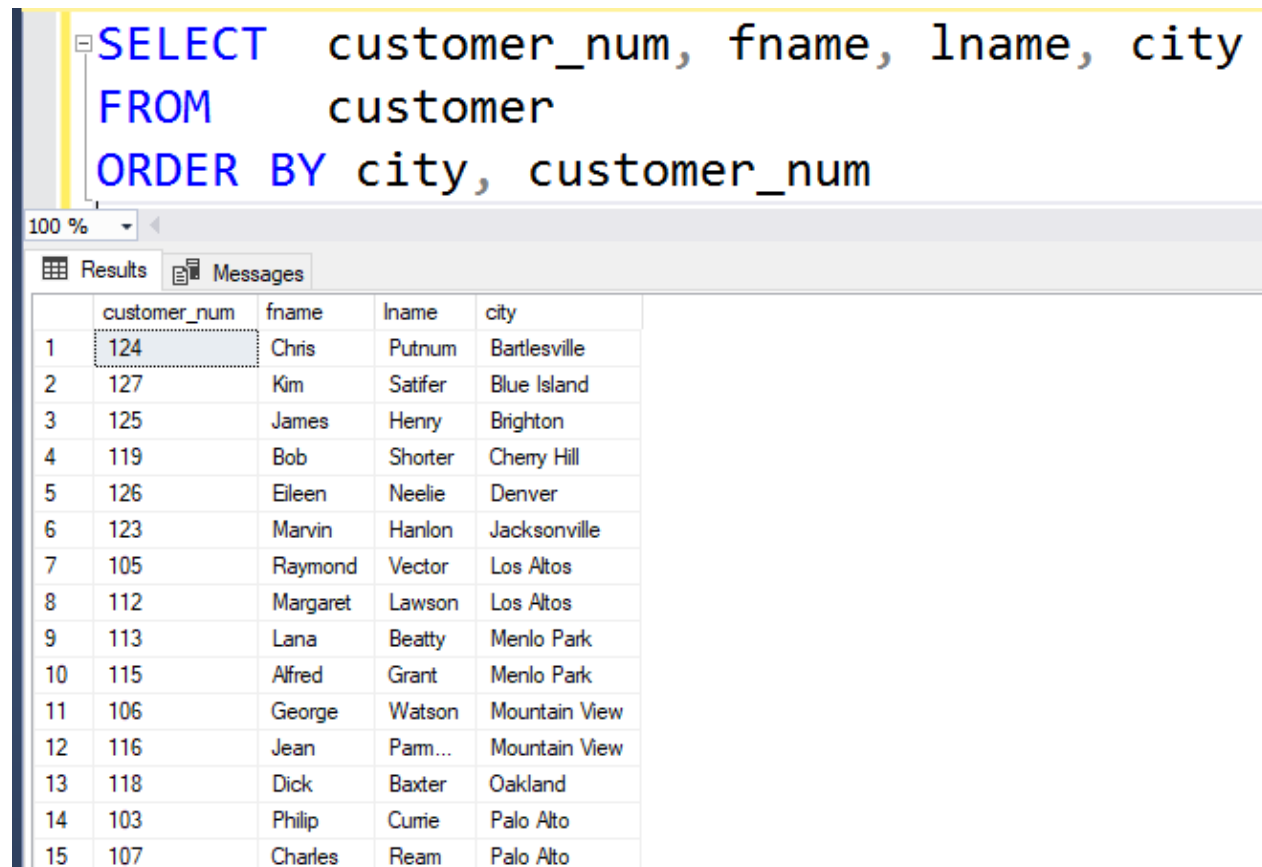
SELECT customer_num, fname, lname, city

FROM customer

ORDER BY city, customer_num

Ordenamiento del resultado de la consulta por una clave o múltiples claves.

```
SELECT customer_num, fname, lname, city
FROM customer
ORDER BY city, customer_num
```



	customer_num	fname	lname	city
1	124	Chris	Putnum	Bartlesville
2	127	Kim	Satifer	Blue Island
3	125	James	Henry	Brighton
4	119	Bob	Shorter	Cherry Hill
5	126	Eileen	Neelie	Denver
6	123	Marvin	Hanlon	Jacksonville
7	105	Raymond	Vector	Los Altos
8	112	Margaret	Lawson	Los Altos
9	113	Lana	Beatty	Menlo Park
10	115	Alfred	Grant	Menlo Park
11	106	George	Watson	Mountain View
12	116	Jean	Parm...	Mountain View
13	118	Dick	Baxter	Oakland
14	103	Philip	Cumie	Palo Alto
15	107	Charles	Ream	Palo Alto

Observamos que las filas están ordenadas por ciudad ascendente y a igual ciudad ordena por customer_num también ascendente.

SQL – Operador SELECT

SELECT customer_num, fname, lname, city

FROM customer

ORDER BY city, customer_num **DESC**

Ordenamiento del resultado de la consulta por una clave o múltiples claves.

```
SELECT customer_num, fname, lname, city
FROM customer
ORDER BY city, customer_num DESC
```

Observamos que las filas están ordenadas por ciudad ASCENDENTE (default) y a igual ciudad ordena por customer_num DESCENDENTE.

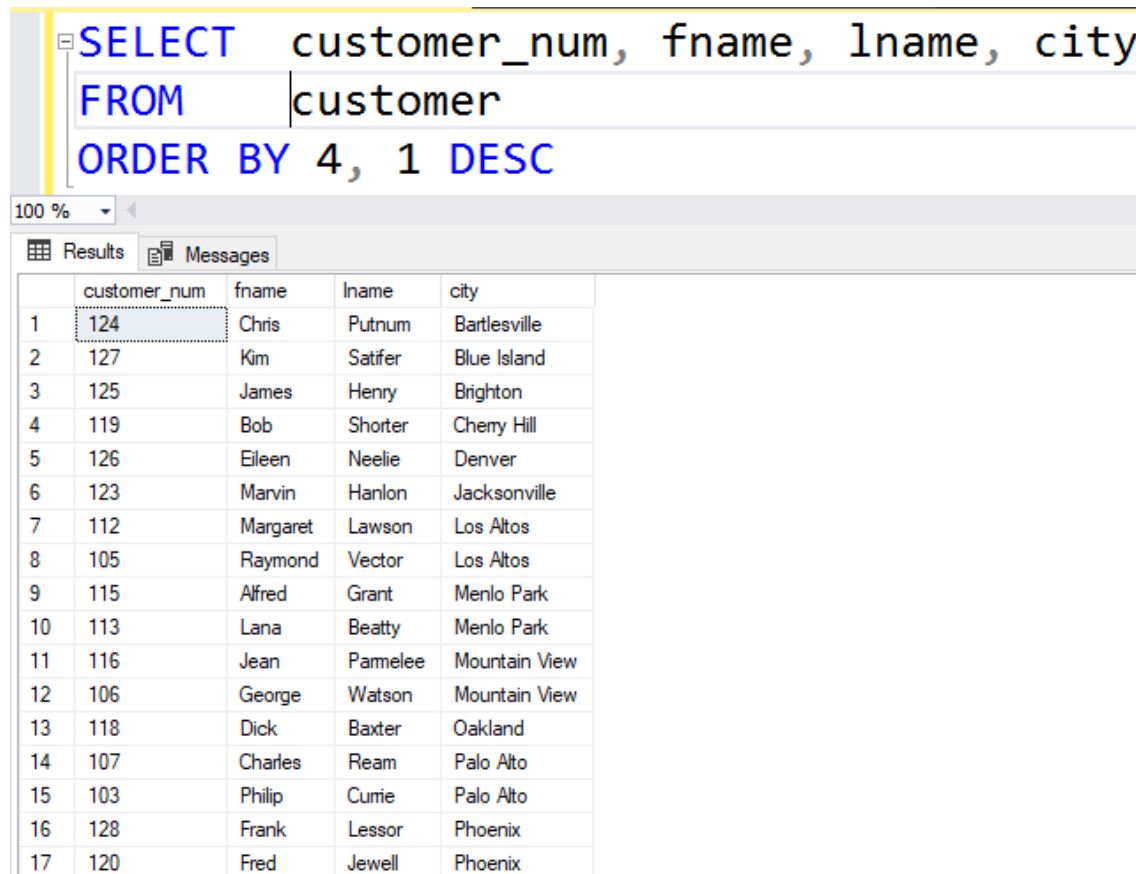
	customer_num	fname	lname	city
1	124	Chris	Putnum	Bartlesville
2	127	Kim	Satifer	Blue Island
3	125	James	Henry	Brighton
4	119	Bob	Shorter	Cherry Hill
5	126	Eileen	Neelie	Denver
6	123	Marvin	Harlon	Jacksonville
7	112	Margaret	Lawson	Los Altos
8	105	Raymond	Vector	Los Altos
9	115	Alfred	Grant	Menlo Park
10	113	Lana	Beatty	Menlo Park
11	116	Jean	Pamelee	Mountain View
12	106	George	Watson	Mountain View
13	118	Dick	Baxter	Oakland
14	107	Charles	Ream	Palo Alto
15	103	Philip	Cumie	Palo Alto

SQL – Operador SELECT

SELECT customer_num, fname, lname, city

FROM customer

ORDER BY city, customer_num **DESC**



The screenshot shows a SQL query editor with the following query:

```
SELECT customer_num, fname, lname, city
FROM customer
ORDER BY 4, 1 DESC
```

Below the query editor, there is a 'Results' tab showing the output of the query. The results are ordered by city (column 4) in descending order, and then by customer_num (column 1) in descending order. The first row is highlighted.

	customer_num	fname	lname	city
1	124	Chris	Putnum	Bartlesville
2	127	Kim	Satfer	Blue Island
3	125	James	Henry	Brighton
4	119	Bob	Shorter	Cherry Hill
5	126	Eileen	Neelie	Denver
6	123	Marvin	Hanlon	Jacksonville
7	112	Margaret	Lawson	Los Altos
8	105	Raymond	Vector	Los Altos
9	115	Alfred	Grant	Menlo Park
10	113	Lana	Beatty	Menlo Park
11	116	Jean	Pamelee	Mountain View
12	106	George	Watson	Mountain View
13	118	Dick	Baxter	Oakland
14	107	Charles	Ream	Palo Alto
15	103	Philip	Cumie	Palo Alto
16	128	Frank	Lessor	Phoenix
17	120	Fred	Jewell	Phoenix

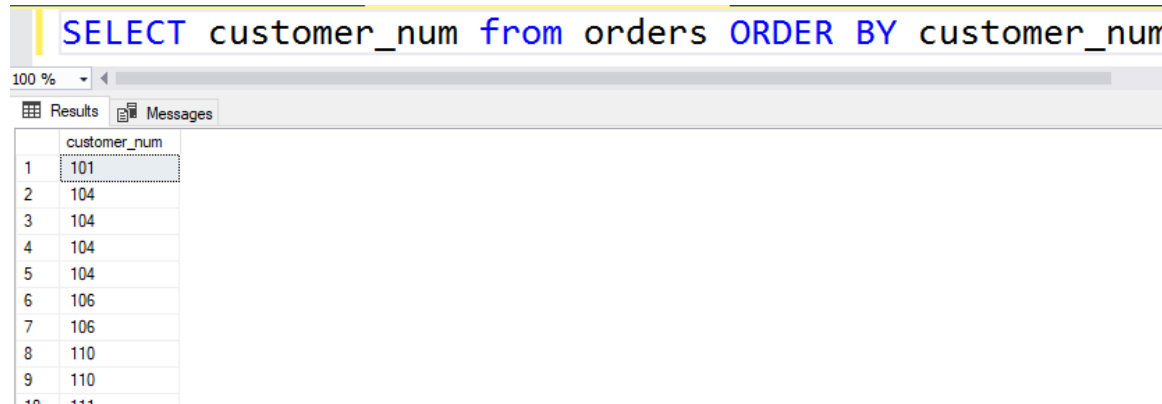
Ordenamiento del resultado de la consulta por una clave o múltiples claves.

Se puede observar en este ejemplo que en el ORDER BY se pueden poner números que indican la posición de la columna en la consulta, en lugar del nombre.

SQL – Operador SELECT

Listar valores únicos para una columna, ante una repetición de valores de esa columna.

SELECT customer_num
FROM orders
ORDER BY 1

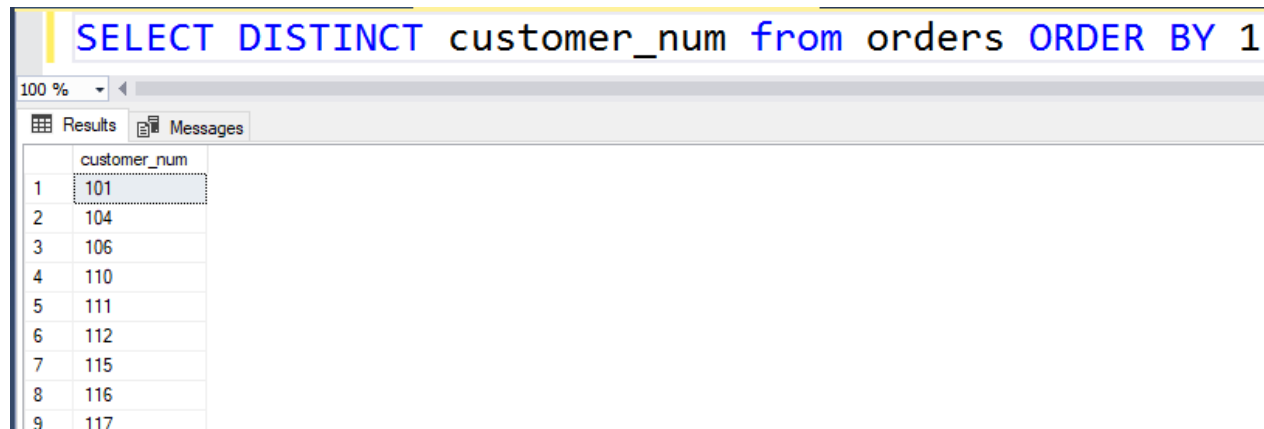


```
SELECT customer_num from orders ORDER BY customer_num
```

	customer_num
1	101
2	104
3	104
4	104
5	104
6	106
7	106
8	110
9	110

vs.
SELECT DISTINCT customer_num

FROM orders
ORDER BY 1



```
SELECT DISTINCT customer_num from orders ORDER BY 1
```

	customer_num
1	101
2	104
3	106
4	110
5	111
6	112
7	115
8	116
9	117

SQL – Operador SELECT

Ejercicio en clase

Listar fname, lname, customer_num, Address1, city y zipcode de los clientes que vivan en el state 'CA', que su zipcode finalice con 025. El apellido deberá estar en mayúsculas (UPPER) y la columna deberá tener el label apellido.

customer		
PK	customer_num	smallint
	fname	varchar(15)
	lname	varchar(15)
	company	varchar(20)
	address1	varchar(20)
	address2	varchar(20)
FK	city	varchar(15)
	state	char(2)
	zipcode	char(5)
	phone	varchar(18)
FK	customer_num_referredBy	smallint
	status	char(1)

SQL – Operador SELECT

Ejercicio en clase

Listar fname, lname, customer_num, Address1, city y zipcode de los clientes que vivan en el state 'CA', que su zipcode finalice con 025. El apellido deberá estar en mayúsculas (UPPER) y la columna deberá tener el label apellido.

customer		
PK	customer_num	smallint
	fname	varchar(15)
	lname	varchar(15)
	company	varchar(20)
	address1	varchar(20)
	address2	varchar(20)
FK	city	varchar(15)
	state	char(2)
	zipcode	char(5)
	phone	varchar(18)
FK	customer_num_referredBy	smallint
	status	char(1)

SQLQuery3.sql - D...HQEMP6\zaffa (51))* SQLQuery2.sql - D...HQEMP6\zaffa (55))*

```
SELECT  fname, UPPER(lname) apellido, customer_num,
        city, zipcode
FROM    customer
where state='CA' and zipcode LIKE '%025'
```

100 %

ResultsMessages

	fname	apellido	customer_num	city	zipcode
1	Lana	BEATTY	113	Menlo Park	94025
2	Alfred	GRANT	115	Menlo Park	94025

SQL – Operador SELECT

Funciones Agregadas

SUM(columna)

COUNT(*) cuenta todas las filas de la tabla

COUNT(columna) cuenta filas con dicha columna No Nula

COUNT(DISTINCT columna) Cuenta solo una vez cada valor.

MIN(columna)

MAX(columna)

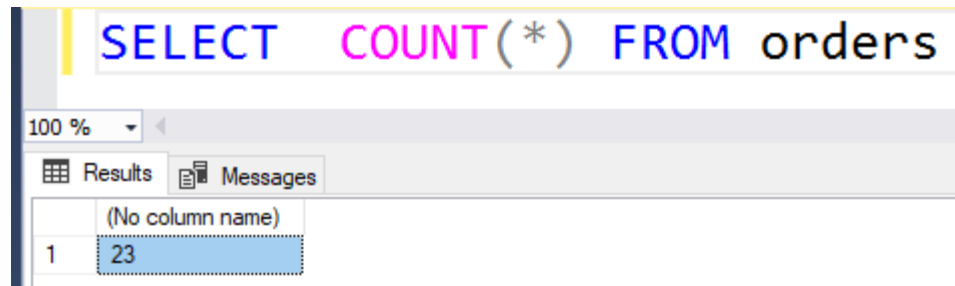
SUM(columna)

AVG(columna)

Son funciones que dado un conjunto de datos realizan operaciones agregadas devolviendo un único valor como resultado.

SQL – Operador SELECT

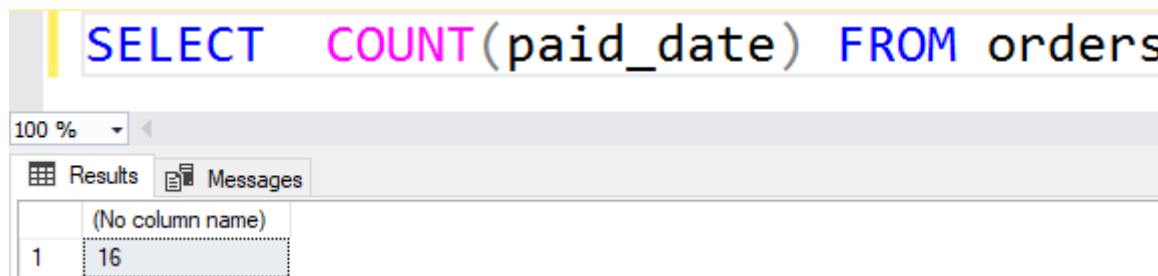
Función COUNT (*) – Mostrar cantidad de Ordenes de Compra.



The screenshot shows a SQL query editor with the text `SELECT COUNT(*) FROM orders`. Below the editor, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one row and one column. The column header is '(No column name)' and the value is 23.

	(No column name)
1	23

Función COUNT (paid_date) – Mostrar cantidad de Ordenes de Compra con fecha de pago No Nula.

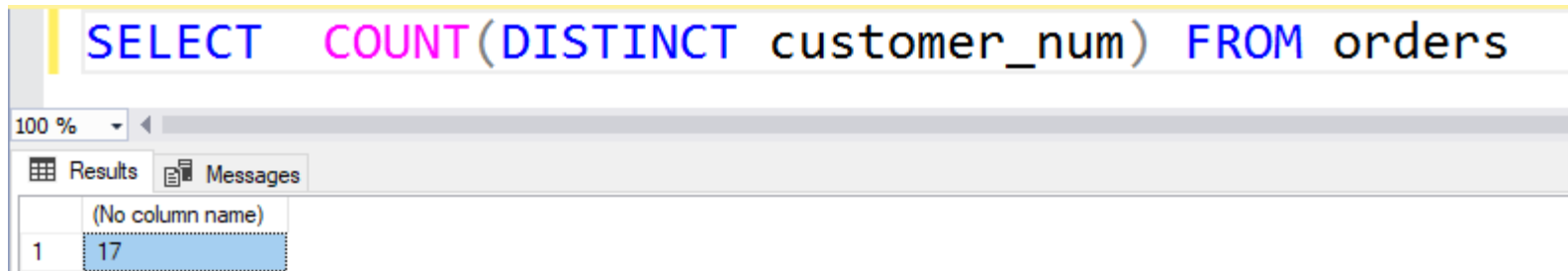


The screenshot shows a SQL query editor with the text `SELECT COUNT(paid_date) FROM orders`. Below the editor, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with one row and one column. The column header is '(No column name)' and the value is 16.

	(No column name)
1	16

SQL – Operador SELECT

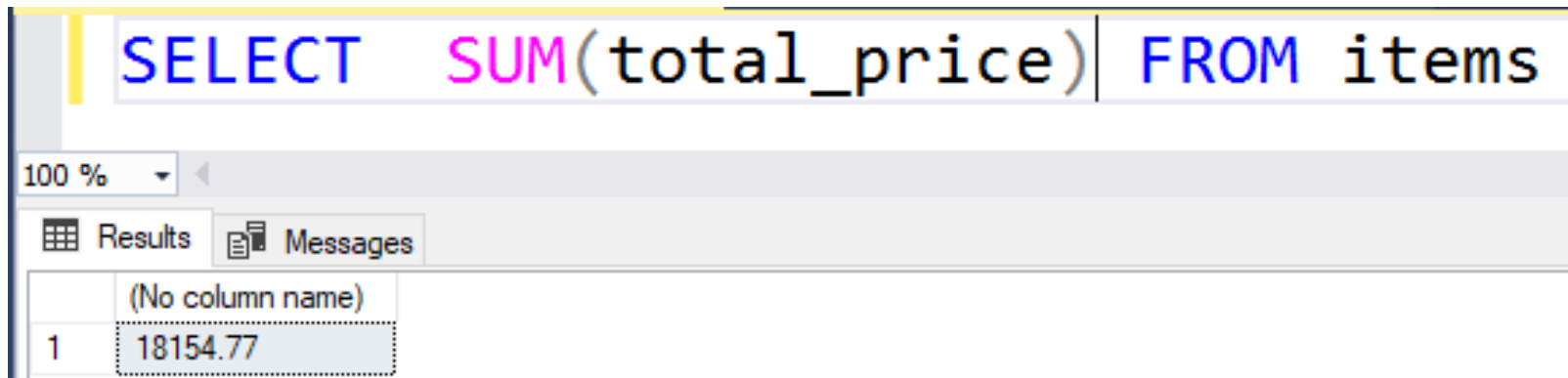
Función COUNT (DISTINCT customer_num) – Mostrar cuantos clientes nos pusieron Ordenes de Compra.



The screenshot shows a SQL query editor with the following query: `SELECT COUNT(DISTINCT customer_num) FROM orders`. Below the query, there is a results pane with a tab labeled 'Results'. The results pane shows a single row with the value 17.

	(No column name)
1	17

Función SUM (total_price) – Mostrar cuanto dinero nos ingreso por los ítems de todas las Ordenes de compra.

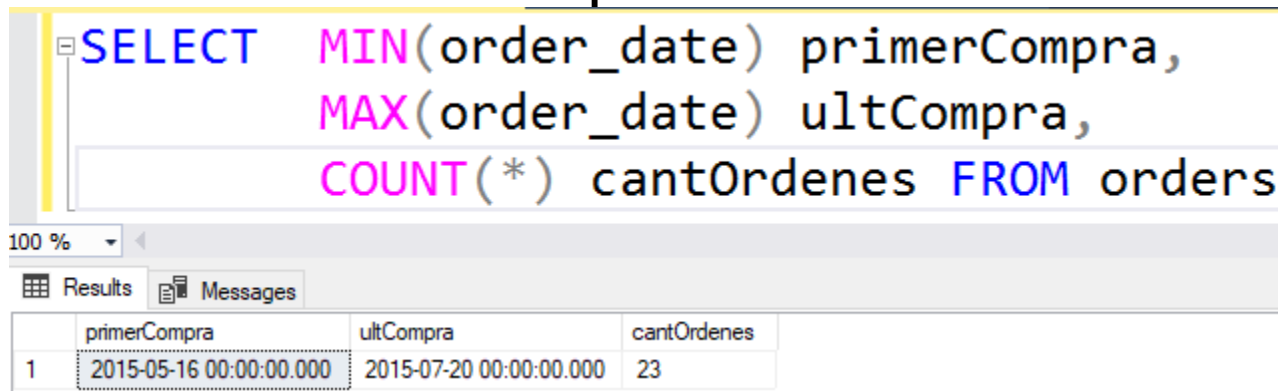


The screenshot shows a SQL query editor with the following query: `SELECT SUM(total_price) FROM items`. Below the query, there is a results pane with a tab labeled 'Results'. The results pane shows a single row with the value 18154.77.

	(No column name)
1	18154.77

SQL – Operador SELECT

Varias funciones combinadas. Informar primer fecha de Orden de Compra, ultima fecha de orden de compra y cantidad de ordenes de compra



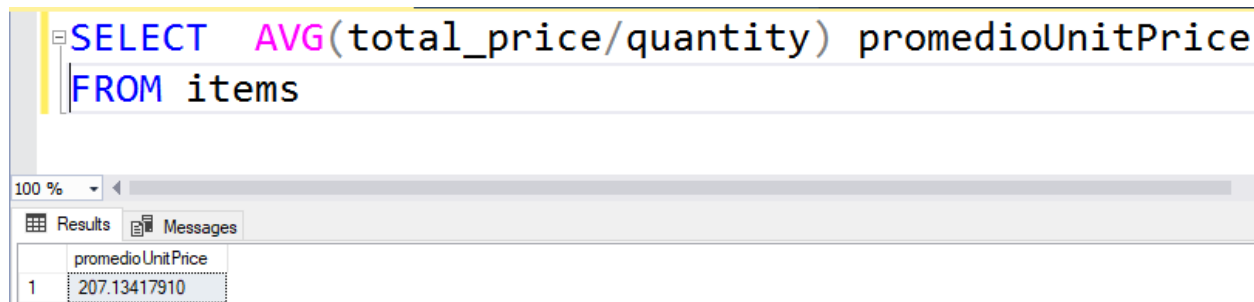
The screenshot shows a SQL query window with the following text:

```
SELECT MIN(order_date) primerCompra,  
       MAX(order_date) ultCompra,  
       COUNT(*) cantOrdenes FROM orders
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	primerCompra	ultCompra	cantOrdenes
1	2015-05-16 00:00:00.000	2015-07-20 00:00:00.000	23

Función agregada de formula matemática. Informar promedio de precios unitarios de los ítems vendidos



The screenshot shows a SQL query window with the following text:

```
SELECT AVG(total_price/quantity) promedioUnitPrice  
FROM items
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	promedioUnitPrice
1	207.13417910

SQL – Operador SELECT

Cláusulas GROUP BY y HAVING

SELECT * | lista de columnas

FROM nom_tabla | lista de tablas

WHERE condiciones ó filtros

Esta clausula de agrupamiento es muy ponderosa en conjunto con la utilización de funciones Agregadas

GROUP BY columnas clave de agrupamiento

HAVING condiciones sobre lo agrupado

ORDER BY columnas clave de ordenamiento

SQL – Operador SELECT

```
SELECT customer_num, COUNT(order_num) cantOrdenes,  
       MIN(order_date) primerCompra, MAX(order_date) ultCompra  
FROM   orders  
GROUP BY customer_num
```

100 %				
Results Messages				
	customer_num	cantOrdenes	primerCompra	ultCompra
1	101	1	2015-05-17 00:00:00.000	2015-05-17 00:00:00.000
2	104	4	2015-05-16 00:00:00.000	2015-06-18 00:00:00.000
3	106	2	2015-05-18 00:00:00.000	2015-06-21 00:00:00.000
4	110	2	2015-06-03 00:00:00.000	2015-06-23 00:00:00.000
5	111	1	2015-06-10 00:00:00.000	2015-06-10 00:00:00.000
6	112	1	2015-05-26 00:00:00.000	2015-05-26 00:00:00.000
7	115	1	2015-06-13 00:00:00.000	2015-06-13 00:00:00.000
8	116	1	2015-05-20 00:00:00.000	2015-05-20 00:00:00.000
9	117	2	2015-05-27 00:00:00.000	2015-06-14 00:00:00.000
10	119	1	2015-06-25 00:00:00.000	2015-06-25 00:00:00.000
11	120	1	2015-07-05 00:00:00.000	2015-07-05 00:00:00.000
12	121	1	2015-07-06 00:00:00.000	2015-07-06 00:00:00.000
13	122	1	2015-07-07 00:00:00.000	2015-07-07 00:00:00.000
14	123	1	2015-07-07 00:00:00.000	2015-07-07 00:00:00.000
15	124	1	2015-07-19 00:00:00.000	2015-07-19 00:00:00.000
16	126	1	2015-07-20 00:00:00.000	2015-07-20 00:00:00.000
17	127	1	2015-07-20 00:00:00.000	2015-07-20 00:00:00.000

Observamos los 17 clientes que compraron, de cada uno tenemos la cantidad de Ordenes de compra, la fecha de la primer y ultima compra.

SQL – Operador SELECT

```
SELECT YEAR(order_date) anio, MONTH(order_date) mes,  
       COUNT(order_num) cantOrdenes,  
       MIN(order_date) primerCompra, MAX(order_date) ultCom  
FROM   orders  
GROUP BY YEAR(order_date), MONTH(order_date)  
ORDER BY 2
```

100 %

Results Messages

	anio	mes	cantOrdenes	primerCompra	ultCompra
1	2015	5	7	2015-05-16 00:00:00.000	2015-05-27 00:00:00.000
2	2015	6	9	2015-06-03 00:00:00.000	2015-06-25 00:00:00.000
3	2015	7	7	2015-07-05 00:00:00.000	2015-07-20 00:00:00.000

Observamos que rápidamente obtenemos la cant. De ordenes, la primer y ultima compra pero ahora agrupado por año y mes.

SQL – Operador SELECT

```
SELECT YEAR(order_date) anio, MONTH(order_date) mes,  
       COUNT(order_num) cantOrdenes,  
       MIN(order_date) primerCompra, MAX(order_date) ultCom  
FROM   orders  
GROUP BY YEAR(order_date), MONTH(order_date)  
ORDER BY 2
```

100 %

Results Messages

	anio	mes	cantOrdenes	primerCompra	ultCompra
1	2015	5	7	2015-05-16 00:00:00.000	2015-05-27 00:00:00.000
2	2015	6	9	2015-06-03 00:00:00.000	2015-06-25 00:00:00.000
3	2015	7	7	2015-07-05 00:00:00.000	2015-07-20 00:00:00.000

Observamos que rápidamente obtenemos la cant. De ordenes, la primer y ultima compra pero ahora agrupado por año y mes.

SQL – Operador SELECT

GROUP BY y ORDER BY

```
SELECT YEAR(order_date) anio, MONTH(order_date) mes,  
       COUNT(order_num) cantOrdenes,  
       MIN(order_date) primerCompra, MAX(order_date) ultCompra  
FROM   orders  
GROUP BY YEAR(order_date), MONTH(order_date)  
ORDER BY cantOrdenes DESC
```

	anio	mes	cantOrdenes	primerCompra	ultCompra
1	2015	6	9	2015-06-03 00:00:00.000	2015-06-25 00:00:00.000
2	2015	7	7	2015-07-05 00:00:00.000	2015-07-20 00:00:00.000
3	2015	5	7	2015-05-16 00:00:00.000	2015-05-27 00:00:00.000

Las filas están ordenadas por cantidad de ordenes en orden Descendente.

Observamos que rápidamente obtenemos la cant. De ordenes, la primer y ultima compra pero ahora agrupado por año y mes.

SQL – Operador SELECT

GROUP BY y HAVING

```
SELECT YEAR(order_date) anio, MONTH(order_date) mes,
       COUNT(order_num) cantOrdenes,
       MIN(order_date) primerCompra,
       MAX(order_date) ultCompra
FROM   orders
GROUP BY YEAR(order_date), MONTH(order_date)
HAVING count(*) >=8
```

100 %

Results Messages

	anio	mes	cantOrdenes	primerCompra	ultCompra
1	2015	6	9	2015-06-03 00:00:00.000	2015-06-25 00:00:00.000

La Clausula HAVING actúa sobre los datos de las filas ya agrupadas y en general se le ponen condiciones con funciones agregadas.

Observamos que rápidamente obtenemos la cant. De ordenes, la primer y ultima compra pero ahora agrupado por año y mes, pero solo las que la cantidad sea \geq a 8.

SQL – Operador SELECT

Ejercicio en clase

Listar por cada código de fabricante, cantidad de ordenes de compra (ante repetidos contar solo una), Suma de quantity, suma de unit_price, para los fabricantes que tengan mas de 5 items comprados (cantidad de filas en table items > 5).
Ordenado por la suma de unit_price*quantity.

items		
PK,FK	order_num	smallint
PK	item_num	smallint
FK	stock_num	smallint
FK	manu_code	char(3)
	quantity	smallint
	unit_price	decimal

SQL – Operador SELECT

Ejercicio en clase

Listar por cada código de fabricante, cantidad de ordenes de compra (ante repetidos contar solo una), Suma de quantity, suma de unit_price, para los fabricantes que tengan mas de 5 items comprados (cantidad de filas en table items > 5).
Ordenado por la suma de unit_price*quantity.

items		
PK,FK	order_num	smallint
PK	item_num	smallint
FK	stock_num	smallint
FK	manu_code	char(3)
	quantity	smallint
	unit_price	decimal

```
SELECT manu_code,
       COUNT(DISTINCT order_num) cantOrdenes,
       SUM(quantity) CantidadComprada,
       SUM(unit_price*quantity) TotalComprado
FROM   items
GROUP BY manu_code
HAVING count(*) >=5
ORDER BY 4 DESC
```

	manu_code	cantOrdenes	CantidadComprada	TotalComprado
1	ANZ	11	59	4701.80
2	HSK	5	7	3748.00
3	HRO	6	10	2882.00
4	SHM	4	10	2337.97
5	KAR	4	13	1373.00
6	SMT	7	15	1269.00
7	PRC	4	13	1198.00

VAMOS A LA PRACTICA