

NIVELL 1

Ex1

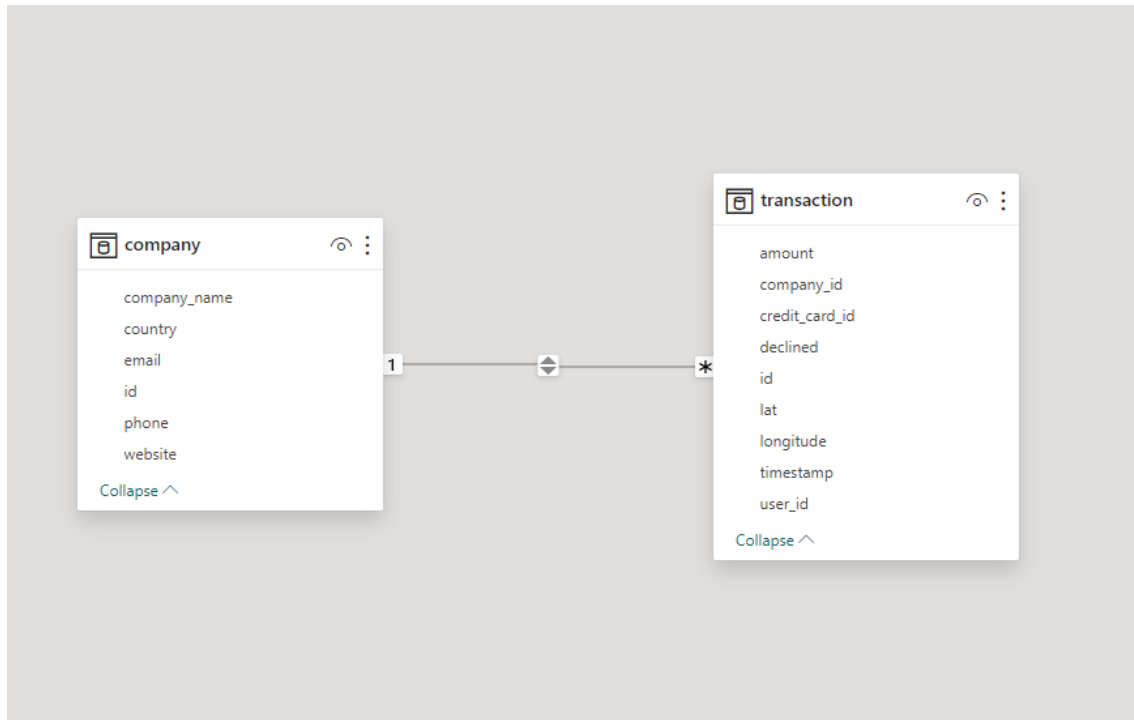


Table Transaction = **FACT TABLE**

Table Company = **DIMENSION TABLE**

Relation = N to 1 (transaction.company_id = company.id)

La mayoría de los elementos son **VARCHAR** (string de letras y/o números) pero también hay Integer (**INT**), Floats (**FLOAT**), una variable de tipo **BOOLEAN** (True/False), una **DECIMAL** y una de tipo fecha (**TIMESTAMP**).

Se establece la relación entre tablas mediante la **FOREIGN KEY** *transaction.company_id*, a través de la instrucción **REFERENCE**.

Se observa que aparecen otras **REFERENCE**, una con *user_id* y otra con *credit_card_id*. Seguramente existan dos tablas más de **DIMENSIONES**, una llamada Users, con información de los usuarios, y otra llamada Credit Cards, con información de las tarjetas de crédito. Así se montaría un esquema en estrella con la tabla Transaction como **FACT TABLE** en el medio.

Ejecutamos los dos archivos proporcionados, uno crea las tablas y las relaciones, el otro las llena de contenido.

Ex2

Se hace un **SELECT** con los datos que se piden, pertenecen todos a la misma tabla, no se requieren JOINS, se ordena con un **ORDER BY** *company_name*.

The screenshot shows a SQL Developer window with a query editor and a results grid. The query is as follows:

```
1 #NIVELL 1
2
3 #Ex2
4 SELECT company_name, email, country FROM transactions.company
5 ORDER BY company_name;
```

The results grid displays a list of companies with their names, emails, and countries. The countries are sorted alphabetically. The list includes companies like A Institute, Ac Fermentum Incorporated, Ac Industries, Ac Libero Inc., Alquam Brat Volutpat LLP, Alquam Incule Lascus Corp., Alquam PC, Alquet Diam Limited, Alquet Sem Limited, Alquet Vel Vulputate Incorporated, Amet Faudibus Ut Foundation, Amet Institute, Amet Lorem LLP, Amet Luctus Vulputate Foundation, Amet Nulla Donec Corporation, Ante Taculis Nec Foundation, Arcu LLP, At Associates, At Pedes Corp., Auctor Mauris Corp., Auctor Mauris Vel LLP, Augue Foundation, Cornallis In Incorporated, and Cras Praesent.

The output pane shows the execution of the query, with a message indicating that 100 rows were returned.

Ex3

En esta ocasión se hace un **SELECT DISTINCT** para conseguir un listado de los países sin repeticiones. Es necesario hacer un **INNER JOIN** para conseguir datos que se encuentran en las dos tablas. Se agrega una cláusula **WHERE** donde *declined* = 0, indicando que la transacción se ha realizado.

The screenshot shows a SQL Developer window with a query editor and a results grid. The query is as follows:

```
7 #Ex3
8 SELECT DISTINCT country FROM company
9 INNER JOIN transaction
10 ON company.id = transaction.company_id
11 WHERE declined = 0;
```

The results grid displays a list of countries: Canada, Germany, Italy, United Kingdom, Sweden, Ireland, United States, Norway, France, New Zealand, Netherlands, Belgium, Australia, China, and Spain.

The output pane shows the execution of the query, with a message indicating that 15 rows were returned.

Ex4

Se puede realizar la misma consulta, pero con un **COUNT** para que nos devuelva el nº de países que realizan compras.

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
13 #Ex4
14 • SELECT COUNT(DISTINCT country) AS 'nº de países realizando compras' FROM company
15 INNER JOIN transaction
16 ON company_id = transaction.company_id
17 WHERE declined = 0;
```

The results pane shows a single row with the value 15, representing the number of countries that have made purchases.

Result Grid
nº de países realizando compras
15

The output pane shows the execution log with the following messages:

#	Time	Action	Message
1	15:27:01	SELECT DISTINCT country FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	15 row(s) returned
2	15:28:54	SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	1 row(s) returned
3	15:29:18	SELECT COUNT(DISTINCT country) AS 'nº de países realizando compras' FROM company INNER JOIN transaction ON company_id = transaction.com...	1 row(s) returned

Ex5

Se realiza un **SELECT** con una clausula **WHERE** donde *id = la id solicitada*.

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```
19 #Ex5
20 • SELECT company_name, country FROM company
21 WHERE id = 'b-2354';
22
23 #Ex6
```

The results pane shows a single row with the company name 'Ac Libero Inc.' and the country 'United Kingdom'.

Result Grid	
company_name	country
Ac Libero Inc.	United Kingdom

The output pane shows the execution log with the following messages:

#	Time	Action	Message
1	15:27:01	SELECT DISTINCT country FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	15 row(s) returned
2	15:28:54	SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	1 row(s) returned
3	15:29:18	SELECT COUNT(DISTINCT country) AS 'nº de países realizando compras' FROM company INNER JOIN transaction ON company_id = transaction.com...	1 row(s) returned
4	15:32:00	SELECT company_name, country FROM company WHERE id = 'b-2354'	1 row(s) returned

Ex6

Se utiliza la función **AVG()** para determinar el valor medio, y se agrupa mediante **GROUP BY** *company_name*. En la cláusula **WHERE** se especifica la relación entre las dos tablas *company.id* = *transaction.company_id*, además de que la transacción se haya realizado, *declined* = 0.

dades_introduirestructura_datossprint1sprint2estructura_bd_transactionssprint3

Don't Limit

22

#Ex6

24 • `SELECT AVG(amount), company_name FROM transaction, company`

25 `WHERE company.id = transaction.company_id AND declined = 0`

26 `GROUP BY company_name`

27 `ORDER BY AVG(amount) DESC;`

Result Grid

AVG(amount)

company_name

481.860000

Eget Ipsum Ltd

477.510000

Sed Id Limited

477.100000

Naeque Telus Incorporated

461.830000

Nunc Sit Incorporated

458.740000

Non Magna LLC

451.290000

Maecenas Malesuada Fringilla Inc.

448.440000

Erat LLP

447.110000

Tortor Nunc Commoda Company

444.160000

Justo Eu Arcu Ltd

442.220000

Pede Cum Ltd

428.400000

Vestibulum Lorem PC

427.710000

Mauris Institute

425.640000

Aliquet Diam Limited

419.970000

Mus Aenean Eget Foundation

415.660000

Sed LLC

414.530000

Viverra Donec Foundation

413.500000

Eget Tincidunt Dui Institute

412.480000

Amet Institute

406.110000

Egestas Nunc Sed Limited

400.630000

Elit Etiam Laoreet Associates

396.150000

Ac Industries

389.620000

Nietus Et Malesuada Ltd

383.730000

Sed Nunc Ltd

379.140000

Gravida Sagittis LLP

376.410000

Quam A Fels Industries

Result 5

×

Output

Action Output

#

Time

Action

Message

1

15:27:01

SELECT DISTINCT country FROM company INNER JOIN transaction ON company.id = transaction.company_id WHERE declined = 0

15 row(s) returned

2

15:28:54

SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company.id = transaction.company_id WHERE declined = 0

1 row(s) returned

3

15:29:18

SELECT COUNT(DISTINCT country) AS "nº de países realizando compras" FROM company INNER JOIN transaction ON company.id = transaction.com...

1 row(s) returned

4

15:32:00

SELECT AVG(amount), country FROM company WHERE id = 'b-2354'

1 row(s) returned

5

15:35:04

SELECT AVG(amount), company_name FROM transaction, company WHERE company.id = transaction.company_id AND declined = 0 GROUP BY co...

100 row(s) returned

NIVELL 2

Ex1

Se comprueba mediante la realización de un **COUNT**, especificando que el valor de dicho **COUNT** sea mayor de 1 a través de la cláusula **HAVING**.

```
27 #NIVELL 2
28
29 #Ex1
30 • SELECT COUNT(id) FROM transactions.company
31   GROUP BY id
32   HAVING COUNT(id) > 1;
```

Result Grid
COUNT(id)

Result 15 x

Output

#	Time	Action	Message
701	16:46:55	SELECT DISTINCT(company_name) FROM company, transaction WHERE company_id = transaction.company_id AND timestamp LIKE (2022-03-16...	6 row(s) returned
702	16:47:08	SELECT DISTINCT(company_name) FROM company, transaction WHERE company_id = transaction.company_id AND timestamp LIKE (2022-03-16...	6 row(s) returned
703	16:59:17	SELECT company_name, email, country FROM transactions.company ORDER BY company_name	100 row(s) returned
704	17:00:25	SELECT company_name, email, country FROM transactions.company ORDER BY company_name	100 row(s) returned
705	17:01:13	SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company_id = transaction.company_id	1 row(s) returned
706	17:02:12	SELECT company_name, country FROM company WHERE id = b-2354	1 row(s) returned
707	17:02:55	SELECT AVG(amount), company_name FROM transaction, company WHERE company_id = transaction.company_id GROUP BY company_name OR...	100 row(s) returned
708	17:03:47	SELECT COUNT(id) FROM transactions.company GROUP BY id HAVING COUNT(id) > 1	0 row(s) returned

Ex2

Aquí se utilizara la fórmula **SUM()** y se agrupará por fechas. Se agrega una cláusula **WHERE** donde *declined* = 0, indicando que la transacción se ha realizado. Para que se muestren las 5 ventas mas grandes se ordena de forma descendiente con un **ORDER BY** y se limitan los resultados a las 5 primeras *rows*.

```
36 #Ex2
37 • SELECT DATE(timestamp), SUM(amount) FROM transaction
38   WHERE declined = 0
39   GROUP BY DATE(timestamp)
40   ORDER BY SUM(amount) DESC
41   LIMIT 5;
```

Result Grid	
DATE(timestamp)	SUM(amount)
2021-12-20	1532.36
2021-04-22	1397.96
2021-05-09	1344.37
2022-02-26	1337.62
2021-03-29	1325.12

Result 7 x

Output

#	Time	Action	Message
1	15:27:01	SELECT DISTINCT country FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	15 row(s) returned
2	15:28:54	SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company_id = transaction.company_id WHERE declined = 0	1 row(s) returned
3	15:29:18	SELECT COUNT(DISTINCT country) AS n° de países realizando compras FROM company INNER JOIN transaction ON company_id = transaction.com...	1 row(s) returned
4	15:32:00	SELECT company_name, country FROM company WHERE id = b-2354	1 row(s) returned
5	15:35:04	SELECT AVG(amount), company_name FROM transaction, company WHERE company_id = transaction.company_id AND declined = 0 GROUP BY co...	100 row(s) returned
6	15:40:38	SELECT COUNT(id) FROM transactions.company GROUP BY id HAVING COUNT(id) > 1	0 row(s) returned
7	15:43:03	SELECT DATE(timestamp), SUM(amount) FROM transaction WHERE declined = 0 GROUP BY DATE(timestamp) ORDER BY SUM(amount) DESC LIM...	5 row(s) returned

Ex3

Se realizara la misma **QUERY** que en el ejercicio anterior pero se ordenara de forma ascendiente para que ahora se muestren los 5 resultado mas pequeños.

The screenshot shows a database IDE with a SQL query editor and a results pane. The query is as follows:

```
40 #Ex3
41
42 • SELECT DATE(timestamp), SUM(amount) FROM transaction
43 GROUP BY DATE(timestamp)
44 ORDER BY SUM(amount) ASC
45 LIMIT 5;
```

The results pane displays a table with two columns: DATE(timestamp) and SUM(amount). The data is as follows:

DATE(timestamp)	SUM(amount)
2022-01-04	15.05
2021-04-27	18.08
2022-01-24	23.86
2022-02-27	30.76
2022-01-14	37.55

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message
703	16:59:17	SELECT company_name, email, country FROM transactions company ORDER BY company_name	100 row(s) returned
704	17:00:25	SELECT company_name, email, country FROM transactions company ORDER BY company_name	100 row(s) returned
705	17:01:13	SELECT COUNT(DISTINCT country) FROM company INNER JOIN transaction ON company_id = transaction.company_id	1 row(s) returned
706	17:02:12	SELECT company_name, country FROM company WHERE id = b-2354	1 row(s) returned
707	17:02:55	SELECT AVG(amount), company_name FROM transaction, company WHERE company_id = transaction.company_id GROUP BY company_name OR...	100 row(s) returned
708	17:03:47	SELECT COUNT(id) FROM transactions company GROUP BY id HAVING COUNT(id) > 1	0 row(s) returned
709	17:04:26	SELECT DATE(timestamp), SUM(amount) FROM transaction GROUP BY DATE(timestamp) ORDER BY SUM(amount) DESC LIMIT 5	5 row(s) returned
710	17:05:11	SELECT DATE(timestamp), SUM(amount) FROM transaction GROUP BY DATE(timestamp) ORDER BY SUM(amount) ASC LIMIT 5	5 row(s) returned

Ex4

Se utilizará la función **AVG** agrupada mediante un **GROUP BY** por países. En la cláusula **WHERE** se especifica la relación entre las dos tablas *company.id = transaction.company_id*, además de que la transacción se haya realizado, *declined = 0*. Finalmente se ordena con un **ORDER BY AVG()**.

The screenshot shows a database IDE with a SQL query editor and a results pane. The query is as follows:

```
49 #Ex4
50
51 • SELECT AVG(amount), country FROM transaction, company
52 WHERE company_id = transaction.company_id AND declined = 0
53 GROUP BY company.country
54 ORDER BY AVG(amount) DESC;
```

The results pane displays a table with two columns: AVG(amount) and country. The data is as follows:

AVG(amount)	country
287.531111	United States
285.825357	Ireland
276.668382	Sweden
271.767527	United Kingdom
261.941930	Canada
255.217500	Belgium
251.114918	Norway
243.342222	Italy
242.239189	Germany
240.940000	Netherlands
222.240000	China
177.331667	Australia
169.410000	France
167.061667	New Zealand
26.220000	Spain

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message
4	15:32:00	SELECT company_name, country FROM company WHERE id = b-2354	1 row(s) returned
5	15:35:04	SELECT AVG(amount), company_name FROM transaction, company WHERE company_id = transaction.company_id AND declined = 0 GROUP BY c...	100 row(s) returned
6	15:40:38	SELECT COUNT(id) FROM transactions company GROUP BY id HAVING COUNT(id) > 1	0 row(s) returned
7	15:43:03	SELECT DATE(timestamp), SUM(amount) FROM transaction WHERE declined = 0 GROUP BY DATE(timestamp) ORDER BY SUM(amount) DESC LI...	5 row(s) returned
8	15:49:06	SELECT AVG(amount), country FROM transaction, company WHERE company_id = transaction.company_id GROUP BY company.country ORDER B...	15 row(s) returned
9	15:49:20	SELECT AVG(amount), country FROM transaction, company WHERE company_id = transaction.company_id GROUP BY company.country ORDER B...	15 row(s) returned
10	15:49:34	SELECT AVG(amount), country FROM transaction, company WHERE company_id = transaction.company_id AND declined = 0 GROUP BY company...	15 row(s) returned

NIVELL 3

Ex1

Este ejercicio requerirá una **SubQUERY**. Queremos calcular la **SUM(amount)**, pero no de todas las compañías, solo de las que han realizado transacciones de entre 100 y 200. Esta condición es la que nos marca la **SubQUERY**. En este caso haremos un **SELECT *** de transaction con la condición de que las transacciones se hayan realizado y de que sean de entre 100 y 200. Este **QUERY** lo llamaremos *transaction100200* y es del cual realizaremos la **SUM(amount)**.

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```
56 #NIVELL 3
57
58 #Ex1
59 SELECT company_name, SUM(amount)
60 FROM company, (SELECT * FROM transaction
61 WHERE declined = 0 AND amount BETWEEN 100 AND 200) AS transaction100200
62 WHERE transaction100200.company_id = company.id
63 GROUP BY company_name
64 ORDER BY SUM(amount) DESC;
65
```

The results grid shows the following data:

company_name	SUM(amount)
Enim Conditimentum Ltd	2747.41
Nunc Interdum Incorporated	2412.55
Ut Semper Foundation	2285.80
Arcu LLP	1589.00
Lorem Eu Incorporated	1544.61
Malesuada PC	1127.09
Non Institute	1078.31
Tristique Nique Venenatis Institute	192.86
Nunc In Foundation	183.84
Cras Vehicula Aliquet Industries	181.87
Auctor Mauris Vel LLP	179.40
Tincidunt Associates	173.34
At Associates	169.10
Dolor Vitae Limited	164.88
Interdum Feugiat Sed Associates	164.86
Mauris Incorporated	158.50
Mattis Foundation	155.57
Quisque Libero LLC	155.44
Non Justo Corp.	151.32
Nec Luctus LLC	124.06
Dinner Privacy Inc	119.68

The output pane shows the execution of the query and other SQL statements, with the final result being 23 rows returned.

Ex2

En este caso haremos un **SELECT DISTINCT** para que nos de los nombres de las compañías. En la cláusula **WHERE** especificaremos las fechas que nos interesan, además de condicionar que la compra se realizó y especificar la relación entre las tablas.

dades_introduir estructura_dades sprint1

Don't Limit

62

#Ex2

64 • `SELECT DISTINCT(company_name) FROM company, transaction`

65 `WHERE company.id = transaction.company_id`

66 `AND (timestamp LIKE ('2022-03-16%') OR timestamp LIKE ('2022-02-26%') OR timestamp LIKE ('2022-02-13%'));`

Result Grid

Filter Rows:

Export:

Wrap Cell Contents

company_name
Sed LLC
Arzu LLP
Nunc Interdum Incorporated
Ut Semper Foundation
Lorem Eu Incorporated
Malesuada PC

Result 20 x

Read Only

Output

Action Output

#	Time	Action	Message
706	17:02:12	SELECT company_name, country FROM company WHERE id = b-2354	1 row(s) returned
707	17:02:55	SELECT AVG(amount), company_name FROM transaction, company WHERE company.id = transaction.company_id GROUP BY company_name OR...	100 row(s) returned
708	17:03:47	SELECT COUNT(id) FROM transactions, company GROUP BY id HAVING COUNT(id) > 1	0 row(s) returned
709	17:04:26	SELECT DATE(timestamp), SUM(amount) FROM transaction GROUP BY DATE(timestamp) ORDER BY SUM(amount) DESC LIMIT 5	5 row(s) returned
710	17:05:11	SELECT DATE(timestamp), SUM(amount) FROM transaction GROUP BY DATE(timestamp) ORDER BY SUM(amount) ASC LIMIT 5	5 row(s) returned
711	17:05:48	SELECT AVG(amount), country FROM transaction, company WHERE company.id = transaction.company_id GROUP BY company, country ORDER B...	15 row(s) returned
712	17:06:58	SELECT company_name, SUM(amount) FROM company, (SELECT * FROM transaction WHERE amount BETWEEN 100 AND 200) AS transaction1...	37 row(s) returned
713	17:07:56	SELECT DISTINCT(company_name) FROM company, transaction WHERE company.id = transaction.company_id AND timestamp LIKE (2022-03-16...	6 row(s) returned