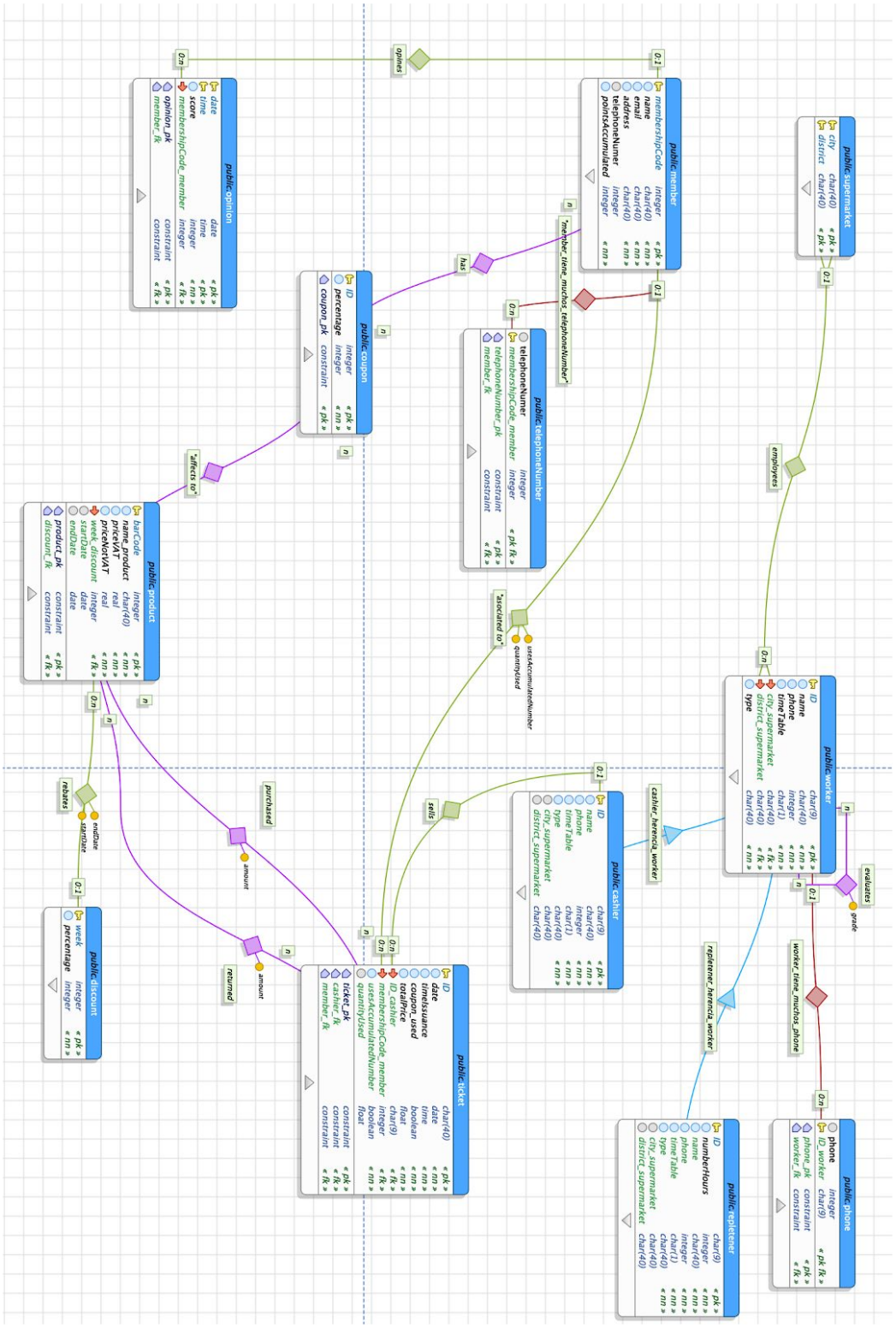
DATA BASES

CAL2

LAURA MAMBRILLA MORENO
ISABEL BLANCO MARTÍNEZ
JESÚS GONZÁLEZ SÁNCHEZ

Universidad de Alcalá

PGMODELER SCHEMA



IMPROVEMENTS AND UPDATES

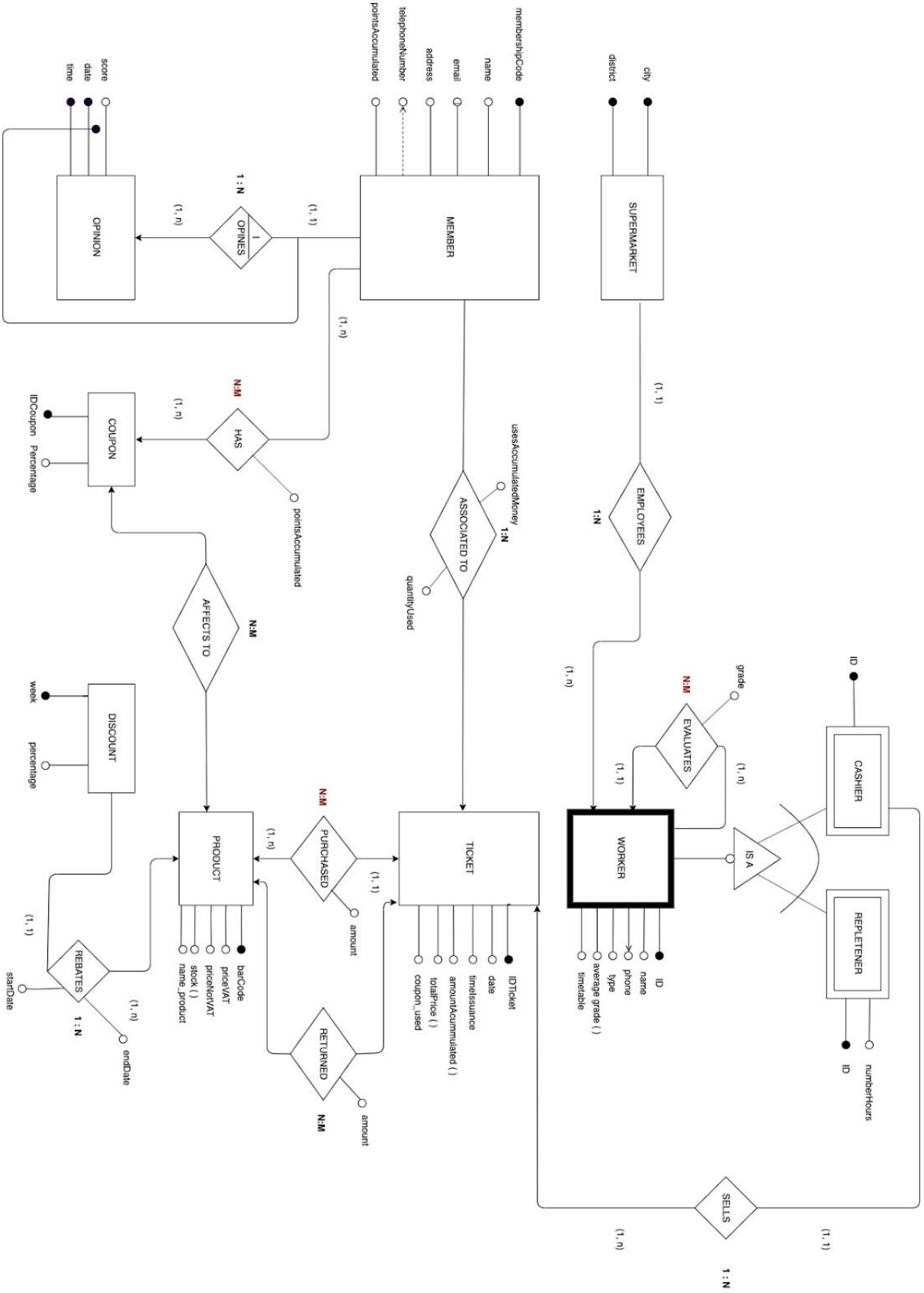
We have done some changes in the entity model since the last assignment, the changes are:

- **Entity worker:**
  - Attribute *mobile* removed
  - Attribute *Phone* is now multivalued
  - Attribute *average grade ( )* is now calculated.
  - Attribute *type* added to make Query 2 much easier. It can be C (cashier) or R (repletener).
- **Entity cashier:**
  - Attribute *numberHours* removed
  - It inherits pk *ID* from worker
- **Entity repletener:**
  - It inherits pk *ID* from worker
- **Entity opinion:**
  - *numberOpinion* removed
- **Entity ticket:**
  - Attribute *name* removed
  - Attribute *amountProduct* removed
  - Attribute *listProduct* removed
  - Attributes *amountAccumulated( )* and *totalPrice( )* now are calculated
  - Attribute *coupon_used* created as a boolean.
  - Attribute *totalPrice* created
- **Entity discount:**
  - Attribute *startDate* removed
  - Attribute *endDate* product removed
  - Attribute *percentage* removed.
- **Entity member:**
  - Attribute *startDate* removed
  - Attribute *endDate* product removed
  - Attribute *telephoneNumber* is now multivalued.
- **Entity product:**
  - Attribute *stock ( )* is now calculated.
  - Attribute name_product is created recognise in an easier way each products.
- **Entity coupon:**
  - Attribute *percentage* added to know the quantity each coupon takes off.

- **Relationship purchased (muchos_ticket_tiene_muchos_product in pgAdmin):**
  - Now is N to M
  - Attribute *amount* that specifies the quantity of products purchased
- **Relationship returned:**
  - Attribute *amount* that specifies the quantity of products returned
- **Relationship has (muchos_member_tiene_muchos_coupon in PgAdmin):**
  - Now is N to M
- **Relationship affects to (muchos_coupon_tiene_muchos_product in PgAdmin):**
  - Now is N to M
- **Relationship evaluates (muchos_worker_tiene_muchos_worker in pgAdmin):**
  - Now is N to M
  - New attribute *grade*

- **Relationship rebates:**
    - New attribute *endDate* (Format: English)
    - New attribute *startDate* (Format: English)
- **Relationships goes to, includes and includes:**
    - Deleted because they’re not necessary
- **New Relationship Associated to:**
    - Attribute *usesAccumulatedMoney* created
    - Attribute *quantityUsed* created

MODIFIED E/R DIAGRAM



ENTITIES

1. Supermarket

	city [PK] character (40)	district [PK] character (40)
1	Madrid	Gran Vía
2	Madrid	Carabanchel
3	Alcalá de Henares	La Garena
4	Alcalá de Henares	Espartales
5	Mallorca	Arenal
6	Málaga	Palma-Palmilla
7	Murcia	La Flota-Vista Alegre
8	Granada	Albaicín

2. Worker

	ID [PK] character (9)	name character (40)	phone integer	timeTable character (1)	city_supermarket character (40)	district_supermarket character (40)	type character (40)
1	11111111A	Aarón	611111111	A	Madrid	Gran Vía	C
2	22222222B	Bernardo	622222222	B	Madrid	Carabanchel	C
3	33333333C	Carmen	633333333	A	Madrid	Carabanchel	C
4	44444444D	Angy	601234567	A	Madrid	Carabanchel	C
5	55555555E	Adrián	601243567	A	Mallorca	Arenal	C
6	66666666F	Aitor	610243567	B	Alcalá de Henares	La Garena	C
7	77777777G	Álvaro	610423567	B	Alcalá de Henares	Espartales	R
8	88888888J	Andrea	610425367	B	Málaga	Palma-Palmilla	R
9	99999999K	Ramón	610535367	A	Murcia	La Flota-Vista Alegre	R
10	45872396L	Ander	610535637	B	Granada	Albaicín	R
11	05794628M	Daniel	613655637	B	Alcalá de Henares	Espartales	R
12	11111111A	Aarón	611111111	A	Madrid	Gran Vía	C
13	22222222B	Bernardo	622222222	B	Madrid	Carabanchel	C
14	33333333C	Carmen	633333333	A	Madrid	Carabanchel	C
15	44444444D	Angy	601234567	A	Madrid	Carabanchel	C
16	55555555E	Adrián	601243567	A	Mallorca	Arenal	C
17	66666666F	Aitor	610243567	B	Alcalá de Henares	La Garena	C
18	77777777G	Álvaro	610423567	B	Alcalá de Henares	Espartales	R
19	88888888J	Andrea	610425367	B	Málaga	Palma-Palmilla	R
20	99999999K	Ramón	610535367	A	Murcia	La Flota-Vista Alegre	R
21	45872396L	Ander	610535637	B	Granada	Albaicín	R
22	05794628M	Daniel	613655637	B	Alcalá de Henares	Espartales	R

3. Cashier

	ID [PK] character (9)	name character (40)	phone integer	timeTable character (1)	city_supermarket character (40)	district_supermarket character (40)	type character (40)
1	11111111A	Aarón	611111111	A	Madrid	Gran Vía	C
2	22222222B	Bernardo	622222222	B	Madrid	Carabanchel	C
3	33333333C	Carmen	633333333	A	Madrid	Carabanchel	C
4	44444444D	Angy	601234567	A	Madrid	Carabanchel	C
5	55555555E	Adrián	601243567	A	Mallorca	Arenal	C
6	66666666F	Aitor	610243567	B	Alcalá de Henares	La Garena	C

4. Repletener

	ID [PK] character (9)	name character (40)	phone integer	timeTable character (1)	city_supermarket character (40)	district_supermarket character (40)	type character (40)	numberHours integer
1	77777777G	Álvaro	610423567	B	Alcalá de Henares	Espartales	R	22
2	88888888J	Andrea	610425367	B	Málaga	Palma-Palmilla	R	16
3	99999999K	Ramón	610535367	A	Murcia	La Flota-Vista Alegre	R	38
4	45872396L	Ander	610535637	B	Granada	Albaicín	R	24
5	05794628M	Daniel	613655637	B	Alcalá de Henares	Espartales	R	18

5. Member

	membershipCode [PK] integer	name character (40)	email character (40)	address character (40)	telephoneNumer integer	pointsAccumulated integer
1	111111111	Laura Garcia	laura.garcia@edu.u...	Calle Mayor	925834875	234
2	222222222	Carlos Bernal	carlos.bernal@edu....	Calle Reliquias	[null]	112
3	333333333	Sergio Muñoz	sergio.muñoz@edu...	Calle del Lago	925676787	433
4	444444444	Almudena Álvarez	almudena.alvarez@...	Calle Lisboa	925633333	333
5	555555555	Israel Plaza	israel.plaza@edu.u...	Calle Burgos	[null]	665
6	666666666	Pablo Gómez	pablo.gomez@edu....	Calle Benidorm	925999999	410
7	777777777	Lucas Martin	lucas.martin@edu....	Calle Severo Ochoa...	925965786	210
8	888888888	Marta Bermejo	marta.bermejo@ed...	Calle Moraleja	925825445	540
9	999999999	Carmen Agüero	carmen.aguero@ed...	Calle Julio Gomez	925825676	200
10	10101010	Clara Manzano	clara.manzano@ed...	Calle Agosto	[null]	231

6. Coupon

	ID [PK] integer	percentage integer
1	123	20
2	132	15
3	231	22
4	213	12
5	312	13

7. Discount

	week [PK] integer	percentage integer
1	40	10
2	4	10
3	23	20
4	12	25
5	20	15
6	22	15
7	18	30

8. Ticket

Data Output Explain Messages

	ID [PK] character (40)	date date	timeIssuance time without time zone	coupon_used boolean	total_price double precision	ID_cashier character (9)	membershipCode_member integer	usesAccumulatedNumber boolean
1	34237	2015-02...	14:15:00	false	21.5	22222222B	[null]	false
2	98695	2019-07...	14:15:00	false	23.4	22222222B	[null]	false
3	56631	2016-11...	10:26:00	true	13.7	44444444D	666666666	true
4	87608	2018-12...	16:52:00	true	9.4	66666666F	999999999	true
5	65812	2017-04...	21:16:00	true	12.8	33333333C	333333333	true
6	1	2011-01...	01:01:00	true	13.9	11111111A	111111111	true
7	2	2012-02...	02:02:00	false	17.7	22222222B	[null]	false
8	3	2013-03...	03:03:00	false	19.9	33333333C	[null]	false
9	4	2014-04...	14:14:00	false	18.29	33333333C	[null]	false
10	5	2015-05...	15:15:00	true	13.2	44444444D	888888888	true
11	6	2016-06...	16:16:00	true	13.2	11111111A	555555555	true
12	7	2017-07...	17:17:00	true	14.36	66666666F	10101010	true
13	21001	2017-06...	01:01:00	true	27.48	66666666F	111111111	false
14	22001	2018-11...	09:30:00	true	12.3	55555555E	999999999	true
15	23001	2019-09...	11:30:00	true	3.26	44444444D	888888888	true
16	24001	2006-05...	12:30:00	true	12.67	55555555E	777777777	true
17	25001	2018-08...	13:31:00	true	32	22222222B	222222222	true
18	21201	2017-07...	05:01:00	false	22.38	66666666F	111111111	false
19	25333	2019-05...	15:31:00	true	3	22222222B	222222222	true
20	25334	2019-05...	19:31:00	true	21	22222222B	777777777	true

9. Product

	barCode [PK] integer	name_product character (40)	priceVAT real	priceNotVAT real	week_discount integer	startDate date	endDate date
1	123456789	Bread ...	0.4	0.38	40	2019-12-02	2019-12-08
2	123456701	Serrano ham ...	10.4	9.2	4	2019-12-09	2019-12-15
3	123111701	Nougat, ...	6	5	40	2019-12-13	2019-12-20
4	123121701	Pantene Shampoo ...	2	1.88	23	2019-01-12	2019-01-19
5	123347701	Goat Cheese ...	4.99	4.1	22	2018-05-24	2018-05-31
6	456456789	Nivea cream ...	5	4.5	40	2019-12-02	2019-12-08
7	789456789	Chicken Pie ...	3.7	3.01	22	2019-05-24	2019-05-30
8	147456789	Barceló 75cl. ...	15.5	14.2	18	2019-05-01	2019-05-05
9	123456715	Lamb leg ...	10.4	9.2	4	[null]	[null]
10	123456720	Loin Spetec ...	18.4	16.34	23	[null]	[null]
11	123456733	Jack Daniel's 1L ...	26.95	25.6	22	[null]	[null]
12	123456731	Prawns 5kg. ...	7.81	6.56	18	[null]	[null]

RELATIONSHIPS

1. **Purchased (muchos_ticket_tiene_muchos_product)**

Data Output Explain Messages

	ID_ticket [PK] character (40)	barCode_product [PK] integer	amount integer
1	34237	123456789	1
2	34237	123456701	3
3	34237	123456720	5
4	98695	123456789	3
5	56631	456456789	3
6	56631	147456789	7
7	87608	456456789	5
8	87608	123111701	2
9	87608	123456789	1
10	65812	123121701	4
11	1	123121701	6
12	1	123456720	2
13	1	123456733	3
14	2	123111701	4
15	2	456456789	2
16	2	123456715	5
17	2	123456720	2
18	3	123456731	2
19	3	789456789	4
20	3	123111701	2
21	4	123456789	5
22	4	123456701	8
23	5	123121701	2
24	5	456456789	3
25	6	123456701	2
26	6	123111701	4
27	6	123121701	2
28	6	123347701	5
29	6	456456789	4
30	7	789456789	2

2. Returned

	barCode_product [PK] integer	ID_ticket [PK] character (40)	amount integer
1	123456733	22001	1
2	789456789	24001	2
3	123456715	21001	1
4	789456789	7	1
5	123347701	6	2
6	123456715	25001	2
7	123456789	4	1
8	123456720	1	1

3. Affects to (muchos_coupon_tiene_muchos_product)

	ID_coupon [PK] integer	barCode_product [PK] integer
1	123	123456789
2	123	123456701
3	123	123347701
4	123	123456720
5	132	123456789
6	132	456456789
7	132	789456789
8	231	123456720
9	213	147456789
10	213	123347701
11	213	123111701
12	312	123456789
13	312	123456731

4. Has (muchos_member_tiene_muchos_coupon)

	ID_coupon [PK] integer	barCode_product [PK] integer
1	123	123456789
2	123	123456701
3	123	123347701
4	123	123456720
5	132	123456789
6	132	456456789
7	132	789456789
8	231	123456720
9	213	147456789
10	213	123347701
11	213	123111701
12	312	123456789
13	312	123456731

5. **Opines (muchos_worker_tiene_muchos_worker)**

	ID_worker [PK] character (9)	ID_worker1 [PK] character (9)	grade double precision
1	11111111A	22222222B	9
2	11111111A	33333333C	8
3	22222222B	33333333C	5
4	22222222B	44444444D	3
5	33333333C	44444444D	8
6	33333333C	77777777G	7
7	33333333C	45872396L	6
8	44444444D	55555555E	2
9	44444444D	88888888J	6
10	44444444D	66666666F	7
11	55555555E	11111111A	5
12	55555555E	05794628M	7
13	55555555E	33333333C	2
14	66666666F	88888888J	8
15	66666666F	22222222B	5
16	66666666F	77777777G	6
17	77777777G	45872396L	4
18	77777777G	44444444D	9
19	77777777G	66666666F	7
20	88888888J	66666666F	2
21	88888888J	33333333C	5
22	88888888J	05794628M	6
23	99999999K	88888888J	8
24	99999999K	11111111A	5
25	99999999K	44444444D	7
26	45872396L	55555555E	3
27	45872396L	05794628M	2
28	45872396L	22222222B	5
29	05794628M	99999999K	8
30	05794628M	33333333C	10
31	05794628M	66666666F	7

QUERIES

1. Obtain the items in the database, showing the barcode and the price without VAT.

```
SELECT "name_product", "barCode", "priceNotVAT"
FROM product;
```

```
#1

SELECT "name_product", "barCode", "priceNotVAT"
FROM product;
```

Data Output | Explain | Messages | Notifications

	name_product character (40)	barCode [PK] integer	priceNotVAT real
1	Bread	123456789	0.38
2	Serrano ham	123456701	9.2
3	Nougat,	123111701	5
4	Pantene Shampoo	123121701	1.88
5	Goat Cheese	123347701	4.1
6	Nivea cream	456456789	4.5
7	Chicken Pie	789456789	3.01
8	Barceló 75cl.	147456789	14.2
9	Lamb leg	123456715	9.2
10	Loin Spetec	123456720	16.34
11	Jack Daniel's 1L	123456733	25.6
12	Prawns 5kg.	123456731	6.56

We are asked to obtain the barcode and the price without VAT among all the items on the database.
We can find all these attributes on the entity named *product.*
As well as the barcode and the price without VAT we obtain the name of the product in order to distinguish easily the product we are referring to.
We can find our attributes with the following name:

- Name of the product as “name_product”
- Barcode as barCode
- Price without VAT as priceNotVAT

We want all of them, so we can easily put select (*all the attributes*) from its entity, *product*.

2. Obtain the name of every worker indicating if they are cashiers or repleteners.

```
SELECT DISTINCT "name", "type"
FROM "worker"
```

```
1  SELECT DISTINCT "name", "type"
2  FROM "worker"
```

Data Output Explain Messages

	name character (40)	type character (40)
1	Daniel	R
2	Aitor	C
3	Bernardo	C
4	Álvaro	R
5	Angy	C
6	Aarón	C
7	Ander	R
8	Adrián	C
9	Ramón	R
10	Andrea	R
11	Carmen	C

We are asked to obtain the name of the worker indicating if they are cashier or repletener. All these attributes can be found on *worker*.
In order to make this query easier we have created a new attribute called “type”. Among its restrictions is to be a char. It can only be ‘C’ (cashier) or ‘R’ (repletener).
We can find our attributes with the following name:

- Name of the worker as “name”
- Cashier or repletener as “type”

As we want to obtain all the workers, we can just place on the select field the attributes of “name” and “type” from *worker*.

After select we must put distinct because if not some of the workers may appear several times.

All the 11 workers will appear, marking if they are C or R.

3. Obtain the name of the repleteners who work more than 20 hours per week.

```
SELECT "ID", "name", "numberHours"
FROM repletener
WHERE "numberHours">20;
```

```
#3

SELECT "ID", "name", "numberHours"
FROM repletener
WHERE "numberHours">20;
```

Data Output | Explain | Messages | Notifications

	ID [PK] character (9)	name character (40)	numberHours integer
1	77777777G	Álvaro	22
2	99999999K	Ramón	38
3	45872396L	Ander	24

We are asked to obtain the name of the repleteners that work more than 20 hours per week.
In this case we will need to use the entity *repletener* in order to access the information of the repleteners.

To facilitate the understanding we will also show the ID and the number of hours that each repletener works, as well as the name.

We can find our attributes with the following name:

- ID of the repletener as “ID”
- Name of the repletener as “name”
- Number of hours as “numberHours”

This query is very similar to the above ones, but in this case we will have to add the condition of being an attribute bigger than an integer.

We will get on the output only the repleteners than work more than 20 hours if we add the WHERE clause after including that we are selecting the attributes from the entity repleteners.

Only three of the repleteners pass this condition, so through the output we only see 3 different lines.

4.Obtain the total money billed by the supermarket since the implementation of the database.

```
SELECT SUM (total_price) as total_billed
FROM ticket;
```

```
1  SELECT SUM (total_price) as total_billed
2  FROM ticket;
3
```

Data Output | Explain | Messages | Notifications

	total_billed double precision
1	279.05999999999995

We are asked to obtain the total money billed by the supermarket since the implementation of the database. This means we will have to take into account all the tickets without any kind of restrictions, so we won't need to include the WHERE clause.

In this case we will need to use the entity *ticket* in order to access the information of the tickets.

In order to obtain the total amount, we will have to make a sumatory using an aggregate function in the SELECT field.

We can find the attribute with the following name:

- Total price of each of the tickets as total_price

We will calculate the total amount by writing SUM(total_price) and we will assign a name to it so that when the output is created, a new name will be shown and the total amount will be the only number shown.

We have tried to round off the total number giving it only two decimals but it won't work, so we have left the original amount indicated. We tried using TRUNCATE and ROUND.

5. Show customer coupons, along with the products they affect and the discount made

```
SELECT coupon."ID", product.name_product,
muchos_member_tiene_muchos_coupon."membershipCode_member", coupon.percentage
FROM coupon INNER JOIN muchos_member_tiene_muchos_coupon ON "ID" =
"ID_coupon",
muchos_coupon_tiene_muchos_product INNER JOIN product ON "barCode_product" =
"barCode" ;
```

```
SELECT coupon."ID", product.name_product, muchos_member_tiene_muchos_coupon."membershipCode_member", coupon.percentage
FROM coupon INNER JOIN muchos_member_tiene_muchos_coupon ON "ID" = "ID_coupon",
muchos_coupon_tiene_muchos_product INNER JOIN product ON "barCode_product" = "barCode";
```

Data Output Explain Messages

	ID integer	name_product character (40)	membershipCode_member integer	percentage integer
1	123	Bread	111111111	20
2	132	Bread	111111111	15
3	231	Bread	333333333	22
4	213	Bread	444444444	12
5	312	Bread	555555555	13
6	123	Bread	555555555	20
7	132	Bread	666666666	15
8	231	Bread	777777777	22
9	213	Bread	777777777	12
10	312	Bread	777777777	13
11	123	Bread	888888888	20
12	132	Bread	10101010	15
13	123	Serrano ham	111111111	20
14	132	Serrano ham	111111111	15
15	231	Serrano ham	333333333	22
16	213	Serrano ham	444444444	12
17	312	Serrano ham	555555555	13
18	123	Serrano ham	555555555	20
19	132	Serrano ham	666666666	15
20	231	Serrano ham	777777777	22
21	213	Serrano ham	777777777	12
22	312	Serrano ham	777777777	13
23	123	Serrano ham	888888888	20
122	132	Nougat,	111111111	15
123	231	Nougat,	333333333	22
124	213	Nougat,	444444444	12
125	312	Nougat,	555555555	13
126	123	Nougat,	555555555	20
127	132	Nougat,	666666666	15
128	231	Nougat,	777777777	22
129	213	Nougat,	777777777	12
130	312	Nougat,	777777777	13
131	123	Nougat,	888888888	20
132	132	Nougat,	10101010	15
133	123	Bread	111111111	20
134	132	Bread	111111111	15
135	231	Bread	333333333	22
136	213	Bread	444444444	12
137	312	Bread	555555555	13
138	123	Bread	555555555	20
139	132	Bread	666666666	15
140	231	Bread	777777777	22
141	213	Bread	777777777	12
142	312	Bread	777777777	13
143	123	Bread	888888888	20
144	132	Bread	10101010	15
145	123	Prawns 5kg.	111111111	20
146	132	Prawns 5kg.	111111111	15
147	231	Prawns 5kg.	333333333	22
148	213	Prawns 5kg.	444444444	12
149	312	Prawns 5kg.	555555555	13
150	123	Prawns 5kg.	555555555	20
151	132	Prawns 5kg.	666666666	15
152	231	Prawns 5kg.	777777777	22
153	213	Prawns 5kg.	777777777	12
154	312	Prawns 5kg.	777777777	13
155	123	Prawns 5kg.	888888888	20

We are asked to show the customers coupons, the products the affect and the discount they made. This query ir a little bit more complex than the previous ones because we have to compare between several entities, that in this case are coupon, muchos_member_tiene_muchos_coupon (has) and product.
We can find the attribute with the following name:

- ID of the coupon as “ID”
- The name of the product as “name_product”
- Each member membership code as “membershipCode_member”
- The percentage of each coupon as “percentage”

In order to compare the information we will use the INNER JOIN clause and then ON to make the reference between the two attributes we want to use.
As we don’t want any specific information to be shown we don’t need to use the WHERE clause and all the results will be on the output.

Since this query implies 156 rows, we have only added the first ones and the last ones on the picture above.

6. Show 5 products on which members have discount coupons

```
SELECT DISTINCT product.name_product,
muchos_member_tiene_muchos_coupon."membershipCode_member"
FROM coupon INNER JOIN muchos_member_tiene_muchos_coupon ON "ID" =
"ID_coupon",
muchos_coupon_tiene_muchos_product INNER JOIN product ON "barCode_product" =
"barCode"
LIMIT 5 ;
```

```
1  SELECT DISTINCT product.name_product, muchos_member_tiene_muchos_coupon."membershipCode_member"
2  FROM coupon INNER JOIN muchos_member_tiene_muchos_coupon ON "ID" = "ID_coupon",
3  muchos_coupon_tiene_muchos_product INNER JOIN product ON "barCode_product" = "barCode"
4  LIMIT 5 ;
5
```

Data Output Explain Messages

	name_product character (40)	membershipCode_member integer
1	Barceló 75cl. ...	10101010
2	Barceló 75cl. ...	111111111
3	Barceló 75cl. ...	333333333
4	Barceló 75cl. ...	444444444
5	Barceló 75cl. ...	555555555

We are asked to show 5 products on which members have discount coupons

As we can see this query is asking for the same thing as the previous exercise, even though we only need to show the products, we will show as well the things we obtained in the last exercise to see the difference between both queries in a clearer way.

We will use the limit clause in order to reduce the rows that will be shown.

In this case instead of 156 rows only 5 will be shown because of the limitation that we have included as it is required in this query.

7. Determine the average degree of satisfaction of the opinions that customers have made online, showing the average score.

```
SELECT round(avg(score), 2) as m_score
FROM opinion ;
```

We are asked to obtain the average grade of satisfaction of the opinions and show the average score. This means we will have to take into account all the opinions without any kind of restrictions, so we won't need to include the WHERE clause.

In this case we will need to use the entity *opinion* in order to access the information of the degree of satisfaction..

In order to obtain the average grade, we will have to make a calculation using an aggregate function in the SELECT field.

We can find the attribute with the following name:

- Degree of satisfaction of the customers as score

We will calculate the total grade by writing avg(score) and we will assign a name to it so that when the output is created, a new name will be shown and the average score will be the only number shown.

We have rounded total number by using round and giving '2' as the number of decimals that will be shown on the created table.

8. Determine the number of tickets that each supermarket has issued, showing the number of tickets, the name of the cashier and the city of the supermarket where the cashier works. Sort the output from highest to lowest.

```
SELECT "name", "city_supermarket", "district_supermarket", count("ticket") as n_ticket
FROM "cashier" INNER JOIN "ticket"
ON ("cashier"."ID" = "ticket"."ID_cashier")
GROUP BY "city_supermarket", "district_supermarket", "name"
ORDER BY count("ticket") desc;
```

```
# 8

SELECT "name", "city_supermarket", "district_supermarket", count("ticket") as n_ticket
FROM "cashier" INNER JOIN "ticket"
ON ("cashier"."ID" = "ticket"."ID_cashier")
GROUP BY "city_supermarket", "district_supermarket", "name"
ORDER BY count("ticket") desc;
```

Data Output | Explain | Messages | Notifications

	name character (40)	city_supermarket character (40)	district_supermarket character (40)	n_ticket bigint
1	Bernardo	Madrid	Carabanchel	4
2	Aitor	Alcalá de Henares	La Garena	3
3	Angy	Madrid	Carabanchel	3
4	Carmen	Madrid	Carabanchel	3
5	Aarón	Madrid	Gran Vía	2
6	Adrián	Mallorca	Arenal	2

We are asked to determine the number of tickets that each supermarket has issued so we have to add the amount of tickets the cashiers of each supermarket have issued and sort them from the highest to the lowest. The attributes used in this query are found on the entities *cashier, ticket* and *supermarket*.

- From Cashier we get its ID and the supermarket were them work.
- From Ticket we get the cashier's ID
- From Supermarket we get the location of the supermarket.

We count the number of tickets grouped by supermarket and then show them ordered from the highest amount to the lowest along with the supermarket city and district, in order to obtain the cashier name we use an inner join between *cashier* and *ticket* where ID on *cashier* equals ID_cashier on *ticket*.

So, for example, we see that Bernardo issued 4 tickets in the supermarket of Carabanchel (Madrid).

9. Determine the number of workers each supermarket has, ordering the exit from lowest to highest.

```
SELECT "city_supermarket", "district_supermarket", count("worker") as n_workers
FROM "worker"
GROUP BY "city_supermarket", "district_supermarket"
ORDER BY count("worker") asc;
```

```
# 9

SELECT "city_supermarket", "district_supermarket", count("worker") as n_workers
FROM "worker"
GROUP BY "city_supermarket", "district_supermarket"
ORDER BY count("worker") asc;
```

Data Output | Explain | Messages | Notifications

	city_supermarket character (40)	district_supermarket character (40)	n_workers bigint
1	Murcia	La Flota-Vista Alegre	2
2	Granada	Albaicín	2
3	Mallorca	Arenal	2
4	Madrid	Gran Vía	2
5	Málaga	Palma-Palmilla	2
6	Alcalá de Henares	La Garena	2
7	Alcalá de Henares	Espartales	4
8	Madrid	Carabanchel	6

We are asked to determine the number of workers each supermarket has, but the output must have a specific order: from the lowest to the highest.

We have needed the entities Worker and Supermarket to get the number of workers of each supermarket.

In order to obtain the required number, we will have to make a calculation using an aggregate function in the SELECT field. We will get it by writing count(“worker”).

We have obtained the order with the instruction “**ORDER BY count**("worker") **asc**”.

10. Show the name and telephone number of the employee with the best score

```
SELECT DISTINCT worker.name, worker.phone,
muchos_worker_tiene_muchos_worker.grade
FROM worker INNER JOIN muchos_worker_tiene_muchos_worker ON "ID" = "ID_worker"
WHERE grade = (SELECT max(grade)
               FROM "muchos_worker_tiene_muchos_worker");
```

```
8  SELECT DISTINCT worker.name, worker.phone, muchos_worker_tiene_muchos_worker.grade
9  FROM worker INNER JOIN muchos_worker_tiene_muchos_worker ON "ID" = "ID_worker"
10 WHERE grade = (SELECT max(grade)
11                FROM "muchos_worker_tiene_muchos_worker");
```

Data Output Explain Messages

	name character (40)	phone integer	grade double precision
1	Daniel	613655637	10

We are asked to show the name and the telephone number of the employee with the best score, so we have needed the entity Worker and the relationship muchos_worker_tiene_muchos_worker (that in our pgModeler document has a different name: opines).

We use WHERE because we have a condition: it must be the worker with the highest score.
In order to getting it we select the highest grade on the relationship "*muchos_worker_tiene_muchos_worker*" from where we get the worker ID with that grade on the relationship, for getting the name and phone we need to use an inner join between *worker* and "*muchos_worker_tiene_muchos_worker"* where ID on *worker* equals ID_worker on "*muchos_worker_tiene_muchos_worker".*

We get that this worker is Daniel with a score of 10.

11. Show the barcode and the discount of the products that were on sale the first week of May 2019.

```
SELECT product.name_product, product."barCode", discount.percentage
FROM product INNER JOIN "discount"
ON (product.week_discount = "discount".week )
WHERE product.week_discount = 18;
```

```
SELECT product.name_product, product."barCode", discount.percentage
FROM product INNER JOIN "discount"
ON (product.week_discount = "discount".week )
WHERE product.week_discount = 18;
```

Data Output Explain Messages Notifications

	name_product character (40)	barCode integer	percentage integer
1	Barceló 75cl.	147456789	30
2	Prawns 5kg.	123456731	30

We are asked to show the barcode and the discount of the products that were on sale the first week of May 2019, so we have to get the week from Discount to check that it is the week number 18 of the year (the first of May). The attributes that we use on this query can be found on the entities *discount* and *product*.

We get the products who have a discount on the week 18 and show their name and bar code, for getting the discount that is applied on them we get the percentage that the discount rebates by using an inner join between *product* and *discount* where week_discount on *product* equals week on *discount* and then show the percentage it rebates.

We get that the products that had a discount that week were Barcelo and Prawns with a 30% discount.

12. Show the name of the members who have benefited from discounts applied to a product the last week of May 2019

SELECT member.name, ticket.date, ticket."timeIssuance"
FROM member **INNER JOIN** ticket
ON member."membershipCode" = ticket."membershipCode_member"
WHERE (ticket.date BETWEEN '05-27-2019' **AND** '05-31-2019') **AND** ticket.coupon_used = TRUE;

```
SELECT member.name, ticket.date, ticket."timeIssuance"
FROM member INNER JOIN ticket
ON member."membershipCode" = ticket."membershipCode_member"
WHERE (ticket.date BETWEEN '05-27-2019' AND '05-31-2019') AND ticket.coupon_used = TRUE;
```

Data Output | Explain | Messages

	name character (40)	date date	timeIssuance time without time zone
1	Carlos Bernal	2019-05...	15:31:00
2	Lucas Martin	2019-05...	19:31:00

We are asked to obtain the name of the members who have benefited from discounts on the las week of May 2019, the attributes used in this query can be found on the entities *ticket*, *member* and the relationship "*muchos_ticket_thiene_muchos_product*" with the names:

- Member name as name on *member*.
- Ticket date and time as name and timeIssuance on *ticket*.

In order to get the members who have benefited from discounts on the last week of May 2019 we select the tickets whose date belongs to this week and where the coupon_used boolean is true, then we select the member name using an inner join between *ticket* and *member* where membershipCode on *member* equals membershipCode_member on *ticket*.

We get that just Carlos Bernal and Lucas Martin benefited from them.

13. Show the name of the workers in alphabetical order of the supermarkets located in cities that begin with “M”.

```
SELECT DISTINCT "name"
FROM "worker" INNER JOIN supermarket
ON "worker".city_supermarket = supermarket.city AND supermarket.city LIKE 'M%'
ORDER BY "worker".name;
```

```
# 13

SELECT DISTINCT "name"
FROM "worker" INNER JOIN supermarket
ON "worker".city_supermarket = supermarket.city AND supermarket.city LIKE'M%'
ORDER BY "name";
```

Data Output | Explain | Messages | Notifications

	name character (40)
1	Aarón
2	Adrián
3	Andrea
4	Angy
5	Bernardo
6	Carmen
7	Ramón

We are asked to obtain the name of the workers who work in a city whose name starts with “M” ordered alphabetically, the attributes used on this query can be found on the entities *supermarket* and *worker* with the name:

- Worker name as name on *worker*.

In order to obtain the names of the workers who work on a city with a name that starts with “M” we select the supermarkets which attribute city starts with “M” and then we select the workers with an inner join between *worker* and *supermarket* where city_supermarket on *worker* equals city on *supermarket*, for sorting them in an alphabetical order we use the command order by.
We need to use the command distinct so the names don’t repeat.

14. Show the email of the member whose total amount accumulated is the highest

```
SELECT email
FROM "member"
WHERE "member"."pointsAccumulated" = (SELECT max("member"."pointsAccumulated")
                                      FROM "member")
```

```
# 14

SELECT email
FROM "member"
WHERE "member"."pointsAccumulated" = (SELECT max("member"."pointsAccumulated")
                                      FROM "member")
```

Data Output | Explain | Messages | Notifications

	email character (40)
1	israel.plaza@edu.u...

We are asked to obtain the email of the member who has accumulated the highest amount of points, the attributes used in this query can be found on the entity *member*.

In order to obtain the email from the member with the highest amount of points accumulated we select the highest number of points from all the members so we found the member with the highest amount of them and then show the email of this member.

we get that the email from the member with the highest amount of points accumulated is israel.plaza@edu.uah.es.

15. Show the product that has been returned the most times.

```
SELECT DISTINCT product.name_product, product."barCode"
FROM product INNER JOIN returned ON "barCode" = "barCode_product"
WHERE amount = (SELECT max(amount)
                FROM "returned");
```

```
1  SELECT DISTINCT product.name_product, product."barCode"
2  FROM product INNER JOIN returned ON "barCode" = "barCode_product"
3  WHERE amount = (SELECT max(amount)
4                  FROM "returned");
```

Data Output Explain Messages

	name_product character (40)	barCode [PK] integer
1	Chicken Pie	789456789
2	Goat Cheese	123347701
3	Lamb leg	123456715

We are asked to obtain the product which has been returned the most times, the attributes used on this query can be found on the entity *product* and the relationship "*returned*" with the names:

- Product name as name on *product*.
- Product bar code as barCode on *product*.

In order to obtain the product that has been returned the most times we select the highest amount from the relationship "*returned*" and we select the name and the bar code of that product with an inner join between *product* and "*returned*" where barCode on *product* equals barCode_product on "*returned*".

We obtain that the products that have been returned the most times have been the chicken pie, the goat cheese and the lab leg.

16. Show the name of the cashier that has issued the most tickets

```
SELECT "name", count("ticket")
FROM "cashier" INNER JOIN "ticket"
ON ("cashier"."ID" = "ticket"."ID_cashier")
GROUP BY "name"
HAVING count("ticket") = (SELECT max(counter)
        FROM (SELECT count("ID_cashier") as counter
                FROM "ticket"
                GROUP BY "ID_cashier")as r);
```

```
#16

SELECT "name", count("ticket")
FROM "cashier" INNER JOIN "ticket"
ON ("cashier"."ID" = "ticket"."ID_cashier")
GROUP BY "name"
HAVING count("ticket") = (SELECT max(counter)
                          FROM (SELECT count("ID_cashier") as counter
                                FROM "ticket"
                                GROUP BY "ID_cashier")as r);
```

Data Output | Explain | Messages | Notifications

	name character (40)	count bigint
1	Bernardo	4

We are asked to obtain the name of the cashier who has issued the most tickets, the attributes used in this query are found on the entities *ticket* and *cashier* with the names:

- Cashier name as name on *cashier*.
- Ticket ID as ID on *ticket*.

In order to obtain the cashier who has issued the most tickets we count the tickets grouping them by the ID of the cashier who issued them and select the bigger group so we obtain the ID of the cashier who has issued the most tickets, but since we need the name and not the ID we do an inner join between *ticket* and *cashier* where ID on *cashier* equals ID_cashier on *ticket*.

We get that the cashier who issued the most tickets is called Bernardo and he has issued 4 tickets.

17. Show the name of the member that has issued the best opinion (the highest score)

```
SELECT "name", "score"
FROM "member" INNER JOIN opinion
ON ("member"."membershipCode" = opinion."membershipCode_member")
WHERE opinion."score" = (SELECT max(opinion."score")
                         FROM opinion);
```

```
#17

SELECT "name", "score"
FROM "member" INNER JOIN opinion
ON ("member"."membershipCode" = opinion."membershipCode_member")
WHERE opinion."score" = (SELECT max(opinion."score")
                         FROM opinion);
```

Data Output | Explain | Messages | Notifications

	name character (40)	score integer
1	Marta Bermejo	8

We are asked to obtain the name of the member who has issued the best opinion, the attributes used in this query can be found on the entities *member* and *opinion* with the names:

- The score given by the member as score on *opinion.*
- Member name as name on *member*.

In order to obtain the member with the best score issued we select highest score from *opinion* and we obtain the name of the member who issued it with an inner join between *opinion* and *member* where membershipCode from *member* equals membershipCode_member on *opinion*.

We get that this member was Marta Bermejo and she issued and opinion with a score of 8.

18. Show the tickets issued by cashiers whose name begins with “A” and works in cities that begin with “M”

```
SELECT DISTINCT ticket."ID", "name", "city"
FROM cashier INNER JOIN "ticket"
ON cashier."ID" = "ticket"."ID_cashier"
INNER JOIN "supermarket"
ON supermarket.city = cashier.city_supermarket
WHERE cashier."name" LIKE 'A%' and "supermarket".city like 'M%';
```

```
1  SELECT DISTINCT ticket."ID", "name", "city"
2  FROM cashier INNER JOIN "ticket"
3  ON cashier."ID" = "ticket"."ID_cashier"
4  INNER JOIN "supermarket"
5  ON supermarket.city = cashier.city_supermarket
6  WHERE cashier."name" LIKE 'A%' and "supermarket".city like 'M%';
```

Data Output Explain Messages

	ID character (40)	name character (40)	city character (40)
1	1	Aarón	Madrid
2	22001	Adrián	Mallorca
3	23001	Angy	Madrid
4	24001	Adrián	Mallorca
5	5	Angy	Madrid
6	56631	Angy	Madrid
7	6	Aarón	Madrid

We are asked to obtain the tickets issued by cashier whose name start with “A” and work on a city whose name start with “M”, the attributes used on this query can be found on the entities *cashier*, *supermarket* and *ticket* under the names:

- Ticket ID as ID on *ticket.*
- Cashier name as name on *cashier*.
- Supermarket city as city on *supermarket*.

In order to obtain the specified tickets we select the cashier who have a name that start with “A” and work in a city that start with “M” and then we show the tickets that have been issued by that cashiers using an inner join between *cashier* and *ticket* where ID from *cashier* is the same that the ID_cashier on *ticket.*

19. Show the id of the tickets issued in the supermarkets of Alcalá de Henares along with the name of the cashier

```
SELECT DISTINCT cashier."name", ticket."ID"
FROM cashier INNER JOIN "ticket"
ON cashier."ID" = "ticket"."ID_cashier"
INNER JOIN "supermarket"
ON supermarket.city = cashier.city_supermarket
WHERE supermarket.city = 'Alcalá de Henares';
```

```
1  SELECT DISTINCT cashier."name", ticket."ID"
2  FROM cashier INNER JOIN "ticket"
3  ON cashier."ID" = "ticket"."ID_cashier"
4  INNER JOIN "supermarket"
5  ON supermarket.city = cashier.city_supermarket
6  WHERE supermarket.city = 'Alcalá de Henares';
```

Data Output Explain Messages

	name character (40)	ID character (40)
1	Aitor	21001
2	Aitor	21201
3	Aitor	7
4	Aitor	87608

We are asked to obtain the ID of the tickets that have been issued on Alcala de Henares and the name of the cashier that issued them, these attributes can be found on the entities *ticket* and *cashier* but we need the entity *supermarket* too so we can select the city.

The attributes used in the query can be found with the names:

- Cashier name as name on the entity *cashier*.
- Ticket ID as ID on the entity *ticket*.
- Supermarket city as city on the entity *supermarket*.

In order to create the query we select the tickets that have been issued on Alcala de Henares by selecting the city from the supermarket entity and then select the cashier who work in that supermarket. Once we have the cashier from that supermarkets we use an inner join between them and the tickets so the cashier ID equals the ID_cashier on the ticket.

When executing this query we obtain that 4 tickets have been issued on Alcala de Henares by the cashier Aitor.

20. Perform the same query as the previous point but for those tickets in which no discount coupons have been used

```
SELECT DISTINCT cashier."name", ticket."ID"
FROM cashier INNER JOIN "ticket"
ON cashier."ID" = "ticket"."ID_cashier"
INNER JOIN "supermarket"
ON supermarket.city = cashier.city_supermarket
WHERE supermarket.city = 'Alcalá de Henares' and ticket.coupon_used = FALSE;
```

```
1  SELECT DISTINCT cashier."name", ticket."ID"
2  FROM cashier INNER JOIN "ticket"
3  ON cashier."ID" = "ticket"."ID_cashier"
4  INNER JOIN "supermarket"
5  ON supermarket.city = cashier.city_supermarket
6  WHERE supermarket.city = 'Alcalá de Henares' and ticket.coupon_used = FALSE;
```

Data Output Explain Messages

	name character (40)	ID character (40)
1	Aitor	21201

We are asked to obtain the ID of the tickets that have been issued on Alcala de Henares and in which no discount has been used along with the name of the cashier that issued them, these attributes can be found on the entities *ticket* and *cashier* but we need the entity *supermarket* too so we can select the city.
We have added a boolean attribute on ticked that indicates if a discount has been used on the ticked.

The attributes used in the query can be found with the names:

- Cashier name as name on the entity *cashier*.
- Ticket ID as ID on the entity *ticket*.
- Supermarket city as city on the entity *supermarket*.
- coupon_used on ticked indicates if a discount has been used.

In order to create the query we select the tickets that have been issued on Alcala de Henares by selecting the city from the supermarket entity then select the cashier who work in that supermarket. Once we have the cashier from that supermarkets we use an inner join between them and the tickets so the cashier ID equals the ID_cashier on the ticket, from that tickets only the ones on which no discount has been used are selected

When executing this query we obtain that 1 tickets have been issued on Alcala de Henares with no discount used on it and it was issued by the cashier Aitor.