

# Proyecto Base de Datos Academia Atentos



Andrea Vieira Hernández  
DAW1 - Mañana  
IES Alixar  
Curso 2021-2022

# Índice de contenidos

Índice de contenidos	2
Introducción del proyecto	4
Presentación de la situación	5
Diagrama Entidad-Relación	6
Diagrama Relacional	7
Creación de la base de datos	8
Carga masiva de datos	10
Consultas - Queries	13
Consulta: El empleado que sea coordinador de todos, porque es el dueño de la academia, debe aparecer toda la información en una sola columna, donde se recoja el nombre completo, la ciudad y provincia donde vive y cuánto cobra.	13
Consulta: Los alumnos de Castilleja de la Cuesta matriculados en Apoyo Escolar que asisten de manera presencial	13
Consulta: Los alumnos que tienen clase los lunes a las 20:00 en cualquier aula y de cualquier asignatura	14
Consulta: Poblaciones de Sevilla con más de una persona relacionada con la academia	14
Consulta: Nombre y apellidos de los alumnos en una misma columna que pagaron más que la media durante el año 2020 en Tomares. Se debe indicar la cantidad pagada y la fecha del recibo	15
Consulta: El profesor que haya estado en activo durante más tiempo, pero que ya no trabaje en la academia	15
Vistas - View	16
Vista Jefe: el empleado que sea coordinador de todos, porque es el dueño de la academia	16
Vista Poblaciones frecuentes: poblaciones de Sevilla con más cantidad de personas relacionadas con la academia	16
Vista Profesor más antiguo: el profesor que haya estado en activo durante más tiempo, pero que ya no trabaje en la academia	17
Funciones - Functions	17
Función: Se crea una función para calcular la cantidad de personas relacionadas con la academia cuya dirección pertenezca a una localidad, pasándole como parámetro el nombre de dicha población.	17
Función: Se crea una función para calcular el balance del mes (que se pasa su número por parámetro) con los usuarios de una población (que se pasa por parámetro).	18
Función: Se crea una función para comprobar si existe un alumno en los registros. Esta función será tremendamente útil en procedimientos, pues nos ayuda a capturar posibles excepciones.	18

Procedimientos - Procedures	19
Procedimiento con función: Genera un pequeño informe sobre las inscripciones que haya podido realizar un alumno en el centro, de manera que se pueda evaluar el cambio de curso, las tarificaciones y las distintas fechas de inicio de curso, además controla la excepción en el caso que se introduzca un código de alumno que no exista.	19
Procedimiento con función: Actualiza el curso de los alumnos solo si el alumno existe, además controla la excepción en el caso que se introduzca un código de alumno que no exista.	20
Procedimiento con cursor: evalúa si en una localidad en concreto, un mes ha tenido un balance superior a 1.000€, a todos los empleados que sean de ese lugar, se les aumentará el salario un 1% siempre y cuando su salario no sea superior a 3.000€.	21
Disparadores - Triggers	22
Disparador: Se inserta en una tabla (AntiguosAlumnos) que funciona a modo de almacén, donde se guardan los datos fundamentales de los alumnos, cuando estos se eliminan de la tabla Alumnos.	22
Disparador: se dispara una inserción en una tabla de seguridad para el histórico de pagos, con los datos fundamentales del alumno y del pago. Añade una excepción si se añade una fecha posterior a la fecha actual.	23
Conclusiones y valoración personal	24

# Introducción del proyecto

En el proyecto que se presenta a continuación, queremos representar una situación lo más cercana posible a la realidad, relacionada con el análisis de datos que se desea almacenar de una empresa para su posterior incorporación a una interfaz gráfica con funcionalidades, que permita interactuar con el sistema y tener acceso a dichos datos.

En este caso particular, se analizan las necesidades de una academia de apoyo escolar, con unas necesidades particulares.

Para la elaboración de este proyecto utilizaremos diversas herramientas según la fase en la que nos encontremos. En primer lugar, para el diseño entidad-relación, utilizaremos la aplicación web de Diagrams.net, pues nos permite crear de manera muy sencilla y gráfica el dibujo de las relaciones.

Por otro lado, para pasar este formato de dibujo al esquema de tablas, es decir, al modelo relacional, utilizaremos la herramienta Workbench, por su capacidad para hacer el diseño relacional con los distintos tipos de relaciones, y que se generen de manera automática el traslado de las claves foráneas y la creación de las tablas que surgen por las relaciones N:M y la definición de los tipos de campos; además, nos permite generar archivos csv y scripts.

Finalmente, para la carga de datos y las consultas utilizaremos DBeaver, un gestor gráfico de base de datos. Cabe resaltar que se trata de un proyecto evolutivo, es decir, que va a evolucionar a medida que avancemos en el proceso de creación. Partiremos de los bocetos creados hasta que, en el punto final, la creación de consultas y ver cómo se relacionan las tablas, hasta dar por cerrada toda la macroestructura de dicho proyecto.

## Presentación de la situación

La Academia desea tener una base de datos donde almacenar los datos relacionados con sus alumnos, sus empleados, los padres de los alumnos, la facturación y los horarios. De todas las personas que están en el centro se desea guardar el nombre y los apellidos, el DNI, el número de teléfono, el correo electrónico y el identificador que asignará el centro. Todas las personas tienen también una dirección de la cual se quiere destacar la población; de cada población se desea conocer el código postal, el nombre y la provincia.

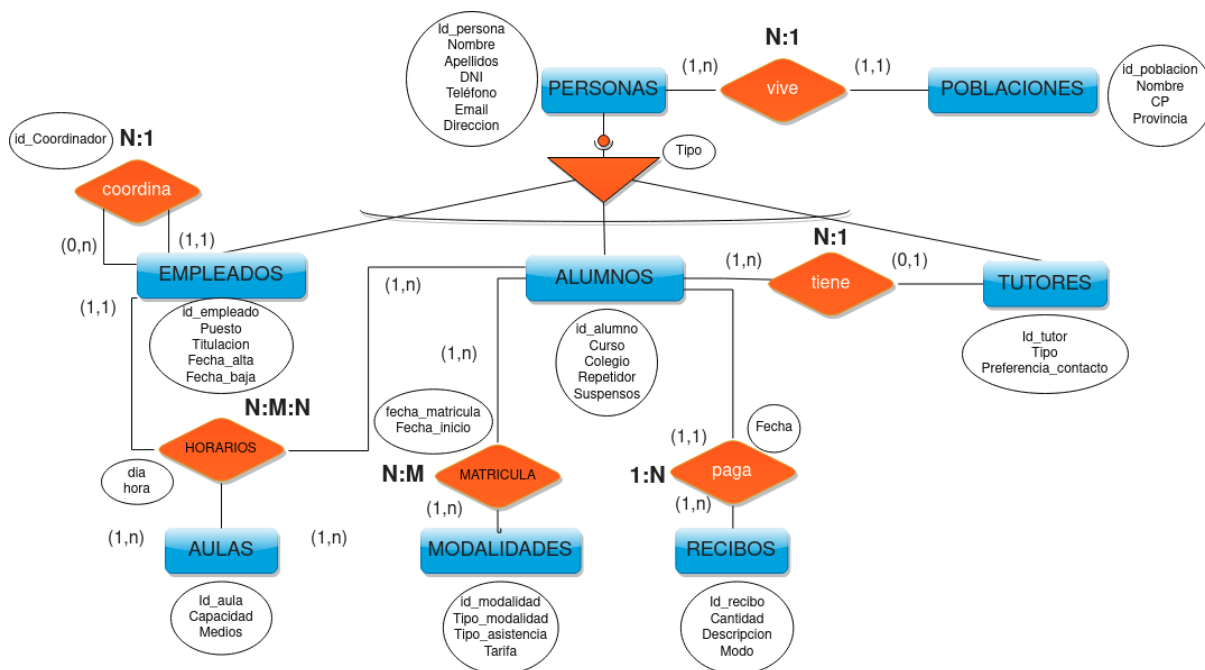
Cada alumno, del que se debe indicar el colegio del que viene, su curso, si es repetidor y la cantidad de asignaturas suspensas, se matricula, en una fecha para un inicio específico, a una de las modalidades de las que dispone la academia y paga los recibos mensualmente. De las modalidades se almacena el tipo de modalidad (general, específica o idiomas), el tipo de asistencia (presencial, online o mixta) y la tarifa que se le aplica. De los recibos se desea saber la cantidad, la fecha en la que se hace, el modo (efectivo, tarjeta o transferencia) y una descripción. Cada alumno tiene un tutor con el que mantiene contacto la academia, de esos tutores se necesita saber si es la madre, el padre u otro familiar, y la forma de contacto que prefieren. Los alumnos se ubican en aulas, que tiene una capacidad determinada y dispone de unos medios, en día y a una hora en concreto, en ese aula tienen a un profesor que es quien imparte las clases.

La academia tiene distintas clases de empleados (profesores, auxiliares, administrativos y coordinadores), los coordinadores pueden realizar cualquiera de las funciones en la empresa, y el resto tiene unas tareas específicas. De todos estos empleados se desea guardar el puesto que tienen, la titulación, la fecha de alta y la fecha de baja, que aparecerá solo si no continúan en la empresa. Cabe destacar, por último, que todos los profesores, auxiliares y administrativos tienen un coordinador.

Estos serían los requisitos iniciales para la elaboración de los bocetos, teniendo en cuenta que luego pueden surgir modificaciones.

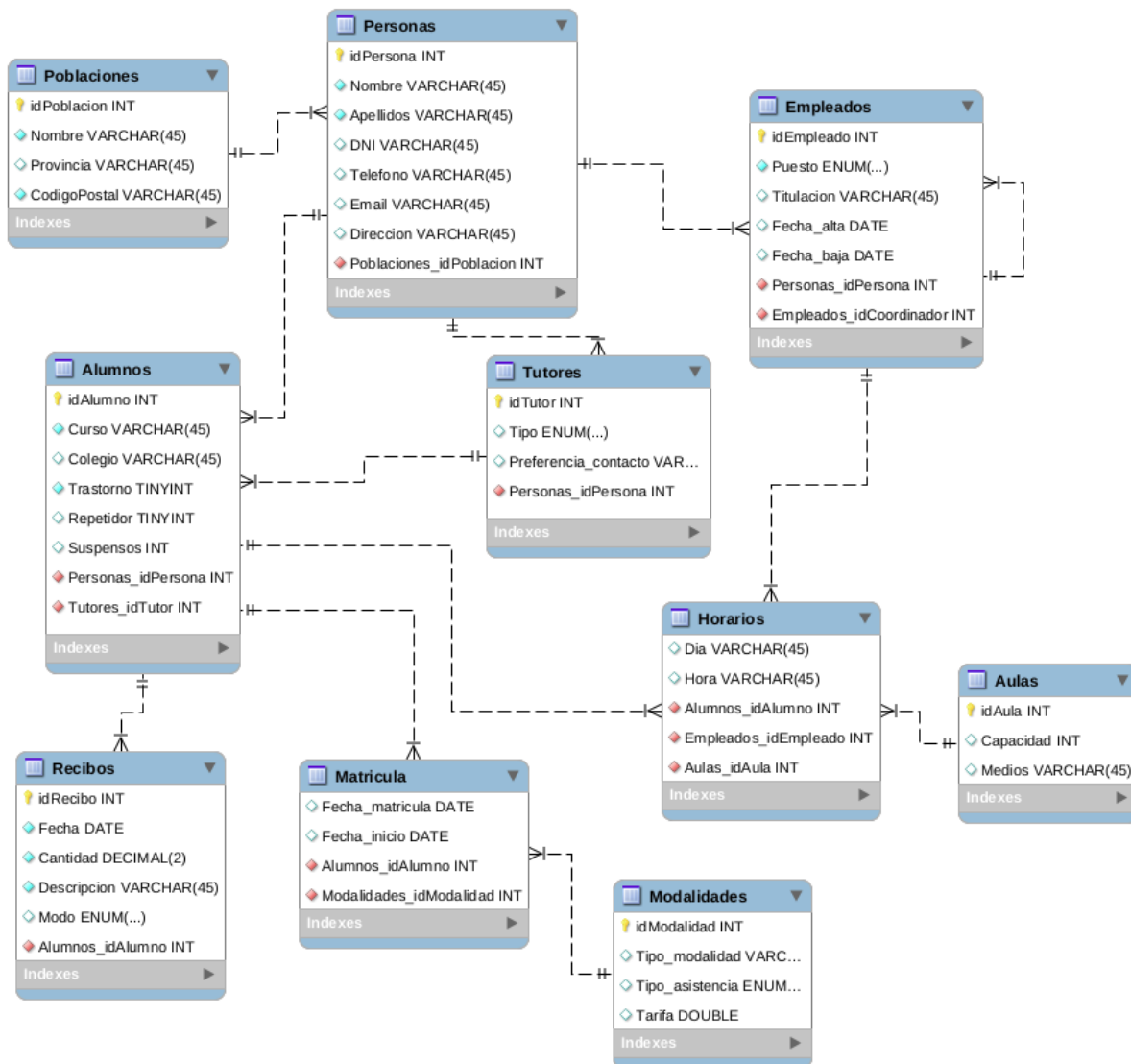
## Diagrama Entidad-Relación

Con el modelo que se presenta a continuación se pretende ilustrar la organización de las entidades y qué relación existe entre ellas. Destacamos la entidad ALUMNOS, pues es sobre la que caerá la mayor parte de la información fundamental que pueda necesitar la empresa.



# Diagrama Relacional

En este modelo, elaborado con Workbench, podemos ver en disposición tabla los elementos que compondrán la base de datos, junto con sus campos y el tipo de valor que se registra en cada uno de ellos. Gracias a este esquema podemos observar el punto de conexión entre las tablas, fundamental para realizar consultas.



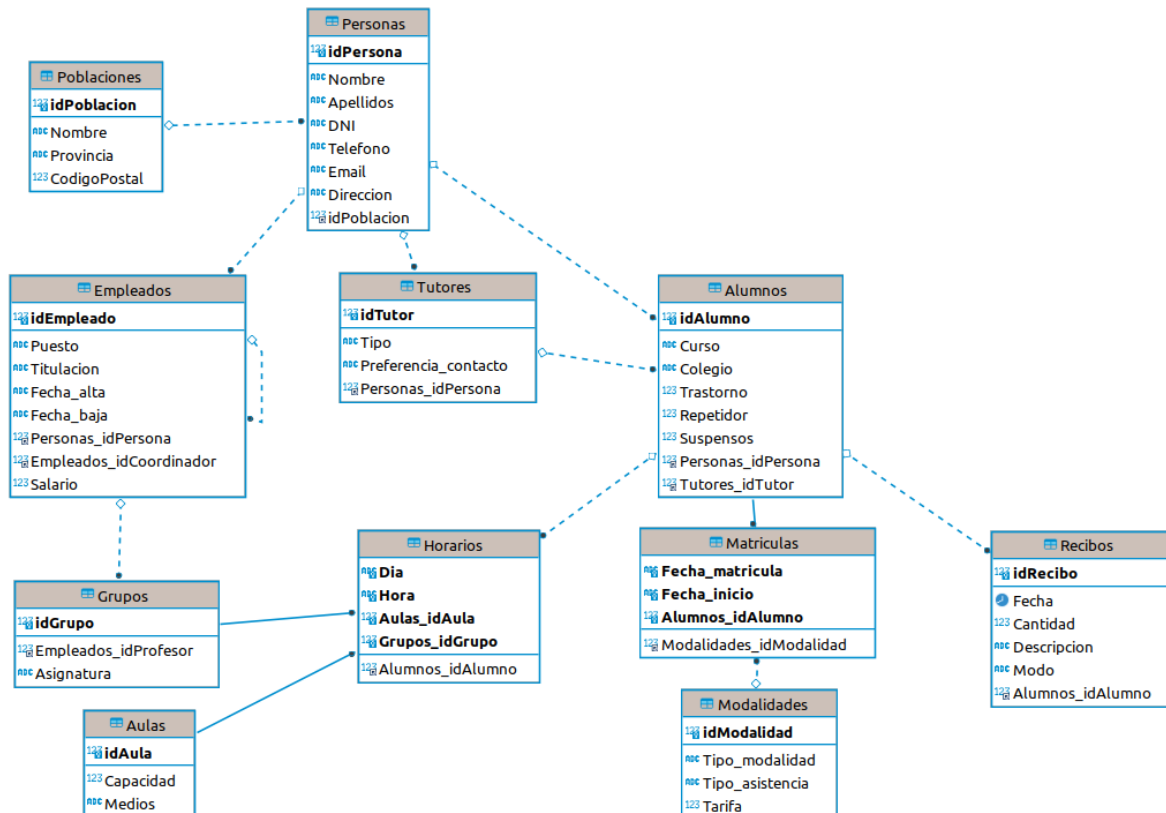
## Creación de la base de datos

Dado el esquema que teníamos previamente planteado, pasamos a poner en marcha realmente el proyecto y, en este paso, se analiza nuevamente cada una de las tablas necesarias para la recopilación de datos y la interrelación entre ellos. En esta fase, hemos añadido una tabla Grupo, pues nos permite relacionar de una forma más óptima la distribución de Alumnos, Profesores y Horarios, de manera que se pueda extraer una organización diaria de los grupos para el centro.

En este punto, hemos utilizado la herramienta DBeaver para la generación de las tablas que vemos a continuación, las cuales se crean automáticamente al subir los datos, proceso que se explica en el siguiente apartado. Para hacer las modificaciones de datos y crear las claves primarias y foráneas, hemos alterado las tablas y hemos añadido las restricciones necesarias. Ejemplo:

```
ALTER TABLE `academia-atentos-db`.Horarios ADD CONSTRAINT Horarios_PK PRIMARY KEY (Dia,Hora,Aulas_idAula,Grupos_idGrupo);
```

```
ALTER TABLE `academia-atentos-db`.Matriculas ADD CONSTRAINT Matriculas_PK PRIMARY KEY (Fecha_matricula,Fecha_inicio,Alumnos_idAlumno);
```







▼ academia-atentos-db	
▼ Tables	
> Alumnos	80K
> Aulas	64K
> Empleados	80K
> Grupos	64K
> Horarios	224K
> Matriculas	288K
> Modalidades	128K
> Personas	160K
> Poblaciones	64K
> Recibos	304K
> Tutores	32K

En la imagen anterior, se muestran todas las tablas creadas, y en el diagrama se puede observar en negritas las claves primarias que hemos asignados y con una flechita y las líneas discontinuas, el punto de relación. En el gráfico se puede apreciar que la tabla más importante es la de Alumnos, pues es el foco de las relaciones.

# Carga masiva de datos

Como el proceso lo hemos hecho cargando las tablas directamente a partir de los archivos csv, es importante explicar cómo se ha realizado la carga de datos masiva.

Para la generación de los datos nos hemos servido principalmente de dos herramientas: un generador automático de datos, Mockaroo (<https://www.mockaroo.com/>) y las hojas de cálculo de Google Drive.

La primera herramienta, Mockaroo, consiste en una plataforma que nos permite configurar el tipo de datos que podemos generar, siendo bastante configurable con el uso de expresiones regulares en los campos introducidos. Cuando hemos seleccionado todos los campos que deseamos y el tipo de dato que contendrán, descargamos el documento en formato csv con los 1000 registros que genera de forma automática en su versión gratuita.

Como algunos de los datos necesitan unas características más específicas, y los campos de las claves foráneas tenían que corresponder con el elemento al que se refieren, hemos tenido que hacer un proceso más “manual” con el uso de las hojas de cálculo, pero que con la propia herramienta de arrastre y copia, hemos conseguido resultados muy rápidos.

En los ejemplos que se muestran a continuación, se puede apreciar un ejemplo de los datos creados en hojas de cálculo, las extensiones de los archivos generados y el proceso de importación de dichos archivos al programa DBeaver y nuestra base de datos.

Datos\_AcademiaAtentos

☆

🔖

☁

Archivo  Editar  Ver  Insertar  Formato  Datos  Herramientas  Extensiones  Ayuda  Última modificación hace 7 días

↶ ↷ 🖨 📄

100% ▾ € % .0 .00 123 ▾

Arial ▾

8 ▾

**B** *I* S **A** 🔍 📊 📈 📉 📉 📉

☰ ▾ ⬇ ▾ ⚡ ▾ 🔗 🔗 📄 ▾

A1 ▾  $\sum$

idPoblacion

	A	B	C	D	E	F	G	H	I	J
1	idPoblacion	Nombre	Provincia	CodigoPostal						
2	1	Aguadulce	Sevilla	41001						
3	2	Alanis	Sevilla	41002						
4	3	Albaida del Aljarafe	Sevilla	41003						
5	4	Alcalá de Guadaira	Sevilla	41004						
6	5	Alcalá del Río	Sevilla	41005						
7	6	Alcolea del Río	Sevilla	41006						
8	7	La Algaba	Sevilla	41007						
9	8	Algámitas	Sevilla	41008						
10	9	Almadén de la Plata	Sevilla	41009						
11	10	Almensilla	Sevilla	41010						
12	11	Arahal	Sevilla	41011						
13	12	Aznalcázar	Sevilla	41012						
14	13	Aznalcóllar	Sevilla	41013						
15	14	Badolatosa	Sevilla	41014						
16	15	Benacazón	Sevilla	41015						
17	16	Bollullos de la Mitad	Sevilla	41016						
18	17	Bormujos	Sevilla	41017						
19	18	Brenes	Sevilla	41018						
20	19	Burguillos	Sevilla	41019						
21	20	Las Cabezas de San Juan	Sevilla	41020						
22	21	Camas	Sevilla	41021						
23	22	La Campana	Sevilla	41022						
24	23	Cantillana	Sevilla	41023						
25	24	Cañada Rosal	Sevilla	41024						
26	25	Carmona	Sevilla	41025						
27	26	Carcasena	Sevilla	41026						
28	27	Carcasena	Sevilla	41027						
29	28	Carcasena	Sevilla	41028						
30	29	Carcasena	Sevilla	41029						
31	30	Carcasena	Sevilla	41030						
32	31	Carcasena	Sevilla	41031						
33	32	Carcasena	Sevilla	41032						
34	33	Carcasena	Sevilla	41033						
35	34	Carcasena	Sevilla	41034						
36	35	Carcasena	Sevilla	41035						
37	36	Carcasena	Sevilla	41036						
38	37	Carcasena	Sevilla	41037						
39	38	Carcasena	Sevilla	41038						
40	39	Carcasena	Sevilla	41039						
41	40	Carcasena	Sevilla	41040						
42	41	Carcasena	Sevilla	41041						
43	42	Carcasena	Sevilla	41042						
44	43	Carcasena	Sevilla	41043						
45	44	Carcasena	Sevilla	41044						
46	45	Carcasena	Sevilla	41045						
47	46	Carcasena	Sevilla	41046						
48	47	Carcasena	Sevilla	41047						
49	48	Carcasena	Sevilla	41048						
50	49	Carcasena	Sevilla	41049						
51	50	Carcasena	Sevilla	41050						
52	51	Carcasena	Sevilla	41051						
53	52	Carcasena	Sevilla	41052						
54	53	Carcasena	Sevilla	41053						
55	54	Carcasena	Sevilla	41054						
56	55	Carcasena	Sevilla	41055						
57	56	Carcasena	Sevilla	41056						
58	57	Carcasena	Sevilla	41057						
59	58	Carcasena	Sevilla	41058						
60	59	Carcasena	Sevilla	41059						
61	60	Carcasena	Sevilla	41060						
62	61	Carcasena	Sevilla	41061						
63	62	Carcasena	Sevilla	41062						
64	63	Carcasena	Sevilla	41063						
65	64	Carcasena	Sevilla	41064						
66	65	Carcasena	Sevilla	41065						
67	66	Carcasena	Sevilla	41066						
68	67	Carcasena	Sevilla	41067						
69	68	Carcasena	Sevilla	41068						
70	69	Carcasena	Sevilla	41069						
71	70	Carcasena	Sevilla	41070						
72	71	Carcasena	Sevilla	41071						
73	72	Carcasena	Sevilla	41072						
74	73	Carcasena	Sevilla	41073						
75	74	Carcasena	Sevilla	41074						
76	75	Carcasena	Sevilla	41075						
77	76	Carcasena	Sevilla	41076						
78	77	Carcasena	Sevilla	41077						
79	78	Carcasena	Sevilla	41078						
80	79	Carcasena	Sevilla	41079						
81	80	Carcasena	Sevilla	41080						
82	81	Carcasena	Sevilla	41081						
83	82	Carcasena	Sevilla	41082						
84	83	Carcasena	Sevilla	41083						
85	84	Carcasena	Sevilla	41084						
86	85	Carcasena	Sevilla	41085						
87	86	Carcasena	Sevilla	41086						
88	87	Carcasena	Sevilla	41087						
89	88	Carcasena	Sevilla	41088						
90	89	Carcasena	Sevilla	41089						
91	90	Carcasena	Sevilla	41090						
92	91	Carcasena	Sevilla	41091						
93	92	Carcasena	Sevilla	41092						
94	93	Carcasena	Sevilla	41093						
95	94	Carcasena	Sevilla	41094						
96	95	Carcasena	Sevilla	41095						
97	96	Carcasena	Sevilla	41096						
98	97	Carcasena	Sevilla	41097						
99	98	Carcasena	Sevilla	41098						
100	99	Carcasena	Sevilla	41099						
101	100	Carcasena	Sevilla	41100						

+

☰

Poblaciones ▾

Personas ▾

Empleados ▾

Tutores ▾

Alumnos ▾





Recibos ▾

Matricula ▾

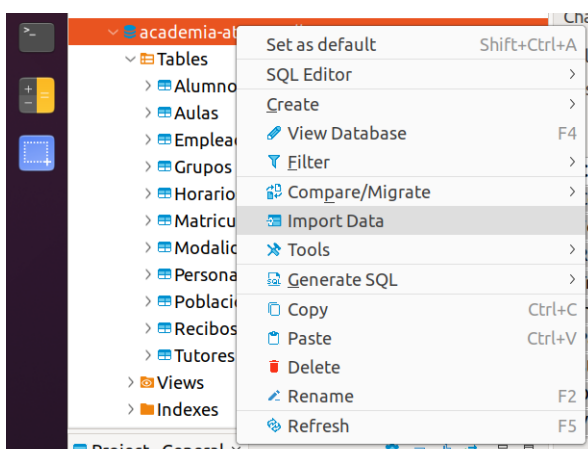
Horarios ▾

Grupos ▾

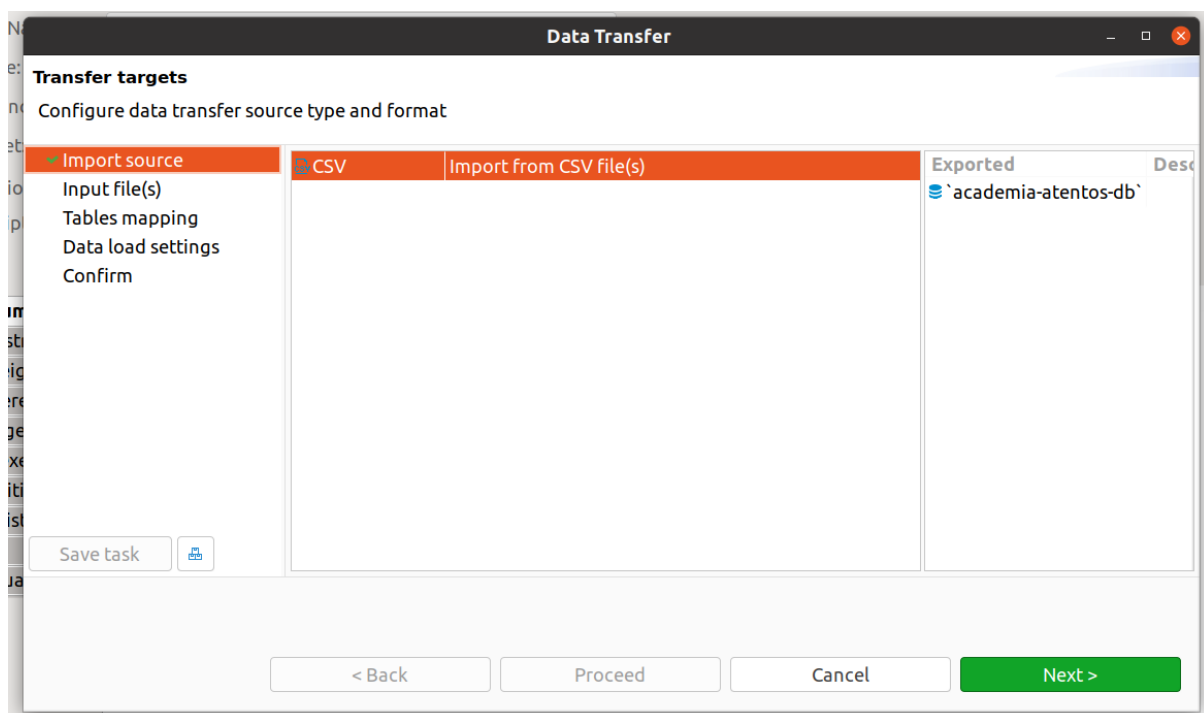
Aquí se puede apreciar un ejemplo de los datos generados en las hojas de cálculo.

	Datos_AcademiaAtentos - Recibos.csv	175,1 kB
	Datos_AcademiaAtentos - Tutores.csv	8,7 kB
	Datos_AcademiaAtentos - Alumnos.csv	16,0 kB
	Datos_AcademiaAtentos - Empleados.csv	15,6 kB

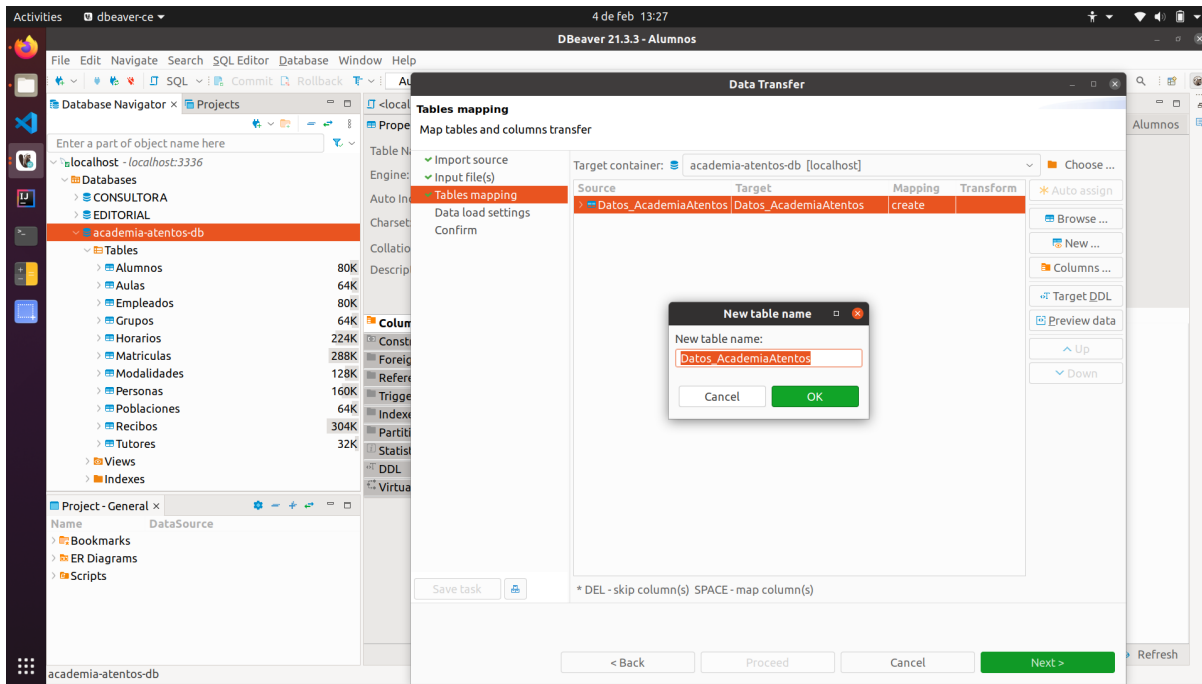
Archivos generados con extensión .csv



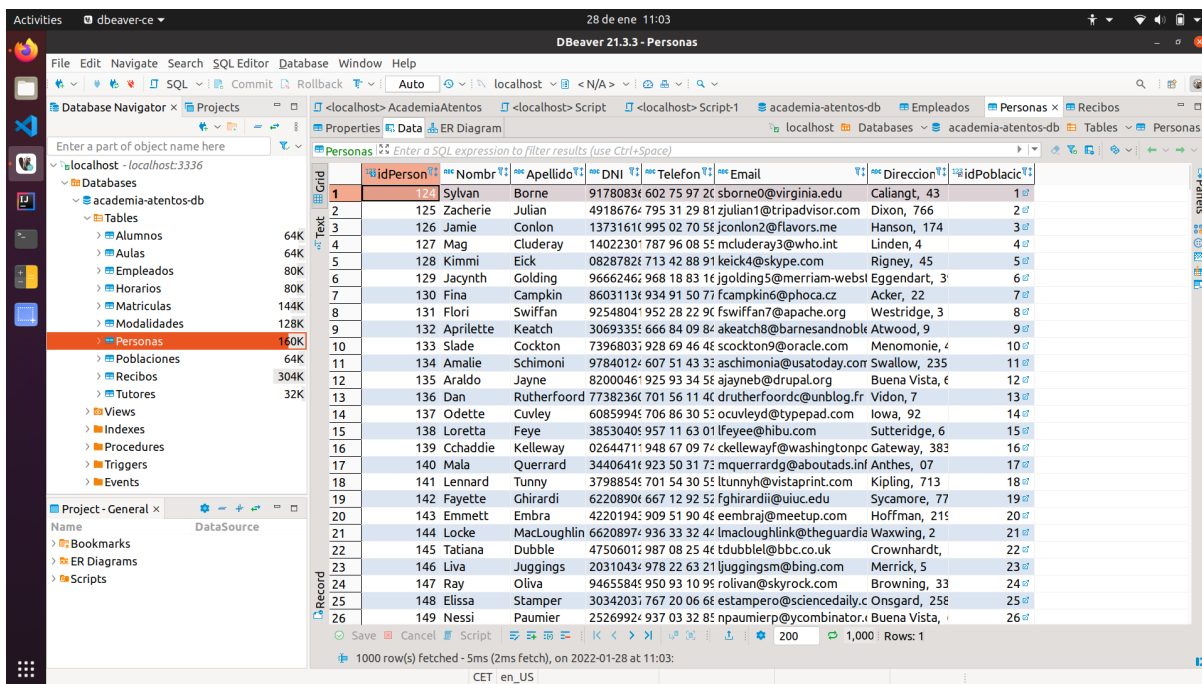
Importación de los archivos .csv al programa DBeaver dentro de nuestra base de datos.



Se indica que se trata de un archivo csv.



Le asignamos el nombre que debe aparecer en nuestra tabla. Cogera por defecto el nombre del archivo.

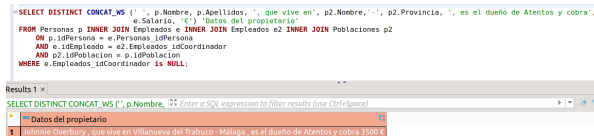


Observamos cómo se han incorporado los datos los datos a nuestra tabla.

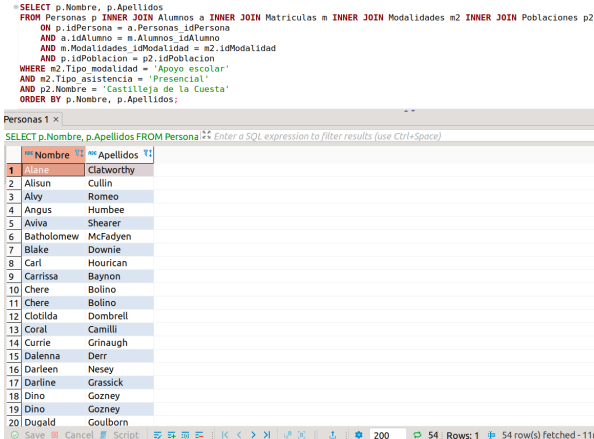
## Consultas - Queries

Hemos realizado 6 consultas donde se puede apreciar la comunicación entre las tablas y el tipo de información interesante para la empresa.

1. Consulta: El empleado que sea coordinador de todos, porque es el dueño de la academia, debe aparecer toda la información en una sola columna, donde se recoja el nombre completo, la ciudad y provincia donde vive y cuánto cobra.

Consulta	Captura
<pre>SELECT DISTINCT CONCAT_WS(' ', p.Nombre, p.Apellidos, ', que vive en', p2.Nombre, '-', p2.Provincia, ', es el dueño de Atentos y cobra', e.Salario, '€') 'Datos del propietario' FROM Personas p INNER JOIN Empleados e INNER JOIN Empleados e2 INNER JOIN Poblaciones p2 ON p.idPersona = e.Personas_idPersona AND e.idEmpleado = e2.Empleados_idCoordinador AND p2.idPoblacion = p.idPoblacion WHERE e.Empleados_idCoordinador is NULL;</pre>	

2. Consulta: Los alumnos de Castilleja de la Cuesta matriculados en Apoyo Escolar que asisten de manera presencial

Consulta	Captura																																										
<pre>SELECT p.Nombre, p.Apellidos FROM Personas p INNER JOIN Alumnos a INNER JOIN Matriculas m INNER JOIN Modalidades m2 INNER JOIN Poblaciones p2 ON p.idPersona = a.Personas_idPersona AND a.idAlumno = m.Alumnos_idAlumno AND m.Modalidades_idModalidad = m2.idModalidad AND p.idPoblacion = p2.idPoblacion WHERE m2.Tipo_modalidad = 'Apoyo escolar' AND m2.Tipo_asistencia = 'Presencial' AND p2.Nombre = 'Castilleja de la Cuesta' ORDER BY p.Nombre, p.Apellidos;</pre>	 <p>The screenshot shows the SQL query and its results. The query is:   <pre>SELECT p.Nombre, p.Apellidos FROM Personas p INNER JOIN Alumnos a INNER JOIN Matriculas m INNER JOIN Modalidades m2 INNER JOIN Poblaciones p2 ON p.idPersona = a.Personas_idPersona AND a.idAlumno = m.Alumnos_idAlumno AND m.Modalidades_idModalidad = m2.idModalidad AND p.idPoblacion = p2.idPoblacion WHERE m2.Tipo_modalidad = 'Apoyo escolar' AND m2.Tipo_asistencia = 'Presencial' AND p2.Nombre = 'Castilleja de la Cuesta' ORDER BY p.Nombre, p.Apellidos;</pre> The results table has 20 rows: <table border="1"> <thead> <tr> <th>Nombre</th> <th>Apellido</th> </tr> </thead> <tbody> <tr><td>Alane</td><td>Clatworthy</td></tr> <tr><td>Alsun</td><td>Cullin</td></tr> <tr><td>Alvy</td><td>Romeo</td></tr> <tr><td>Angus</td><td>Humbree</td></tr> <tr><td>Aviva</td><td>Shearer</td></tr> <tr><td>Batholomew</td><td>McFadyen</td></tr> <tr><td>Blake</td><td>Downie</td></tr> <tr><td>Carl</td><td>Hourican</td></tr> <tr><td>Carrissa</td><td>Baynon</td></tr> <tr><td>Chere</td><td>Bolino</td></tr> <tr><td>Chere</td><td>Bolino</td></tr> <tr><td>Clotilda</td><td>Dombrell</td></tr> <tr><td>Coral</td><td>Camilli</td></tr> <tr><td>Currie</td><td>Grinaugh</td></tr> <tr><td>Dalenna</td><td>Derr</td></tr> <tr><td>Darleen</td><td>Nesey</td></tr> <tr><td>Darline</td><td>Grassick</td></tr> <tr><td>Dino</td><td>Gozney</td></tr> <tr><td>Dino</td><td>Gozney</td></tr> <tr><td>Dugald</td><td>Goulborn</td></tr> </tbody> </table></p>	Nombre	Apellido	Alane	Clatworthy	Alsun	Cullin	Alvy	Romeo	Angus	Humbree	Aviva	Shearer	Batholomew	McFadyen	Blake	Downie	Carl	Hourican	Carrissa	Baynon	Chere	Bolino	Chere	Bolino	Clotilda	Dombrell	Coral	Camilli	Currie	Grinaugh	Dalenna	Derr	Darleen	Nesey	Darline	Grassick	Dino	Gozney	Dino	Gozney	Dugald	Goulborn
Nombre	Apellido																																										
Alane	Clatworthy																																										
Alsun	Cullin																																										
Alvy	Romeo																																										
Angus	Humbree																																										
Aviva	Shearer																																										
Batholomew	McFadyen																																										
Blake	Downie																																										
Carl	Hourican																																										
Carrissa	Baynon																																										
Chere	Bolino																																										
Chere	Bolino																																										
Clotilda	Dombrell																																										
Coral	Camilli																																										
Currie	Grinaugh																																										
Dalenna	Derr																																										
Darleen	Nesey																																										
Darline	Grassick																																										
Dino	Gozney																																										
Dino	Gozney																																										
Dugald	Goulborn																																										

Consulta: Los alumnos que tienen clase los lunes a las 20:00 en cualquier aula y de cualquier asignatura

## Consulta

```
SELECT CONCAT_WS(' ',p.Nombre, p.Apellidos) Alumno,
g.Asignatura, CONCAT('Aula: ', a2.idAula) Aula
FROM Personas p INNER JOIN Alumnos a INNER JOIN Horarios h
INNER JOIN Aulas a2 INNER JOIN Grupos g
ON p.idPersona = a.Personas_idPersona
AND h.Alumnos_idAlumno = a.idAlumno
AND h.Aulas_idAula = a2.idAula
AND h.Grupos_idGrupo = g.idGrupo
WHERE h.Dia = 'Lunes' AND h.Hora = '20:00'
ORDER BY a2.idAula;
```

## Captura

```
SELECT CONCAT_WS(' ',p.Nombre, p.Apellidos) Alumno, g.Asignatura, CONCAT('Aula: ', a2.idAula) Aula
FROM Personas p INNER JOIN Alumnos a INNER JOIN Horarios h INNER JOIN Aulas a2 INNER JOIN Grupos g
ON p.idPersona = a.Personas_idPersona
AND h.Alumnos_idAlumno = a.idAlumno
AND h.Aulas_idAula = a2.idAula
AND h.Grupos_idGrupo = g.idGrupo
WHERE h.Dia = 'Lunes' AND h.Hora = '20:00'
ORDER BY a2.idAula;
```

Grupos 1 x

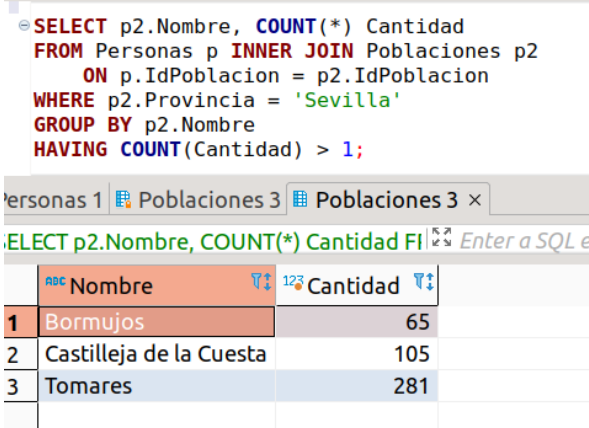
SELECT CONCAT\_WS(' ',p.Nombre, p.Apellido) AS Alumno, g.Asignatura AS Asignatura, a2.idAula AS Aula

	Alumno	Asignatura	Aula
6	Lothario Filan	Geología	Aula: 41
7	Lamar Sambals	Naturales	Aula: 42
8	Erhart Aldred	Música	Aula: 46
9	Clotilda Dombrell	Música	Aula: 52
10	Sutherland Pechell	Educación física	Aula: 53
11	Ernaline Paslow	Biología	Aula: 62
12	Tabbatha Abrahms	Latín	Aula: 64
13	Dyna Sandeson	Filosofía	Aula: 75
14	Alvin Albisser	Física	Aula: 83
15	Currie Grinaugh	Naturales	Aula: 96
16	Gerhard Wilden	Geología	Aula: 98
17	Alejandrina McLae	Educación audiovisual	Aula: 100
18	Reginald Etching	Música	Aula: 101
19	Jesselyn Burrass	Catalán	Aula: 103
20	Philly Baker	Catalán	Aula: 104
21	Clyde Pisco	Sociales	Aula: 105
22	Lishe Braidon	Música	Aula: 107
23	Rik Oller	Matemáticas aca	Aula: 113
24	Lyn Hansen	Francés	Aula: 118
25	Berte Sawl	Cultura clásica	Aula: 121

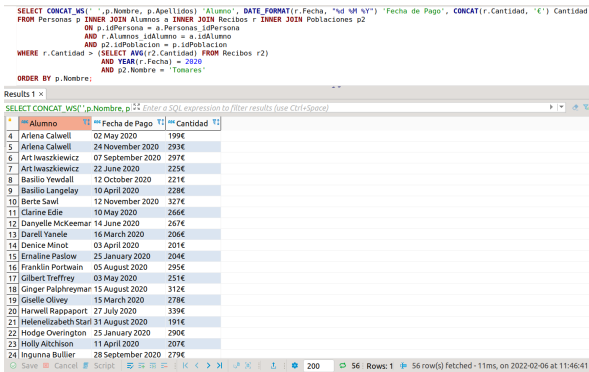
Save Cancel Script 38 Rows: 1

38 row(s) fetched - Sms, on 2022-02-06 at 20:48:29

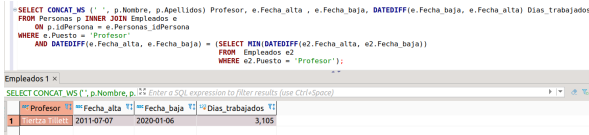
Consulta: Poblaciones de Sevilla con más de una persona relacionada con la academia

Consulta	Captura								
<pre>SELECT p2.Nombre, COUNT(*) Cantidad FROM Personas p INNER JOIN Poblaciones p2 ON p.IdPoblacion = p2.IdPoblacion WHERE p2.Provincia = 'Sevilla' GROUP BY p2.Nombre HAVING COUNT(Cantidad) &gt; 1;</pre>	 <pre>SELECT p2.Nombre, COUNT(*) Cantidad FROM Personas p INNER JOIN Poblaciones p2 ON p.IdPoblacion = p2.IdPoblacion WHERE p2.Provincia = 'Sevilla' GROUP BY p2.Nombre HAVING COUNT(Cantidad) &gt; 1;</pre> <table border="1"> <thead> <tr> <th>Nombre</th> <th>Cantidad</th> </tr> </thead> <tbody> <tr><td>1 Bormujos</td><td>65</td></tr> <tr><td>2 Castilleja de la Cuesta</td><td>105</td></tr> <tr><td>3 Tomares</td><td>281</td></tr> </tbody> </table>	Nombre	Cantidad	1 Bormujos	65	2 Castilleja de la Cuesta	105	3 Tomares	281
Nombre	Cantidad								
1 Bormujos	65								
2 Castilleja de la Cuesta	105								
3 Tomares	281								

Consulta: Nombre y apellidos de los alumnos en una misma columna que pagaron más que la media durante el año 2020 en Tomares. Se debe indicar la cantidad pagada y la fecha del recibo

Consulta	Captura
<pre>SELECT CONCAT_WS(' ', p.Nombre, p.Apellidos) 'Alumno', DATE_FORMAT(r.Fecha, "%d %M %Y") 'Fecha de Pago', CONCAT(r.Cantidad, '€') Cantidad FROM Personas p INNER JOIN Alumnos a INNER JOIN Recibos r INNER JOIN Poblaciones p2 ON p.idPersona = a.Personas_idPersona AND r.Alumnos_idAlumno = a.idAlumno AND p2.idPoblacion = p.idPoblacion WHERE r.Cantidad &gt; (SELECT AVG(r2.Cantidad) FROM Recibos r2) AND YEAR(r.Fecha) = 2020 AND p2.Nombre = 'Tomares'  ORDER BY p.Nombre;</pre>	 <pre>SELECT CONCAT_WS(' ', p.Nombre, p.Apellidos) 'Alumno', DATE_FORMAT(r.Fecha, "%d %M %Y") 'Fecha de Pago', CONCAT(r.Cantidad, '€') Cantidad FROM Personas p INNER JOIN Alumnos a INNER JOIN Recibos r INNER JOIN Poblaciones p2 ON p.idPersona = a.Personas_idPersona AND r.Alumnos_idAlumno = a.idAlumno AND p2.idPoblacion = p.idPoblacion WHERE r.Cantidad &gt; (SELECT AVG(r2.Cantidad) FROM Recibos r2) AND YEAR(r.Fecha) = 2020 AND p2.Nombre = 'Tomares'  ORDER BY p.Nombre;</pre>

Consulta: El profesor que haya estado en activo durante más tiempo, pero que ya no trabaje en la academia

Consulta	Captura
<pre>SELECT CONCAT_WS(' ', p.Nombre, p.Apellidos) Profesor, e.Fecha_alta, e.Fecha_baja, DATEDIFF(e.Fecha_baja, e.Fecha_alta) Dias_trabajados FROM Personas p INNER JOIN Empleados e ON p.idPersona = e.Personas_idPersona WHERE e.Puesto = 'Profesor' AND DATEDIFF(e.Fecha_alta, e.Fecha_baja) = (SELECT MIN(DATEDIFF(e2.Fecha_alta, e2.Fecha_baja)) FROM Empleados e2 WHERE e2.Puesto = 'Profesor');</pre>	 <pre>SELECT CONCAT_WS(' ', p.Nombre, p.Apellidos) Profesor, e.Fecha_alta, e.Fecha_baja, DATEDIFF(e.Fecha_baja, e.Fecha_alta) Dias_trabajados FROM Personas p INNER JOIN Empleados e ON p.idPersona = e.Personas_idPersona WHERE e.Puesto = 'Profesor' AND DATEDIFF(e.Fecha_alta, e.Fecha_baja) = (SELECT MIN(DATEDIFF(e2.Fecha_alta, e2.Fecha_baja)) FROM Empleados e2 WHERE e2.Puesto = 'Profesor');</pre>

## Vistas - View

A partir de las consultas que hemos creado en el apartado anterior, creamos las vistas. Se han creado 3 vistas, una para los diferentes tipos de consultas que se han planteado.


### Views

> Jefe

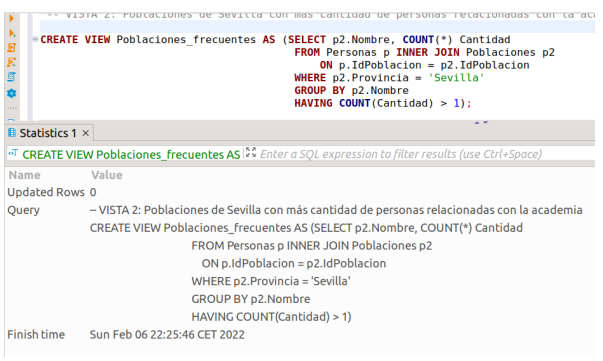
> Poblaciones\_frecuentes

> Profesor\_mas\_antiguo

1. Vista Jefe: el empleado que sea coordinador de todos, porque es el dueño de la academia

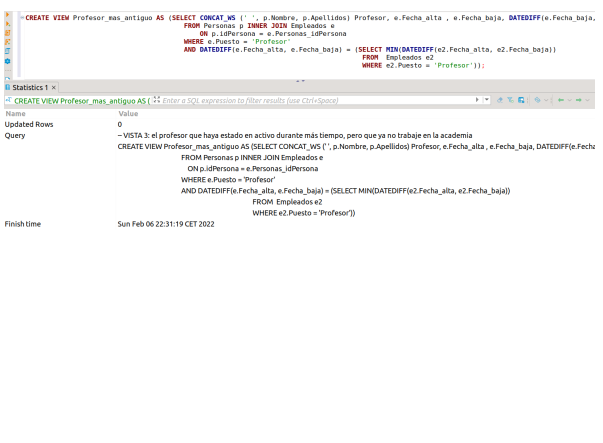
Consulta	Captura
<pre>CREATE VIEW Jefe AS (SELECT DISTINCT CONCAT_WS(' ', p.Nombre, p.Apellidos, ' que vive en', p2.Nombre, '-', p2.Provincia, ', es el dueño de Atentos y cobra', e.Salario, '€') 'Datos del propietario' FROM Personas p INNER JOIN Empleados e INNER JOIN Empleados e2 INNER JOIN Poblaciones p2 ON p.idPersona = e.Personas_idPersona AND e.idEmpleado = e2.Empleados_idCoordinador AND p2.idPoblacion = p.IdPoblacion WHERE e.Empleados_idCoordinador IS NULL);</pre>	

2. Vista Poblaciones frecuentes: poblaciones de Sevilla con más cantidad de personas relacionadas con la academia

Consulta	Captura
<pre>CREATE VIEW Poblaciones_frecuentes AS (SELECT p2.Nombre, COUNT(*) Cantidad FROM Personas p INNER JOIN Poblaciones p2 ON p.IdPoblacion = p2.IdPoblacion WHERE p2.Provincia = 'Sevilla' GROUP BY p2.Nombre HAVING COUNT(Cantidad) &gt; 1);</pre>	




3. Vista Profesor más antiguo: el profesor que haya estado en activo durante más tiempo, pero que ya no trabaje en la academia

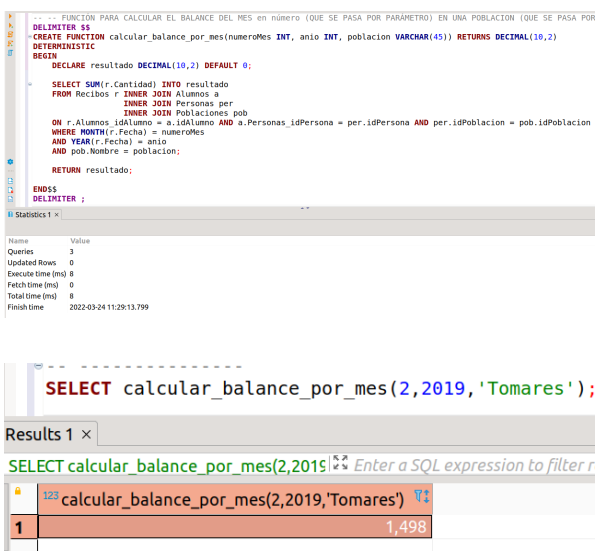
Consulta	Captura														
<pre>CREATE VIEW Profesor_mas_antiguo AS (SELECT CONCAT_WS (' ', p.Nombre, p.Apellidos) Profesor, e.Fecha_alta, e.Fecha_baja, DATEDIFF(e.Fecha_baja, e.Fecha_alta) Dias_trabajados FROM Personas p INNER JOIN Empleados e ON p.idPersona = e.Personas_idPersona WHERE e.Puesto = 'Profesor' AND DATEDIFF(e.Fecha_alta, e.Fecha_baja) = (SELECT MIN(DATEDIFF(e2.Fecha_alta, e2.Fecha_baja)) FROM Empleados e2 WHERE e2.Puesto = 'Profesor'));</pre>	 <p>Statistics 1 x</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Queries</td> <td>3</td> </tr> <tr> <td>Updated Rows</td> <td>0</td> </tr> <tr> <td>Execute time (ms)</td> <td>11</td> </tr> <tr> <td>Fetch time (ms)</td> <td>0</td> </tr> <tr> <td>Total time (ms)</td> <td>11</td> </tr> <tr> <td>Finish time</td> <td>Sun Feb 06 22:31:19 CET 2022</td> </tr> </tbody> </table>	Name	Value	Queries	3	Updated Rows	0	Execute time (ms)	11	Fetch time (ms)	0	Total time (ms)	11	Finish time	Sun Feb 06 22:31:19 CET 2022
Name	Value														
Queries	3														
Updated Rows	0														
Execute time (ms)	11														
Fetch time (ms)	0														
Total time (ms)	11														
Finish time	Sun Feb 06 22:31:19 CET 2022														

## Funciones - Functions

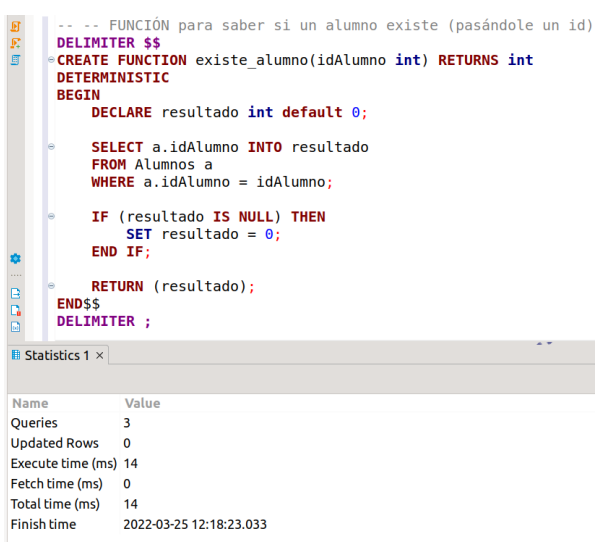
1. Función: Se crea una función para calcular la cantidad de personas relacionadas con la academia cuya dirección pertenezca a una localidad, pasándole como parámetro el nombre de dicha población.

Consulta	Captura																		
<pre>DELIMITER \$\$ CREATE FUNCTION calcular_personas_por_poblacion(nombrePoblacion VARCHAR(45)) RETURNS int DETERMINISTIC BEGIN DECLARE resultado int DEFAULT 0; SELECT COUNT(*) INTO resultado FROM Personas p INNER JOIN Poblaciones p2 ON p.IdPoblacion = p2.IdPoblacion WHERE p2.Nombre = nombrePoblacion GROUP BY p2.Nombre; RETURN resultado;  END\$\$  DELIMITER ;  SELECT calcular_personas_por_poblacion('Tomares');</pre>	 <p>Statistics 1 x</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Queries</td> <td>3</td> </tr> <tr> <td>Updated Rows</td> <td>0</td> </tr> <tr> <td>Execute time (ms)</td> <td>11</td> </tr> <tr> <td>Fetch time (ms)</td> <td>0</td> </tr> <tr> <td>Total time (ms)</td> <td>11</td> </tr> <tr> <td>Finish time</td> <td>2022-03-24 10:53:40.641</td> </tr> </tbody> </table> <p>Results 1 x</p> <table border="1"> <thead> <tr> <th>1</th> <th>281</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>281</td> </tr> </tbody> </table>	Name	Value	Queries	3	Updated Rows	0	Execute time (ms)	11	Fetch time (ms)	0	Total time (ms)	11	Finish time	2022-03-24 10:53:40.641	1	281	1	281
Name	Value																		
Queries	3																		
Updated Rows	0																		
Execute time (ms)	11																		
Fetch time (ms)	0																		
Total time (ms)	11																		
Finish time	2022-03-24 10:53:40.641																		
1	281																		
1	281																		

2. Función: Se crea un función para calcular el balance del mes (que se pasa su número por parámetro) con los usuarios de una población (que se pasa por parámetro).


Consulta	Captura				
<pre> DELIMITER \$\$ CREATE FUNCTION calcular_balance_por_mes(numeroMes INT, anio INT, poblacion VARCHAR(45)) RETURNS DECIMAL(10,2) DETERMINISTIC BEGIN DECLARE resultado DECIMAL(10,2) DEFAULT 0; SELECT SUM(r.Cantidad) INTO resultado FROM Recibos r INNER JOIN Alumnos a INNER JOIN Personas per INNER JOIN Poblaciones pob ON r.Alumnos_idAlumno = a.idAlumno AND a.Personas_idPersona = per.idPersona AND per.idPoblacion = pob.idPoblacion WHERE MONTH(r.Fecha) = numeroMes AND YEAR(r.Fecha) = anio AND pob.Nombre = poblacion; RETURN resultado; END\$\$ DELIMITER ;  SELECT calcular_balance_por_mes(2,2019,'Tomares');</pre>	 <pre> -- -- FUNCIÓN PARA CALCULAR EL BALANCE DEL MES EN NÚMERO (QUE SE PASA POR PARÁMETRO) EN UNA POBLACION (QUE SE PASA POR DELIMITER \$\$ CREATE FUNCTION calcular_balance_por_mes(numeroMes INT, anio INT, poblacion VARCHAR(45)) RETURNS DECIMAL(10,2) DETERMINISTIC BEGIN DECLARE resultado DECIMAL(10,2) DEFAULT 0; SELECT SUM(r.Cantidad) INTO resultado FROM Recibos r INNER JOIN Alumnos a INNER JOIN Personas per INNER JOIN Poblaciones pob ON r.Alumnos_idAlumno = a.idAlumno AND a.Personas_idPersona = per.idPersona AND per.idPoblacion = pob.idPoblacion WHERE MONTH(r.Fecha) = numeroMes AND YEAR(r.Fecha) = anio AND pob.Nombre = poblacion; RETURN resultado; END\$\$ DELIMITER ;  -- Statistics 1 x Name      Value Queries   3 Updated Rows 0 Execute time (ms) 8 Fetch time (ms) 0 Total time (ms) 8 Finish time 2022-03-24 11:28:13.799  SELECT calcular_balance_por_mes(2,2019,'Tomares');</pre> <p>Results 1 x</p> <p>SELECT calcular_balance_por_mes(2,2019,'Tomares')</p> <table border="1"> <thead> <tr> <th>1</th> <th>calcular_balance_por_mes(2,2019,'Tomares')</th> </tr> </thead> <tbody> <tr> <td></td> <td>1,498</td> </tr> </tbody> </table>	1	calcular_balance_por_mes(2,2019,'Tomares')		1,498
1	calcular_balance_por_mes(2,2019,'Tomares')				
	1,498				

3. Función: Se crea una función para comprobar si existe un alumno en los registros. Esta función será tremendamente útil en procedimientos, pues nos ayuda a capturar posibles excepciones.

Consulta	Captura
<pre> DELIMITER \$\$ CREATE FUNCTION existe_alumno(idAlumno int) RETURNS int DETERMINISTIC BEGIN DECLARE resultado int default 0;  SELECT a.idAlumno INTO resultado FROM Alumnos a WHERE a.idAlumno = idAlumno;  IF (resultado IS NULL) THEN SET resultado = 0; END IF;  RETURN (resultado);  END\$\$ DELIMITER ;  SELECT existe_alumno(5);</pre>	 <pre> -- -- FUNCIÓN para saber si un alumno existe (pasándole un id) DELIMITER \$\$ CREATE FUNCTION existe_alumno(idAlumno int) RETURNS int DETERMINISTIC BEGIN DECLARE resultado int default 0;  SELECT a.idAlumno INTO resultado FROM Alumnos a WHERE a.idAlumno = idAlumno;  IF (resultado IS NULL) THEN SET resultado = 0; END IF;  RETURN (resultado); END\$\$ DELIMITER ;  -- Statistics 1 x Name      Value Queries   3 Updated Rows 0 Execute time (ms) 14 Fetch time (ms) 0 Total time (ms) 14 Finish time 2022-03-25 12:18:23.033</pre>

## Procedimientos - Procedures

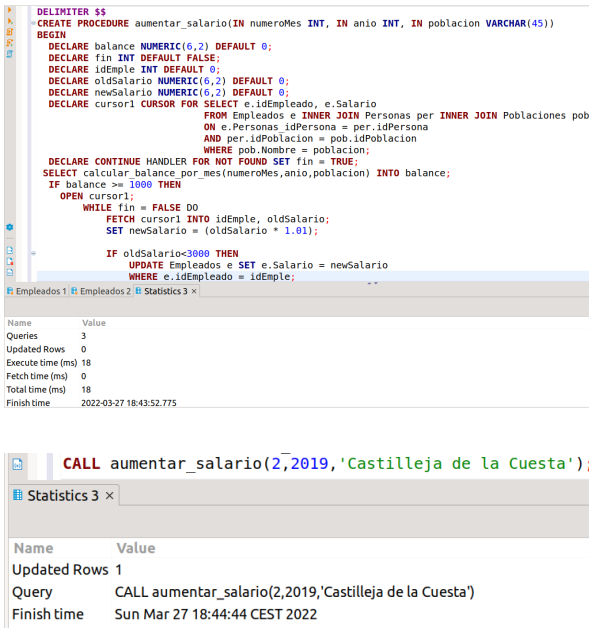
1. Procedimiento con función: Genera un pequeño informe sobre las inscripciones que haya podido realizar un alumno en el centro, de manera que se pueda evaluar el cambio de curso, las tarificaciones y las distintas fechas de inicio de curso, además controla la excepción en el caso que se introduzca un código de alumno que no exista.

Consulta	Captura
<pre> DELIMITER \$\$ CREATE PROCEDURE informe_alumno(IN idAlumno INT) BEGIN     IF(existe_alumno(idAlumno) &lt;&gt; 0) THEN         SELECT CONCAT('+++++++ INFORME ++++++', '\n', '\n', 'Alumno: ', p.Nombre, ' ', p.Apellidos, '\n', '-----', '\n', 'Colegio: ', a.Colegio, ' Curso: ', a.Curso, '\n', '-----', '\n', 'Modalidad: ', m2.Tipo_modalidad, ' Asistencia: ', m2.Tipo_asistencia, '\n', '-----', '\n', 'Fecha de inicio: ', m.Fecha_inicio, ' Tarifa: ', m2.Tarifa, '€', '\n', '+++++++')         FROM Personas p INNER JOIN Alumnos a             INNER JOIN Matriculas m             INNER JOIN Modalidades m2         ON p.idPersona = a.Personas_idPersona         AND a.idAlumno = m.Alumnos_idAlumno         AND m.Modalidades_idModalidad = m2.idModalidad         WHERE a.idAlumno = idAlumno         ORDER BY p.Apellidos, p.Nombre, m.Fecha_inicio;     ELSE         SIGNAL SQLSTATE '45001'         SET MESSAGE_TEXT = 'El alumno no consta en nuestros registros';     END IF; END \$\$ DELIMITER ;  CALL informe_alumno(5); </pre>	 <pre> DELIMITER \$\$ CREATE PROCEDURE informe_alumno(IN idAlumno INT) BEGIN     IF(existe_alumno(idAlumno) &lt;&gt; 0) THEN         SELECT CONCAT('+++++++ INFORME ++++++', '\n', '\n', 'Alumno: ', p.Nombre, ' ', p.Apellidos, '\n', '-----', '\n', 'Colegio: ', a.Colegio, ' Curso: ', a.Curso, '\n', '-----', '\n', 'Modalidad: ', m2.Tipo_modalidad, ' Asistencia: ', m2.Tipo_asistencia, '\n', '-----', '\n', 'Fecha de inicio: ', m.Fecha_inicio, ' Tarifa: ', m2.Tarifa, '€', '\n', '+++++++')         FROM Personas p INNER JOIN Alumnos a             INNER JOIN Matriculas m             INNER JOIN Modalidades m2         ON p.idPersona = a.Personas_idPersona         AND a.idAlumno = m.Alumnos_idAlumno         AND m.Modalidades_idModalidad = m2.idModalidad         WHERE a.idAlumno = idAlumno         ORDER BY p.Apellidos, p.Nombre, m.Fecha_inicio;     ELSE         SIGNAL SQLSTATE '45001'         SET MESSAGE_TEXT = 'El alumno no consta en nuestros registros';     END IF; END \$\$ DELIMITER ; </pre> <p><b>CALL informe_alumno(56586);</b></p> <p>Results 1 x</p> <p>CALL informe_alumno(56586) Enter a SQL expression to filter results (us</p> <p>SQL Error [1644] [45001]: El alumno no consta en nuestros registros</p> <pre> 1 "+++++++ INFORME +++++++ 2 3 Alumno: Morgen Aitken 4 ----- 5 Colegio: Murainagrass School Curso: ESO 2 6 ----- 7 Modalidad: Asignatura específica Asistencia: Mixto 8 ----- 9 Fecha de inicio: 2010-05-05 Tarifa: 174€ 10 ++++++++ 11 "+++++++ INFORME +++++++ 12 13 Alumno: Morgen Aitken 14 ----- 15 Colegio: Murainagrass School Curso: ESO 2 16 ----- 17 Modalidad: Apoyo escolar Asistencia: Online 18 ----- 19 Fecha de inicio: 2017-04-06 Tarifa: 138€ 20 ++++++++ </pre>

2. Procedimiento con función: Actualiza el curso de los alumnos solo si el alumno existe, además controla la excepción en el caso que se introduzca un código de alumno que no exista.

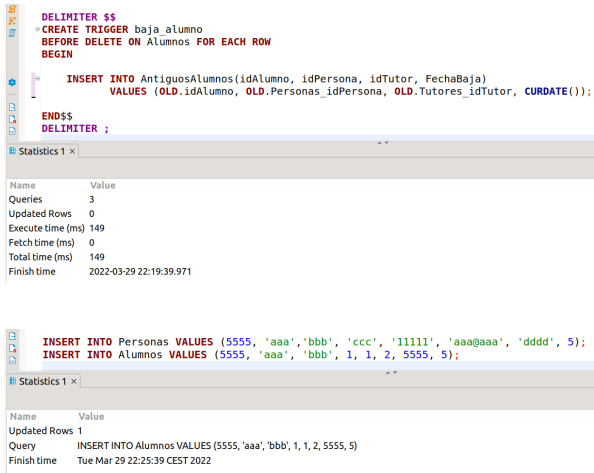
Consulta	Captura																						
<pre>DELIMITER \$\$ CREATE PROCEDURE actualizar_curso(idAlumno INT, curso varchar(13)) BEGIN   IF(existe_alumno(idAlumno) &lt;&gt; 0) THEN     UPDATE Alumnos a     SET a.Curso = curso     WHERE a.idAlumno = idAlumno;   ELSE     SIGNAL SQLSTATE '45001'     SET MESSAGE_TEXT = 'El alumno no consta en nuestros registros';   END IF; END\$\$ DELIMITER ;  CALL actualizar_curso(1, 'ESO 3');</pre>	<pre>-- PROCEDURE PARA ACTUALIZAR EL CURSO DE UN ALUMNO DELIMITER \$\$ CREATE PROCEDURE actualizar_curso(idAlumno INT, curso varchar(13)) BEGIN   IF(existe_alumno(idAlumno) &lt;&gt; 0) THEN     UPDATE Alumnos a     SET a.Curso = curso     WHERE a.idAlumno = idAlumno;   ELSE     SIGNAL SQLSTATE '45001'     SET MESSAGE_TEXT = 'El alumno no consta en nuestros registros';   END IF; END\$\$ DELIMITER ;</pre> <table><tr><th>Name</th><th>Value</th></tr><tr><td>Queries</td><td>3</td></tr><tr><td>Updated Rows</td><td>0</td></tr><tr><td>Execute time (ms)</td><td>10</td></tr><tr><td>Fetch time (ms)</td><td>0</td></tr><tr><td>Total time (ms)</td><td>10</td></tr><tr><td>Finish time</td><td>2022-03-27 16:23:09.203</td></tr></table> <pre>CALL actualizar_curso(1, 'ESO 3');</pre> <table><tr><th>Name</th><th>Value</th></tr><tr><td>Updated Rows</td><td>1</td></tr><tr><td>Query</td><td>CALL actualizar_curso(1, 'ESO 3')</td></tr><tr><td>Finish time</td><td>Sun Mar 27 16:23:57 CEST 2022</td></tr></table>	Name	Value	Queries	3	Updated Rows	0	Execute time (ms)	10	Fetch time (ms)	0	Total time (ms)	10	Finish time	2022-03-27 16:23:09.203	Name	Value	Updated Rows	1	Query	CALL actualizar_curso(1, 'ESO 3')	Finish time	Sun Mar 27 16:23:57 CEST 2022
Name	Value																						
Queries	3																						
Updated Rows	0																						
Execute time (ms)	10																						
Fetch time (ms)	0																						
Total time (ms)	10																						
Finish time	2022-03-27 16:23:09.203																						
Name	Value																						
Updated Rows	1																						
Query	CALL actualizar_curso(1, 'ESO 3')																						
Finish time	Sun Mar 27 16:23:57 CEST 2022																						

3. Procedimiento con cursor: evalúa si en una localidad en concreto, un mes ha tenido un balance superior a 1.000€, a todos los empleados que sean de ese lugar, se les aumentará el salario un 1% siempre y cuando su salario no sea superior a 3.000€.

Consulta	Captura																						
<pre> DELIMITER \$\$ CREATE PROCEDURE aumentar_salario(IN numeroMes INT, IN anio INT, IN poblacion VARCHAR(45)) BEGIN     DECLARE balance NUMERIC(6,2) DEFAULT 0;     DECLARE fin INT DEFAULT FALSE;     DECLARE idEmple INT DEFAULT 0;     DECLARE oldSalario NUMERIC(6,2) DEFAULT 0;     DECLARE newSalario NUMERIC(6,2) DEFAULT 0;     DECLARE cursor1 CURSOR FOR         SELECT e.idEmpleado, e.Salario         FROM Empleados e INNER JOIN Personas per         INNER JOIN Poblaciones pob         ON e.Personas_idPersona = per.idPersona         AND per.idPoblacion = pob.idPoblacion         WHERE pob.Nombre = poblacion;     DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;      SELECT     calcular_balance_por_mes(numeroMes,anio,poblacion)     INTO balance;      IF balance &gt;= 1000 THEN         OPEN cursor1;         WHILE fin = FALSE DO             FETCH cursor1 INTO idEmple, oldSalario;             SET newSalario = (oldSalario * 1.01);             IF oldSalario &lt; 3000 THEN                 UPDATE Empleados e                 SET e.Salario = newSalario                 WHERE e.idEmpleado = idEmple;             ELSE                 SIGNAL SQLSTATE '45003'                 SET MESSAGE_TEXT = 'No se puede aumentar el salario';             END IF;         END WHILE;         CLOSE cursor1;     ELSE         SIGNAL SQLSTATE '45003'         SET MESSAGE_TEXT = 'No se puede aumentar el salario';     END IF; END \$\$ DELIMITER ;  CALL aumentar_salario(2,2019,'Castilleja de la Cuesta');</pre>	 <pre> DELIMITER \$\$ CREATE PROCEDURE aumentar_salario(IN numeroMes INT, IN anio INT, IN poblacion VARCHAR(45)) BEGIN     DECLARE balance NUMERIC(6,2) DEFAULT 0;     DECLARE fin INT DEFAULT FALSE;     DECLARE idEmple INT DEFAULT 0;     DECLARE oldSalario NUMERIC(6,2) DEFAULT 0;     DECLARE newSalario NUMERIC(6,2) DEFAULT 0;     DECLARE cursor1 CURSOR FOR SELECT e.idEmpleado, e.Salario     FROM Empleados e INNER JOIN Personas per INNER JOIN Poblaciones pob     ON e.Personas_idPersona = per.idPersona     AND per.idPoblacion = pob.idPoblacion     WHERE pob.Nombre = poblacion;     DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;     SELECT calcular_balance_por_mes(numeroMes,anio,poblacion) INTO balance;     IF balance &gt;= 1000 THEN         OPEN cursor1;         WHILE fin = FALSE DO             FETCH cursor1 INTO idEmple, oldSalario;             SET newSalario = (oldSalario * 1.01);             IF oldSalario &lt; 3000 THEN                 UPDATE Empleados e SET e.Salario = newSalario                 WHERE e.idEmpleado = idEmple;             ELSE                 SIGNAL SQLSTATE '45003'                 SET MESSAGE_TEXT = 'No se puede aumentar el salario';             END IF;         END WHILE;         CLOSE cursor1;     ELSE         SIGNAL SQLSTATE '45003'         SET MESSAGE_TEXT = 'No se puede aumentar el salario';     END IF; END \$\$ DELIMITER ;  CALL aumentar_salario(2,2019,'Castilleja de la Cuesta');</pre> <table border="1"> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Queries</td><td>3</td></tr> <tr> <td>Updated Rows</td><td>0</td></tr> <tr> <td>Execute time (ms)</td><td>18</td></tr> <tr> <td>Fetch time (ms)</td><td>0</td></tr> <tr> <td>Total time (ms)</td><td>18</td></tr> <tr> <td>Finish time</td><td>2022-03-27 18:43:52.775</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th>Name</th><th>Value</th></tr> </thead> <tbody> <tr> <td>Updated Rows</td><td>1</td></tr> <tr> <td>Query</td><td>CALL aumentar_salario(2,2019,'Castilleja de la Cuesta')</td></tr> <tr> <td>Finish time</td><td>Sun Mar 27 18:44:44 CEST 2022</td></tr> </tbody> </table>	Name	Value	Queries	3	Updated Rows	0	Execute time (ms)	18	Fetch time (ms)	0	Total time (ms)	18	Finish time	2022-03-27 18:43:52.775	Name	Value	Updated Rows	1	Query	CALL aumentar_salario(2,2019,'Castilleja de la Cuesta')	Finish time	Sun Mar 27 18:44:44 CEST 2022
Name	Value																						
Queries	3																						
Updated Rows	0																						
Execute time (ms)	18																						
Fetch time (ms)	0																						
Total time (ms)	18																						
Finish time	2022-03-27 18:43:52.775																						
Name	Value																						
Updated Rows	1																						
Query	CALL aumentar_salario(2,2019,'Castilleja de la Cuesta')																						
Finish time	Sun Mar 27 18:44:44 CEST 2022																						

# Disparadores - Triggers

1. Disparador: Se inserta en una tabla (AntiguosAlumnos) que funciona a modo de almacén, donde se guardan los datos fundamentales de los alumnos, cuando estos se eliminan de la tabla Alumnos.

Consulta	Captura
<pre>CREATE TABLE `academia-atentos-db`.AntiguosAlumnos (   idAlumno INTEGER NULL,   idPersona INTEGER NULL,   idTutor INTEGER NULL,   FechaBaja DATE NULL);  DELIMITER \$\$ CREATE TRIGGER baja_alumno BEFORE DELETE ON Alumnos FOR EACH ROW BEGIN   INSERT INTO AntiguosAlumnos(idAlumno, idPersona, idTutor,   FechaBaja)   VALUES (OLD.idAlumno, OLD.Personas_idPersona,   OLD.Tutores_idTutor, CURDATE()); END\$\$ DELIMITER ;  INSERT INTO Personas VALUES (5555, 'aaa', 'bbb', 'ccc', '11111', 'aaa@aaa', 'dddd', 5); INSERT INTO Alumnos VALUES (5555, 'aaa', 'bbb', 1, 1, 2, 5555, 5);  DELETE FROM Alumnos WHERE idAlumno = 5555;</pre>	 <p>The screenshot shows the execution of the SQL script. It includes the trigger creation, the insertion of data into the 'AntiguosAlumnos' table, and the deletion of a row from the 'Alumnos' table. The results show that the trigger successfully captured the data before deletion.</p>

2. Disparador: se dispara una inserción en una tabla de seguridad para el histórico de pagos, con los datos fundamentales del alumno y del pago. Añade una excepción si se añade una fecha posterior a la fecha actual.

Consulta

```

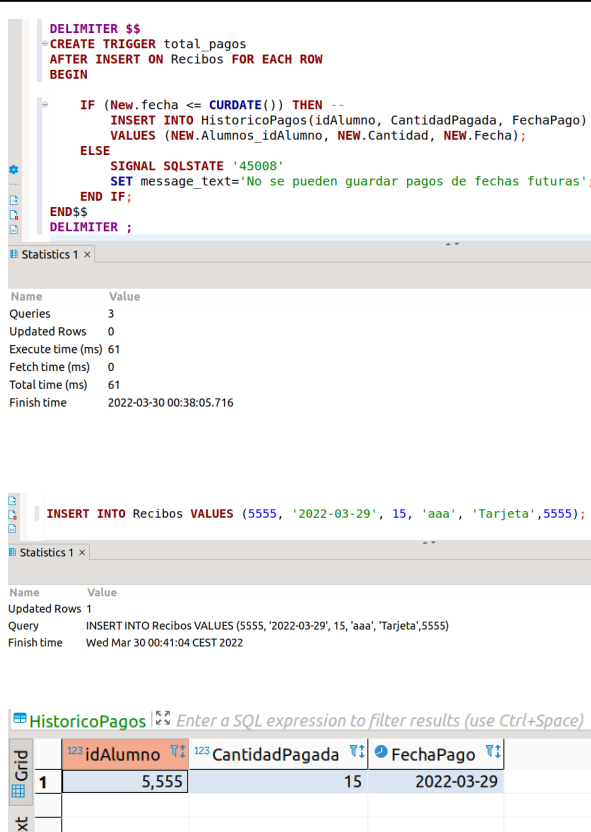
CREATE TABLE `academia-atentos-db`.HistoricoPagos (
    idAlumno INTEGER NULL,
    CantidadPagada DECIMAL(10,2) NULL,
    FechaPago DATE);

DELIMITER $$
CREATE TRIGGER total_pagos
AFTER INSERT ON Recibos FOR EACH ROW
BEGIN
    IF (New.fecha <= CURDATE()) THEN
        INSERT INTO HistoricoPagos(idAlumno, CantidadPagada,
            FechaPago)
            VALUES (NEW.Alumnos_idAlumno, NEW.Cantidad, NEW.Fecha);
    ELSE
        SIGNAL SQLSTATE '45008'
        SET message_text='No se pueden guardar pagos de fechas
futuras';
    END IF;
END$$
DELIMITER ;

INSERT INTO Recibos VALUES (5555, '2022-03-29', 15, 'aaa',
'Tarjeta',5555);

```

Captura



```

DELIMITER $$
CREATE TRIGGER total_pagos
AFTER INSERT ON Recibos FOR EACH ROW
BEGIN
    IF (New.fecha <= CURDATE()) THEN --
        INSERT INTO HistoricoPagos(idAlumno, CantidadPagada, FechaPago)
        VALUES (NEW.Alumnos_idAlumno, NEW.Cantidad, NEW.Fecha);
    ELSE
        SIGNAL SQLSTATE '45008'
        SET message_text='No se pueden guardar pagos de fechas futuras';
    END IF;
END$$
DELIMITER ;

INSERT INTO Recibos VALUES (5555, '2022-03-29', 15, 'aaa', 'Tarjeta',5555);

```

Name	Value
Queries	3
Updated Rows	0
Execute time (ms)	61
Fetch time (ms)	0
Total time (ms)	61
Finish time	2022-03-30 00:38:05.716

Name	Value
Updated Rows	1
Query	INSERT INTO Recibos VALUES (5555, '2022-03-29', 15, 'aaa', 'Tarjeta',5555)
Finish time	Wed Mar 30 00:41:04 CEST 2022

idAlumno	CantidadPagada	FechaPago
5,555	15	2022-03-29

## Conclusiones y valoración personal

El proyecto que en el presente trabajo se ha mostrado sobre la academia Atentos, es el resultado de largas horas de trabajo y de un proceso de aprendizaje aún por asimilar, lo que nos hace reflexionar sobre todo el proceso que hay detrás del diseño, la puesta en marcha y el mantenimiento que puede llegar a tener una base de datos.

Se ha pretendido elaborar un almacén de datos lo más cercano posible a la realidad del mundo de las aplicaciones y las necesidades del cliente. El objetivo que se había pautado desde primera hora, que era tener una gran cantidad de datos y gestionarlos, lo hemos conseguido, pero no sin antes haber sufrido grandes dificultades durante el proceso.

Este proyecto lo podríamos entender como una composición de distintas partes, en concreto, se pueden apreciar tres grandes fases. Una primera parte, en la hemos hecho un análisis de las necesidades de nuestro cliente, donde hemos diseñado un modelo de las entidades existentes y los tipos de datos que almacenaban cada una. En esta parte, la gran dificultad con la que nos hemos encontrado ha sido el decidir todos los elementos necesarios y entender de qué manera sería conveniente que estuviesen almacenados; por ejemplo, en el caso de relacionar los profesores con los alumnos, en un grupo, y en un horario.

La segunda parte, la comprendemos como la fase de la carga masiva de datos, donde hemos visto una pequeña muestra de lo que resultaría tratar con una gran cantidad de datos y su dificultad en cuanto a los formatos y a la referenciación a otras tablas. Las dificultades que hemos encontrado en esta parte, han sido, en primer lugar, el asegurarnos la configuración de los tipos de datos, concretamente en los de tipo decimal (para el salario, por ejemplo), las fechas (asegurarnos que estén en el formato correcto), y, por último, lo que ha resultado más complicado, asegurarnos de que los identificadores a otras tablas, realmente correspondan.

Finalmente, en la tercera fase, una vez definidos bien los pasos previos y entendiendo cómo se relacionan las tablas, cómo están organizados los datos y asegurándonos de que estos realmente sean los tipo correctos, ya entramos en el manejo y gestión de la información. En este punto, hemos hecho consultas que extraen datos que el cliente puede necesitar; hemos hecho consultas dinámicas, es decir, que trabajan con funciones, procedimientos y cursores, y hemos recogido algunas de las excepciones que pueden surgir al cumplir con las condiciones en cada una se requiere. En esta parte, las dificultades que hemos encontrado se pueden clasificar en dos, en primer lugar, en idear cuáles podrían ser las más útiles para el cliente; y, por otro lado, la elaboración de los disparadores, el cursor y las excepciones, pues requieren de un alto análisis de los elementos a tratar, comprender el funcionamiento de las estructuras repetitivas y condicionales, y saber qué necesitamos para conseguir un objetivo determinado.

Como podemos apreciar, la complejidad va aumentando a lo largo del proyecto, un buen planteamiento inicial es la clave del éxito y dejar abierta la posibilidad de añadir y modificar con facilidad también es importante, pues a medida que vamos trabajando nos vamos percatando de que necesitamos otros elementos que quizá no se hubiesen tenido en cuenta desde un principio.



Como broche final para este proyecto, cabe destacar que esto es una pequeña muestra, que tiene muchos puntos a mejorar, ajustar y llevarlo a una realidad más cercana, y podríamos implementar miles de funciones, procedimientos, cursores y disparadores más, que irán surgiendo con la gestión del día a día. Así pues, como mejoras podríamos añadir un formato de ticket con un procedimiento y un cursor, ampliar algunas tablas más y algunos campos, pues a lo largo del proyecto he pensado en otro tipo de datos que también podrían resultar interesantes de recoger para consultas, como por ejemplo, en Alumnos, indicar el momento de su baja, o si da clases grupales o individuales, pues modificaría la tarifa.

Este proyecto ha sido todo un reto personal, que ha implicado mucho trabajo y esfuerzo, donde la creatividad ha sido fundamental y una buena planificación muy necesaria y, por la poca experiencia que tengo, encontrarme con este primer impacto me ha servido para reflexionar sobre el impacto que pueden llegar a tener las decisiones que se tomen desde primera hora. Asimismo, y para concluir, me ha servido para darme cuenta de la magnitud que pueda llegar a tener la base de datos incluso en una pequeña empresa.