



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

ENTREGABLE 2

Alumnos:

Javier Alejandro Rivera Zavala - 311288876

Abraham Jimenez Reyes - 318230577

PROFESOR:

Victor Manuel Corza Vargas

AYUDANTES

Alexis Hernandez Castro

Diana Irma Canchola Hernandez

Gibran Aguilar Zuñiga

Rogelio Alcantar Arenas

ASIGNATURA:

Fundamentos de bases de datos

12 DE JUNIO DE 2023

## **1. Lista de requerimientos para la base de datos propuesta.**

La cadena "Ciencias Buster" ha identificado la oportunidad de aprovechar la demanda de películas en formato DVD y ha decidido abrir centros de renta. Como parte del proceso de contratación, se nos ha encargado desarrollar un ejemplo de bases de datos que serán utilizadas por los sistemas de la compañía. Estas bases de datos simuladas representan una versión simplificada de las bases de datos reales que se utilizarán en las computadoras para proporcionar servicios a los clientes. Es importante tener en cuenta que parte de la información almacenada en estas bases de datos se traslada a otra base de datos después de ser procesada por el sistema. La base de datos creada y almacenada será para los empleados de Ciencias Buster, tendremos información general de cada empleado e información personal.

La tabla "Departamentos" contiene información sobre los diferentes departamentos de la empresa, como su ID, nombre y descripción. Esto nos permitirá tener un registro de los departamentos existentes en la organización.

La tabla "Supervisor" almacena detalles sobre los supervisores en la empresa. Cada supervisor tiene un ID único, nombre, descripción y se encuentra asociado a un empleado en la tabla "Empleados". Esta relación nos permite establecer quién es el supervisor de cada empleado.

La tabla "Salarios" guarda información relacionada con los salarios de los empleados. Cada registro en esta tabla tiene un ID único, un ID de empleado asociado, el salario correspondiente, así como las fechas de inicio y término del salario. Esto nos permitirá llevar un registro histórico de los salarios de cada empleado.

La tabla "Direcciones" almacena los detalles de las direcciones de los empleados, incluyendo la calle, ciudad, estado y código postal. Cada dirección tiene un ID único y está asociada a un empleado en la tabla "Empleados". Esta relación nos permite tener un registro completo de las direcciones de los empleados.

La tabla "Dependientes" contiene información sobre los dependientes de los empleados, como sus nombres, parentesco y fecha de nacimiento. Cada dependiente tiene un ID único y está asociado a un empleado en la tabla "Empleados". Esta relación nos permite mantener un registro de los dependientes de cada empleado.

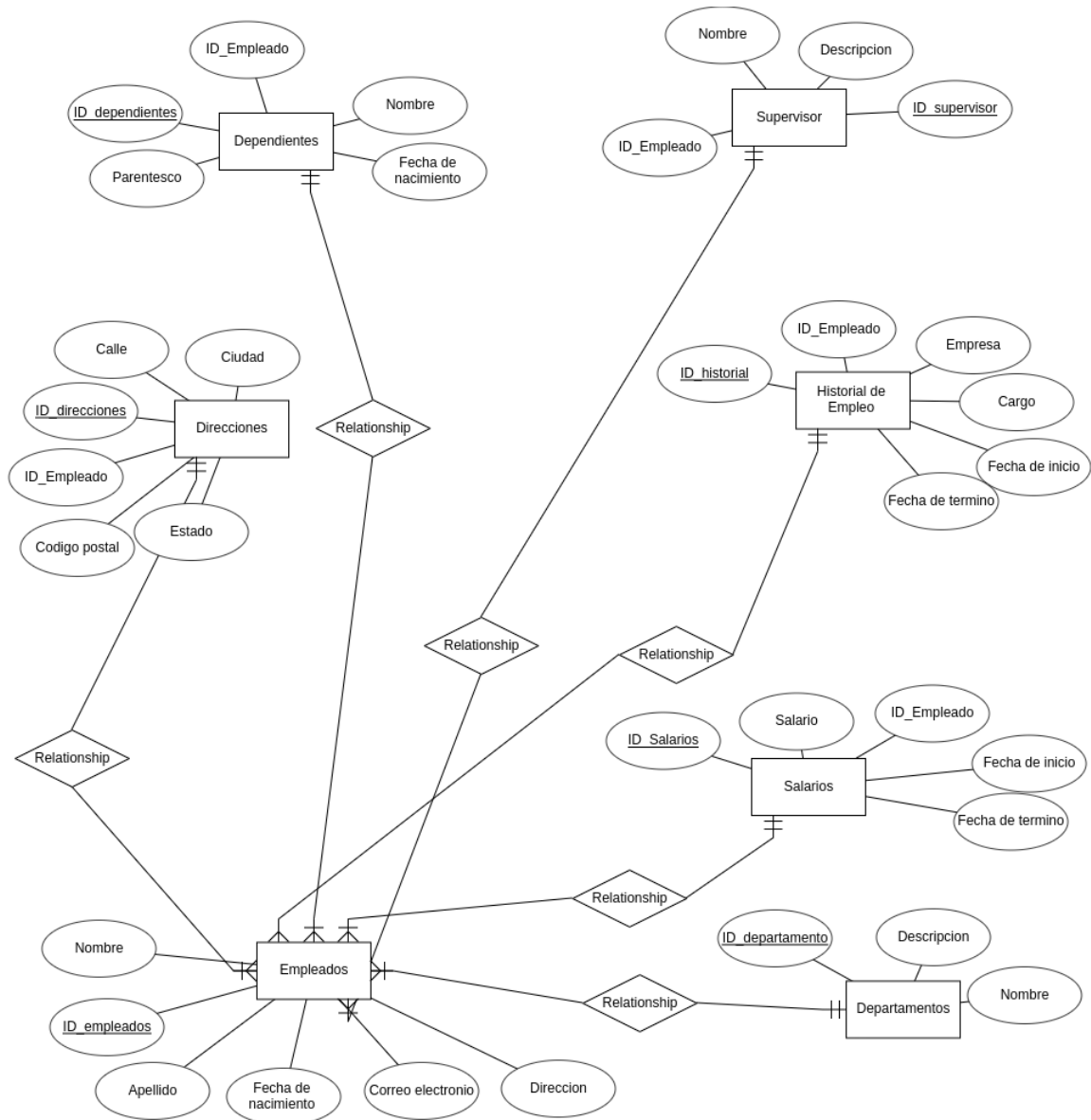
La tabla "Historial\_de\_Empleo" guarda los detalles del historial laboral de los empleados, incluyendo el nombre de la empresa en la que trabajaron, el cargo que ocupan y las fechas de inicio y término de su empleo. Cada registro en esta tabla

tiene un ID único y está asociado a un empleado en la tabla "Empleados". Esta relación nos permite rastrear el historial laboral de cada empleado.

Por último, la tabla "Empleados" actúa como la tabla principal que integra la información de los empleados y establece las relaciones con las otras tablas. Almacena datos como el ID del empleado, nombre, apellido, fecha de nacimiento, dirección, correo electrónico, así como los ID de departamento, salario, dirección, dependientes, historial y supervisor asociados. Las claves externas en esta tabla aseguran la integridad referencial y permiten acceder a la información relacionada en otras tablas.

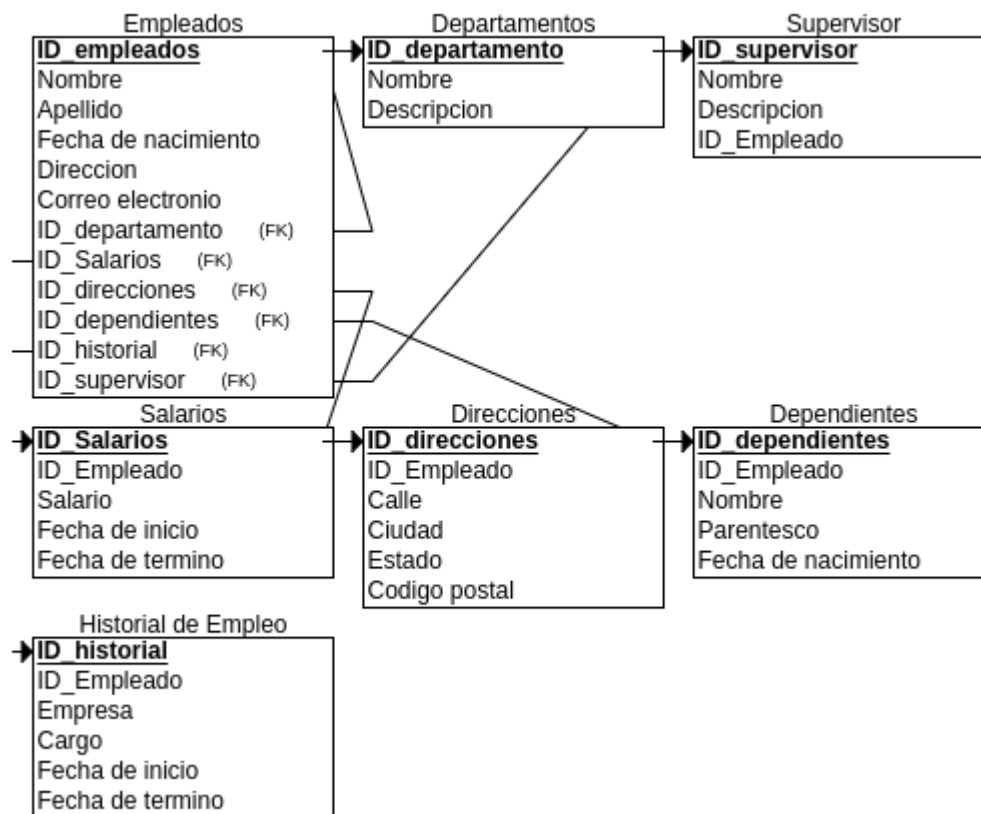
## **2. Modelo conceptual de la base de datos, respetando la nomenclatura de Peter Chen.**

Modelo conceptual de la base de datos



### 3. Modelo relacional, deben incluirse:

- Llaves primarias
- Llaves foráneas
- Utilizar el software de modelado de su preferencia.



En este modelo, se pueden identificar las siguientes relaciones:

- La tabla "Empleados" tiene una relación de uno a muchos con las tablas "Salarios", "Direcciones", "Dependientes", "Historial\_de\_Empleo" y "Supervisor", ya que un empleado puede tener múltiples registros asociados en cada una de estas tablas.
- La tabla "Departamentos" no tiene relaciones directas con otras tablas en el modelo actual. Sin embargo, podría establecerse una relación de uno a muchos con la tabla "Empleados" si se desea asignar a cada departamento varios empleados.
- La tabla "Supervisor" tiene una relación de uno a uno con la tabla "Empleados", ya que un supervisor está asociado a un empleado específico a través del campo "ID\_Empleado".

#### 4. Script completo y sin errores para la creación de todos los elementos que conforman el esquema de la base de datos.

- El Script debe estar diseñado para la versión 14 de Postgres.

- **Deben estar contempladas todas las llaves primarias, llaves candidatas y llaves foráneas; todas las llaves foráneas deben contar con un trigger de integridad referencial (SET NULL, CASCADE o SET DEFAULT).**

```
CREATE DOMAIN FechaInicio AS INTEGER CHECK (VALUE >= 19500101 AND
VALUE <= 21000101);
```

```
CREATE DOMAIN FechaTermino AS INTEGER CHECK (VALUE >= 19500201 AND
VALUE <= 21000101);
```

```
CREATE DOMAIN CorreoElectronico AS VARCHAR(50) CHECK (VALUE ~*
'^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$');
```

```
CREATE TABLE public.departamentos (
    id_departamento integer NOT NULL,
    nombre character varying(30) NOT NULL,
    descripcion character varying(30) NOT NULL,
    PRIMARY KEY (id_departamento)
);
```

```
CREATE TABLE public.dependientes (
    id_dependientes integer NOT NULL,
    id_empleado integer NOT NULL,
    nombre character varying(20) NOT NULL,
    parentesco character varying(10) NOT NULL,
    fecha_de_nacimiento integer,
    CONSTRAINT chk_nombre_parentesco CHECK (((nombre IS NULL) AND
(parentesco IS NULL)) OR ((nombre IS NOT NULL) AND (parentesco IS NOT
NULL))),
    PRIMARY KEY (id_dependientes)
);
```

```
CREATE TABLE public.direcciones (
    id_direcciones integer NOT NULL,
    id_empleado integer NOT NULL,
    calle character varying(50) NOT NULL,
    ciudad character varying(30) NOT NULL,
    estado character varying(30) NOT NULL,
    codigo_postal character varying(5) NOT NULL,
    CONSTRAINT chk_codigo_postal_length CHECK ((length((codigo_postal)::text)
<= 5)),
    PRIMARY KEY (id_direcciones)
);
```

```

CREATE TABLE public.historial_de_empleo (
    id_historial integer NOT NULL,
    id_empleado integer NOT NULL,
    empresa character varying(20) NOT NULL,
    cargo character varying(50) NOT NULL,
    fecha_de_inicio integer,
    fecha_de_termino integer,
    CONSTRAINT chk_empresa_cargo CHECK (((empresa IS NULL) AND (cargo IS
NULL)) OR ((empresa IS NOT NULL) AND (cargo IS NOT NULL))),
    CONSTRAINT chk_fecha_inicio CHECK ((fecha_de_inicio > 19500101)),
    CONSTRAINT chk_fecha_termino CHECK ((fecha_de_termino < 21000101)),
    PRIMARY KEY (id_historial)
);

```

```

CREATE TABLE public.salarios (
    id_salarios integer NOT NULL,
    id_empleado integer NOT NULL,
    salario integer,
    fecha_de_inicio public.FechaInicio NOT NULL,
    fecha_de_termino public.FechaTermino,
    CONSTRAINT chk_salario_positive CHECK ((salario >= 0)),
    PRIMARY KEY (id_salarios)
);

```

```

CREATE TABLE public.supervisor (
    id_supervisor integer NOT NULL,
    nombre character varying(20) NOT NULL,
    descripcion character varying(60) NOT NULL,
    id_empleado integer NOT NULL,
    PRIMARY KEY (id_supervisor)
);

```

```

CREATE TABLE public.empleados (
    id_empleados integer NOT NULL,
    nombre character varying(10) NOT NULL,
    apellido character varying(20) NOT NULL,
    fecha_de_nacimiento integer,
    direccion character varying(45) NOT NULL,
    correo_electronico public.correoelectronico NOT NULL,
    id_departamento integer NOT NULL,
    id_salarios integer NOT NULL,
    id_direcciones integer NOT NULL,
    id_dependientes integer NOT NULL,

```

```

    id_historial integer NOT NULL,
    id_supervisor integer NOT NULL,
    PRIMARY KEY (id_empleados),
    FOREIGN KEY (id_departamento) REFERENCES
public.departamentos(id_departamento) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (id_dependientes) REFERENCES
public.dependientes(id_dependientes) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (id_direcciones) REFERENCES
public.direcciones(id_direcciones) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (id_historial) REFERENCES
public.historial_de_empleo(id_historial) ON UPDATE CASCADE ON DELETE
CASCADE,
    FOREIGN KEY (id_salarios) REFERENCES public.salarios(id_salarios) ON
UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (id_supervisor) REFERENCES public.supervisor(id_supervisor)
ON UPDATE CASCADE ON DELETE CASCADE
);

```

**5. Script con las Instrucciones “Insert” que permitan poblar la base de datos (Este Script no se debe incluir en el reporte final, sólo en los entregables):**

- Se deben generar 100 registros para cada tabla.
- Si para el buen funcionamiento de la base de datos se requieren más de 100 registros o menos de 100 registros en una tabla, se debe explicar claramente la razón, sólo en este caso sí se debe incluir un apartado en el reporte final.

**6. Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.**

- Tablas involucradas en la restricción.
- FK de la tabla que referencia y PK de la tabla referenciada.
- Justificación del trigger de integridad referencial elegido.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.
- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

**A) Tablas Empleados y departamentos**

Restricción de integridad referencial: Empleados.ID\_departamento -> Departamentos.ID\_departamento



FK Tabla referencia: Empleados

Pk Tabla referenciada: Departamentos

Justificación del trigger de integridad referencial elegido, ON DELETE CASCADE: Cuando se elimina un departamento de la tabla Departamentos, se desea eliminar automáticamente todos los empleados asociadas a ese departamento en la tabla empleados.

Instrucción DELETE que evidencia el funcionamiento de la restricción:  
DELETE FROM Departamentos WHERE id\_departamento = 2;

Primero revisamos ambas tablas sin aplicar el DELETE

Actividades pgAdmin 4 10 de jun 23:03

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Processes jqdcxclm/jqdcxclm@Abraham jimenez\*

jqdcxclm/jqdcxclm@Abraham jimenez

Query Query History

```

1 SELECT *
2 FROM departamentos;

```

Scratch Pad x

Data Output Messages Notifications

	id_departamento [FK] integer	nombre character varying	descripcion character varying
1	1	Logística	Envíos
2	2	Logística	Envíos
3	3	Logística	Envíos
4	4	Logística	Envíos
5	5	Logística	Envíos
6	6	Logística	Envíos
7	7	Logística	Envíos

Total rows: 100 of 100 Query complete 00:00:00.891 Ln 2, Col 19

Actividades pgAdmin 4 8 de jun 21:58

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Processes Servers jqdcxclm/jqdcxclm@Abraham jimenez\*

jqdcxclm/jqdcxclm@Abraham jimenez

Query Query History

```

1
2 SELECT *
3 FROM empleados;
4
5
6
7

```

Scratch Pad x

Data Output Messages Notifications

	id_empleados [PK] integer	nombre character varying	apellido character varying	fecha_de_nacimiento numeric	direccion character varying	correo_electronico character varying	id_departamento integer	id_salarios integer	id_direc integer
1	1	Elihu	Castañeda	20011003	Avenida sur NO.20	eli.hu.castañeda@hotmail.com	1	1	
2	2	Alejandra	García	19950515	Calle 5 de Mayo No. 123	ale.garcia@gmail.com	2	2	
3	3	Juan	Hernández	19881230	Avenida Juárez No. 456	juan.hdez@yahoo.com	3	3	
4	4	Maria	López	19900725	Calle Reforma No. 789	maria.lopez@gmail.com	4	4	
5	5	Carlos	Martínez	19930218	Avenida Hidalgo No. 234	carlos.mtz@hotmail.com	5	5	
6	6	Laura	Rodríguez	19920910	Calle Morelos No. 567	laura.rodriguez@yahoo.com	6	6	

Total rows: 100 of 100 Query complete 00:00:00.746 Ln 3, Col 15

En ambas tablas esta el id\_departamento =2

Aplicamos el DELETE

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Processes jdcxclm/jdcxclm@Abraham jimenez\*

Query Query History

```

1 DELETE
2 FROM DEPARTAMENTOS
3 WHERE id_departamento = 2;
4
5 SELECT *
6 FROM empleados

```

Data Output Messages Notifications

	id_empleados [PK] integer	nombre character varying	apellido character varying	fecha_de_nacimiento numeric	direccion character varying	correo_electronico character varying	id_departamento integer	id_salarios integer	id_direc integer
1+	1	abraham	Jimenez	20010805	Calle cadete No.04	abraham.jmnz@gmail.com	1	1	1
2	3	Juan	Hernández	19881230	Avenida Juárez No. 456	juan.hdez@yahoo.com	3	3	3
3	4	Maria	López	19900725	Calle Reforma No. 789	maria.lopez@gmail.com	4	4	4
4	5	Carlos	Martínez	19930218	Avenida Hidalgo No. 234	carlos.mtz@hotmail.com	5	5	5
5	6	Laura	Rodríguez	19920910	Calle Morelos No. 567	laura.rodriguez@yahoo.com	6	6	6
6	7	Pedro	González	19931205	Calle Principal No. 789	pedro.gonzalez@gmail.com	7	7	7

Total rows: 97 of 97 Query complete 00:00:00.785 Changes staged: Added: 1 Ln 6, Col 15

pgAdmin 4

File Object Tools Help

Object Explorer

- FS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (7)
  - departamentos
    - Columns (3)
      - id\_departamento
      - nombre
      - descripcion
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - dependientes
  - direcciones
  - empleados
  - historial\_de\_empleo
  - salarios
  - supervisor
  - Trigger Functions
  - Types
  - Views

Query Query History

```

1 DELETE
2 FROM DEPARTAMENTOS
3 WHERE id_departamento = 2;
4
5 SELECT *
6 FROM departamentos

```

Data Output Messages Notifications

	id_departamento [PK] integer	nombre character varying	descripcion character varying
1	1	Logística	Envíos
2	3	Logística	Envíos
3	4	Logística	Envíos
4	5	Logística	Envíos
5	6	Logística	Envíos
6	7	Logística	Envíos
7	8	Logística	Envíos

Total rows: 99 of 99 Query complete 00:00:00.828 Ln 6, Col 19

Se eliminó el empleado con el id\_departamento = 2

## B) Tablas empleados y direcciones

Restricción de integridad referencial:

Empleados.id\_direcciones-> Direcciones.id\_direcciones

FK Tabla referenciadora: empleados

PK Tabla referenciada: direcciones

Justificación del trigger de integridad referencial elegido, ON DELETE CASCADE:

Cuando se elimina una direccion de la tabla direcciones se elimina al empleado con esa direccion.

Instrucción DELETE que evidencia el funcionamiento de la restricción:  
DELETE FROM Direcciones WHERE id\_direcciones = 10;

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 DELETE
2 FROM direcciones
3 WHERE id_direcciones = 10;
4
5 SELECT *
6 FROM direcciones
```

The 'Data Output' tab shows the result of the query, which is a table with 6 columns: id\_direcciones [PK] integer, id\_empleado integer, calle character varying, ciudad character varying, estado character varying, and codigo\_postal character varying. The table contains 12 rows of data.

id_direcciones [PK] integer	id_empleado integer	calle character varying	ciudad character varying	estado character varying	codigo_postal character varying
6	6	Calle del Bosque No. 987	Toluca	Estado de México	50090
7	7	Avenida Reforma No. 654	Mérida	Yucatán	97000
8	8	Calle del Río No. 321	Cancún	Quintana Roo	77500
9	9	Avenida de los Pinos No. 567	Acapulco	Guerrero	39670
10	11	Avenida del Sol No. 456	Hermosillo	Sonora	83000
11	12	Calle Juárez No. 654	Morelia	Michoacán	58090
12	13	Avenida de la Luna No. 321	Tijuana	Baja California	22000

Total rows: 99 of 99 Query complete 00:00:01.017 Ln 6, Col 17

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 DELETE
2 FROM direcciones
3 WHERE id_direcciones = 10;
4
5 SELECT *
6 FROM empleados
```

The 'Data Output' tab shows the result of the query, which is a table with 10 columns: id\_empleados [PK] integer, nombre character varying, apellido character varying, fecha\_de\_nacimiento numeric, direccion character varying, correo\_electronico character varying, id\_departamento integer, id\_salarios integer, and id\_direc integer. The table contains 11 rows of data.

id_empleados [PK] integer	nombre character varying	apellido character varying	fecha_de_nacimiento numeric	direccion character varying	correo_electronico character varying	id_departamento integer	id_salarios integer	id_direc integer
6	Ana	Sánchez	19970620	Avenida Libertad No. 890	ana.sanchez@hotmail.com	8	8	
7	Miguel	Torres	19910508	Calle 8 No. 234	miguel.torres@yahoo.com	9	9	
8	Javier	Vargas	19981203	Calle Juárez No. 890	javier.vargas@hotmail.com	11	11	
9	Valentina	Morales	19930128	Avenida Morelia No. 234	valentina.morales@yahoo.com	12	12	
10	Fernando	Lara	19901215	Calle 9 No. 345	fernando.lara@gmail.com	13	13	
11	Adriana	Pérez	19980928	Avenida México No. 678	adriana.perez@yahoo.com	14	14	

Total rows: 96 of 96 Query complete 00:00:01.074 Ln 6, Col 15

se elimino correctamente el empleado con el id\_direcciones = 10

### C) Tablas empleados y Salarios

Restricción de integridad referencial: Empleados.id\_salarios -> Salarios.id\_salarios

FK Tabla referenciadora: Empleados

PK Tabla referenciada: Salarios

Justificación del trigger de integridad referencial elegido, ON DELETE CASCADE:

Cuando se elimina un salario de la tabla salarios se elimina al empleado con ese id\_salario.

Instrucción DELETE que evidencia el funcionamiento de la restricción:

DELETE FROM Salarios WHERE id\_Salarios = 20

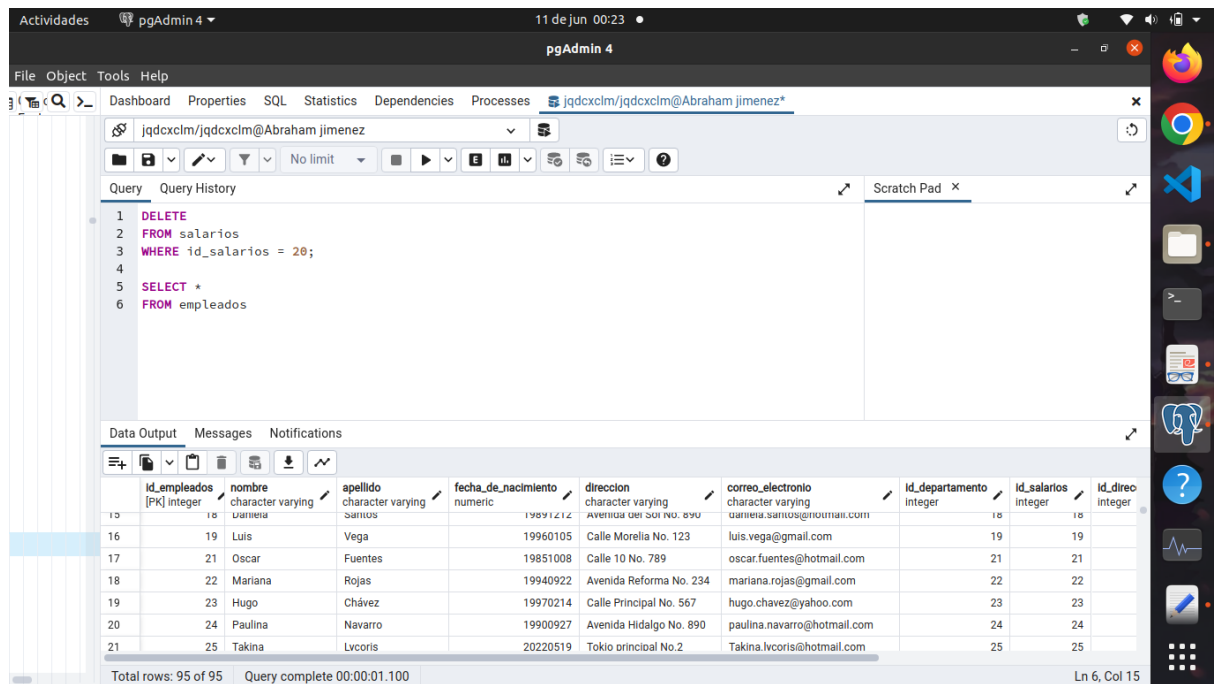
The screenshot shows the pgAdmin 4 interface. The top menu bar includes File, Object, Tools, and Help. The main window displays a SQL query in the 'Query' tab:

```
1 DELETE
2 FROM salarios
3 WHERE id_salarios = 20;
4
5 SELECT *
6 FROM salarios
```

Below the query, the 'Data Output' tab shows the results of the query. The table has 6 columns: id\_salarios [PK] integer, id\_empleado integer, salario integer, fecha\_de\_inicio integer, and fecha\_de\_termino integer. The data is as follows:

id_salarios [PK] integer	id_empleado integer	salario integer	fecha_de_inicio integer	fecha_de_termino integer
15	15	22000	20120120	20170630
16	16	19000	20100615	20111231
17	17	21000	20140101	[null]
18	18	24000	20130510	20171231
19	19	23000	20110105	20140331
20	21	15000	20040620	[null]
21	22	18000	20100115	20131231

The status bar at the bottom indicates 'Total rows: 99 of 99' and 'Query complete 00:00:00.928'. The right sidebar shows a 'Scratch Pad' tab.



se eliminó correctamente el empleado con id\_salarios = 20

#### D) Tablas empleados y dependientes

Restriccion de integridad referencial:

Empleados.id\_dependientes -> Dependientes.id\_dependientes

FK Tabla referenciadora: Empleados

PK Tabla referenciada: Dependientes

Justificación del trigger de integridad referencial elegido, ON DELETE CASCADE:

Cuando se elimina un salario de la tabla salarios se elimina al empleado con ese id\_salario.

Instrucción DELETE que evidencia el funcionamiento de la restricción:

DELETE FROM dependientes WHERE id\_dependientes = 35;

Actividades pgAdmin 4 11 de jun 00:24

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Processes jdcxclm/jdcxclm@Abraham jimenez\*

jdcxclm/jdcxclm@Abraham jimenez

Query Query History Scratch Pad x

```

1 DELETE
2 FROM dependientes
3 WHERE id_dependientes = 35;
4
5 SELECT *
6 FROM dependientes

```

Data Output Messages Notifications

	id_dependientes [PK] integer	id_empleado integer	nombre character varying	parentesco character varying	fecha_de_nacimiento integer
30	30	30	Liz	Sobrina	19890325
31	31	31	Lizabeth	Hija	19890324
32	32	32	Zuri	Hija	19890323
33	33	33	Nayelly	Hija	19890322
34	34	34	Cansas	Hermana	19890321
35	36	36	Alondra	Hija	19890323
36	37	37	Katia	Prima	19890324

Total rows: 99 of 99 Query complete 00:00:00.832 Ln 3, Col 22

Actividades pgAdmin 4 11 de jun 00:24

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Processes jdcxclm/jdcxclm@Abraham jimenez\*

jdcxclm/jdcxclm@Abraham jimenez

Query Query History Scratch Pad x

```

1 DELETE
2 FROM dependientes
3 WHERE id_dependientes = 35;
4
5 SELECT *
6 FROM empleados

```

Data Output Messages Notifications

	id	do	fecha_de_nacimiento	direccion	correo_electronico	id_departamento	id_salarios	id_direcciones	id_dependientes	id_historial
30	ja		19900317	Calle Juárez No. 789	lorena.ortega@hotmail.com	34	34	34	34	3
31			19871123	Calle Principal No. 567	lucia.vega@yahoo.com	36	36	36	36	3
32	os		19931202	Avenida del Sol No. 890	diego.santos@hotmail.com	37	37	37	37	3
33	les		19940817	Avenida Morelia No. 123	valentina.morales@gmail.com	38	38	38	38	3
34	ández		19920729	Calle 12 No. 456	javier.fernandez@yahoo.com	39	39	39	39	3
35	z		19900112	Avenida Libertad No. 789	camila.lopez@hotmail.com	30	40	40	40	4

Total rows: 94 of 94 Query complete 00:00:01.489 Ln 6, Col 15

se elimino correctamente el empleado con id\_dependientes = 35;

## 7. Evidencia del funcionamiento de al menos 3 restricciones check para “atributos” de varias tablas.

- Tabla elegida
- Atributo elegido
- Breve descripción de la restricción
- Instrucción para la creación de la restricción.

- Instrucción que permita evidenciar que la restricción esta funcionando.
- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

### A) Ejemplo 1

Tabla elegida: Salarios

Atributo elegido: id\_salarios

Breve descripción de la restricción:

Verificamos que el salario sea menor o igual a cero.

Instrucción para la creación de la restricción:

ALTER TABLE Salarios

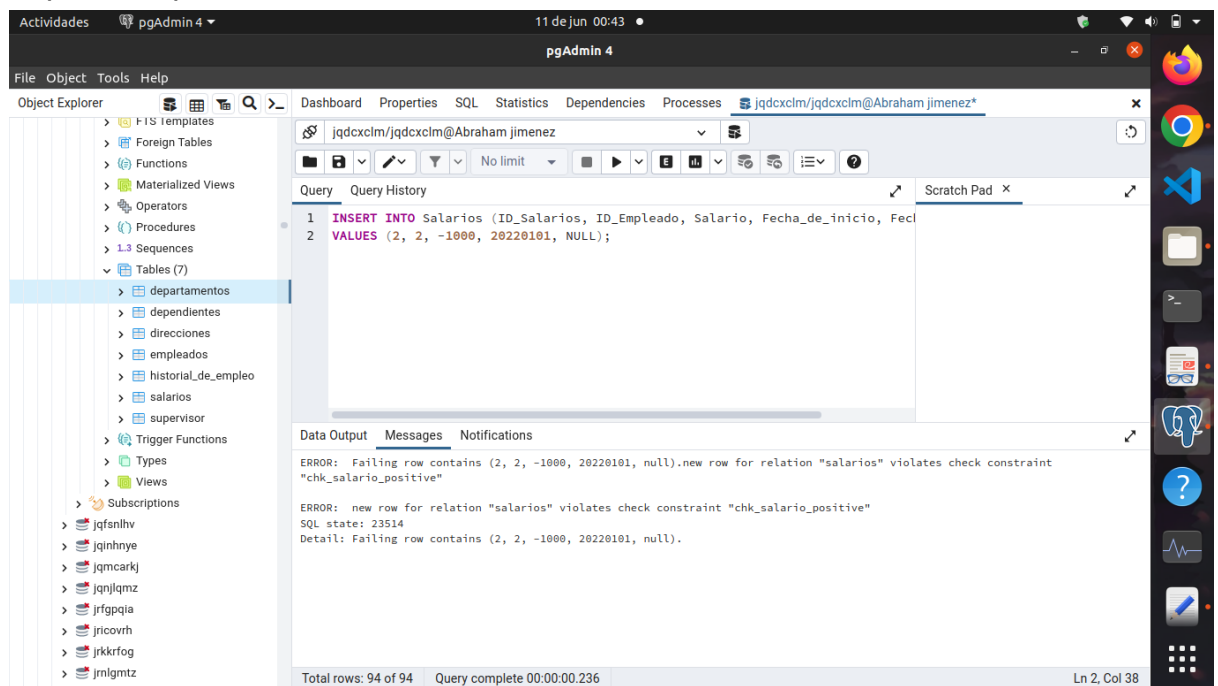
ADD CONSTRAINT CHK\_Salario\_Positive CHECK (Salario >= 0);

Instrucción que evidencia si la restricción esta funcionando:

INSERT INTO Salarios (ID\_Salarios, ID\_Empleado, Salario, Fecha\_de\_inicio, Fecha\_de\_termino)

VALUES (2, 2, -1000, 20220101, NULL);

Captura de pantalla:



Al ejecutar esta instrucción, se producirá un error debido a que el salario es menor que cero esto muestra que la restricción CHECK está funcionando correctamente.

### B) Ejemplo 2.



Tabla elegida: Historial\_de\_empleo

Atributo elegido: fecha\_de\_termino

Breve descripción de la restricción: Verificamos que la fecha de termino sea menor al año de la empresa.

Instrucción para la creación de la restricción:

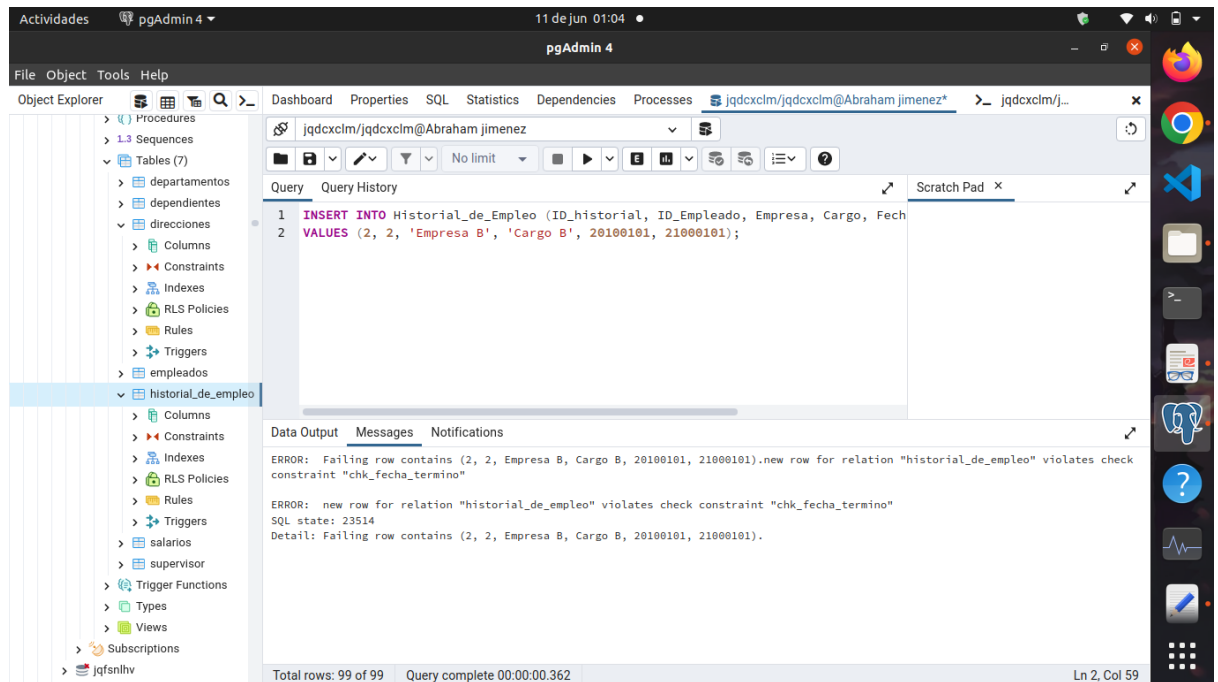
```
ALTER TABLE Historial_de_Empleo
```

```
ADD CONSTRAINT CHK_Fecha_Termino CHECK (Fecha_de_termino < 21000101);
```

Instrucción que evidencia si la restricción está funcionando:

```
INSERT INTO Historial_de_Empleo (ID_historial, ID_Empleado, Empresa, Cargo, Fecha_de_inicio, Fecha_de_termino)
VALUES (2, 2, 'Empresa B', 'Cargo B', 20100101, 21000101);
```

Captura de pantalla:



Al ejecutar esta instrucción, se producirá un error debido a que la fecha de termino es igual al año de la empresa esto muestra que la restricción CHECK está funcionando correctamente.

### C) Ejemplo 3

Tabla elegida: Historial\_de\_empleo

Atributo elegido: fecha\_de\_inicio

Breve descripción de la restricción: Verificamos que la fecha de inicio sea mayor a la fecha de apertura de la empresa.

Instrucción para la creación de la restricción:

```
ALTER TABLE Historial_de_Empleo
```

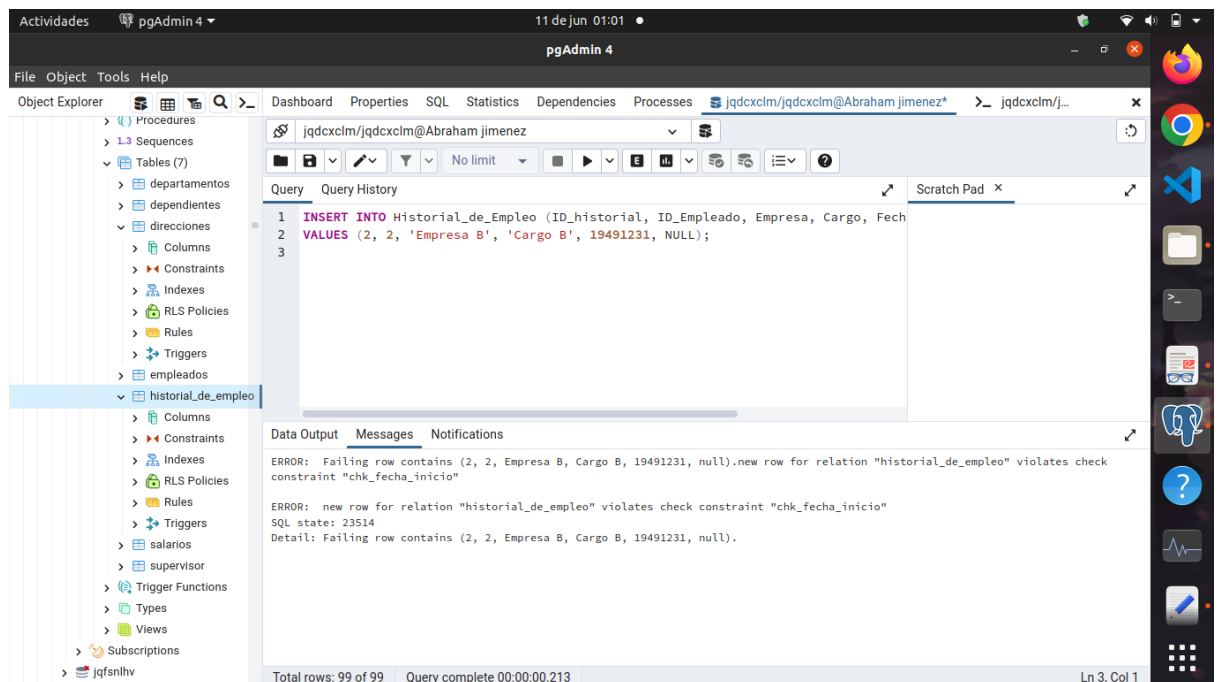
```
ADD CONSTRAINT CHK_Fecha_Inicio CHECK (Fecha_de_inicio > 19500101);
```

Instrucción que evidencia si la restricción está funcionando:

```
INSERT INTO Historial_de_Empleo (ID_historial, ID_Empleado, Empresa, Cargo, Fecha_de_inicio, Fecha_de_termino)
```

```
VALUES (2, 2, 'Empresa B', 'Cargo B', 19491231, NULL);
```

Captura de pantalla:



Al ejecutar esta instrucción, se producirá un error debido a que la fecha de inicio es menor al año de apertura de la empresa esto muestra que la restricción CHECK está funcionando correctamente.

**8. Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.**

- **Tabla elegida**
- **Atributo elegido**
- **Breve descripción del dominio y de la restricción check propuesta.**
- **Instrucción para la creación del dominio personalizado.**
- **Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.**

**A) Dominio personalizado 1**

Tabla elegida: Salarios

Atributo elegido: Fecha de inicio

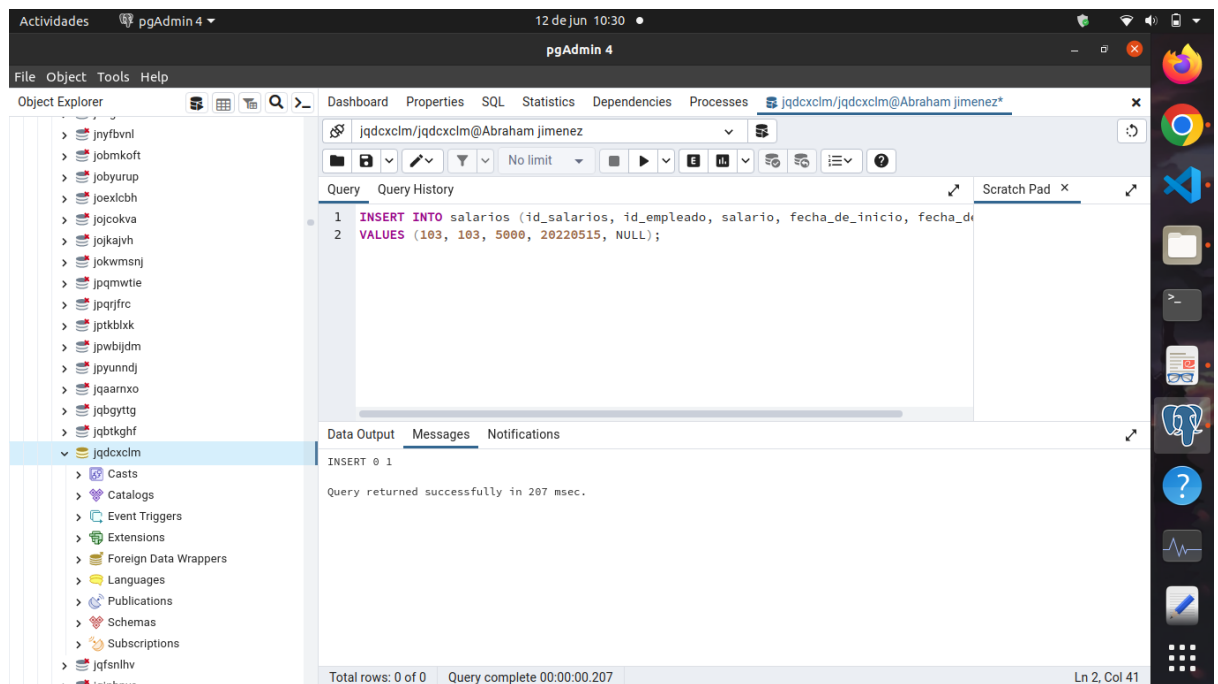
Breve descripción del dominio y de la restricción check propuesta:

El domain Fechalnicio tiene una restricción CHECK que especifica que el valor ingresado debe ser mayor o igual a 19500101 (que representa el 1 de enero de 1950) y menor o igual a 21000101 (que representa el 1 de enero de 2100). Esto garantiza que solo se puedan ingresar fechas de inicio válidas dentro de ese rango.

Instrucción para la creación del dominio personalizado:

```
CREATE DOMAIN Fechalnicio AS INTEGER CHECK (VALUE >= 19500101 AND VALUE <= 21000101);
```

Captura de pantalla:



En este ejemplo, se está insertando un nuevo registro en la tabla salarios. El valor 20220515 se está asignando a la columna fecha\_de\_inicio, que utiliza el dominio Fechalnicio. Dado que 20220515 cumple con la restricción CHECK del dominio, que especifica que el valor debe estar entre 19500101 y 21000101, por lo tanto cumple con la restricción.

## B) Dominio personalizado 2

Tabla elegida: Salarios

Atributo elegido: Fecha\_de\_termino

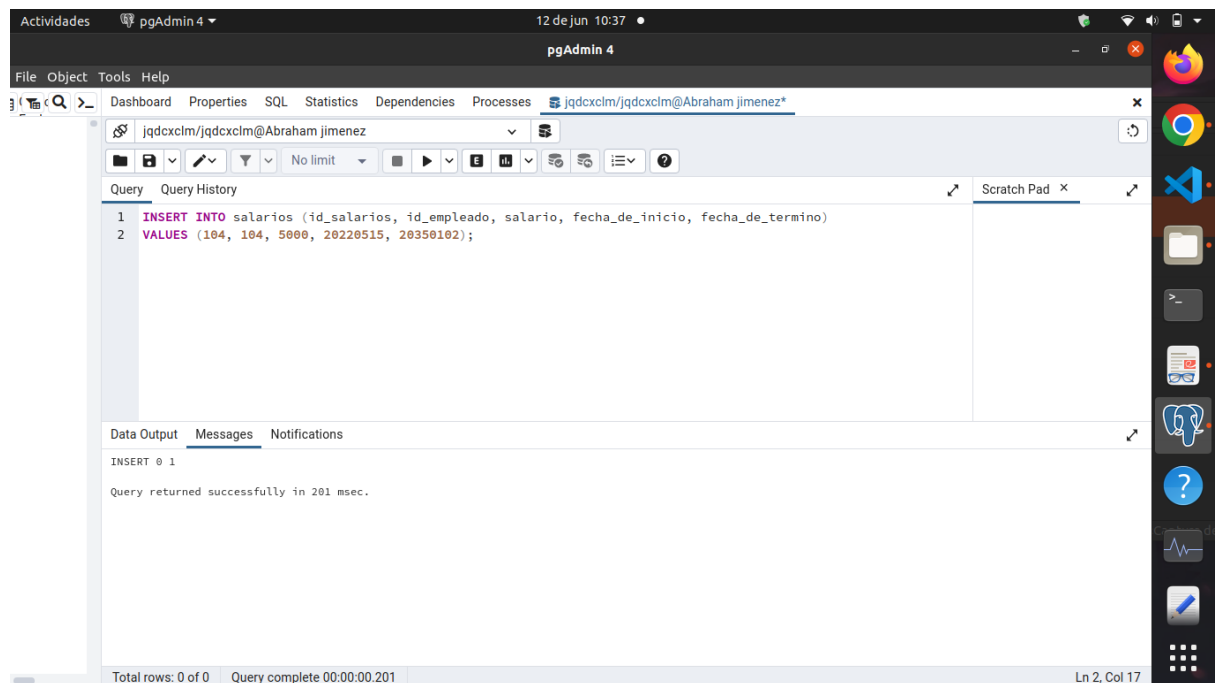
Breve descripción del dominio y de la restricción check propuesta:

El domain FechaTermino tiene la clausula CHECK que se utiliza para aplicar una restricción en el dominio. En este caso, la restricción especifica que el valor asignado a cualquier columna o variable basada en este dominio debe ser mayor o igual a 19500201 y menor o igual a 21000101.

Instrucción para la creación del dominio personalizado:

```
CREATE DOMAIN FechaTermino AS INTEGER CHECK (VALUE >= 19500201 AND VALUE <= 21000101);
```

Captura de pantalla:



En este ejemplo, se está insertando un nuevo registro en la tabla salarios. El valor 20350102 se está asignando a la columna fecha\_de\_termino, que utiliza el dominio FechaTermino. Dado que 20350102 cumple con la restricción CHECK del dominio, que especifica que el valor debe estar entre 19500201 y 21000101, por lo tanto cumple con la restricción.

### C) Dominio personalizado 3

Tabla elegida empleados

Atributo elegido: Correo\_electronico

Breve descripción del dominio y de la restricción check propuesta:

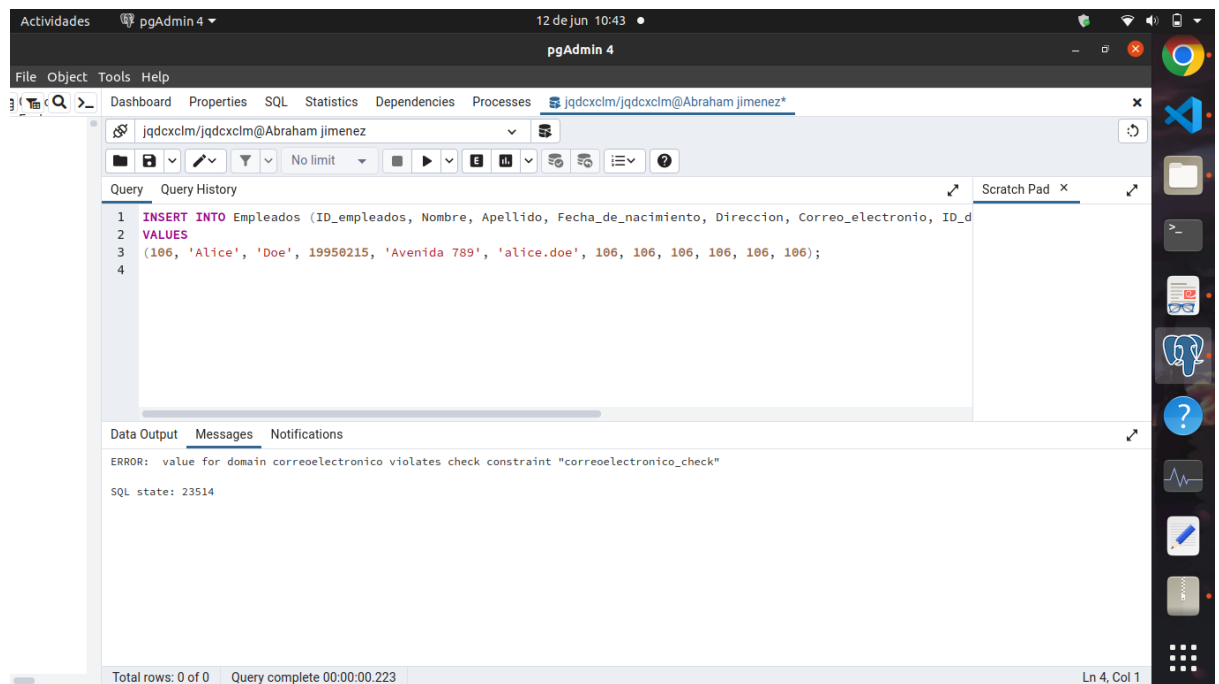
El domain utilizado es CorreoElectronico, la restricción del dominio se especifica mediante una expresión regular. En este caso, la expresión regular

es `^[A-Za-z0-9._%+~]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$`, que valida que el valor de la columna "correo\_electronico" cumpla con el formato de una dirección de correo electrónico válida.

Instrucción para la creación del dominio personalizado:

```
CREATE DOMAIN CorreoElectronico AS VARCHAR(50) CHECK (VALUE ~*
'^[A-Za-z0-9._%+~]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$');
```

Captura de pantalla:



Aquí podemos ver como el correo electronico no cumple con el domain permitido lo que nos arroja un error por lo cual el domain cumple con la restricción del dominio.

## 9. Evidencia del funcionamiento de al menos 2 restricciones para “tuplas” en diferentes tablas.

- Tabla elegida
- Breve descripción de la restricción.
- Instrucción para la creación de la restricción.
- Instrucción “Insert” o “Update” que permita evidenciar que la restricción esta funcionando.
- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

## A) Evidencia 1

Tabla elegida: Dependientes

Restriccion check llamada "chk\_nombre\_parentesco" que verifica las siguientes condiciones:

Si el campo "Nombre" es nulo, entonces el campo "Parentesco" también debe ser nulo.

Si el campo "Nombre" no es nulo, entonces el campo "Parentesco" tampoco debe ser nulo.

Instruccion para la creacion de la restriccion:

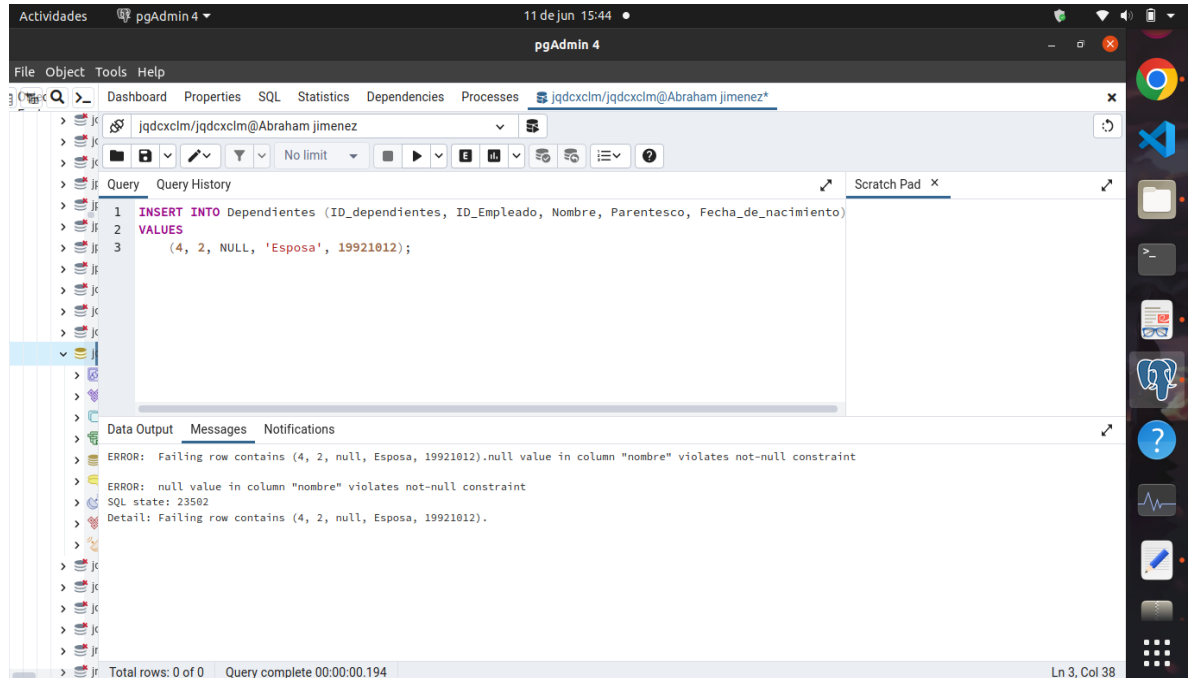
ALTER TABLE Dependientes

ADD CONSTRAINT chk\_nombre\_parentesco CHECK ((Nombre IS NULL AND Parentesco IS NULL) OR (Nombre IS NOT NULL AND Parentesco IS NOT NULL));

Instruccion insert :

INSERT INTO Dependientes (ID\_dependientes, ID\_Empleado, Nombre, Parentesco, Fecha\_de\_nacimiento)  
VALUES(4, 2, NULL, 'Esposa', 19921012);

Captura de pantalla:



En este ejemplo tenemos un valor null pero en el parentesco no tenemos un null entonces nos arroja un error ya que los dos deben ser null o deben tener valores.

## B) Evidencia 2.

Tabla elegida: Historial\_de\_empleo

Descripcion : En este ejemplo, se agrega una restricción CHECK llamada "chk\_empresa\_cargo" que verifica las siguientes condiciones:

Si el campo "Empresa" es nulo, entonces el campo "Cargo" también debe ser nulo.

Si el campo "Empresa" no es nulo, entonces el campo "Cargo" tampoco debe ser nulo.

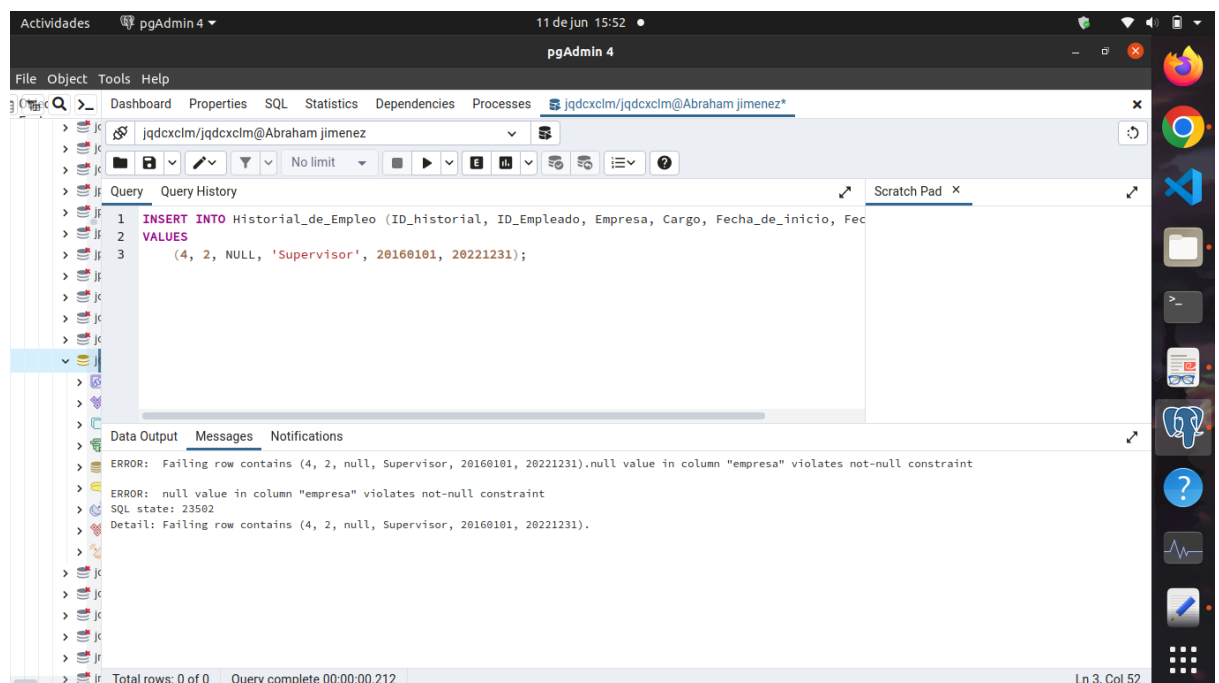
Instruccion para la creacion de la restricción:

```
ALTER TABLE Historial_de_Empleo
ADD CONSTRAINT chk_empresa_cargo
CHECK ((Empresa IS NULL AND Cargo IS NULL) OR
      (Empresa IS NOT NULL AND Cargo IS NOT NULL));
```

Instruccion Insert para evidenciar la restricción:

```
INSERT INTO Historial_de_Empleo (ID_historial, ID_Empleado, Empresa,
Cargo, Fecha_de_inicio, Fecha_de_termino) VALUES (4, 2, NULL,
'Supervisor', 20160101, 20221231);
```

Captura de pantalla :



En este ejemplo tenemos el cargo pero en el nombre de empresa tenemos null, entonces tenemos un error ya que la restricción nos dice que debemos tener 2 null o dos valores.

**10. Plantea 3 consultas que consideres relevantes para la base de datos propuesta. Para cada consulta planteada, incluir en el reporte los siguientes incisos:**

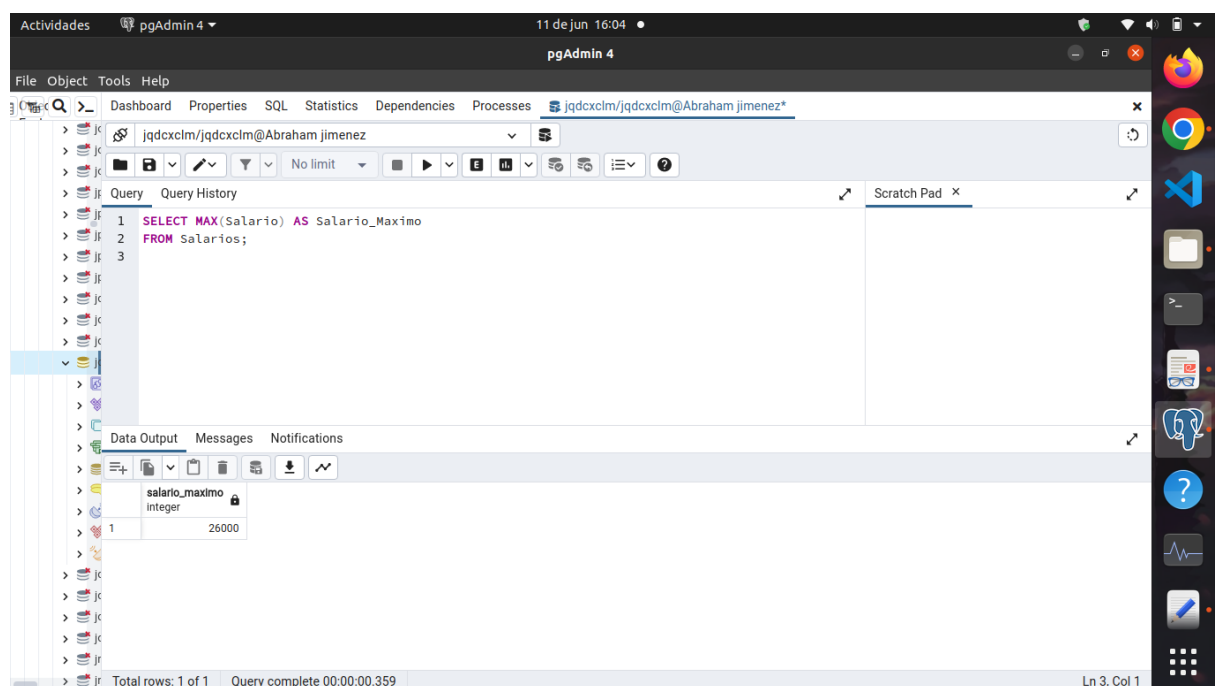
- a) Redacción clara de la consulta.
- b) Código en lenguaje SQL de la consulta.
- c) Ejecutar la consulta en Postgres e incluir una captura de pantalla con el resultado de la consulta.

**A) Consulta 1. Consultamos el salario máximo de la tabla salarios.**

Código en SQL de la consulta:

```
SELECT MAX(Salario) AS Salario_Maximo  
FROM Salarios;
```

Captura de pantalla de la consulta:



**B) Consulta 2.**

En esta consulta, se seleccionan todos los campos (usando el asterisco \*) de la tabla "Historial\_de\_Empleo" usando la cláusula SELECT. Luego, se utiliza la cláusula WHERE para filtrar los resultados y obtener solamente los registros donde el valor del campo "Empresa" sea igual a "Global Solutions"

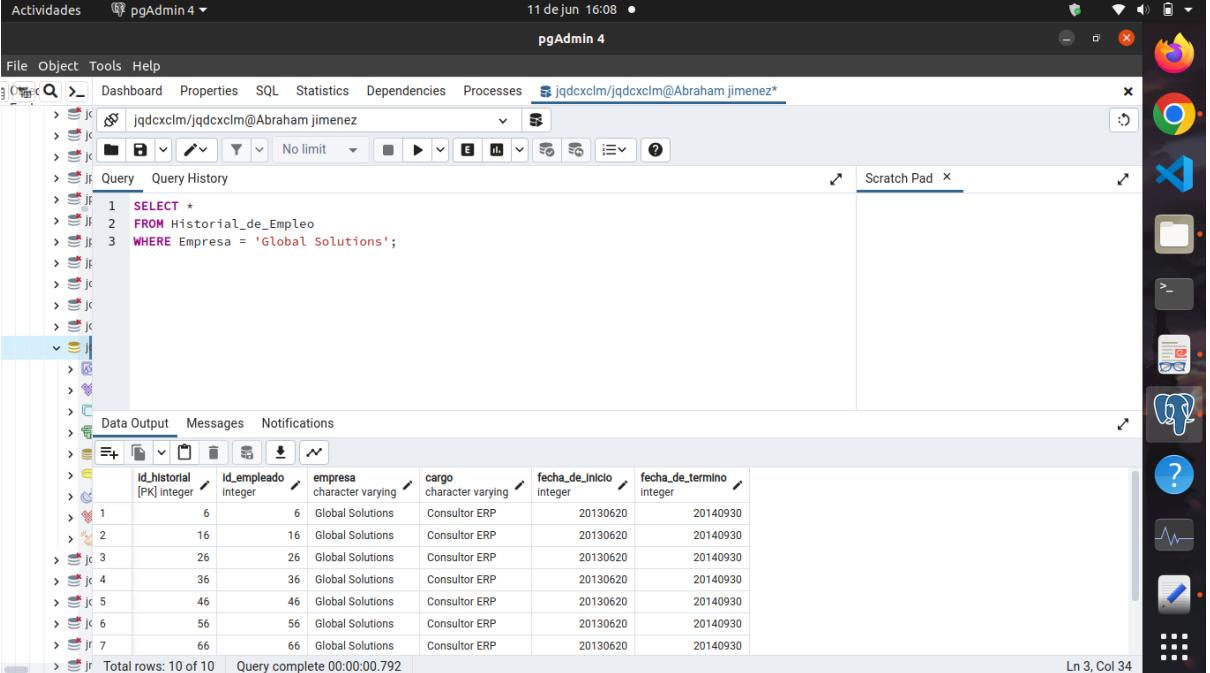
Código en SQL de la consulta:

```
SELECT *  
FROM Historial_de_Empleo  
WHERE Empresa = 'Global Solutions';
```



WHERE Empresa = 'Global Solutions';

Captura de pantalla de la consulta:



The screenshot shows the pgAdmin 4 interface. The SQL query editor contains the following query:

```
1 SELECT *
2 FROM Historial_de_Empleo
3 WHERE Empresa = 'Global Solutions';
```

The Data Output tab shows the results of the query in a table with 7 columns and 7 rows of data.

	Id_historial [PK] integer	Id_empleado integer	empresa character varying	cargo character varying	fecha_de_inicio integer	fecha_de_termino integer
1	6	6	Global Solutions	Consultor ERP	20130620	20140930
2	16	16	Global Solutions	Consultor ERP	20130620	20140930
3	26	26	Global Solutions	Consultor ERP	20130620	20140930
4	36	36	Global Solutions	Consultor ERP	20130620	20140930
5	46	46	Global Solutions	Consultor ERP	20130620	20140930
6	56	56	Global Solutions	Consultor ERP	20130620	20140930
7	66	66	Global Solutions	Consultor ERP	20130620	20140930

Total rows: 10 of 10 Query complete 00:00:00.792 Ln 3, Col 34

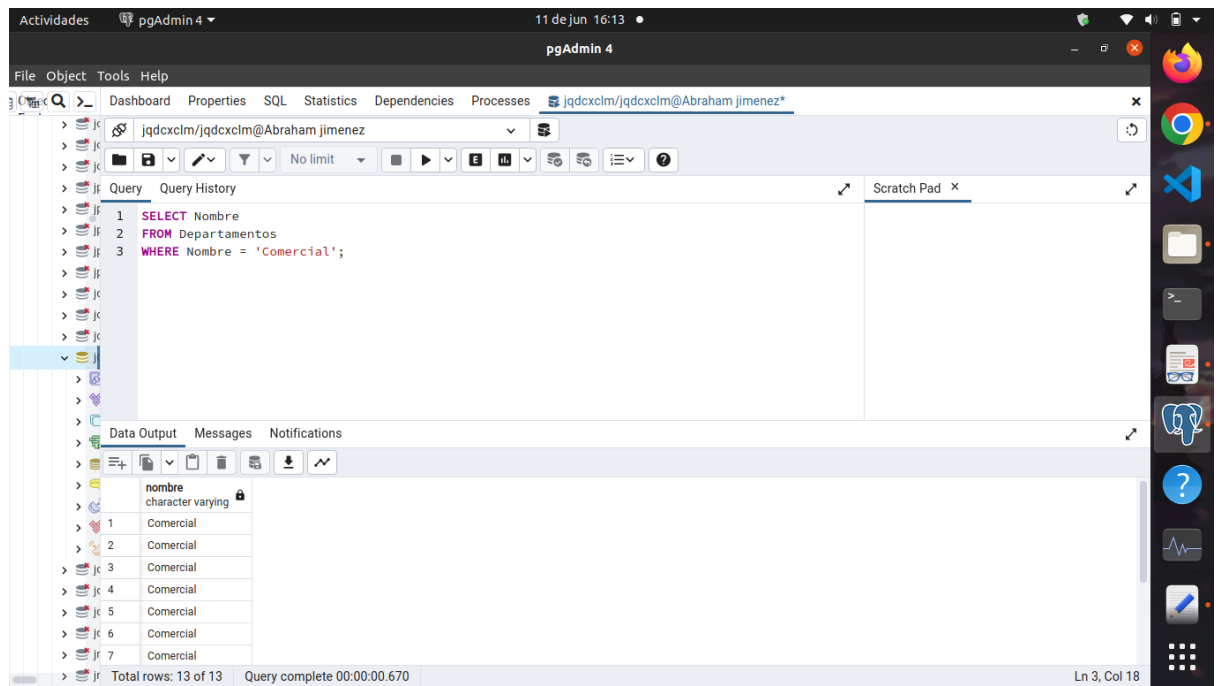
### C) Consulta 3.

En esta consulta, se selecciona el campo "Nombre" de la tabla "Departamentos" utilizando la cláusula SELECT. Luego, se utiliza la cláusula WHERE para filtrar los resultados y obtener solamente los registros donde el valor del campo "Nombre" sea igual a 'Comercial'.

Código en SQL de la consulta:

```
SELECT Nombre
FROM Departamentos
WHERE Nombre = 'Comercial';
```

Captura de pantalla de la consulta:



**11. Plantea 3 vistas que consideres relevantes para la base de datos propuesta. Para cada vista planteada, incluir en el reporte los siguientes incisos:**

- Redacción clara de la vista planteada.
- Código en lenguaje SQL que permita crear la vista solicitada.
- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla.
- Incluir un ejemplo que los evaluadores puedan ejecutar para verificar el funcionamiento de las vistas.

#### A) 1.Vista de la información de empleados

Redacción:

La vista "Vista\_Empleados" se crea a partir de una consulta SELECT que combina información de varias tablas: "Empleados", "Departamentos" y "Salarios". La consulta utiliza las cláusulas JOIN para unir las tablas en función de las claves primarias y foráneas correspondientes.

La estructura de la vista incluye las siguientes columnas:

ID\_empleados: Es el identificador único de cada empleado.

Nombre: Es el nombre del empleado.

Apellido: Es el apellido del empleado.

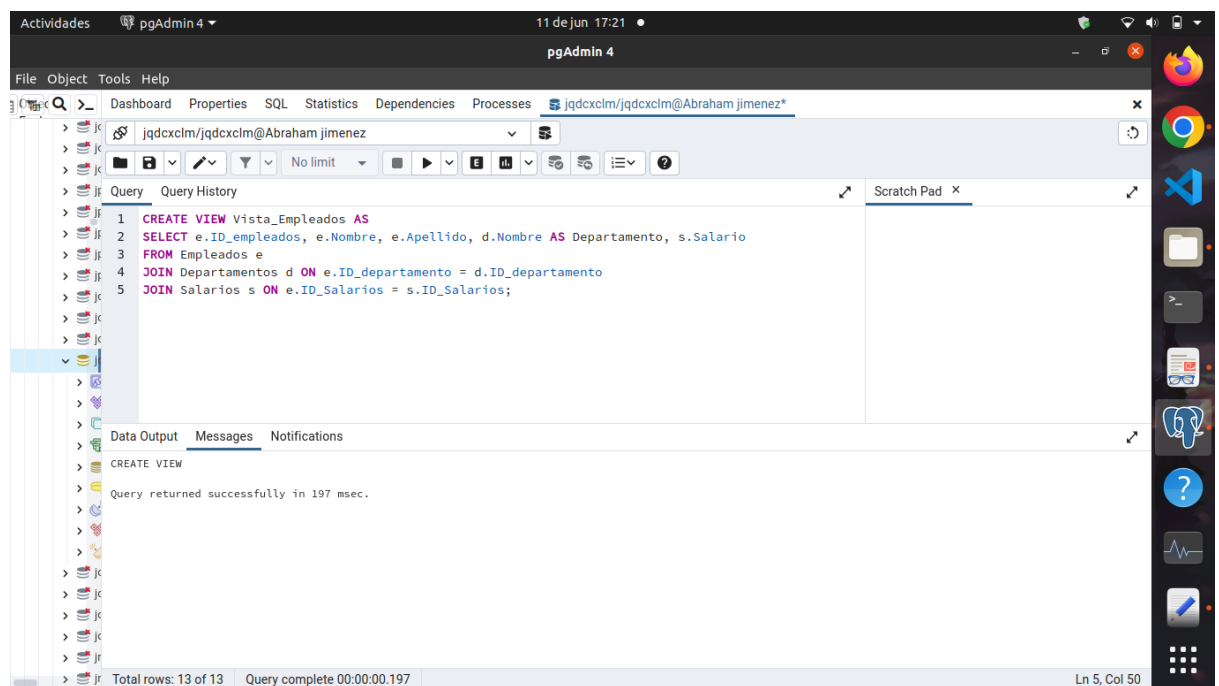
Departamento: Es el nombre del departamento al que pertenece el empleado.

Salario: Es el salario del empleado.

Código en SQL de la vista:

```
CREATE VIEW Vista_Empleados AS
SELECT e.ID_empleados, e.Nombre, e.Apellido, d.Nombre AS
Departamento, s.Salario
FROM Empleados e
JOIN Departamentos d ON e.ID_departamento = d.ID_departamento
JOIN Salarios s ON e.ID_Salarios = s.ID_Salarios;
```

Creación de la vista



EJEMPLO:

```
SELECT *
FROM Vista_Empleados;
```

## B) Vista de direccion y dependientes de empleados

Redacción:

Crea una vista llamada "Vista\_Direccion\_Dependientes". Esta vista combina información de las tablas "Empleados", "Direcciones" y "Dependientes" utilizando las cláusulas JOIN para unir las tablas en función de las claves primarias y foráneas correspondientes.

La estructura de la vista incluye las siguientes columnas:

ID\_empleados: Es el identificador único de cada empleado.

Nombre: Es el nombre del empleado.

Calle: Es la calle de la dirección asociada al empleado.

Ciudad: Es la ciudad de la dirección asociada al empleado.

Estado: Es el estado de la dirección asociada al empleado.

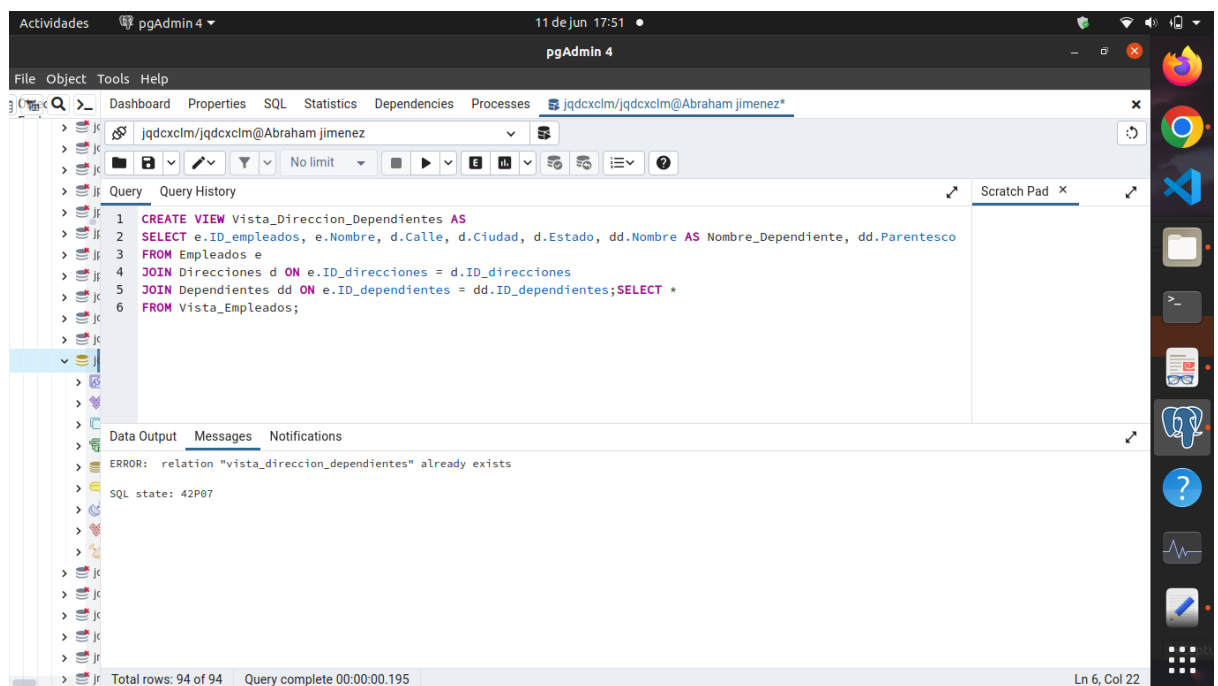
Nombre\_Dependiente: Es el nombre del dependiente asociado al empleado.

Parentesco: Es el parentesco del dependiente con el empleado.

Código SQL de la vista:

```
CREATE VIEW Vista_Direccion_Dependientes AS
SELECT e.ID_empleados, e.Nombre, d.Calle, d.Ciudad, d.Estado,
dd.Nombre AS Nombre_Dependiente, dd.Parentesco
FROM Empleados e
JOIN Direcciones d ON e.ID_direcciones = d.ID_direcciones
JOIN Dependientes dd ON e.ID_dependientes = dd.ID_dependientes;
```

Creacion de la vista:( ya lo habíamos escrito y no tome captura de pantalla sorry).



**Ejemplo.**

```
SELECT *
FROM Vista_Direccion_Dependientes;
```

### C) Vista 3. Vista historial\_de\_empleo con supervisor

Redacción:

Creo una vista llamada "Vista\_Historial\_Supervisor". Esta vista combina información de las tablas "Historial\_de\_Empleo", "Empleados" y "Supervisor"

utilizando las cláusulas JOIN para unir las tablas en función de las claves primarias y foráneas correspondientes.

La estructura de la vista incluye las siguientes columnas:

ID\_historial: Es el identificador único de cada registro en el historial de empleo.

Empresa: Es el nombre de la empresa asociada al historial de empleo.

Cargo: Es el cargo o puesto que el empleado ocupó durante ese periodo de empleo.

Fecha\_de\_inicio: Es la fecha de inicio del periodo de empleo.

Fecha\_de\_termino: Es la fecha de término del periodo de empleo.

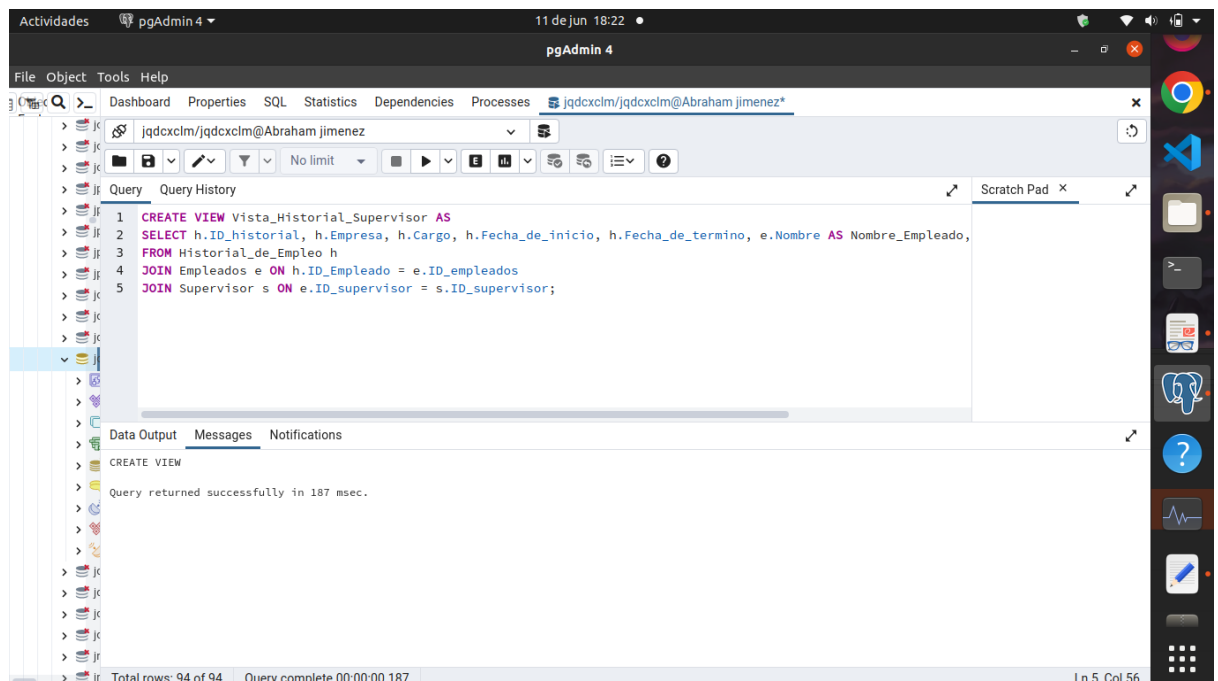
Nombre\_Empleado: Es el nombre del empleado asociado al historial de empleo.

Nombre\_Supervisor: Es el nombre del supervisor asociado al empleado en ese periodo de empleo.

Código en SQL:

```
CREATE VIEW Vista_Historial_Supervisor AS
SELECT h.ID_historial, h.Empresa, h.Cargo, h.Fecha_de_inicio,
h.Fecha_de_termino, e.Nombre AS Nombre_Empleado, s.Nombre AS
Nombre_Supervisor
FROM Historial_de_Empleo h
JOIN Empleados e ON h.ID_Empleado = e.ID_empleados
JOIN Supervisor s ON e.ID_supervisor = s.ID_supervisor;
```

Creación de la vista:



```
EJEMPLO;  
SELECT *  
FROM Vista_Historial_Supervisor;
```