Base de Datos II

2024

Tecnólogo en Informática

Laboratorio

Grupo: LEDHDF

Carlos Gil - 5.499.582-7

Salvador Vanoli - 5.620.198-5

Valentin Veintemilla - 5.419.774-4

UTEC

Docente

Deborah Rivas

Índice

Letra propuesta

BASE DE DATOS II // LABORATORIO // 2024
INFORMACIÓN del LABORATORIO

-El obligatorio será realizado en forma grupal (con un mínimo de 2 integrantes y un
máximo de 3 integrantes por grupo).
-No se aceptan trabajos individuales, salvo casos de fuerza mayor los cuales deben ser
expuestos al docente, los cuales serán evaluados por el mismo.
-Vale un total de 10 puntos, un mínimo 6 puntos para aprobar (60%) y tiene su defensa
final.
-Si se pierde implica perder el total del curso. Puede pasar que en un mismo grupo
luego de la defensa obtengan calificaciones diferentes dentro del mismo grupo

El plazo para la entrega es hasta el 21/06/2024 hasta las 12:00 horas. Una vez culminado el plazo estipulado, no se aceptarán más obligatorios por parte del docente.

FORMATO de la ENTREGA

La entrega se realizará en carpeta, con el formato solicitado:
UTEC - Carrera Tecnólogo en Informática
Título Base de Datos II - Laboratorio
Nombre del grupo
Nombre, apellido y cédula de cada integrante
Paginado
Letra uniforme en todo el trabajo
Letra fuente Verdana, tamaño fuente 11/12, interlineado (0,0, sencillo)
Letra propuesta
MER, DR, restricciones de integridad, datos de cada tabla, el sistema de gestión
Resultado del trabajo, con los comandos SQL, para las consultas propuestas
Resultado del trabajo, con los comandos SQL, para la definición y manejo de datos

Resultado del trabajo, con las operaciones de Álgebra Relacional

OBJETIVO

El objetivo principal de esta actividad grupal, es comprender cómo se realizan consultas sobre los datos de una base de datos concreta. Al momento de la corrección se tendrán en cuenta tanto que la solución sea correcta.

LETRA PROPUESTA

Se desea mantener su sistema de información, donde los datos que se desean organizar y modelar son los siguientes:

Existe una empresa, que produce varios productos, que son suministrados por distribuidores, los vende a varios clientes y donde trabajan varios funcionarios.

De la empresa se conoce su nombre, dirección, teléfono, email, código postal, ciudad, país.

De los productos se conoce el número de producto o identificador, su nombre, descripción, tipo de producto, cantidad en stock, la cantidad de stock mínima que precisa y precio unitario.

La empresa tiene contacto con varios distribuidores, que son los que suministran sus productos en distintos puntos geográficos o ciudades. Un mismo tipo de producto puede ser suministrado por varios distribuidores. De cada distribuidor se desea mantener su cedula, nombre apellido, dirección, email, teléfono celular y ciudad.

De los clientes se desea conocer sus datos personales como cedula, nombre, apellido, fecha de nacimiento, dirección, email, teléfono celular y ciudad. Dentro de la empresa, los clientes se diferencian por un código interno, que se incrementa automáticamente cuando un cliente se da de alta en ella.

Un cliente puede comprar tantos productos como desee a la empresa y un mismo tipo de producto puede ser comprado por varios clientes diferentes. Cada vez que se realiza una compra/venta, la misma quedará registrada en la base de datos. El registro de compra debe corresponder a

la empresa, donde se sabe el código, la fecha en la que se ha comprado, el porcentaje de IVA aplicado, una descripción de la misma y la forma de pago. Cada compra puede tener muchas líneas de compra, de las cuales se sabe el código de linea, el número de producto comprado, la cantidad comprada y el precio de compra asociado. Cada linea debe tener un producto asociado y a su vez, el producto puede aparecer en varias compras o registros de compras. De las forma de pago se sabe, el código de forma de pago y el tipo (efectivo, tarjeta de crédito, tarjeta de débito, cheque, PayPal, etc.).

A futuro, la empresa desea vender por catálogo online. El cliente podrá pedir el producto, pagarlo y preguntar sobre el estado de su pedido. Podrá preguntar una vez realizado el pedido, cada 3 días, sino ha recibido el encargo.

El cliente puede presentar quejas o reclamaciones. Cuando se recibe su reclamación, se desea registrarla en el sistema de información, con un código, una descripción, una clasificación (simple, media o compleja), una fecha de alta y una fecha de resolución. Cuando la resolución se da, se envía un mensaje o carta al cliente y el proceso finaliza.

Cada funcionario trabaja para una sola empresa, vende un único producto de la misma y trabaja en un área de trabajo (una y solo una por funcionario). De los funcionarios se conoce su número de cédula, su nombre, apellido, fecha de nacimiento, dirección, email, teléfono celular, título obtenido, número de área de trabajo, cargo en la misma y la fecha de alta en la empresa. Cada funcionario tiene un único jefe funcionario de la empresa. Entonces un funcionario puede ser jefe de varios funcionarios y un subordinado es gestionado por un único jefe. Se conoce la descripción de dicha relación, donde no todos tienen jefe.

Cada funcionario realiza muchas tareas, de las cuales se sabe el código, nombre y descripción de la misma. De cada funcionario también interesa saber el número de hijos que tiene, su sueldo o salario y comisión. De cada hijo se desea conocer su cédula, nombre y fecha de nacimiento.

También la información relativa a las habilidades del funcionario, por ejemplo: marketing, mercadotecnia, programación. Cada habilidad tendrá un código y una descripción y un funcionario puede tener varias habilidades, donde una misma habilidad, puede ser realizada por varios funcionarios diferentes.

Del área de trabajo se sabe que hay varios cargos y los mismos se pueden repetir. De las áreas de trabajo se sabe su número de área, su nombre, empresa, ciudad y su presupuesto anual.

La misma auto gestiona sus propias ofertas de empleo, entonces cada vez que es necesario, abre un llamado a interesados. De los llamados se conoce su número, una descripción, la fecha de posteo y la fecha límite de presentación al mismo. De cada persona interesada, inscripta, se registra la fecha de inscripción y su curriculum vitae. En caso que se decida no contratar a nadie, el llamado se declara como desierto y se registra el motivo de dicha resolución, para tenerlo en cuenta en futuros llamados. También se puede dar la situación, de que ninguna persona se inscriba al llamado, en cuyo caso también es declarado como desierto. De lo contrario, se registran las personas contratadas al mismo.

NOTAS:

-de todos los atributos a mantener, los mismos contienen datos, no hay ninguno en NULL.
-se tomará en cuenta la creatividad a la hora de definir datos y trabajo con comandos SQL (definición, manejo de datos y consultas) y AR.
-tomar en cuenta que no todos los manejadores de bded permiten la agrupación de funciones matemáticas.
-resolver las consultas SQL, con consultas, SubConsultas y AR.

SE SOLICITA

a) presentar el MER, DR o MR, restricciones de integridad y datos para cada tabla.
Para cada atributo, seleccionar el tipo de campo, que mejor se adapte a los datos del mismo. Tomar en cuenta que todos los atributos contienen datos (valores), en principio no hay ninguno en NULL. Cada tabla debe tener la cantidad necesaria de datos, para la resolución de las consultas propuestas.

b) implementar la solución, utilizando una base de datos relacional, en el sistema de gestión de base de datos elegido
- ejecutar y presentar el resultado del trabajo, con los comandos SQL, para las consultas propuestas. Presentar resultados de las mismas también.

- ejecutar y presentar el resultado del trabajo, con los comandos SQL, para la definición y manejo de datos de una base de datos (creación, inserción, borrado, modificación, etc.)

c) presentar el resultado del trabajo, con las operaciones de Álgebra Relacional (AR), para TODAS las consultas propuestas

d) Consultas

01) Mostrar la lista de los funcionarios (nombre, apellido, cargo), que tienen el mismo cargo que "Juan Pedro". Puede haber más de un funcionario con nombre "Juan Pedro".
SUBCONSULTA

02) Mostrar la lista de los funcionarios (nombre, apellido, sueldo, número de área de trabajo), que ganan igual o más, que el máximo sueldo, de los funcionarios del área de trabajo número "1404". SUBCONSULTA

03) Para cada cargo, mostrar la suma total de los sueldos, de los funcionarios del área de trabajo de nombre "SQA". Presentar cargo y la suma total como sumaSueldos.
SUBCONSULTA

04) Presentar la cantidad total de funcionarios, con cargo "Tester", del área de trabajo de nombre "SQA". Presentar la cantidad total como totalTesters. SUBCONSULTA

05) Presentar el cargo, el sueldo máximo y el sueldo mínimo, de cada grupo de cargos, pero sólo para aquellos grupos, con más de dos tuplas (registros) y con un sueldo máximo mayor o igual a 1404.

06) Presentar la lista de funcionarios (cedula de identidad, nombre, apellido, titulo, sueldo), cuyo sueldo supere, el sueldo medio de todos los funcionarios. SUBCONSULTA

07) Mostrar la lista de los funcionarios cuyo sueldo, es mayor, que el de los "Supervisores" o "Directores". Presentar nombre, apellido, sueldo y número de área de trabajo del funcionario. SUBCONSULTA

08) Presentar la lista de funcionarios, con cargo "Tester", cuyo sueldo es mayor, que el de los "Supervisores" o "Directores". Presentar nombre,

apellido, sueldo y número de área de trabajo del funcionario. SUBCONSULTA

09) Listar, en orden alfabético ascendente por nombre, a los funcionarios que NO trabajen ni en la ciudad de “Montevideo” ni en “Maldonado”. Presentar nombre, apellido y número de área de trabajo. SUBCONSULTA

10) Listar, en orden alfabético descendente por nombre, a los funcionarios que NO trabajen ni en la ciudad de “Montevideo”, ni en “Minas”, ni en “Colonia del Sacramento”. Presentar nombre, apellido, número de área de trabajo y cargo. En la solución trabajar con los comandos IN y NOT IN. SUBCONSULTA

11) Listar el número y el nombre, de las áreas de trabajo, que tengan algún funcionario con fecha de alta posterior al año “2016”. SUBCONSULTA

12) Listar el número y el nombre de las áreas de trabajo, que tengan algún funcionario con fecha de alta posterior al año “2014” y con una comisión mayor o igual al 20%. SUBCONSULTA

13) REALIZAR CON COMANDOS IN y EXISTS (por separado las SubConsultas). Mostrar el número, nombre y ciudad, de las áreas de trabajo, donde existan funcionarios, cuya comisión, sea mayor al 10% del sueldo (es por cada funcionario). SUBCONSULTA

14) REALIZAR CON COMANDOS IN y =ANY (por separado las SubConsultas). Presentar que todos los funcionarios, tengan asignado un número de área de trabajo, existente en la tabla AREA de TRABAJO. Presentar el número y nombre del área de trabajo. SUBCONSULTAS

15) REALIZAR CON COMANDO =ANY. Presentar el número, nombre y ciudad, de aquellas áreas de trabajo, en los que al menos, exista un funcionario con comisión mayor a “4”. SUBCONSULTA

16) REALIZAR CON COMANDO HAVING. Mostrar el número de área de trabajo y el sueldo máximo del área, para todas las áreas cuyo sueldo máximo supere, el sueldo medio de todos los funcionarios. SUBCONSULTA

17) Presentar los números de área de trabajo, con más funcionarios asignados. Presentar: número de área de trabajo y la cantidad total de funcionarios asignados por área de trabajo. SUBCONSULTA

18) Resolver con subConsulta (de 3 niveles). Presentar el número, el nombre y la ciudad, del área de trabajo, con más funcionarios asignados, con el cargo “Tester”.

19) Resolver con subConsulta (de 3 niveles). Presentar el número, el nombre y la ciudad, del área de trabajo, con más funcionarios asignados, con el cargo “administrativo”, utilizando las dos tablas presentadas.

20) Presentar el número y el nombre de las áreas de trabajo, siempre que haya más de dos funcionarios trabajando en ellas. SUBCONSULTA

21) Presentar los números de las áreas de trabajo, en las que el sueldo medio de sus funcionarios (por área), sea menor o igual, que la media de todos los sueldos de todos los funcionarios. Presentar número y sueldo medio por área de trabajo. SUBCONSULTA

22) Presentar la cédula, nombre, apellido y título de todos los funcionarios, cuyo sueldo es mayor, que el sueldo medio de todos los funcionarios con el mismo título. Presentar la información ordenada en forma descendente por título.

23) Presentar el número de área de trabajo, con más presupuesto asignado (sueldo + comisión), para pagar el sueldo y la comisión de sus funcionarios. Presentar número de área y mayor presupuesto. SUBCONSULTA

24) Presentar los números de área de trabajo, cuyo presupuesto asignado, sea menor al presupuesto medio total. Presentar número de área y presupuesto asignado (sueldo + comisión). SUBCONSULTA

25) Presentar la cédula, el número de área de trabajo, el cargo y el sueldo, de los cargos con mayor sueldo de cada área de trabajo.

26) Presentar los datos de los funcionarios (cédula, nombre y apellido), con los datos de su área de trabajo (número, nombre y ciudad).

27) Presentar la cédula, el apellido y el cargo de los funcionarios que pertenezcan al área de trabajo con nombre “SQA”.

28) Presentar la lista de los funcionarios (nombre y apellido) con los nombres de sus jefes.

29) Presentar los datos de los funcionarios (cédula, nombre y apellido), cuyo cargo sea “tester”, con los datos de sus jefes (cédula, nombre y apellido).

30) Presentar los datos de los funcionarios y de sus áreas de trabajo, incluyendo las áreas que no tienen todavia funcionarios asignados. Resolver con reunión x izquierda.

31) Presentar los datos de los funcionarios (cédula, nombre y apellido) y de sus áreas de trabajo (número, nombre y ciudad), incluyendo a los funcionarios que no tienen áreas de trabajo asignadas aún. Resolver con reunion por derecha.

32) Presentar los datos de los funcionarios de las áreas de trabajo diferentes al área de trabajo con nombre “Marketing”.

33) Presentar los datos de los funcionarios de las áreas de trabajo con números menores, que el número de área de trabajo con nombre “Contabilidad”.

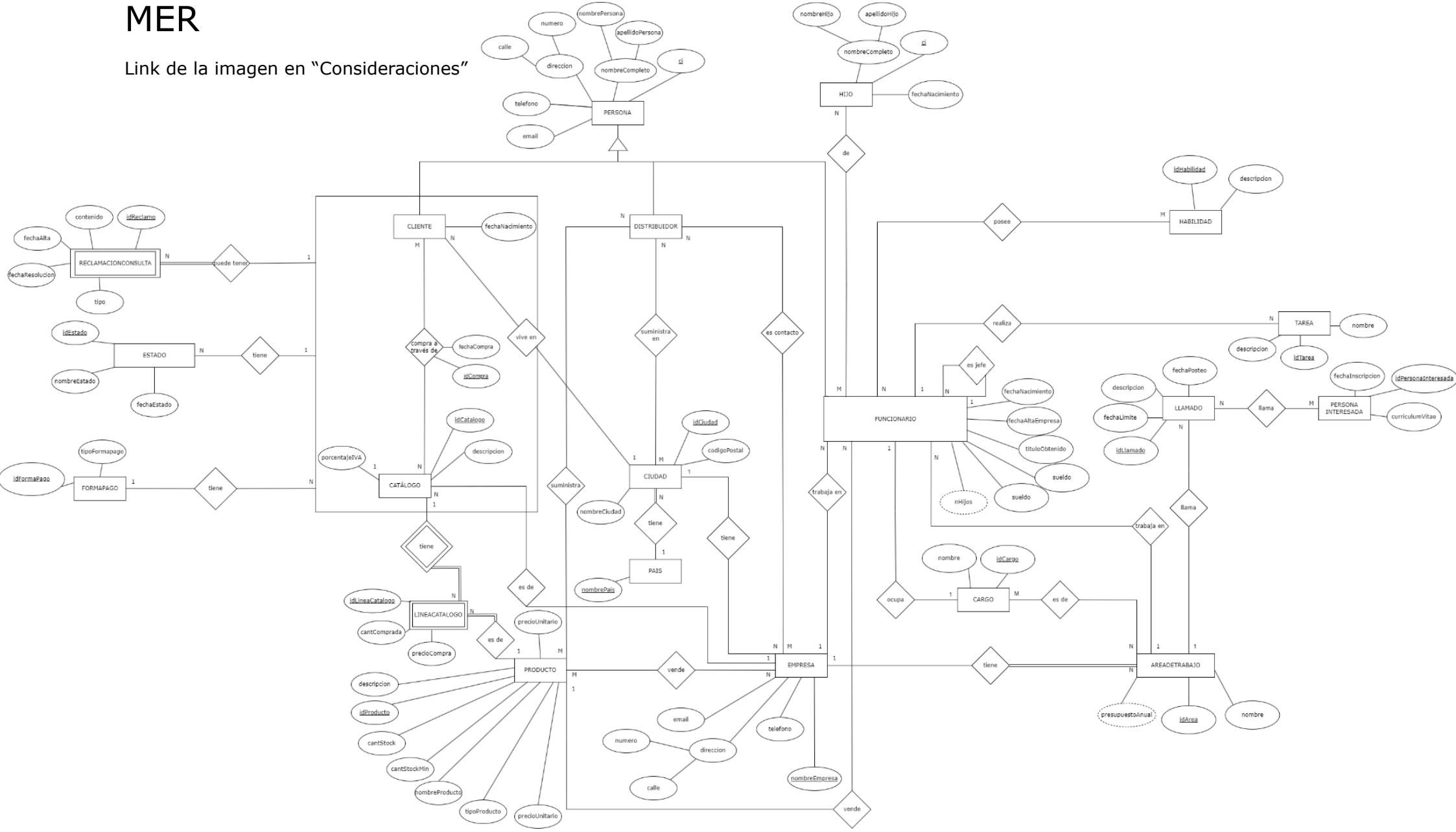
34) Resolver con subConsulta (de 3 niveles) y además, con subConsulta y consulta con JOIN. Presentar los datos de los funcionarios, que tienen el mismo cargo, que los cargos del número de área de trabajo “SQA”. Presentar cédula, nombre, apellido y nombre del cargo.

Consideraciones

- La tabla HIJO repite datos ya que no puede heredar de personas. Personas tienen datos necesarios para funcionario, cliente y distribuidor que no puede tener la tabla Hijo.
- Eliminamos el identificador interno de los clientes ya que formaban una superclave al unirlos con el CI.
- A los inserts agregamos algunos con valor NULL ya que de otra manera no se podrían probar las consultas con LEFT y RIGHT JOIN.
- Para mayor claridad visual, en las consultas con AR utilizamos {} para designar los atributos por los cuales hacemos GROUP BY. Estos los ponemos antes de las “G”.
- Link de la imagen del MER a tamaño completo: https://drive.google.com/file/d/1WKks5gxtkWKmu1OcLxhBOyIjfx6bnBlX/view?usp=sharing

MER

Link de la imagen en “Consideraciones”



Modelo Relacional (MR)

Claves primarias -> Clave primaria
Atributos foráneos -> Foránea

PAIS(nombrePais)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
nombrePais	varchar	PK	50	NOT NULL

CIUDAD(idCiudad, nombreCiudad, codigoPostal, nombrePais)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idCiudad	varchar	PK	50	NOT NULL
nombreCiudad	varchar		50	NOT NULL
codigoPostal	varchar		50	NOT NULL
nombrePais	varchar	FK	50	

RNE

- nombrePais(CIUDAD) pertenece nombrePais(PAIS)

EMPRESA(nombreEmpresa, calle, numero, teléfono, email, idCiudad)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
nombreEmpresa	varchar	PK	50	NOT NULL
calle	varchar		50	
numero	varchar		50	
teléfono	varchar		50	
email	varchar		50	
idCiudad	varchar	FK	50	

RNE

- idCiudad(EMPRESA) pertenece idCiudad(CIUDAD)

PRODUCTO(idProducto, cantStock, cantStockMin, nombreProducto, descripción, tipoProducto, precioUnitario)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idProducto	varchar	PK	50	NOT NULL
cantStock	integer			
cantStockMin	integer			
nombreProducto	varchar		50	
descripción	varchar		50	
tipoProducto	varchar		50	
precioUnitario	integer			

PERSONA(ci, nombrePer, apellidoPer, numero, calle, teléfono, email)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ci	varchar	PK	50	NOT NULL
nombrePersona	varchar		50	
apellidoPersona	varchar		50	
numero	varchar		50	
calle	varchar		50	
teléfono	varchar		8	
email	varchar		50	

HIJO(ci, nombreHijo, apellidoHijo, fechaNacimiento)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ci	varchar	PK	50	NOT NULL
fechaNacimiento	date			
nombreHijo	varchar		50	
apellidoHijo	varchar		50	

CLIENTE(ci, fechaNacimiento, idCiudad)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ci	varchar	PK, FK	50	NOT NULL
fechaNacimiento	date			
idCiudad	varchar	FK	50	

RNE

- ci(CLIENTE) pertenece ci(PERSONA)
- idCiudad(CLIENTE) pertenece idCiudad(CIUDAD)

CATALOGO(idCatalogo, descripcion, porcentajeIVA, idEmpresa)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idCatalogo	varchar	PK	50	NOT NULL
descripcion	varchar		50	
porcentajeIVA	integer			
idEmpresa	varchar	FK	50	NOT NULL

RNE

- idEmpresa(PEDIDO) pertenece idEmpresa(EMPRESA)

CLIENTE_CATALOGO(idCatálogo, ciCliente, idCompra, fechaCompra, idFormaPago)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idCatalogo	varchar	PK, FK	50	NOT NULL
ciCliente	varchar	PK, FK	50	NOT NULL
idCompra	varchar	PK	50	NOT NULL
fechaCompra	date			
idFormaPago	varchar	FK	50	

RNE

- ciCliente(CLIENTE_CATALOGO) pertenece ci(CLIENTE)
- idCatalogo(CLIENTE_CATALOGO) pertenece idCatalogo(CATALOGO)

DISTRIBUIDOR(ci)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ci	varchar	PK, FK	50	NOT NULL

RNE

- ci(DISTRIBUIDOR) pertenece ci(PERSONA)

DISTRIBUIDOR_PRODUCTO(ci, idProducto)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ciDistribuidor	varchar	PK, FK	50	NOT NULL
idProducto	varchar	FK, FK	50	NOT NULL

RNE

- ciDistribuidor(DISTRIBUIDOR_PRODUCTO) pertenece ci(DISTRIBUIDOR)
- idProducto(DISTRIBUIDOR_PRODUCTO) pertenece idProducto(PRODUCTO)

DISTRIBUIDOR_CIUDAD(ci, idCiudad)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ciDistribuidor	varchar	PK, FK	50	NOT NULL
idCiudad	varchar	FK, FK	50	NOT NULL

RNE

- ciDistribuidor(DISTRIBUIDOR_CIUDAD) pertenece ci(DISTRIBUIDOR)
- idCiudad(DISTRIBUIDOR_CIUDAD) pertenece idCiudad(CIUDAD)

DISTRIBUIDOR_EMPRESA(ci, nombreEmpresa)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ciDistribuidor	varchar	PK, FK	50	NOT NULL
nombreEmpresa	varchar	FK, FK	50	NOT NULL

RNE

- ciDistribuidor(DISTRIBUIDOR_EMPRESA) pertenece ci(DISTRIBUIDOR)
- nombreEmpresa(DISTRIBUIDOR_EMPRESA) pertenece nombreEmpresa(EMPRESA)

FUNCIONARIO(ci, fechaNac, titulo, fechaAltaEmpresa, sueldo, comision nombreEmpresa, idProducto, ciJefe, idCargo, idCiudad)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ci	varchar	PK, FK	50	NOT NULL
fechaNac	date		11	
titulo	varchar		50	
fechaAlta	date		50	
sueldo	integer			
comision	integer			
nombreEmpresa	varchar	FK	50	
idProducto	varchar	FK	50	
ciJefe	varchar	FK	50	
idCargo	varchar	FK	50	
idCiudad	varchar	FK	50	
idArea	varchar	FK	50	

RNE

- ci(FUNCIONARIO) pertenece ci(PERSONA)
- nombreEmpresa(FUNCIONARIO) pertenece nombreEmpresa(EMPRESA)
- idProducto(FUNCIONARIO) pertenece idProducto(PRODUCTO)
- ciJefe(FUNCIONARIO) pertenece ci(FUNCIONARIO)
- idCargo(FUNCIONARIO) pertenece idCargo(CARGO)
- idCiudad(FUNCIONARIO) pertenece idCiudad(CIUDAD)
- idArea(FUNCIONARIO) pertenece idArea(AREADETRABAJO)

FUNCIONARIO_HIJO(ciFuncionario, ciHijo)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ciFuncionario	varchar	PK, FK	50	NOT NULL
ciHijo	varchar	PK, FK	50	NOT NULL

PRODUCTO_EMPRESA(idProducto, nombreEmpresa)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idProducto	varchar	PK, FK	50	NOT NULL
nombreEmpresa	varchar	PK, FK	50	NOT NULL

RNE

- idProducto(PRODUCTO_EMPRESA) pertenece idProducto(PRODUCTO)
- nombreEmpresa(PRODUCTO_EMPRESA) pertenece nombreEmpresa(EMPRESA)

HABILIDAD(idHabilidad, descripcion)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idHabilidad	varchar	PK	50	NOT NULL
descripcion	varchar		50	

FUNCIONARIO_HABILIDAD(ci, idHabilidad)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
ciFuncionario	varchar	PK, FK	50	NOT NULL
idHabilidad	varchar	PK, FK	50	NOT NULL

RNE

- ciFuncionario(FUNCIONARIO_HABILIDAD) pertenece ci(FUNCIONARIO)
- idHabilidad(FUNCIONARIO_HABILIDAD) pertenece idHabilidad(HABILIDAD)

TAREA(idTarea, nombre, descripción, ciFuncionario)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idTarea	varchar	PK	50	NOT NULL
nombre	varchar		50	NOT NULL
descripción	varchar		50	NOT NULL
ciFuncionario	varchar	FK	50	NOT NULL

RNE

- ciFuncionario(HABILIDAD) pertenece ci(FUNCIONARIO)

AREADETRABAJO(idArea, presupuestoAnual, nombre, nombreEmpresa)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idArea	varchar	PK	50	NOT NULL
presupuestoAnual	integer			
nombre	varchar		50	
nombreEmpresa	varchar	FK	50	

RNE

- nombreEmpresa(AREADETRABAJO) pertenece nombreEmpresa(EMPRESA)

LLAMADO(idLlamado, descripcion, fechaPosteo, fechaLimite, idArea)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idLlamado	varchar	PK	50	NOT NULL
descripcion	varchar		50	
fechaPosteo	date			
fechaLimite	integer			
idArea	varchar	FK	50	

RNE

- idArea(LLAMADO) pertenece a idArea(AREADETRABAJO)

PERSONAINTERESADA(idPersonaInteresada, fechaInscripcion, curriculumVitae)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idPersonaInteresada	varchar	PK	50	NOT NULL
fechaInscripcion	varchar		50	NOT NULL
curriculumVitae	bytea			NOT NULL

LLAMADO_PERSONAINTERESADA(idLlamado, idPersonaInteresada)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idLlamado	varchar	PK, FK	50	NOT NULL
idPersonaInteresada	varchar	PK, FK	50	NOT NULL

RNE

- idLlamado(LLAMADO_PERSONAINTERESADA) pertenece idLlamado(LLAMADO)
- idPersonaInteresada(LLAMADO_PERSONAINTERESADA) pertenece idLlamado(LLAMADO)

CARGO(idCargo, nombre)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idCargo	varchar	PK	50	NOT NULL
nombre	varchar		50	

CARGO_AREADETRABAJO(idCargo, idArea)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idCargo	varchar	PK, FK	50	NOT NULL
idArea	varchar	PK, FK	50	NOT NULL

RNE

- idArea(CARGO_AREADETRABAJO) pertenece a idArea(AREADETRABAJO)
- idCargo(CARGO_AREADETRABAJO) pertenece a idCargo(CARGO)

FORMAPAGO(idFormaPago, tipo)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idFormaPago	varchar	PK	50	NOT NULL
tipoFormaPago	varchar		50	

LINEACATALOGO(idLineaCatalogo, idPedido, cantComprada, precioCompra, idProducto)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idLineaCatalogo	varchar	PK	50	NOT NULL
idCatalogo	varchar	PK, FK	50	NOT NULL
cantComprada	integer		3	
precioCompra	float		10	
idProducto	varchar	FK	50	NOT NULL

RNE

- idCatalogo(LINEACATALOGO) pertenece idCatalogo(CATALOGO)
- idProducto(LINEACATALOGO) pertenece idProducto(PRODUCTO)

ESTADO(idEstado, nombreEstado, idCompra, ciCliente, idCatalogo)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idEstado	varchar	PK	50	NOT NULL
nombreEstado	varchar		50	NOT NULL
idCompra	varchar	FK	50	NOT NULL
ciCliente	varchar	FK	50	NOT NULL
idCatalogo	varchar	FK	50	NOT NULL

RNE

- idCompra(LINEACATALOGO) pertenece idCompra(CLIENTE_CATALOGO)
- ciCliente(LINEACATALOGO) pertenece ci(CLIENTE)
- idCatalogo(LINEACATALOGO) pertenece idCatalogo(CATALOGO)

RECLAMACIONCONSULTA(idReclamo, ciCliente, idCatalogo, idCompra, fechaAlta, fechaResolucion, tipo, contenido)

nombreAtributo	TipoCampo	Descripción	Tamaño	Restricción
idReclamo	varchar	PK	50	NOT NULL
fechaAlta	date			
fechaResolucion	date			
tipo	varchar		50	
contenido	varchar		500	
ciCliente	varchar	PK, FK		NOT NULL
idCatalogo	varchar	PK, FK	50	NOT NULL
idCompra	varchar	PK, FK	50	NOT NULL

RNE

- ciCliente(RECLAMACIONCONSULTA) pertenece a ci(CLIENTE_CATALOGO)
- idCatalogo(RECLAMACIONCONSULTA) pertenece a idCatalogo(CLIENTE_CATALOGO)
- idCompra(RECLAMACIONCONSULTA) pertenece a idCompra(CLIENTE_CATALOGO)

Restricciones no estructurales (RNE)

- Si la diferencia entre la fecha actual y la fechaCompra de la relación entre CLIENTE y CATÁLOGO es mayor a 3, el cliente no podrá realizar consultas.
- En RECLAMACIONCONSULTA, fechaAlta debe ser posterior a fechaResolución.
- En Llamado, fechaLímite debe ser posterior a fechaPosteo.
- En PERSONAINTERESADA, fechaInscripción debe ser posterior a fechaPosteo, y anterior a fechaLímite, del Llamado correspondiente.
- Un FUNCIONARIO no puede ser jefe de sí mismo.
- Los CATÁLOGO pueden ser web o físicos. Cuando se realiza una compra a través de un catálogo físico, la tabla ESTADO tendrá una tupla con nombre de estado “Retirado en local”.

Datos del Sistema de Gestión

Nombre: PostgreSQL

Versión: 16

Año: 2024

/*

SQL definición y manejo de datos

*/

```
-- Crear tabla FORMAPAGO
CREATE TABLE FORMAPAGO (
    idFormaPago VARCHAR(50) NOT NULL PRIMARY KEY,
    tipoFormaPago VARCHAR(50)
);

-- Crear tabla CARGO
CREATE TABLE CARGO (
    idCargo VARCHAR(50) NOT NULL PRIMARY KEY,
    nombre VARCHAR(50)
);

-- Crear tabla PAIS
CREATE TABLE PAIS (
    nombrePais VARCHAR(50) NOT NULL PRIMARY KEY
);

-- Crear tabla CIUDAD
CREATE TABLE CIUDAD (
    idCiudad VARCHAR(50) NOT NULL PRIMARY KEY,
    nombreCiudad VARCHAR(50) NOT NULL,
    codigoPostal VARCHAR(50) NOT NULL,
    nombrePais VARCHAR(50),
    CONSTRAINT fk_pais FOREIGN KEY (nombrePais) REFERENCES
PAIS(nombrePais)
);

-- Crear tabla EMPRESA
CREATE TABLE EMPRESA (
    nombreEmpresa VARCHAR(50) NOT NULL PRIMARY KEY,
    calle VARCHAR(50),
    numero VARCHAR(50),
    telefono VARCHAR(50),
    email VARCHAR(50),
    idCiudad VARCHAR(50),
```

```
    CONSTRAINT fk_ciudad FOREIGN KEY (idCiudad) REFERENCES
CIUDAD(idCiudad)
);

-- Crear tabla AREADETRABAJO
CREATE TABLE AREADETRABAJO (
    idArea VARCHAR(50) NOT NULL PRIMARY KEY,
    presupuestoAnual INTEGER,
    nombre VARCHAR(50),
    nombreEmpresa VARCHAR(50),
    CONSTRAINT fk_empresa FOREIGN KEY (nombreEmpresa)
REFERENCES EMPRESA(nombreEmpresa)
);

-- Crear tabla PRODUCTO
CREATE TABLE PRODUCTO (
    idProducto VARCHAR(50) NOT NULL PRIMARY KEY,
    cantStock INTEGER,
    cantStockMin INTEGER,
    nombreProducto VARCHAR(50),
    descripcion VARCHAR(50),
    tipoProducto VARCHAR(50),
    precioUnitario INTEGER
);

-- Crear tabla PERSONA
CREATE TABLE PERSONA (
    ci VARCHAR(50) NOT NULL PRIMARY KEY,
    nombrePersona VARCHAR(50),
    apellidoPersona VARCHAR(50),
    numero VARCHAR(50),
    calle VARCHAR(50),
    telefono VARCHAR(8),
    email VARCHAR(50)
);

-- Crear tabla HIJO
CREATE TABLE HIJO (
    ci VARCHAR(50) NOT NULL PRIMARY KEY,
    fechaNacimiento DATE,
    nombreHijo VARCHAR(50),
```

```
    apellidoHijo VARCHAR(50)
);

-- Crear tabla CLIENTE
CREATE TABLE CLIENTE (
    ci VARCHAR(50) NOT NULL PRIMARY KEY,
    fechaNacimiento DATE,
    idCiudad VARCHAR(50),
    CONSTRAINT fk_ciudad_cliente FOREIGN KEY (idCiudad) REFERENCES
CIUDAD(idCiudad),
    CONSTRAINT fk_persona_cliente FOREIGN KEY (ci) REFERENCES
PERSONA(ci)
);

-- Crear tabla CATALOGO
CREATE TABLE CATALOGO (
    idCatalogo VARCHAR(50) NOT NULL PRIMARY KEY,
    descripcion VARCHAR(50),
    porcentajeIVA INTEGER,
    idEmpresa VARCHAR(50) NOT NULL,
    CONSTRAINT fk_empresa_catalogo FOREIGN KEY (idEmpresa)
REFERENCES EMPRESA(nombreEmpresa)
);

-- Crear tabla CLIENTE_CATALOGO
CREATE TABLE CLIENTE_CATALOGO (
    idCatalogo VARCHAR(50) NOT NULL,
    ciCliente VARCHAR(50) NOT NULL,
    idCompra VARCHAR(50) NOT NULL,
    fechaCompra DATE,
    idFormaPago VARCHAR(50),
    PRIMARY KEY (idCatalogo, ciCliente, idCompra),
    CONSTRAINT fk_catalogo FOREIGN KEY (idCatalogo) REFERENCES
CATALOGO(idCatalogo),
    CONSTRAINT fk_cliente FOREIGN KEY (ciCliente) REFERENCES
CLIENTE(ci),
    CONSTRAINT fk_forma_pago FOREIGN KEY (idFormaPago)
REFERENCES FORMAPAGO(idFormaPago)
);

-- Crear tabla DISTRIBUIDOR
```

```
CREATE TABLE DISTRIBUIDOR (
    ci VARCHAR(50) NOT NULL PRIMARY KEY,
    CONSTRAINT fk_persona_distribuidor FOREIGN KEY (ci) REFERENCES
PERSONA(ci)
);

-- Crear tabla DISTRIBUIDOR_PRODUCTO
CREATE TABLE DISTRIBUIDOR_PRODUCTO (
    ciDistribuidor VARCHAR(50) NOT NULL,
    idProducto VARCHAR(50) NOT NULL,
    PRIMARY KEY (ciDistribuidor, idProducto),
    CONSTRAINT fk_distribuidor FOREIGN KEY (ciDistribuidor)
REFERENCES DISTRIBUIDOR(ci),
    CONSTRAINT fk_producto FOREIGN KEY (idProducto) REFERENCES
PRODUCTO(idProducto)
);

-- Crear tabla DISTRIBUIDOR_CIUDAD
CREATE TABLE DISTRIBUIDOR_CIUDAD (
    ciDistribuidor VARCHAR(50) NOT NULL,
    idCiudad VARCHAR(50) NOT NULL,
    PRIMARY KEY (ciDistribuidor, idCiudad),
    CONSTRAINT fk_distribuidor_ciudad FOREIGN KEY (ciDistribuidor)
REFERENCES DISTRIBUIDOR(ci),
    CONSTRAINT fk_ciudad_distribuidor FOREIGN KEY (idCiudad)
REFERENCES CIUDAD(idCiudad)
);

-- Crear tabla DISTRIBUIDOR_EMPRESA
CREATE TABLE DISTRIBUIDOR_EMPRESA (
    ciDistribuidor VARCHAR(50) NOT NULL,
    nombreEmpresa VARCHAR(50) NOT NULL,
    PRIMARY KEY (ciDistribuidor, nombreEmpresa),
    CONSTRAINT fk_distribuidor_empresa FOREIGN KEY (ciDistribuidor)
REFERENCES DISTRIBUIDOR(ci),
    CONSTRAINT fk_empresa_distribuidor FOREIGN KEY (nombreEmpresa)
REFERENCES EMPRESA(nombreEmpresa)
);

-- Crear tabla FUNCIONARIO
CREATE TABLE FUNCIONARIO (
```

```
    ci VARCHAR(50) NOT NULL PRIMARY KEY,
    fechaNac DATE,
    titulo VARCHAR(50),
    fechaAlta DATE,
    sueldo INTEGER,
    comision INTEGER,
    nombreEmpresa VARCHAR(50),
    idProducto VARCHAR(50),
    ciJefe VARCHAR(50),
    idCargo VARCHAR(50),
    idCiudad VARCHAR(50),
    idArea VARCHAR(50),
    CONSTRAINT fk_empresa_funcionario FOREIGN KEY (nombreEmpresa)
REFERENCES EMPRESA(nombreEmpresa),
    CONSTRAINT fk_producto_funcionario FOREIGN KEY (idProducto)
REFERENCES PRODUCTO(idProducto),
    CONSTRAINT fk_jefe FOREIGN KEY (ciJefe) REFERENCES
FUNCIONARIO(ci),
    CONSTRAINT fk_persona_funcionario FOREIGN KEY (ci) REFERENCES
PERSONA(ci),
    CONSTRAINT fk_cargo_funcionario FOREIGN KEY (idCargo)
REFERENCES CARGO(idCargo),
    CONSTRAINT fk_ciudad_funcionario FOREIGN KEY (idCiudad)
REFERENCES CIUDAD(idCiudad),
    CONSTRAINT fk_area_funcionario FOREIGN KEY (idArea) REFERENCES
AREADETRABAJO(idArea)
);

-- Crear tabla FUNCIONARIO_HIJO
CREATE TABLE FUNCIONARIO_HIJO (
    ciFuncionario VARCHAR(50) NOT NULL,
    ciHijo VARCHAR(50) NOT NULL,
    PRIMARY KEY (ciFuncionario, ciHijo),
    CONSTRAINT fk_funcionario_hijo FOREIGN KEY (ciFuncionario)
REFERENCES FUNCIONARIO(ci),
    CONSTRAINT fk_hijo_funcionario FOREIGN KEY (ciHijo) REFERENCES
HIJO(ci)
);

-- Crear tabla PRODUCTO_EMPRESA
CREATE TABLE PRODUCTO_EMPRESA (
```

```
    idProducto VARCHAR(50) NOT NULL,
    nombreEmpresa VARCHAR(50) NOT NULL,
    PRIMARY KEY (idProducto, nombreEmpresa),
    CONSTRAINT fk_producto_empresa FOREIGN KEY (idProducto)
REFERENCES PRODUCTO(idProducto),
    CONSTRAINT fk_empresa_producto FOREIGN KEY (nombreEmpresa)
REFERENCES EMPRESA(nombreEmpresa)
);

-- Crear tabla HABILIDAD
CREATE TABLE HABILIDAD (
    idHabilidad VARCHAR(50) NOT NULL PRIMARY KEY,
    descripcion VARCHAR(50)
);

-- Crear tabla FUNCIONARIO_HABILIDAD
CREATE TABLE FUNCIONARIO_HABILIDAD (
    ciFuncionario VARCHAR(50) NOT NULL,
    idHabilidad VARCHAR(50) NOT NULL,
    PRIMARY KEY (ciFuncionario, idHabilidad),
    CONSTRAINT fk_funcionario_habilidad FOREIGN KEY (ciFuncionario)
REFERENCES FUNCIONARIO(ci),
    CONSTRAINT fk_habilidad_funcionario FOREIGN KEY (idHabilidad)
REFERENCES HABILIDAD(idHabilidad)
);

-- Crear tabla TAREA
CREATE TABLE TAREA (
    idTarea VARCHAR(50) NOT NULL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    descripcion VARCHAR(50) NOT NULL,
    ciFuncionario VARCHAR(50) NOT NULL,
    CONSTRAINT fk_funcionario_tarea FOREIGN KEY (ciFuncionario)
REFERENCES FUNCIONARIO(ci)
);

-- Crear tabla LLAMADO
CREATE TABLE LLAMADO (
    idLlamado VARCHAR(50) NOT NULL PRIMARY KEY,
    descripcion VARCHAR(50),
    fechaPosteo DATE,
```

```
    fechaLimite DATE,
    idArea VARCHAR(50),
    CONSTRAINT fk_area_llamado FOREIGN KEY (idArea) REFERENCES
AREADETRABAJO(idArea)
);

-- Crear tabla PERSONAINTERESADA
CREATE TABLE PERSONAINTERESADA (
    idPersonaInteresada VARCHAR(50) NOT NULL PRIMARY KEY,
    fechaInscripcion DATE NOT NULL,
    curriculumVitae BYTEA NOT NULL
);

-- Crear tabla LLAMADO_PERSONAINTERESADA
CREATE TABLE LLAMADO_PERSONAINTERESADA (
    idLlamado VARCHAR(50) NOT NULL,
    idPersonaInteresada VARCHAR(50) NOT NULL,
    PRIMARY KEY (idLlamado, idPersonaInteresada),
    CONSTRAINT fk_llamado_persona_interesada FOREIGN KEY
(idLlamado) REFERENCES LLAMADO(idLlamado),
    CONSTRAINT fk_persona_interesada_llamado FOREIGN KEY
(idPersonaInteresada) REFERENCES
PERSONAINTERESADA(idPersonaInteresada)
);

-- Crear tabla CARGO_AREADETRABAJO
CREATE TABLE CARGO_AREADETRABAJO (
    idCargo VARCHAR(50) NOT NULL,
    idArea VARCHAR(50) NOT NULL,
    PRIMARY KEY (idCargo, idArea),
    CONSTRAINT fk_cargo_area FOREIGN KEY (idCargo) REFERENCES
CARGO(idCargo),
    CONSTRAINT fk_area_cargo FOREIGN KEY (idArea) REFERENCES
AREADETRABAJO(idArea)
);

-- Crear tabla LINEACATALOGO
CREATE TABLE LINEACATALOGO (
    idLineaCatalogo VARCHAR(50) NOT NULL PRIMARY KEY,
    idCatalogo VARCHAR(50) NOT NULL,
    cantComprada INTEGER,
```

```
    precioCompra FLOAT,
    idProducto VARCHAR(50) NOT NULL,
    CONSTRAINT fk_catalogo_linea FOREIGN KEY (idCatalogo)
REFERENCES CATALOGO(idCatalogo),
    CONSTRAINT fk_producto_linea FOREIGN KEY (idProducto)
REFERENCES PRODUCTO(idProducto)
);

-- Crear tabla ESTADO
CREATE TABLE ESTADO (
    idEstado VARCHAR(50) NOT NULL PRIMARY KEY,
    nombreEstado VARCHAR(50) NOT NULL,
    idCompra VARCHAR(50) NOT NULL,
    ciCliente VARCHAR(50) NOT NULL,
    idCatalogo VARCHAR(50) NOT NULL,
    CONSTRAINT fk_estado_cliente_catalogo FOREIGN KEY (ciCliente,
idCatalogo, idCompra) REFERENCES CLIENTE_CATALOGO(ciCliente,
idCatalogo, idCompra)
);

-- Crear tabla RECLAMACIONCONSULTA
CREATE TABLE RECLAMACIONCONSULTA (
    idReclamo VARCHAR(50) NOT NULL PRIMARY KEY,
    fechaAlta DATE,
    fechaResolucion DATE,
    tipo VARCHAR(50),
    contenido VARCHAR(500),
    ciCliente VARCHAR(50) NOT NULL,
    idCatalogo VARCHAR(50) NOT NULL,
    idCompra VARCHAR(50) NOT NULL,
    CONSTRAINT fk_cliente_catalogo_reclamo FOREIGN KEY (ciCliente,
idCatalogo, idCompra) REFERENCES CLIENTE_CATALOGO(ciCliente,
idCatalogo, idCompra)
);
```

```
-- Insertar datos en FORMAPAGO
INSERT INTO FORMAPAGO (idFormaPago, tipoFormaPago) VALUES
('FP001', 'Tarjeta de Crédito'),
('FP002', 'Efectivo'),
('FP003', 'Transferencia Bancaria');

-- Insertar datos en CARGO
INSERT INTO CARGO (idCargo, nombre) VALUES
('C001', 'Gerente'),
('C002', 'Asistente'),
('C003', 'Desarrollador'),
('C004', 'Analista'),
('C005', 'Técnico'),
('C006', 'Consultor'),
('C007', 'Supervisor'),
('C008', 'Operario'),
('C009', 'Director'),
('C010', 'Tester'),
('C011', 'Administrativo');

INSERT INTO PAIS (nombrePais) VALUES
('Uruguay'),
('Argentina'),
('Brasil'),
('Chile'),
('Paraguay'),
('Perú'),
('Bolivia'),
('Ecuador'),
('Colombia'),
('Venezuela');

-- Insertar datos en CIUDAD
INSERT INTO CIUDAD (idCiudad, nombreCiudad, codigoPostal,
nombrePais) VALUES
('C001', 'Montevideo', '11000', 'Uruguay'),
('C002', 'Buenos Aires', '1000', 'Argentina'),
('C003', 'São Paulo', '01000', 'Brasil'),
('C004', 'Santiago', '8320000', 'Chile'),
('C005', 'Asunción', '1209', 'Paraguay'),
```

```
('C006', 'Lima', '15000', 'Perú'),
('C007', 'La Paz', '2000', 'Bolivia'),
('C008', 'Quito', '17000', 'Ecuador'),
('C009', 'Bogotá', '110111', 'Colombia'),
('C010', 'Caracas', '1010', 'Venezuela'),
('C011', 'Maldonado', '91350', 'Uruguay'),
('C012', 'Colonia del Sacramento', '7500', 'Uruguay'),
('C013', 'Minas', '8340', 'Uruguay');

-- Insertar datos en EMPRESA
INSERT INTO EMPRESA (nombreEmpresa, calle, numero, telefono, email,
idCiudad) VALUES
('Empresa A', 'Calle 1', '123', '12345678', 'empresaA@example.com',
'C001'),
('Empresa B', 'Calle 2', '456', '87654321', 'empresaB@example.com',
'C002'),
('Empresa C', 'Calle 3', '789', '12348765', 'empresaC@example.com',
'C003'),
('Empresa D', 'Calle 4', '321', '23456789', 'empresaD@example.com',
'C004'),
('Empresa E', 'Calle 5', '654', '98765432', 'empresaE@example.com',
'C005'),
('Empresa F', 'Calle 6', '987', '34567890', 'empresaF@example.com',
'C006'),
('Empresa G', 'Calle 7', '147', '45678901', 'empresaG@example.com',
'C007'),
('Empresa H', 'Calle 8', '258', '56789012', 'empresaH@example.com',
'C008'),
('Empresa I', 'Calle 9', '369', '67890123', 'empresaI@example.com',
'C009'),
('Empresa J', 'Calle 10', '741', '78901234', 'empresaJ@example.com',
'C010'),
('Empresa W', 'Calle 11', 543, '12341323', 'empresaW@example.com',
'C011'),
('Empresa K', 'Calle 12', 398, '95472064', 'empresaK@example.com',
'C011'),
('Empresa L', 'Calle 13', 195, '94876403', 'empresaL@example.com',
'C012'),
('Empresa M', 'Calle 14', 307, '74315609', 'empresaM@example.com',
'C013');
```

```
-- Insertar datos en AREADETRABAJO
INSERT INTO AREADETRABAJO (idArea, presupuestoAnual, nombre,
nombreEmpresa) VALUES
('A001', 100000, 'Área de Ventas', 'Empresa A'),
('A002', 200000, 'Área de Desarrollo', 'Empresa B'),
('A003', 150000, 'Área de Marketing', 'Empresa C'),
('A004', 120000, 'Área de Recursos Humanos', 'Empresa D'),
('A005', 130000, 'Área de Finanzas', 'Empresa E'),
('A006', 140000, 'Contabilidad', 'Empresa F'),
('A007', 110000, 'Área de Investigación', 'Empresa G'),
('A008', 170000, 'Área de Producción', 'Empresa H'),
('A009', 160000, 'Marketing', 'Empresa I'),
('A010', 180000, 'SQA', 'Empresa J'),
('A011', 130000, 'Área de Prueba', 'Empresa W'),
('1404', 185000, 'Área de Infraestructura', 'Empresa B'),
('A012', 200000, 'Área de Desarrollo', 'Empresa K'),
('A013', 110000, 'Área de Investigación', 'Empresa L'),
('A014', 140000, 'Área de Ventas', 'Empresa M');

-- Insertar datos en PRODUCTO
INSERT INTO PRODUCTO (idProducto, cantStock, cantStockMin,
nombreProducto, descripcion, tipoProducto, precioUnitario) VALUES
('P001', 50, 5, 'Producto 1', 'Descripción 1', 'Tipo A', 100),
('P002', 100, 10, 'Producto 2', 'Descripción 2', 'Tipo B', 200),
('P003', 75, 7, 'Producto 3', 'Descripción 3', 'Tipo C', 150);

-- Insertar datos en PERSONA
INSERT INTO PERSONA (ci, nombrePersona, apellidoPersona, numero,
calle, telefono, email) VALUES
('CI001', 'Juan', 'Pérez', '101', 'Calle A', '12345678',
'juan.perez@example.com'),
('CI002', 'María', 'Gómez', '202', 'Calle B', '87654321',
'maria.gomez@example.com'),
('CI003', 'Luis', 'Rodríguez', '303', 'Calle C', '12348765',
'luis.rodriguez@example.com'),
('CI004', 'Ana', 'Fernández', '404', 'Calle D', '23456789',
'ana.fernandez@example.com'),
('CI005', 'Carlos', 'López', '505', 'Calle E', '98765432',
'carlos.lopez@example.com'),
('CI006', 'Laura', 'Martínez', '606', 'Calle F', '34567890',
'laura.martinez@example.com'),
```

```
('CI007', 'Pedro', 'González', '707', 'Calle G', '45678901',
'pedro.gonzalez@example.com'),
('CI008', 'Lucía', 'Sánchez', '808', 'Calle H', '56789012',
'lucia.sanchez@example.com'),
('CI009', 'Jorge', 'Díaz', '909', 'Calle I', '67890123',
'jorge.diaz@example.com'),
('CI010', 'Sofía', 'Pereira', '1001', 'Calle J', '78901234',
'sofia.pereira@example.com'),
('CI011', 'Salvador', 'Veintemilla', '3243', 'Calle K', '78902334',
'salvador.veintemilla@example.com'),
('CI012', 'Juan Pedro', 'Ramirez', 4231, 'Calle L', '32456734',
'juanpablo.ramirez@example.com'),
('CI013', 'Juan Pedro', 'Lopez', '9452', 'Calle M', '75923041',
'juanpablo.lopez@example.com'),
('CI014', 'Luca', 'Prodan', '7381', 'Calle O', '83094812',
'luca.prodan@example.com'),
('CI015', 'Jeff', 'Porcaro', '9067', 'Calle P', '50392836',
'jeff.porcaro@example.com'),
('CI016', 'Mick', 'Jagger', '4825', 'Calle Q', '94724587',
'mick.jagger@example.com'),
('CI017', 'George', 'Harrison', '9043', 'Calle R', '15098436',
'george.harrison@example.com'),
('CI018', 'Carlos', 'Gil', '2343', 'Calle G', '42343233',
'carlos.gil@example.com');

-- Insertar datos en HIJO
INSERT INTO HIJO (ci, fechaNacimiento, nombreHijo, apellidoHijo)
VALUES
('CI001H1', '2010-01-01', 'Pedro', 'Pérez'),
('CI002H1', '2012-02-02', 'Ana', 'Gómez'),
('CI003H1', '2015-03-03', 'Carlos', 'Rodríguez'),
('CI004H1', '2011-04-04', 'Luisa', 'Fernández'),
('CI005H1', '2013-05-05', 'Jorge', 'López'),
('CI006H1', '2016-06-06', 'María', 'Martínez'),
('CI007H1', '2017-07-07', 'Sofía', 'González'),
('CI008H1', '2014-08-08', 'Pablo', 'Sánchez'),
('CI009H1', '2018-09-09', 'Lucía', 'Díaz'),
('CI010H1', '2019-10-10', 'Miguel', 'Pereira');

-- Insertar datos en CLIENTE
INSERT INTO CLIENTE (ci, fechaNacimiento, idCiudad) VALUES
```

```
('CI001', '1980-01-01', 'C001'),
('CI002', '1990-02-02', 'C002'),
('CI003', '1985-03-03', 'C003');

-- Insertar datos en CATALOGO
INSERT INTO CATALOGO (idCatalogo, descripcion, porcentajeIVA,
idEmpresa) VALUES
('CAT001', 'Catálogo 1', 10, 'Empresa A'),
('CAT002', 'Catálogo 2', 15, 'Empresa B'),
('CAT003', 'Catálogo 3', 20, 'Empresa C');

-- Insertar datos en CLIENTE_CATALOGO
INSERT INTO CLIENTE_CATALOGO (idCatalogo, ciCliente, idCompra,
fechaCompra, idFormaPago) VALUES
('CAT001', 'CI001', 'COMP001', '2024-01-01', 'FP001'),
('CAT002', 'CI002', 'COMP002', '2024-02-01', 'FP002'),
('CAT003', 'CI003', 'COMP003', '2024-03-01', 'FP003');

-- Insertar datos en DISTRIBUIDOR
INSERT INTO DISTRIBUIDOR (ci) VALUES
('CI001'),
('CI002'),
('CI003');

-- Insertar datos en DISTRIBUIDOR_PRODUCTO
INSERT INTO DISTRIBUIDOR_PRODUCTO (ciDistribuidor, idProducto)
VALUES
('CI001', 'P001'),
('CI002', 'P002'),
('CI003', 'P003');

-- Insertar datos en DISTRIBUIDOR_CIUDAD
INSERT INTO DISTRIBUIDOR_CIUDAD (ciDistribuidor, idCiudad) VALUES
('CI001', 'C001'),
('CI002', 'C002'),
('CI003', 'C003');

-- Insertar datos en DISTRIBUIDOR_EMPRESA
INSERT INTO DISTRIBUIDOR_EMPRESA (ciDistribuidor, nombreEmpresa)
VALUES
('CI001', 'Empresa A'),
```

```
('CI002', 'Empresa B'),
('CI003', 'Empresa C');

-- Insertar datos en FUNCIONARIO
INSERT INTO FUNCIONARIO (ci, fechaNac, titulo, fechaAlta, sueldo,
comision, nombreEmpresa, idProducto, ciJefe, idCargo, idCiudad, idArea)
VALUES
('CI001', '1980-01-01', 'Licenciado', '2020-01-01', 50000, 5000, 'Empresa
A', 'P001', NULL, 'C001', 'C001', 'A001'),
('CI002', '1990-02-02', 'Ingeniero', '2021-01-01', 60000, 6000, 'Empresa
B', 'P002', 'CI001', 'C002', 'C002', 'A002'),
('CI003', '1985-03-03', 'Doctor', '2022-01-01', 70000, 7000, 'Empresa C',
'P003', 'CI002', 'C003', 'C003', 'A003'),
('CI004', '1988-04-04', 'Master', '2019-01-01', 55000, 5500, 'Empresa D',
'P001', 'CI003', 'C004', 'C004', 'A004'),
('CI005', '1992-05-05', 'Técnico', '2018-01-01', 45000, 4500, 'Empresa
E', 'P002', 'CI004', 'C005', 'C005', 'A005'),
('CI006', '1983-06-06', 'Licenciado', '2017-01-01', 50000, 5000, 'Empresa
F', 'P003', 'CI005', 'C006', 'C006', 'A006'),
('CI007', '1995-07-07', 'Ingeniero', '2016-01-01', 60000, 7000, 'Empresa
G', 'P001', 'CI006', 'C007', 'C007', 'A007'),
('CI008', '1986-08-08', 'Doctor', '2015-01-01', 70000, 7000, 'Empresa H',
'P002', 'CI007', 'C008', 'C008', 'A008'),
('CI009', '1993-09-09', 'Master', '2014-01-01', 55000, 5500, 'Empresa I',
'P003', 'CI008', 'C009', 'C009', 'A009'),
('CI010', '1984-10-10', 'Técnico', '2013-01-01', 45000, 4500, 'Empresa J',
'P001', 'CI009', 'C010', 'C010', 'A010'),
('CI011', '2004-10-10', 'Técnico', '2015-01-01', 100000, 10000, 'Empresa
J', 'P001', 'CI009', 'C010', 'C010', 'A010'),
('CI012', '1978-04-11', 'Ingeniero', '2002-01-01', 120000, 12000,
'Empresa J', 'P002', 'CI003', 'C003', 'C010', 'A010'),
('CI013', '1989-07-03', 'Licenciado', '2007-01-01', 65000, 6500, 'Empresa
B', 'P003', 'CI004', 'C007', 'C002', '1404'),
('CI014', '1983-04-12', 'Master', '1997-08-02', 80000, 8000, 'Empresa J',
'P002', 'CI012', 'C008', 'C010', 'A010'),
('CI015', '1969-02-06', 'Doctor', '1995-04-11', 110000, 11000, 'Empresa
K', 'P001', 'CI009', 'C006', 'C011', 'A012'),
('CI016', '1975-12-01', 'Tecnico', '1999-12-01', 90000, 9000, 'Empresa L',
'P002', 'CI001', 'C002', 'C012', null),
('CI017', '1982-08-10', 'Doctor', '2015-11-08', 50000, 11000, 'Empresa
M', 'P002', 'CI007', 'C006', 'C013', 'A014'),
```

```
('CI018', '1932-08-10', 'Doctor', '2015-01-08', 50000, 6000, 'Empresa J',
'P002', 'CI007', 'C011', 'C013', 'A010');


-- Insertar datos en FUNCIONARIO_HIJO
INSERT INTO FUNCIONARIO_HIJO (ciFuncionario, ciHijo) VALUES
('CI001', 'CI001H1'),
('CI002', 'CI002H1'),
('CI003', 'CI003H1');

-- Insertar datos en PRODUCTO_EMPRESA
INSERT INTO PRODUCTO_EMPRESA (idProducto, nombreEmpresa)
VALUES
('P001', 'Empresa A'),
('P002', 'Empresa B'),
('P003', 'Empresa C');

INSERT INTO HABILIDAD (idHabilidad, descripcion) VALUES
('HAB001', 'Programación'), ('HAB002', 'Marketing'),
('HAB003', 'Ventas'), ('HAB004', 'Gestión de Proyectos'),
('HAB005', 'Diseño Gráfico'), ('HAB006', 'Soporte Técnico'),
('HAB007', 'Análisis de Datos'), ('HAB008', 'Redes Sociales'),
('HAB009', 'Servicio al Cliente'), ('HAB010', 'Finanzas');

-- Insertar datos en FUNCIONARIO_HABILIDAD
INSERT INTO FUNCIONARIO_HABILIDAD (ciFuncionario, idHabilidad)
VALUES
('CI001', 'HAB001'),
('CI002', 'HAB002'),
('CI003', 'HAB003');

INSERT INTO TAREA (idTarea, nombre, descripcion, ciFuncionario)
VALUES
('TAR001', 'Tarea 1', 'Descripción de tarea 1', 'CI001'),
('TAR002', 'Tarea 2', 'Descripción de tarea 2', 'CI002'),
('TAR003', 'Tarea 3', 'Descripción de tarea 3', 'CI003'),
('TAR004', 'Tarea 4', 'Descripción de tarea 4', 'CI004'),
('TAR005', 'Tarea 5', 'Descripción de tarea 5', 'CI005'),
('TAR006', 'Tarea 6', 'Descripción de tarea 6', 'CI006'),
('TAR007', 'Tarea 7', 'Descripción de tarea 7', 'CI007'),
('TAR008', 'Tarea 8', 'Descripción de tarea 8', 'CI008'),
```

```
('TAR009', 'Tarea 9', 'Descripción de tarea 9', 'CI009'),
('TAR010', 'Tarea 10', 'Descripción de tarea 10', 'CI010');

INSERT INTO LLAMADO (idLlamado, descripcion, fechaPosteo,
fechaLimite, idArea) VALUES
('LL001', 'Llamado 1', '2024-01-01', '2024-01-31', 'A001'),
('LL002', 'Llamado 2', '2024-02-01', '2024-02-28', 'A002'),
('LL003', 'Llamado 3', '2024-03-01', '2024-03-31', 'A003'),
('LL004', 'Llamado 4', '2024-04-01', '2024-04-30', 'A004'),
('LL005', 'Llamado 5', '2024-05-01', '2024-05-31', 'A005'),
('LL006', 'Llamado 6', '2024-06-01', '2024-06-30', 'A006'),
('LL007', 'Llamado 7', '2024-07-01', '2024-07-31', 'A007'),
('LL008', 'Llamado 8', '2024-08-01', '2024-08-31', 'A008'),
('LL009', 'Llamado 9', '2024-09-01', '2024-09-30', 'A009'),
('LL010', 'Llamado 10', '2024-10-01', '2024-10-31', 'A010');

INSERT INTO PERSONAINTERESADA (idPersonaInteresada,
fechaInscripcion, curriculumVitae) VALUES
('PI001', '2024-01-01', decode('48656c6c6f', 'hex')),
('PI002', '2024-02-01', decode('576f726c64', 'hex')),
('PI003', '2024-03-01', decode('50726573746f', 'hex')),
('PI004', '2024-04-01', decode('4a617661', 'hex')),
('PI005', '2024-05-01', decode('435353', 'hex')),
('PI006', '2024-06-01', decode('506974686f6e', 'hex')),
('PI007', '2024-07-01', decode('536f667477617265', 'hex')),
('PI008', '2024-08-01', decode('5265616374', 'hex')),
('PI009', '2024-09-01', decode('486f746d61696c', 'hex')),
('PI010', '2024-10-01', decode('4d617468', 'hex'));

-- Insertar datos en LLAMADO_PERSONAINTERESADA
INSERT INTO LLAMADO_PERSONAINTERESADA (idLlamado,
idPersonaInteresada) VALUES
('LL001', 'PI001'),
('LL002', 'PI002'),
('LL003', 'PI003');

-- Insertar datos en CARGO_AREADETRABAJO
INSERT INTO CARGO_AREADETRABAJO (idCargo, idArea) VALUES
('C001', 'A001'),
('C002', 'A002'),
('C003', 'A003'),
```

```
('C004', 'A004'),
('C005', 'A005'),
('C006', 'A006'),
('C007', 'A007'),
('C008', 'A008'),
('C009', 'A010'),
('C010', 'A009'),
('C011', 'A009');

-- Insertar datos en LINEACATALOGO
INSERT INTO LINEACATALOGO (idLineaCatalogo, idCatalogo,
cantComprada, precioCompra, idProducto) VALUES
('LC001', 'CAT001', 10, 1000.00, 'P001'),
('LC002', 'CAT002', 20, 2000.00, 'P002'),
('LC003', 'CAT003', 30, 3000.00, 'P003');

-- Insertar datos en ESTADO
INSERT INTO ESTADO (idEstado, nombreEstado, idCompra, ciCliente,
idCatalogo) VALUES
('E001', 'Pendiente', 'COMP001', 'CI001', 'CAT001'),
('E002', 'Completado', 'COMP002', 'CI002', 'CAT002'),
('E003', 'En Proceso', 'COMP003', 'CI003', 'CAT003');

-- Insertar datos en RECLAMACIONCONSULTA
INSERT INTO RECLAMACIONCONSULTA (idReclamo, fechaAlta,
fechaResolucion, tipo, contenido, ciCliente, idCatalogo, idCompra) VALUES
('R001', '2024-01-15', '2024-01-20', 'Reclamo', 'Contenido del reclamo 1',
'CI001', 'CAT001', 'COMP001'),
('R002', '2024-02-15', '2024-02-20', 'Consulta', 'Contenido del consulta
2', 'CI002', 'CAT002', 'COMP002'),
('R003', '2024-03-15', '2024-03-20', 'Reclamo', 'Contenido del reclamo 3',
'CI003', 'CAT003', 'COMP003');
```

SQL Resolución de consultas y AR

**/*
01) Mostrar la lista de los funcionarios (nombre, apellido, cargo), que tienen el mismo cargo que "Juan Pedro". Puede haber más de un funcionario con nombre "Juan Pedro".
SUBCONSULTA
*/**

```
SELECT pe.nombrepersona, pe.apellidopersona, ca.nombre
FROM funcionario f
JOIN persona pe ON f.ci = pe.ci
JOIN cargo ca ON f.idcargo = ca.idcargo
WHERE f.idcargo IN (
        SELECT idcargo
        FROM funcionario f
        JOIN PERSONA pe ON pe.ci = f.ci
        WHERE pe.nombrepersona = 'Juan Pedro'
);
```

R1 ← π funcionario.idcargo
(σ persona.nombrepersona = 'Juan Pedro')
(funcionario ⋈ persona)

π (persona.nombrepersona, persona.apellidopersona, cargo.nombre
(σ funcionario.idcargo IN (R1))
(funcionario ⋈ persona ⋈ cargo))

**/*
02) Mostrar la lista de los funcionarios (nombre, apellido, sueldo, número de área de trabajo), que ganan igual o más, que el máximo sueldo, de los funcionarios del área de trabajo número "1404". SUBCONSULTA
*/**

```
SELECT p.nombrepersona, p.apellidopersona, f.sueldo, f.idarea
FROM persona p
JOIN funcionario f ON p.ci = f.ci
WHERE f.sueldo >= (
        SELECT MAX(sueldo)
        FROM funcionario
        WHERE idarea = '1404'
);
```

```
R1 ← π G MAX(sueldo)
        (σ idarea = '1404')
        (funcionario)

π (persona.nombrepersona, persona.apellidopersona, funcionario.sueldo,
funcionario.idarea
(σ funcionario.sueldo >= (R1))
(persona ⋈ funcionario))
```

**/*
03) Para cada cargo, mostrar la suma total de los sueldos, de los funcionarios del área de trabajo de nombre "SQA". Presentar cargo y la suma total como sumaSueldos.
SUBCONSULTA
*/**

```
SELECT SUM(f.sueldo) AS sumaSueldos, c.nombre AS nombreCargo
FROM funcionario f
JOIN cargo c ON f.idCargo = c.idCargo
WHERE f.idArea IN (
        SELECT idArea
        FROM areadetrabajo
        WHERE nombre = 'SQA'
) GROUP BY f.idcargo, c.nombre;
```

```
R1 ← π idArea
        (σ nombre = 'SQA')
        (areadetrabajo)

π ({funcionario.idcargo, cargo.nombre} G SUM(f.sueldo) AS
sumaSueldos, cargo.nombre AS nombreCargo
(σ funcionario.idArea IN (R1))
(funcionario ⋈ cargo))
```

**/*
04) Presentar la cantidad total de funcionarios, con cargo "Tester", del área de trabajo de nombre "SQA". Presentar la cantidad total como totalTesters. SUBCONSULTA
*/**

```
SELECT COUNT(*) AS totalTesters
FROM funcionario f
WHERE f.idCargo IN (
        SELECT idCargo
        FROM cargo
        WHERE nombre = 'Tester'
) AND f.idArea IN (
        SELECT idArea
        FROM areadetrabajo
        WHERE nombre = 'SQA'
);
```

```
R1 ← π idCargo
        (σ nombre = 'Tester')
        (cargo)

R2 ← π idArea
        (σ nombre = 'SQA')
        (areadetrabajo)

π (G COUNT(funcionario.ci) AS totalTesters
(σ funcionario.idCargo IN (R1) Y funcionario.idArea IN (R2))
(funcionario))
```

**/*
05) Presentar el cargo, el sueldo máximo y el sueldo mínimo, de cada grupo de cargos, pero sólo para aquellos grupos, con más de dos tuplas (registros) y con un sueldo máximo mayor o igual a 1404.
*/**

```
SELECT c.nombre, MAX(f.sueldo) AS mayorSueldo, MIN(f.sueldo) AS
menorSueldo
FROM funcionario f
JOIN cargo c ON f.idcargo = c.idcargo
GROUP BY f.idcargo, c.nombre
HAVING MAX(f.sueldo) > 1404 AND COUNT (*) > 2;
```

```
π ({funcionario.idcargo, c.nombre} G MAX(funcionario.sueldo) AS
mayorSueldo, {funcionario.idcargo, c.nombre} G MIN(funcionario.sueldo)
AS menorSueldo, cargo.nombre
(σ {funcionario.idcargo, c.nombre} G MAX(funcionario.sueldo) > 1404 Y
{funcionario.idcargo, c.nombre} G COUNT(funcionario.ci) > 2)
(funcionario ⋈ cargo))
```

**/*
06) Presentar la lista de funcionarios (cedula de identidad, nombre, apellido, título, sueldo), cuyo sueldo supere, el sueldo medio de todos los funcionarios. SUBCONSULTA
*/**

```
SELECT p.ci, p.nombrepersona, p.apellidopersona, f.titulo, f.sueldo
FROM persona p
JOIN funcionario f ON p.ci = f.ci
WHERE f.sueldo > (
        SELECT AVG(sueldo)
        FROM funcionario
);
```

```
R1 ← π (G AVG(sueldo)
        (funcionario))
```

```
π (persona.ci, persona.nombrepersona, persona.apellidopersona,
funcionario.titulo, funcionario.sueldo
(σ funcionario.sueldo > (R1))
(persona ⋈ funcionario))
```

**/*
07) Mostrar la lista de los funcionarios cuyo sueldo, es mayor, que el de los "Supervisores" o "Directores". Presentar nombre, apellido, sueldo y número de área de trabajo del funcionario.
SUBCONSULTA
*/**

```
SELECT p.nombrepersona, p.apellidopersona, f.sueldo, f.idarea
FROM persona p
JOIN funcionario f ON p.ci = f.ci
WHERE f.sueldo > (
        SELECT MAX(sueldo)
        FROM funcionario f
        JOIN cargo c ON f.idcargo = c.idcargo
        WHERE c.nombre = 'Supervisor'
)
OR f.sueldo > (
        SELECT MAX(sueldo)
        FROM funcionario f
        JOIN cargo c ON f.idcargo = c.idcargo
        WHERE c.nombre = 'Director'
);
```

MAX_SUELDO_SUPERVISOR ← π (G MAX(sueldo)
(σ cargo.nombre = 'Supervisor')
(funcionario ⋈ cargo))

MAX_SUELDO_DIRECTOR ← π (G MAX(sueldo)
(σ cargo.nombre = 'Director')
(funcionario ⋈ cargo))

π (persona.nombrepersona, persona.apellidopersona, funcionario.sueldo, funcionario.idarea
(σ funcionario.sueldo > (MAX_SUELDO_SUPERVISOR) O
funcionario.sueldo > (MAX_SUELDO_DIRECTOR))
(persona ⋈ funcionario))

**/*
08) Presentar la lista de funcionarios, con cargo "Tester", cuyo sueldo es mayor, que el de los "Supervisores" o "Directores". Presentar nombre, apellido, sueldo y número de área de trabajo del funcionario. SUBCONSULTA
*/**

```
SELECT p.nombrepersona, p.apellidopersona, f.sueldo, f.idarea
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo c ON f.idcargo = c.idcargo
WHERE c.nombre = 'Tester'
AND (f.sueldo > (
        SELECT MAX(sueldo)
        FROM funcionario f
        JOIN cargo c ON f.idcargo = c.idcargo
        WHERE c.nombre = 'Supervisor'
)
OR f.sueldo > (
        SELECT MAX(sueldo)
        FROM funcionario f
        JOIN cargo c ON f.idcargo = c.idcargo
        WHERE c.nombre = 'Director'
));
```

MAX_SUELDO_SUPERVISOR ← π (G MAX(sueldo)
(σ cargo.nombre = 'Supervisor')
(funcionario ⋈ cargo))

MAX_SUELDO_DIRECTOR ← π (G MAX(sueldo)
(σ cargo.nombre = 'Director')
(funcionario ⋈ cargo))

π (persona.nombrepersona, persona.apellidopersona, funcionario.sueldo, funcionario.idarea
(σ cargo.nombre = 'Tester' Y (funcionario.sueldo >
(MAX_SUELDO_SUPERVISOR) O funcionario.sueldo >
(MAX_SUELDO_DIRECTOR)))
(persona ⋈ funcionario ⋈ cargo))

**/*
09) Listar, en orden alfabético ascendente por nombre, a los funcionarios que NO trabajen ni en la ciudad de "Montevideo" ni en "Maldonado". Presentar nombre, apellido y número de área de trabajo. SUBCONSULTA
*/**

```
SELECT p.nombrePersona, p.apellidoPersona, f.idArea
FROM persona p
JOIN funcionario f ON p.ci = f.ci
WHERE f.nombreEmpresa IN (
        SELECT e.nombreEmpresa
        FROM empresa e
        JOIN ciudad c ON e.idCiudad = c.idCiudad
        WHERE c.nombreCiudad NOT IN ('Montevideo', 'Maldonado')
);
```

R1 ← π (empresa.nombreEmpresa
(σ ciudad.nombreciudad != 'Montevideo' Y ciudad.nombreciudad != 'Maldonado')
(empresa ⋈ ciudad))

π (persona.nombrePersona, persona.apellidoPersona, funcionario.idArea
(σ funcionario.nombreEmpresa IN (R1))
(persona ⋈ funcionario))

**/*
10) Listar, en orden alfabético descendente por nombre, a los funcionarios que NO trabajen ni en la ciudad de "Montevideo", ni en "Minas", ni en "Colonia del Sacramento". Presentar nombre, apellido, número de área de trabajo y cargo. En la solución trabajar con los comandos IN y NOT IN. SUBCONSULTA
*/**

```
SELECT p.nombrepersona, p.apellidopersona, f.idarea, ca.nombre
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo ca ON f.idcargo = ca.idcargo
WHERE f.nombreEmpresa NOT IN(
	SELECT e.nombreEmpresa
	FROM empresa e
	JOIN ciudad c ON e.idCiudad = c.idCiudad
	WHERE c.nombreCiudad = 'Montevideo' OR c.nombreCiudad =
	'Minas' OR c.nombreCiudad = 'Colonia del Sacramento'
)
ORDER BY p.nombrepersona DESC, p.apellidopersona DESC;
```

```
NOMBRES_EMPRESAS <- π (empresa.nombreempresa
                       (σ ciudad.nombreciudad = 'Montevideo' O
                       ciudad.nombreciudad = 'Minas' O
                       ciudad.nombreciudad = 'Colonia del
                       Sacramento')
                       (empresa ⋈ ciudad))

π (persona.nombrepersona, persona.apellidopersona, funcionario.idarea,
cargo.nombre
(σ funcionario.nombreEmpresa NOT IN (NOMBRES_EMPRESAS))
(persona ⋈ funcionario ⋈ cargo))
```

```
SELECT p.nombrepersona, p.apellidopersona, f.idarea, ca.nombre
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo ca ON f.idcargo = ca.idcargo
WHERE f.nombreempresa IN (
	SELECT nombreempresa
	FROM empresa e
	JOIN ciudad c ON e.idciudad = c.idciudad
```

```
        WHERE c.nombreCiudad != 'Montevideo' AND c.nombreCiudad !=
        'Minas' AND c.nombreCiudad != 'Colonia del Sacramento'
)
ORDER BY p.nombrepersona DESC, p.apellidopersona DESC;
```

NOMBRES_EMPRESAS <- π (empresa.nombreempresa
(σ ciudad.nombreciudad != ‘Montevideo’ Y
ciudad.nombreciudad != ‘Minas’ Y
ciudad.nombreciudad != ‘Colonia del
Sacramento’)
(empresa ⋈ ciudad))

π (persona.nombrepersona, persona.apellidopersona, funcionario.idarea, cargo.nombre
(σ funcionario.nombreEmpresa IN (NOMBRES_EMPRESAS))
(persona ⋈ funcionario ⋈ cargo))

**/*
11) Listar el número y el nombre, de las áreas de trabajo, que tengan algún funcionario con fecha de alta posterior al año “2016”. SUBCONSULTA
*/**

```
SELECT DISTINCT ar.idarea, ar.nombre
FROM areadetrabajo ar
WHERE 2016 < ANY(
        SELECT EXTRACT(year from f.fechaAlta)
        FROM funcionario f
        WHERE f.idArea = ar.idArea
);
```

π (areadetrabajo.idArea, areadetrabajo.nombre (σ 2016 < ANY (π(EXTRACT(year from funcionario.fechaAlta) (σ funcionario.idArea = areadetrabajo.idArea) (funcionario))) (areadetrabajo))

**/*
12) Listar el número y el nombre de las áreas de trabajo, que tengan algún funcionario con fecha de alta posterior al año "2014" y con una comisión mayor o igual al 20%. SUBCONSULTA
*/**

```
SELECT DISTINCT ar.idarea, ar.nombre
FROM areadetrabajo ar
WHERE 2014 < ANY(
	SELECT EXTRACT(year from f.fechaAlta)
	FROM funcionario f
	WHERE f.idArea = ar.idArea
	AND f.comision >= (f.sueldo * 0.20)
);
```

```
π (DISTINCT areadetrabajo.idArea, areadetrabajo.nombre
(σ 2014 < ANY (
	π (EXTRACT(year from funcionario.fechaAlta) (σ funcionario.idArea
	= areadetrabajo.idArea Y funcionario.comision > (funcionario.sueldo
* 0.20)) (funcionario))
)
(areadetrabajo))
```

**/*
13) REALIZAR CON COMANDOS IN y EXISTS (por separado las SubConsultas). Mostrar el número, nombre y ciudad, de las áreas de trabajo, donde existan funcionarios, cuya comisión, sea mayor al 10% del sueldo (es por cada funcionario). SUBCONSULTA
*/**

```
SELECT DISTINCT ar.idarea, ar.nombre, ci.nombreciudad
FROM ciudad ci
JOIN empresa em ON ci.idciudad = em.idciudad
JOIN areadetrabajo ar ON ar.nombreempresa = em.nombreempresa
WHERE ar.idarea IN (
   SELECT ar2.idarea
   FROM areadetrabajo ar2
   JOIN cargo_areadetrabajo caar2 ON caar2.idarea = ar2.idarea
   JOIN cargo ca2 ON ca2.idcargo = caar2.idcargo
   JOIN funcionario f2 ON f2.idcargo = ca2.idcargo
   WHERE f2.comision > (0.10 * f2.sueldo)
);
```

R1 ← funcionario ⋈ cargo ⋈ cargo_areadetrabajo ⋈ areadetrabajo
R2 ← π (areadetrabajo.idarea (σ funcionario.sueldo > 0.10 * funcionario.sueldo (R1)))
R3 ← ciudad ⋈ empresa ⋈ (σ areadetrabajo.nombreempresa = empresa.nombreempresa (areadetrabajo))
R4 ← R3 ⋈ areadetrabajo.idarea = R2.idarea (R2)

π (areadetrabajo.idarea, areadetrabajo.nombre, ci.nombreciudad (R4))

```
SELECT DISTINCT ar.idarea, ar.nombre, ci.nombreciudad
FROM ciudad ci
JOIN empresa em ON ci.idciudad = em.idciudad
JOIN areadetrabajo ar ON ar.nombreempresa = em.nombreempresa
WHERE EXISTS (
   SELECT *
   FROM areadetrabajo ar2
   JOIN cargo_areadetrabajo caar2 ON caar2.idarea = ar2.idarea
   JOIN cargo ca2 ON ca2.idcargo = caar2.idcargo
   JOIN funcionario f2 ON f2.idcargo = ca2.idcargo
   WHERE ar2.idarea = ar.idarea
     AND f2.comision > (0.10 * f2.sueldo)
```

);

R1 ← π (areadetrabajo.idarea(σ funcionario.comision > 0.10 * funcionario.sueldo (funcionario ⋈ cargo ⋈ cargo_areadetrabajo ⋈ areadetrabajo)))

R2 ← ciudad ⋈ empresa ⋈ σ areadetrabajo.nombreempresa = empresa.nombreempresa (areadetrabajo)

π (areadetrabajo.idarea, areadetrabajo.nombre, ciudad.nombreciudad (R2 ⋈ (σ areadetrabajo.idarea ∈ R1 (areadetrabajo))))

**/*
14) REALIZAR CON COMANDOS IN y =ANY (por separado las SubConsultas). Presentar que todos los funcionarios, tengan asignado un número de área de trabajo, existente en la tabla AREA de TRABAJO. Presentar el número y nombre del área de trabajo. SUBCONSULTAS
*/**

```
SELECT DISTINCT f.ci, pe.nombrepersona, f.titulo, f.nombreempresa,
f.idcargo, ar.idarea, ar.nombre
FROM persona pe
JOIN funcionario f ON f.ci = pe.ci
JOIN cargo ca ON f.idcargo = ca.idcargo
JOIN cargo_areadetrabajo caar ON ca.idcargo = caar.idcargo
JOIN areadetrabajo ar ON caar.idarea = ar.idarea
WHERE ar.idarea IN (
   SELECT idarea
   FROM areadetrabajo
);
```

```
R1 ← π (idarea (areadetrabajo))
R2 ← persona ⋈ persona.ci = funcionario.ci (funcionario)
R3 ← R2 ⋈ funcionario.idcargo = cargo.idcargo (cargo)
R4 ← R3 ⋈ cargo.idcargo=cargo_areadetrabajo.idcargo
(cargo_areadetrabajo)
R5 ← R4 ⋈ cargo_areadetrabajo.idarea=areadetrabajo.idarea
(areadetrabajo)
R6 ← σ areadetrabajo.idarea ∈ R1 (R5)

π (funcionario.ci, persona.nombreper, funcionario.fechanac,
funcionario.titulo, funcionario.fechaalta, funcionario.nombreempresa,
funcionario.idproducto, funcionario.cijefe, funcionario.idcargo,
funcionario.idciudad, areadetrabajo.idarea, areadetrabajo.nombre(r6))
```

```
SELECT DISTINCT f.ci, pe.nombrepersona, f.titulo, f.nombreempresa,
f.idcargo, ar.idarea, ar.nombre
FROM persona pe
JOIN funcionario f ON f.ci = pe.ci
JOIN cargo ca ON f.idcargo = ca.idcargo
JOIN cargo_areadetrabajo caar ON ca.idcargo = caar.idcargo
JOIN areadetrabajo ar ON caar.idarea = ar.idarea
```

```
WHERE ar.idarea = ANY (
   SELECT idarea
   FROM areadetrabajo
);
```

R1←π (idarea (areadetrabajo))
R2←persona ⋈ funcionario ⋈ cargo ⋈ cargo_areadetrabajo ⋈ areadetrabajo
R3←σ areadetrabajo.idarea ∈ R1 (R2 ⋈ R1)

π (funcionario.ci, persona.nombreper, funcionario.fechanac, funcionario.titulo, funcionario.fechaalta, funcionario.nombreempresa, funcionario.idproducto, funcionario.cijefe, funcionario.idcargo, funcionario.idciudad, areadetrabajo.idarea, areadetrabajo.nombre (R3))

/*
15) REALIZAR CON COMANDO =ANY. Presentar el número, nombre y ciudad, de aquellas áreas de trabajo, en los que al menos, exista un funcionario con comisión mayor a "4".
SUBCONSULTA
***/**

```
SELECT DISTINCT ar.idarea, ar.nombre, ci
FROM funcionario f
JOIN cargo ca ON f.idcargo = ca.idcargo
JOIN cargo_areadetrabajo caar ON ca.idcargo = caar.idcargo
JOIN areadetrabajo ar ON caar.idarea = ar.idarea
JOIN empresa em ON ar.nombreempresa = em.nombreempresa
WHERE f.comision = ANY (
        SELECT comision
        FROM funcionario
        WHERE comision > 4
);
```

R1 ← π (comision (σ comision > 4 (funcionario)))
R2 ← funcionario ⋈ cargo
R3 ← R2 ⋈ cargo_areadetrabajo
R4 ← R3 ⋈ areadetrabajo
R5 ← R4 ⋈ empresa
R6 ← σ funcionario.comision ∈ R1 (R5)

π (areadetrabajo.idarea, areadetrabajo.nombre, funcionario.ci (R6))

**/*
16) REALIZAR CON COMANDO HAVING.
Mostrar el número de área de trabajo y el sueldo máximo del área, para todas las áreas cuyo sueldo máximo supere,
el sueldo medio de todos los funcionarios. SUBCONSULTA
*/**

```
SELECT f.idarea, MAX(f.sueldo) AS sueldo_maximo
FROM funcionario f
GROUP BY f.idarea
HAVING MAX(f.sueldo) > (
        SELECT AVG(sueldo) AS sueldo
        FROM funcionario
);
```

R1 ← π (G AVG (funcionario.sueldo) AS sueldo (funcionario))

π (areadetrabajo.idArea, {areadetrabajo.idArea} G
MAX(funcionario.sueldo) AS sueldo_maximo (σ {funcionario.idArea} G
MAX(funcionario.sueldo) > R1.sueldo) (funcionario ⋈ R1 ⋈ cargo ⋈
cargo_areadetrabajo ⋈ areadetrabajo))

**/*
17) Presentar los números de área de trabajo, con más funcionarios asignados.**

Presentar: número de área de trabajo y la cantidad total de funcionarios asignados por área de trabajo.
***/**

```
SELECT ar.idarea, COUNT(f.ci) AS cantidad_funcionarios
FROM funcionario f
JOIN cargo ca ON f.idcargo = ca.idcargo
JOIN cargo_areadetrabajo caar ON ca.idcargo = caar.idcargo
JOIN areadetrabajo ar ON caar.idarea = ar.idarea
GROUP BY ar.idarea;
```

```
ⴖ (areadetrabajo.idArea, {idArea} G COUNT(funcionario.ci) AS
cantidad_funcionarios (σ {idArea} G COUNT(ci) > cantidad_funcionarios)
(funcionario ⋈ cargo ⋈ cargo_areadetrabajo ⋈ areadetrabajo))
```

/*
18) Resolver con subConsulta (de 3 niveles). Presentar el número, el nombre y la ciudad, del área de trabajo, con más funcionarios asignados, con el cargo "Tester".
***/**

```
SELECT ar.idarea, ar.nombre, ci.nombreciudad
FROM areadetrabajo ar
JOIN empresa em ON ar.nombreempresa = em.nombreempresa
JOIN ciudad ci ON em.idciudad = ci.idciudad
WHERE ar.idarea = (
        SELECT idarea
        FROM (
            SELECT ar.idarea, COUNT(*) AS num_funcionarios
            FROM areadetrabajo ar
            JOIN cargo_areadetrabajo caar ON ar.idarea = caar.idarea
            JOIN cargo ca ON caar.idcargo = ca.idcargo
            JOIN funcionario f ON ca.idcargo = f.idcargo
            WHERE ca.nombre = 'Tester'
            GROUP BY ar.idarea
```

```
    ) AS subconsulta1
    ORDER BY num_funcionarios DESC
);
```

R1 ← π (areadetrabajo.idarea
(σ cargo.nombre = ‘Tester’)
(areadetrabajo ⋈ cargo_areadetrabajo ⋈ cargo ⋈
funcionario))

R2 ← π (idarea, ({R1.idArea} G COUNT(*)) AS num_funcionarios
(σ cargo.nombre = ‘Tester’)
(R1))

π (areadetrabajo.idarea, areadetrabajo.nombre, ciudad.nombreciudad
(σ areadetrabajo.idarea = (R2))
(areadetrabajo ⋈ empresa ⋈ ciudad))

**/*
19) Resolver con subConsulta (de 3 niveles). Presentar el número, el nombre y la ciudad, del área de trabajo, con más funcionarios asignados, con el cargo "administrativo", utilizando las dos tablas presentadas.
*/**

```
SELECT ar.idArea, ar.nombre, ci.nombreCiudad
FROM areadetrabajo ar
JOIN empresa em ON ar.nombreEmpresa = em.nombreEmpresa
JOIN ciudad ci ON em.idCiudad = ci.idCiudad
WHERE ar.idArea = (
   SELECT idarea
   FROM (
      SELECT caar.idarea, COUNT(*) AS num_funcionarios
      FROM cargo_areadetrabajo caar
      JOIN cargo ca ON caar.idCargo = ca.idCargo
      JOIN funcionario f ON ca.idCargo = f.idCargo
      WHERE ca.nombre = 'Administrativo'
      GROUP BY caar.idArea
      ORDER BY num_funcionarios DESC
   ) AS subconsulta1
);
```

```
R1 ← π (cargo_areadetrabajo.idArea, {cargo_areadetrabajo.idArea} G
       COUNT(*) AS num_funcionarios
       (σ cargo.nombre = 'Administrativo')
       (cargo_areadetrabajo ⋈ cargo ⋈ funcionario))

R2 ← π (idArea
       (R1))

π (areadetrabajo.idarea, areadetrabajo.nombre, ciudad.nombreCiudad
(σ areadetrabajo.idArea IN (R2))
(areadetrabajo ⋈ empresa ⋈ ciudad))
```

**/*
20) Presentar el número y el nombre de las áreas de trabajo, siempre que haya más de dos funcionarios trabajando en ellas.
*/**

```
SELECT ar.idarea, ar.nombre
FROM areadetrabajo ar
JOIN cargo_areadetrabajo caar ON caar.idarea = ar.idarea
JOIN cargo ca ON ca.idcargo = caar.idcargo
JOIN funcionario f ON f.idcargo = ca.idcargo
GROUP BY ar.idarea, ar.nombre
HAVING COUNT(f.ci) > 2;
```

```
π (areadetrabajo.idArea, areadetrabajo.nombre
(σ {areadetrabajo.idArea, areadetrabajo.nombre} G
COUNT(funcionario.ci) > 2)
(areadetrabajo ⋈ cargo_areadetrabajo ⋈ cargo ⋈ funcionario))
```

**/*
21) Presentar los números de las áreas de trabajo, en las que el sueldo medio de sus funcionarios (por área), sea menor o igual, que la media de todos los sueldos de todos los funcionarios. Presentar número y sueldo medio por área de trabajo.
SUBCONSULTA
*/**

```
SELECT f.idArea, AVG(f.sueldo)
FROM funcionario f
GROUP BY f.idArea
HAVING AVG(f.sueldo) <= (
        SELECT AVG(f.sueldo)
        FROM funcionario f
);
```

```
R1 ← π(G AVG(funcionario.sueldo)
        (funcionario))
```

π (areadetrabajo.idArea, {funcionario.idArea} G AVG(funcionario.sueldo)
σ (G AVG(funcionario.sueldo) <= R1)
(funcionario))

**/*
22) Presentar la cédula, nombre, apellido y título de todos los funcionarios, cuyo sueldo es mayor, que el sueldo medio de todos los funcionarios con el mismo título. Presentar la información ordenada en forma descendente por título.
*/**

```
SELECT f.ci, p.nombrePersona, p.apellidoPersona, f.titulo
FROM funcionario f
JOIN persona p ON f.ci = p.ci
JOIN (
    SELECT titulo, AVG(sueldo) AS sueldo_medio
    FROM funcionario
    GROUP BY titulo
) AS s ON f.titulo = s.titulo
WHERE f.sueldo > s.sueldo_medio
ORDER BY f.titulo DESC;
```

R1 ← π (titulo, {titulo} G AVG(sueldo) AS sueldo_medio (funcionario))

π (funcionario.ci, persona.nombrePersona, persona.apellidoPersona, funcionario.titulo
(σ funcionario.sueldo > R1.sueldo_medio)
(funcionario ⋈ persona ⋈ R1))

**/*
23) Presentar el número de área de trabajo, con más presupuesto asignado (sueldo + comisión), para pagar el sueldo y la comisión de sus funcionarios. Presentar número de área y mayor presupuesto. SUBCONSULTA
*/**

```
SELECT f.idArea, SUM(f.sueldo + f.comision) AS presupuestoAnual
FROM funcionario f
GROUP BY f.idArea
HAVING SUM(f.sueldo + f.comision) >= ALL(
        SELECT SUM(f.sueldo + f.comision)
        FROM funcionario f
        GROUP BY f.idArea
);
```

R1 ← π({funcionaro.idArea} G SUM(funcionario.sueldo + funcionario.comision) (funcionario))

π (funcionario.idArea, {funcionario.idArea} G SUM(funcionario.sueldo + funcionario.comision) AS presupuestoAnual
(σ {funcionario.idArea} G SUM(funcionario.sueldo + funcionario.comision) >= ALL(R1))
(funcionario))

**/*
24) Presentar los números de área de trabajo, cuyo presupuesto asignado, sea menor al presupuesto medio total. Presentar número de área y presupuesto asignado (sueldo + comisión).
SUBCONSULTA
*/**

```
SELECT f.idArea, SUM(f.sueldo + f.comision) AS presupuestoAsignado
FROM funcionario f
GROUP BY f.idArea
HAVING SUM(f.sueldo + f.comision) < (
        SELECT SUM(f.sueldo + f.comision)/COUNT(*) AS presupuesto
        FROM funcionario f
);
```

```
R1 ← π(G SUM(funcionario.sueldo + funcionario.comision)/COUNT(*)
        (funcionario))

π(funcionario.idArea, {funcionario.idArea} G SUM(funcionario.sueldo +
funcionario.comision) AS presupuestoAsignado
(σ SUM(funcionario.sueldo + funcionario.comision) < R1.presupuesto)
(funcionario))
```

**/*
25) Presentar la cédula, el número de área de trabajo, el cargo y el sueldo, de los cargos con mayor sueldo de cada área de trabajo.
*/**

```
SELECT f.ci, f.idArea, f.idCargo, f.sueldo
FROM FUNCIONARIO f
WHERE f.sueldo = (
        SELECT MAX(f2.sueldo)
        FROM FUNCIONARIO f2
        WHERE f.idarea = f2.idarea
);
```

```
R1 ← G MAX(funcionario.sueldo) (funcionario)
```

π (funcionario.ci, funcionario.idArea, funcionario.idCargo,
funcionario.sueldo
(σ R1.idArea = funcionario.idArea Y R1.sueldo = funcionario.sueldo)
(R1 ⋈ funcionario))

/*
26) Presentar los datos de los funcionarios (cédula, nombre y apellido), con los datos de su área de trabajo (número, nombre y ciudad).
***/**

SELECT f.ci, p.nombrePersona, p.apellidoPersona, at.idArea, at.nombre,
c.nombreCiudad
FROM funcionario f
JOIN persona p ON p.ci = f.ci
JOIN areadetrabajo at ON at.idArea = f.idArea
JOIN empresa e ON e.nombreEmpresa = at.nombreEmpresa
JOIN ciudad c ON c.idCiudad = e.idCiudad;

π (funcionario.ci, persona.nombrePersona, persona.apellidoPersona,
areadetrabajo.idArea, areadetrabajo.nombre, ciudad.nombreCiudad
(funcionario ⋈ persona ⋈ areadetrabajo ⋈ empresa ⋈ ciudad))

/*
27) Presentar la cédula, el apellido y el cargo de los funcionarios que pertenezcan al área de trabajo con nombre "SQA".
***/**

SELECT f.ci, p.apellidoPersona, ca.nombre
FROM funcionario f
JOIN persona p ON p.ci = f.ci
JOIN cargo ca ON f.idCargo = ca.idCargo
JOIN areadetrabajo at ON f.idArea = at.idArea
WHERE at.nombre = 'SQA';

π (funcionario.ci, persona.apellidoPersona, cargo.nombre
(σ areadetrabajo.nombre = 'SQA')
(funcionario ⋈ persona ⋈ cargo ⋈ areadetrabajo))

**/*
28) Presentar la lista de los funcionarios (nombre y apellido) con los nombres de sus jefes.
*/**

```
SELECT p.nombrePersona AS nombre_funcionario, p.apellidoPersona AS
apellido_funcionario, p2.nombrePersona AS nombre_jefe
FROM funcionario f
JOIN funcionario f2 ON f.ciJefe = f2.ci
JOIN persona p ON f.ci = p.ci
JOIN persona p2 ON f2.ci = p2.ci;
```

jefe ← (funcionario ⋈ persona)
empleado ← (funcionario ⋈ persona)

π (empleado.nombrePersona, empleado.apellidoPersona,
jefe.nombrePersona
(σ jefe.ci = empleado.ciJefe)
(jefe ⋈ empleado))

**/*
29) Presentar los datos de los funcionarios (cédula, nombre y apellido), cuyo cargo sea "Tester", con los datos de sus jefes (cédula, nombre y apellido).
*/**

```
SELECT p.ci AS cedula_funcionario, p.nombrePersona AS
nombre_funcionario, p.apellidoPersona AS apellido_funcionario, p2.ci AS
cedula_jefe, p2.nombrePersona AS nombre_jefe, p2.apellidoPersona AS
apellido_jefe
FROM funcionario f1
JOIN funcionario f2 ON f1.ciJefe = f2.ci
JOIN persona p ON f1.ci = p.ci
JOIN persona p2 ON f2.ci = p2.ci
JOIN cargo ca ON f1.idCargo = ca.idCargo
WHERE ca.nombre = 'Tester';
```

```
jefe ← (funcionario ⋈ persona)
empleado ← (funcionario ⋈ persona)

π (empleado.ci,empleado.nombrePersona, empleado .apellidoPersona,
jefe.ci, jefe.nombrePersona, jefe.apellidoPersona
(σ empleado.idCargo = π (idCargo ((σ cargo.nombre = 'Tester') (cargo)))
Y empleado.ciJefe = jefe.ci)
(jefe ⋈ empleado))
```

**/*
30) Presentar los datos de los funcionarios y de sus áreas de trabajo, incluyendo las áreas que no tienen todavía funcionarios asignados. Resolver con reunión x izquierda.
*/**

```
SELECT *
FROM areadetrabajo at
LEFT JOIN funcionario f ON at.idArea = f.idArea;
```

```
(areadetrabajo ]X| funcionario)
```

**/*
31) Presentar los datos de los funcionarios (cédula, nombre y apellido) y de sus áreas de trabajo (número, nombre y ciudad), incluyendo a los funcionarios que no tienen áreas de trabajo asignadas aún. Resolver con reunion por derecha.
*/**

```
SELECT f.ci, p.nombrePersona, p.apellidoPersona, at.idArea AS
numero_area, at.nombre AS nombre_area, c.nombreCiudad
FROM areadetrabajo at
RIGHT JOIN funcionario f ON f.idArea = at.idArea
LEFT JOIN persona p ON f.ci = p.ci
LEFT JOIN empresa e ON at.nombreEmpresa = e.nombreEmpresa
LEFT JOIN ciudad c ON e.idCiudad = c.idCiudad;
```

```
π (funcionario.ci, persona.nombrePersona, persona.apellidoPersona,
areadetrabajo.idArea, areadetrabajo.nombre, ciudad.nombreCiudad
(areadetrabajo |X[ funcionario ]X| persona ]X| empresa ]X| ciudad))
```

**/*
32) Presentar los datos de los funcionarios de las áreas de trabajo diferentes al área de trabajo con nombre “Marketing”.
*/**

```
SELECT *
FROM funcionario f
JOIN areadetrabajo at ON f.idArea = at.idArea
WHERE at.nombre != 'Marketing';
```

```
(σ idArea != π (idArea ((σ nombre = ‘Marketing’) (areadetrabajo))))
(funcionario)
```

**/*
33) Presentar los datos de los funcionarios de las áreas de trabajo con números menores, que el número de área de trabajo con nombre "Contabilidad".
*/**

```
SELECT *
FROM funcionario f
JOIN persona p ON f.ci = p.ci
JOIN areadetrabajo at ON f.idArea = at.idArea
WHERE f.idArea < (
        SELECT idArea
        FROM areadetrabajo
        WHERE nombre = 'Contabilidad'
);
```

R1 ← σ (nombre = 'Contabilidad')
(areadetrabajo)

(σ idArea < π (idArea (R1)))
(funcionario ⋈ persona)

**/*
34) Resolver con subConsulta (de 3 niveles) y además, con subConsulta y consulta con JOIN. Presentar los datos de los funcionarios, que tienen el mismo cargo, que los cargos del número de área de trabajo "SQA". Presentar cédula, nombre, apellido y nombre del cargo.
*/**

```
SELECT p.ci, p.nombrePersona, p.apellidoPersona, c.nombre
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo c ON f.idCargo = c.idCargo
WHERE f.idCargo IN (
	SELECT idCargo
	FROM funcionario
	WHERE idArea IN (
		SELECT idArea
		FROM areadetrabajo
		WHERE nombre = 'SQA'
	)
);
```

```
R1 ← π idArea
	(σ nombre = 'SQA')
	(areadetrabajo)

R2 ← π idCargo
	(σ idarea IN (R1))
	(funcionario)

π persona.ci, persona.nombrePersona, persona.apellidoPersona,
cargo.nombre
(σ idCargo IN (R2))
(persona ⋈ funcionario ⋈ cargo)
```

```
SELECT p.ci, p.nombrePersona, p.apellidoPersona, c.nombre
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo c ON f.idCargo = c.idCargo
WHERE f.idCargo IN (
```

```
        SELECT idCargo
        FROM funcionario f
        JOIN areadetrabajo a ON f.idarea = a.idarea
        WHERE a.nombre = 'SQA'
);
```

```
R1 ← π funcionario.idCargo
        (σ areadetrabajo.nombre = 'SQA')
        (funcionario ⋈ areadetrabajo)

π persona.ci, persona.nombrePersona, persona.apellidoPersona,
cargo.nombre
(σ funcionario.idCargo IN (R1))
(persona ⋈ funcionario ⋈ cargo)

SELECT p.ci, p.nombrePersona, p.apellidoPersona, c.nombre
FROM persona p
JOIN funcionario f ON p.ci = f.ci
JOIN cargo c ON f.idCargo = c.idCargo
JOIN cargo_areadetrabajo ca ON c.idCargo = ca.idCargo
JOIN areadetrabajo at ON ca.idArea = at.idArea
WHERE at.nombre = 'SQA';

π persona.ci, persona.nombrePersona, persona.apellidoPersona,
cargo.nombre
(σ areadetrabajo.nombre = 'SQA')
(persona ⋈ funcionario ⋈ cargo ⋈ cargo_areadetrabajo ⋈ areadetrabajo)
```