Juan Menduiña

Trabajo Práctico N° 1:
Introducción al Modelado Conceptual.

Resolver los siguientes ejercicios, representando entidades, relaciones, jerarquías (en caso de que existan) y atributos involucrados.

Ejercicio 1.

Crear la entidad Cliente con los siguientes atributos: DNI, apellido, nombre, CUIL, domicilio detallado, teléfono y email. De cada cliente, se registran sus compras. De cada compra, se guarda la fecha, el monto total, el medio de pago y un número de comprobante único.

Ejercicio 2.

Se deben crear las entidades Socio, Libro, Copia de Libro y Préstamo, incluyendo sus correspondientes relaciones y los atributos que se detallan a continuación. El socio se caracteriza por su información personal: DNI, apellido, nombre, fecha de nacimiento y dirección. El libro, en cambio, cuenta con una descripción, un número de ISBN único, género y año de edición. Las copias poseen un número de copia único y un estado (bueno, regular, malo), y cada copia corresponde a un único libro. El préstamo tendrá fecha y hora, el socio involucrado y la/las copia/s que el socio se lleva.

Ejercicio 3.

Crear la entidad Cuenta Bancaria con los siguientes atributos: número único de cuenta, CBU (código único), titular de la cuenta, tipo de cuenta (si corresponde a una cuenta corriente o caja de ahorro) y saldo. Si es una cuenta cuyo tipo es caja de ahorro, se necesita conocer el número máximo de extracciones permitidas en un mes. En cambio, si es una cuenta corriente, se necesita el saldo en descubierto permitido. Del titular de la cuenta, se registran sus datos personales: DNI, apellido, nombre, fecha de nacimiento, dirección, teléfono de contacto y correo electrónico.

Ejercicio 4.

Se debe modelar la información necesaria para un gimnasio. Se necesita registrar los datos correspondientes a los abonados. De cada abonado, se registra: DNI, apellido, nombre, peso, estatura, información de contacto, las disciplinas que desea realizar (libre, pilates, spinning, etc.), antecedentes médicos (una descripción general), si está activo o no y si está al día con la cuota mensual. De cada disciplina, se registra un nombre único y el costo mensual de la misma.

Ejercicio 5.

Se debe modelar la información necesaria para el área de recursos humanos de la Facultad de Informática. Interesa registrar la información sobre los empleados de la facultad y el lugar y función de trabajo dentro de la misma. La facultad se divide en áreas. De cada área, se conoce su nombre, no se repite para otras áreas, y una descripción asociada. De los empleados, se conocen sus datos personales: DNI, apellido, nombre, fecha de nacimiento, dirección, teléfono de contacto, email, fecha de ingreso y cantidad de hijos. Un empleado, a lo largo del tiempo, puede ir cambiando de área, debe quedar registro de cada área por la que pasa un empleado indicando una breve descripción de las funciones en esa área y el período de tiempo (desde/hasta) que trabajó (o lleva trabajando) en el área. Discutir, al menos, tres soluciones diferentes para resolver el historial de este ejercicio.

Ejercicio 6.

Se debe modelar la información para un club donde se realizan diversos deportes. De cada deportista, se registra: DNI, apellido, nombre, fecha de nacimiento, email, teléfonos, fecha de ingreso al club, deportes que desea realizar (un deportista puede desarrollar varios deportes). De los deportes, se registra un nombre único y descripción asociada. Un deporte puede tener un representante, en tal caso, interesa registrar de la persona que lo representa su DNI, nombre, apellido y email. Además, resulta necesario almacenar información sobre las lesiones que el deportista ha tenido a lo largo de su carrera. En caso de haber tenido, sobre la lesión, interesa conocer: nombre de la lesión (por ejemplo, esguince de tobillo derecho grado II), fecha en que sucedió la lesión, tiempo de rehabilitación y tipo de lesión (esguince, fractura, desgarro muscular, etc.). Nota: Se podrían pedir estadísticas de tipos de lesiones más comunes en determinado deporte; discutir la mejor solución.

Ejercicio 7.

Modelar la información necesaria para una bicicletería que brinda servicio de alquiler de bicicletas. Existen clientes que pueden alquilar bicicletas de forma temporal. De los clientes, se registra: número único de cliente, nombre, apellido, fecha de nacimiento y domicilio. Existen diferentes tipos de bicicletas para alquiler. Del alquiler, se debe registrar: la bicicleta alquilada, el cliente y la fecha de comienzo y finalización del alquiler. Cada bicicleta tiene un número único asociado, el tipo de bicicleta y una descripción. De los tipos de bicicletas, se registra: el nombre correspondiente (el nombre del tipo de bicicletas no se repite) y una descripción.

Ejercicio 8.

Se debe modelar la información necesaria para una agencia de turismo que vende pasajes aéreos. De los clientes, se registran: código único de cliente, DNI, nombre, apellido, nacionalidad y fecha de nacimiento. Los clientes compran los pasajes aéreos para vuelos a diferentes destinos. De cada vuelo, se registra: ciudad origen, ciudad destino, fecha y hora de salida, fecha y hora de llegada, aerolínea y un número de vuelo asociado. El número de vuelo se puede reutilizar en diferentes días de salida (es decir, se repite en diferentes fechas). Cada pasaje que la agencia tiene disponible para vender se caracteriza por el número de fila y butaca y el vuelo al que corresponde. Cuando se vende un pasaje, se debe dejar registro del cliente que realizó la compra correspondiente.

Ejercicio 9.

Se debe modelar la información de una casa de electrodomésticos. La misma registra información sobre sus empleados y clientes. De cada empleado, se registra: DNI, CUIL, nombre, apellido, fecha de nacimiento, fecha de ingreso, fecha de egreso, email/s y sueldo básico. De los clientes, se registra: DNI, nombre, apellido y, opcionalmente, teléfono y dirección de mail. También se dispone de información sobre los electrodomésticos a la venta. De cada uno, se registra un código interno único, nombre, modelo, stock, precio regular y precio online (si está disponible). Cada electrodoméstico pertenece a una categoría (climatización, lavado, cocina, etc.; el nombre de la categoría es único). Es importante registrar las ventas realizadas. De cada venta, se registra: código de venta (único), fecha, cliente, electrodomésticos incluidos, precio total y el empleado que realizó dicha venta. Nota: Tener en cuenta que se podrían pedir estadísticas sobre mejor o peor cliente, tipos de electrodomésticos más vendidos, promedios de ventas por clientes entre otros; discutir la mejor solución.

Juan Menduiña

Trabajo Práctico N° 2:
Modelado Conceptual, Lógico y Físico.

PARTE I: Modelado Conceptual, Lógico y Físico.

Para cada ejercicio, realizar el correspondiente modelo conceptual, pasaje al modelo lógico y pasaje al modelo físico.

Ejercicio 1.

Se debe modelar la información necesaria para una Inmobiliaria de la ciudad de La Plata. Es necesario modelar la información de clientes y empleados de la inmobiliaria. De ambos, se conoce DNI, nombre, apellido y dirección detallada. Además, de los empleados, se conoce el número de legajo, el cual no se repite entre diferentes empleados y el área donde trabaja cada uno. Los empleados pueden ir rotando de área a lo largo del tiempo y es necesario modelar por las distintas áreas que pasó un empleado. De las áreas, se conoce código único de área, descripción, teléfonos y ubicación. De cada cliente, además, se debe almacenar sexo, nacionalidad, número de pasaporte (si tuviera) y los inmuebles de los que es dueño. Un cliente no puede trabajar en la inmobiliaria.

La inmobiliaria maneja diferentes inmuebles, de los cuales se conoce dirección detallada, código único de inmueble, cantidad de ambientes, si posee balcón, si posee lavadero, cantidad de baños, si se alquila, se vende o ambas cosas, precio de venta y precio de alquiler, tipo de inmueble (casa, dúplex, depto., ...).

Debe quedar registrado todo alquiler y venta que realiza la inmobiliaria, detallando, para los alquileres, inmueble, cliente (dueño), fechas de inicio y fin de alquiler, empleado que alquiló y precio. De las ventas, se registran fecha y hora de venta, dueño y empleado que vendió la propiedad, precio de venta y comisión de venta.

Nota: Tener en cuenta que podría pedirse promedio de ventas de un semestre del año, vendedor más exitoso del año, tipo de inmueble más alquilado o más vendido, entre otros.

Ejercicio 2.

Se desea modelar el manejo de la información referente a una casa de comidas. De los clientes, se conoce DNI, nombre, apellido, teléfonos, dirección detallada, email y CUIL. La casa vende diferentes platos, de los que se conoce código único, precio actual y descripción. Cuando un cliente llama, se le toma el pedido, el cual puede estar formado por uno o varios platos. Debe quedar registro de la fecha y hora que se realizó el llamado, la fecha y hora de entrega del pedido, el precio total y los precios que fue vendido cada plato. Los repartos se realizan por zonas, conociéndose, de cada zona, el número único de zona y el costo de envío hasta esa zona. Es necesario poder obtener el costo de envío de un pedido dado.

Por otro lado, se debe modelar la reposición de materia prima con la que se elaboran los platos. Cada plato está formado por varios productos, de los que se conoce código único, descripción, precio actual, stock y cantidad mínima. Se realizan compras de productos a proveedores, de los cuales se conoce nro. único de proveedor, DNI, nombre, apellido y razón social (único). Por cada compra, se debe poder obtener los productos involucrados, el proveedor, la fecha, la hora y el precio al que se compró cada producto.

Nota: Tener en cuenta que se podría pedir obtener los productos que conforman cada plato, como así también aquellos pedidos que fueron entregados, los pedidos que fueron rechazados, pedidos mayores a $10.000 y pedidos que se encuentran pendientes, entre otras estadísticas y consultas.

Ejercicio 3.

Se desea modelar la información necesaria para el tratamiento de pacientes con COVID-19. Es necesario representar tanto a los pacientes que han contraído el virus como a los médicos que atienden los casos. Tanto para los médicos como para los pacientes, es necesario almacenar: DNI, número de pasaporte (si poseen), dirección detallada, nombre, apellido y fecha de nacimiento.

Para los pacientes, se debe almacenar, además, una descripción de enfermedades preexistentes e información referente al episodio de contagio y, además, cantidad de dosis de la vacuna anti COVID. Tener en cuenta que hay pacientes que pueden contagiarse más de una vez, por lo tanto tendrán más de un episodio. De cada episodio, se registra: número único de episodio, fecha de detección, síntomas y médicos que llevaron a cabo la atención. Los síntomas pueden variar entre los diferentes episodios, sabiendo que cada síntoma tiene un código único y una descripción asociada.

Para los médicos, es necesario almacenar, además, el código de matrícula, especialidades y sala en la que atiende. Un médico puede rotar de salas y es necesario modelar el historial de rotaciones de cada médico. De la sala, se conoce número de sala (único), piso y capacidad. De las especialidades, se almacena un nombre único y descripción. Es necesario que se almacene la nota promedio obtenida por el médico en cada especialidad.

Tener en cuenta que un médico también puede contagiarse COVID-19 y el modelo debe permitir representar esto.

Nota: Podría pedirse médico que tiene máximo de atenciones por COVID, médico que no atendió más de 10 pacientes, paciente con más contagios, entre otros.

Ejercicio 4.

Se desea modelar la información necesaria para una cadena de tiendas de indumentaria. De cada tienda, se desea almacenar: la razón social, su dirección completa, un teléfono de contacto, instagram y facebook de la misma (si posee). Las tiendas están conformadas por un grupo de empleados y un supervisor a cargo de la misma. De los empleados, se registran: DNI, CUIL, nombre, apellido, fecha de nacimiento, dirección, uno o varios teléfonos de contacto, fecha de ingreso, cantidad de hijos y estado civil.

Las tiendas están divididas en sectores de trabajo: personal, proveedores, ventas, entre otros. De cada sector, se registra nombre, descripción y un código único relativo a la tienda; el mismo código de sector puede estar en varias tiendas. Cada empleado está asignado a un sector determinado, pero, con el transcurso del tiempo, va cambiando de sector; se debe poder determinar los sectores por los que pasó un empleado en orden cronológico. Cada empleado trabaja en una y sólo una tienda. El supervisor sólo podrá estar a cargo de una tienda.

De los productos que se comercializan en las tiendas, se debe registrar: tipo de producto, marca, modelo, talle, descripción, color, precio de venta y el stock del mismo en la tienda y un código único de producto. Por último, se deben registrar las ventas realizadas, indicando fecha, número de ticket fiscal, empleado que efectúa la venta, el/los productos involucrados y el total de la misma.

Notas:

- *El supervisor es un empleado de la tienda que se desempeña como tal.*
- *Tener en cuenta que el stock de un producto y el precio de venta del producto podrían variar en las diferentes tiendas.*
- *Se debe poder consultar la información mediante diferentes alternativas: marca y/o modelo más vendidos, tipo de productos más vendidos, entre otras.*

Ejercicio 5.

Se desea modelar la información necesaria para una red social. La red social permite a los usuarios compartir imágenes de diferentes temáticas y realizar comentarios sobre las mismas. De los usuarios, se registran: nombre y apellido, usuario en la red social (que es único), clave de acceso, correo electrónico, dirección detallada y un teléfono de contacto.

En la red social, los usuarios pueden subir imágenes, o bien comentar, descargar o compartir una publicación (imagen) de otros usuarios vinculados.

Cuando dos usuarios se vinculan, se debe almacenar información de este vínculo: fecha y hora, tipo de vínculo y, opcionalmente, una descripción estandarizada en la red social que indica de donde se conocen (lugar de vínculo). En la misma fecha y hora, el mismo usuario no podrá generar dos vínculos.

De las imágenes, se debe almacenar: fecha y hora de publicación, temática de la imagen, título, usuario que realiza el posteo y el nombre del archivo correspondiente, además de un conjunto de palabras claves que caracterizan la imagen. Un usuario no puede subir dos imágenes con título idéntico. De las temáticas, se registra un nombre único y una descripción asociada.

De las imágenes, se debe poder determinar los usuarios que compartieron la imagen o la descargaron, indicando fecha y hora y si la compartió o descargó. Así mismo, las imágenes pueden recibir reacciones de los usuarios (por ejemplo: "me gusta", "me encanta", "no me gusta", etc.), debiendo almacenarse el tipo de reacción, el usuario que la realizó y la fecha y hora en que se registró.

Así mismo, los usuarios pueden realizar comentarios en la red social, se debe almacenar un texto, la fecha y hora de creación del comentario, qué imagen comentó o a qué comentario respondió. Un comentario podría ser respuesta a otro comentario existente. Un usuario no podrá realizar dos comentarios en la misma fecha y hora. Los comentarios también pueden recibir reacciones de los usuarios, debiendo almacenarse el tipo de reacción, el usuario que la realizó y la fecha y hora en que se registró.

Ejercicio 6.

Se debe modelar la información necesaria para un vivero muy importante de la ciudad. Resulta indispensable conocer la información de los empleados y clientes del vivero. De ambos, se registra: DNI, CUIT, apellido, nombre, teléfonos de contacto, email si posee y dirección detallada. En particular, para los empleados, debe registrarse, además: fecha de ingreso, fecha de nacimiento, cantidad de hijos y un número único de empleado. De los clientes, se registra, además, el código único de cliente.

El vivero ofrece diferentes artículos para jardinería: plantas, herramientas, macetas, sustratos, entre otros. De todos ellos, se registra un número único de artículo, precio, tamaño, nombre, stock y descripción.

Se deben registrar las ventas realizadas. De cada venta, se registra fecha, hora, cliente y el empleado que se encarga de realizar la venta; además, se deben registrar el/los artículos vendidos y un número de ticket fiscal. De los artículos vendidos, se debe almacenar precio de venta y cantidad vendida.

El cliente puede abonar las compras en efectivo, con débito o crédito. Si abona con débito o crédito, debe quedar registro del número de tarjeta, banco y entidad que la emite (Visa, Mastercard, etc.). Del banco y entidad emisora, se registra un nombre único y descripción. Si el pago es con crédito, se debe registrar, además, la cantidad de cuotas en que realizó el pago. Tener en cuenta que la tarjeta se debe poder reutilizar en otros pagos y con otro nro. de cuotas. Si abona en efectivo, se deberá dejar constancia de esto.

Ejercicio 7.

Se desea modelar la información necesaria para una empresa dedicada a la realización de eventos gastronómicos al aire libre. Para cada evento, se alquilan cierta cantidad de lugares, donde el inquilino podrá vender productos comestibles, o bien utilizarlo para difundir su actividad; además, se brindan servicios tales como electricidad, agua corriente y demás, si el inquilino lo requiere. De los lugares, se registra: número, descripción, metros cuadrados (m2), ubicación, distancia a la puerta de acceso al predio y precio por día. De los servicios, en cambio, se registra: nombre único, descripción y precio del servicio.

De cada evento, se registra: nombre del evento, fecha y hora de inicio, dirección del evento, duración del evento y el staff de personas encargadas de la organización y difusión del evento. De cada integrante del staff, se debe registrar: DNI, nombre completo, dirección detallada, fecha de nacimiento, correo electrónico, teléfonos de contacto y el rol que cumple dentro del evento.

De los inquilinos, se registran: DNI, nombre completo, dirección detallada, teléfonos de contacto, razón social (si el inquilino representa a una empresa) y tipo de gastronomía (si vende comestibles). Además, se debe registrar información de los alquileres: fecha, hora, quién es el inquilino, el evento al que corresponde el alquiler, el o los lugares que alquila y, en caso de que alquile servicios, los servicios alquilados.

Nota: El nombre del evento no se podrá repetir en una misma fecha. Tener en cuenta que debe poder determinar el costo de todos los alquileres (valor al que se alquila o alquiló cada lugar o servicio), actuales e histórico.

Ejercicio 8.

Se desea modelar una base de datos para el manejo de una unidad del servicio penitenciario (cárcel en lenguaje coloquial). En la unidad trabajan empleados, de los cuales se conoce nombre, apellido, dirección completa, legajo (único), DNI, teléfonos y el área donde trabaja. Cada empleado puede trabajar en una única área a la vez, pero puede, a lo largo del tiempo, trabajar en las diferentes áreas; se debe modelar el historial de áreas en donde ha trabajado un empleado. Existen dos tipos de áreas, las áreas administrativas y los pabellones.

En las áreas administrativas, trabajan sólo empleados administrativos y, en los pabellones, sólo empleados penitenciarios. Un empleado administrativo no puede ser penitenciario. Además, de los empleados penitenciarios, es necesario conocer el número de matrícula, el cual es único. Los empleados administrativos pueden tener uno o varios títulos.

De los internos, se conoce, nombre, apellido, apodo, número de causa (puede ser compartida por varios internos), listado estandarizados de delitos en esa causa y DNI. Los internos son alojados en pabellones; es necesario conocer el pabellón donde se encuentra un interno. Si un interno es movido a otro pabellón, debe quedar registrado el historial. De los pabellones, se conoce, número único de pabellón, ubicación, cantidad máxima de internos y una descripción. Las áreas administrativas tienen un número único y una descripción.

Por último, es necesario registrar los posibles incidentes ocurridos en la unidad, detallando fecha y hora del incidente, internos involucrados, descripción del incidente, el pabellón donde ocurrió y el penitenciario responsable.

Ejercicio 9.

Se debe modelar la información correspondiente para la gestión de personal y proyectos de una empresa de software con sede en la ciudad de La Plata. La empresa se encuentra dividida en diferentes áreas; de cada una de ellas, se conoce: nombre del área (único), un código de área único y una descripción de las funciones de la misma. Los empleados de la empresa corresponden a un área de la empresa, pudiendo prestar servicios en uno o varios proyectos. De cada empleado, se conoce: DNI, CUIL, fecha de nacimiento, cantidad de hijos, fecha de ingreso, una descripción de el/los título/s si posee, dirección detallada, uno o varios teléfonos de contacto y el área en la que se desempeña. Cada área cuenta con un gerente encargado de la misma que forma parte de los empleados de la empresa. Todos los empleados pueden rotar dentro de las áreas de la empresa; se debe poder determinar todos los empleados que trabajaron en una determinada área en orden cronológico; del mismo modo, todos los gerentes que tuvo un área. La empresa, además, lleva adelante varios proyectos; de cada proyecto, se conoce: nombre único, fecha de comienzo, fecha estimada de finalización, costo estimado, tipo de proyecto y los empleados que intervienen en su realización. Cuando un empleado es asignado a un proyecto, se debe indicar fecha inicio, fecha de fin opcional, cargo que desempeñará dentro del proyecto, cantidad de horas dedicadas. De los tipos de proyecto, se conoce: nombre único del tipo y una descripción asociada. Además, se tiene estandarizado los posibles cargos a asumir dentro de un proyecto; de cada cargo, se almacena: código único, nombre y una descripción del mismo.

Nota: Tener en cuenta que podría consultarse cuántos empleados se desempeñan con cargo programador, o bien cuántos proyectos de tipo "X" finalizaron este año.

Ejercicio 10.

Se debe modelar la información necesaria para una herramienta que permita realizar el seguimiento de tareas que desempeña determinada empresa en sus proyectos. La herramienta debe permitir almacenar información básica de cada proyecto, el conjunto de tareas que involucra el proyecto e información sobre los empleados que se desempeñan implementando las tareas, o bien coordinando proyectos y los servicios necesarios para cada proyecto.

De los empleados, se registra DNI, apellido, nombre, fecha de nacimiento, dirección detallada, email y teléfono de contacto. Además, para cada empleado, se debe registrar un nombre de usuario y clave para utilizar la herramienta.

De cada proyecto, se registra un código único, un título, descripción, fecha de inicio, fecha estimada de finalización, fecha efectiva de finalización, el presupuesto asignado, el empleado coordinador del proyecto y, si requiere servicios externos, información de los mismos, indicando, además, fecha de inicio y fin del servicio.

Cada proyecto se divide en tareas; de cada tarea, se registra: número único de tarea, nombre, descripción, tipo de tarea, porcentaje realizado, fecha de inicio y fecha de fin de la misma, estado de la tarea (pendiente, ejecución, finalizada, cancelada, etc.), empleado que cargó la tarea y empleado asignado para realizarla (si posee). Además, la tarea puede tener uno o varios empleados seguidores de la misma; se debe dejar registro de los seguidores de cada tarea, indicando, por cada seguidor, si está interesado en recibir notificaciones sobre cambios en la tarea. Las tareas se pueden dividir en varias subtareas; se debe registrar, para cada tarea, la o las tareas que la componen.

De los servicios externos, se registra: código de servicio, nombre, costo y la empresa que lo brinda. El código de servicio es relativo a la empresa, es decir, no se repite dentro de la empresa. De las empresas, se detalla razón social, teléfonos de contacto y dirección detallada de la misma. Tener en cuenta que ese servicio podría ser utilizado, posteriormente, en otro proyecto.

Nota: La herramienta debe permitir realizar estadísticas tales como cuáles son las tareas en determinado estado, qué empleado tiene más tareas asignadas, cuáles son las tareas de un determinado proyecto, entre otras.

Ejercicio 11.

Se trata de modelar la información de clientes de una peluquería. De cada cliente, debe modelarse la información personal: nombre, apellido, fecha de nacimiento, DNI, dirección y teléfonos, además de una descripción si es alérgico a algún componente químico. De cada cliente, se tiene una ficha, la misma permite determinar las atenciones que se le realizaron al cliente. De cada atención, se debe registrar fecha de atención, qué peluquero lo atendió, si se le realizó tratamiento detalle del o los mismos. Se deberá indicar, para cada tratamiento aplicado, qué componentes se utilizaron (en caso de ser necesario) y cantidad de cada componente.

Existen diferentes tratamientos. Los tratamientos se identifican por su nombre y, además, se registra una descripción, precio, duración y los componentes necesarios para el mismo (algunos tratamientos pueden no requerir componentes), indicando cantidad sugerida por cada componente. De los componentes, se registra código único de componente, nombre y marca que lo fabrica.

De cada peluquero, se registran: DNI, apellido, nombre, domicilio detallado y teléfono de contacto.

Tener en cuenta que el valor de los tratamientos varía con el transcurso del tiempo; se debe poder determinar cuánto pagó un cliente "x" por el tratamiento "y" en una fecha determinada.

El cliente puede abonar la atención en efectivo, con débito o crédito, debiéndose registrar el modo de pago. Si abona con débito o crédito, debe quedar registro del número de tarjeta (número único), código de seguridad, banco y entidad que la emite (Visa, Mastercard, etc.). Si el pago es con crédito, se debe registrar, además, la cantidad de cuotas en que realizó el pago.

Nota: Tener en cuenta que la tarjeta se puede usar en N pagos y en diferente cantidad de cuotas en cada caso.

Ejercicio 12.

Se debe modelar la información necesaria para la gestión de un colegio de la ciudad de La Plata, el mismo cuenta con tres niveles de enseñanza: jardín de infantes, primaria y secundaria.

Se debe almacenar información de los alumnos, indicando: nombre, apellido, DNI, fecha de nacimiento, dirección detallada, teléfonos, descripción de alergias (si posee), si es celíaco, diabético o intolerante a la lactosa y el nivel de enseñanza que se encuentra.

De los alumnos, se conoce, además, información de sus tutores, indicando: DNI, nombre, apellido, teléfonos de contacto, email, dirección detallada, si permite contactar por WhatsApp (nro. de teléfono para WhatsApp) y vínculo con el alumno. Además, los alumnos pueden tener personas autorizadas a retirarlos; de estos, se almacena la misma información que la de los tutores. Si un alumno es retirado, se debe dejar constancia de fecha y hora de retiro, alumno y el tutor o persona autorizada que lo retira.

El colegio cuenta con un comedor que brinda a sus alumnos un menú que dispone de diferentes platos. De cada plato ofrecido en el menú, se conoce su código (único), descripción, costo, si es apto para celíacos, si es apto para diabéticos y si es apto para intolerantes a la lactosa. Se debe dejar constancia de los platos consumidos por los alumnos, registrando la fecha y el precio del plato al momento del consumo.

Además, se debe registrar información relacionada a los pagos realizados al colegio, indicando: fecha y hora, tutor que abona, a qué alumno/s corresponde el pago, monto, la forma de pago con la que realiza el mismo y qué concepto está pagando: si es un pago de matrícula o si es un pago del servicio de comedor (se debe almacenar el período de tiempo de los consumos pagados), o bien si es un pago de cuota mensual (se debe almacenar número de cuota y año que abona). Si el pago se realiza con tarjeta de débito o crédito, se debe dejar constancia de: nro. de tarjeta, código de seguridad, entidad emisora de la tarjeta (Visa, Mastercard, etc.), banco de la tarjeta y cantidad de cuotas en las que realiza el pago.

Nota: Tener en cuenta que la tarjeta se puede usar en N pagos y en diferente cantidad de cuotas en cada caso.

Ejercicio 13.

Se desean modelar los datos relacionados con vuelos, pasajeros, tripulación y reservas de Aerolíneas Argentinas. De cada vuelo, se conoce número de vuelo (que se repite para diferentes fechas de salida), aeropuerto de origen, aeropuerto de destino, fecha y hora de salida, fecha y hora de llegada (se cargan una vez que arriba), el avión que realizará el vuelo, los miembros de la tripulación asignados y los pasajeros que viajarán. De cada avión, se conoce su matrícula (única), su marca (Boeing, Airbus, Embraer, etc.), su modelo, su año de fabricación y su capacidad (cantidad de asientos para pasajeros). De cada aeropuerto, se conoce un código único (por ejemplo, para el Aeropuerto Internacional de Punta Cana, el código es PUJ), el nombre del aeropuerto, la ciudad y el país en el que se encuentra. De cada ciudad, se conoce su nombre y población. Tener en cuenta que un mismo nombre de ciudad se puede repetir para diferentes países.

De los miembros de la tripulación, se conoce su nombre, apellido, CUIL, número de legajo (único), horas de vuelo acumuladas y uno o más roles que pueden cumplir. En cambio, de los pasajeros que viajarán, se conoce su nombre, apellido, CUIL, número de pasaporte, teléfono, email y, en caso de estar asociados al programa de beneficios de la aerolínea, los puntos acumulados hasta el momento y el número de la tarjeta de beneficios.

Para viajar en un vuelo, los pasajeros necesitan hacer una reserva. De la reserva, se conocen el número único de reserva, fecha de la reserva, los asientos asignados a cada pasajero y el estado de la reserva (pendiente de pago, pagada, cancelada, etc.). Una misma reserva puede incluir varios pasajeros que viajan juntos en el mismo vuelo. Además, se quiere llevar un registro histórico de los cambios de estado de la reserva, registrando el período de tiempo en que la reserva permaneció en cada estado. Esto permitirá llevar un seguimiento detallado del ciclo de vida de cada reserva.

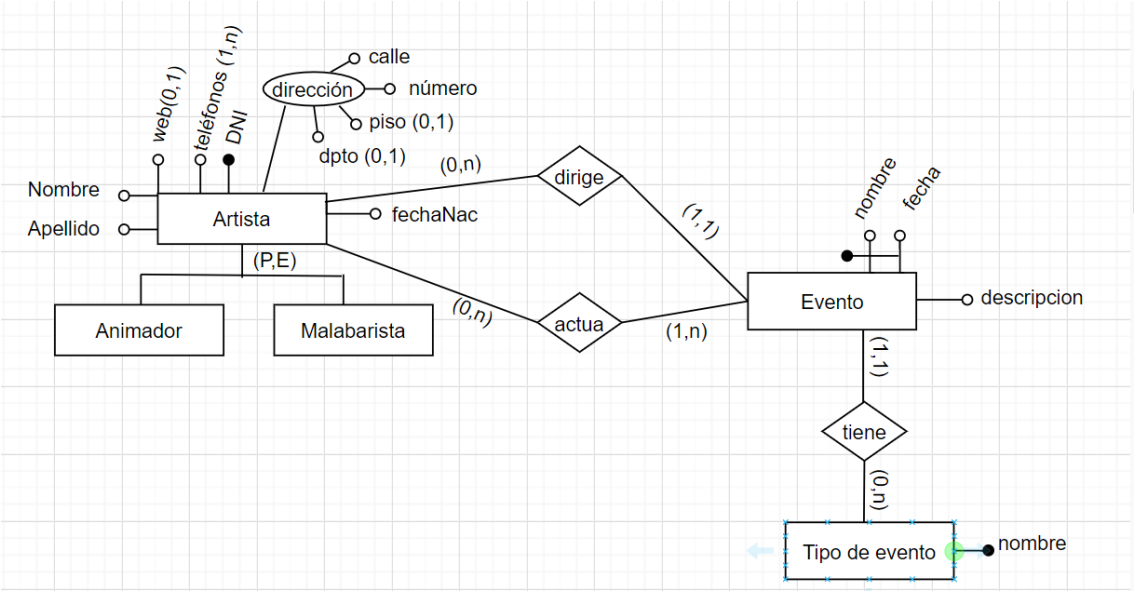
PARTE II: Derivación a Modelos Lógico y Físico.

Para cada ejercicio, plantear el correspondiente pasaje al modelo lógico y al modelo físico.

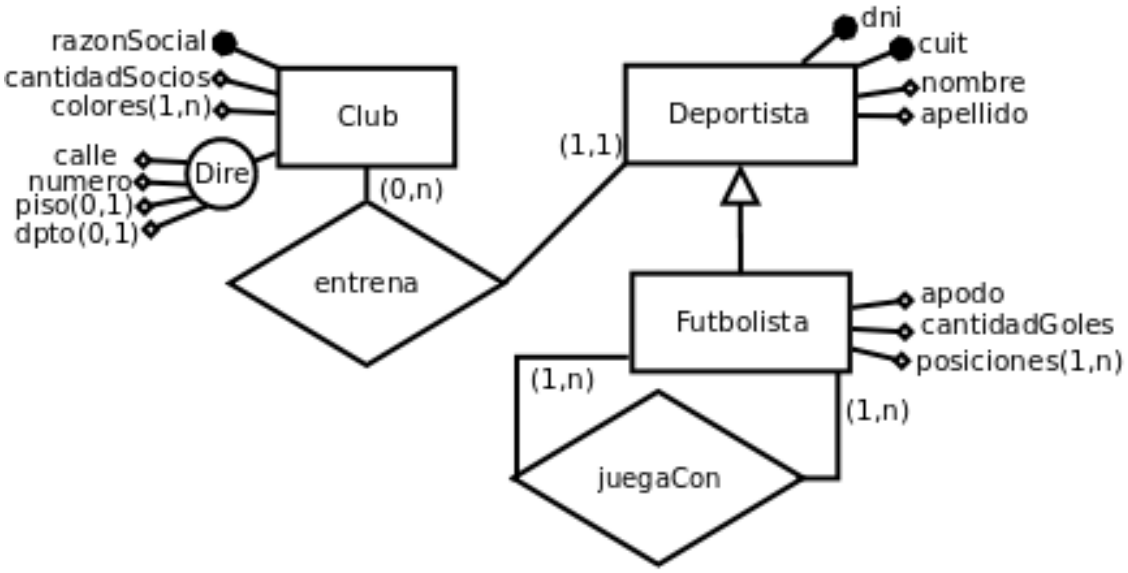
Convenciones para el modelo físico:

- *Subrayar con una única línea las claves primarias.*
- *Denotar como fk a las claves externas.*
- *Denotar atributos opcionales con el signo de interrogación (por ejemplo, b3?).*

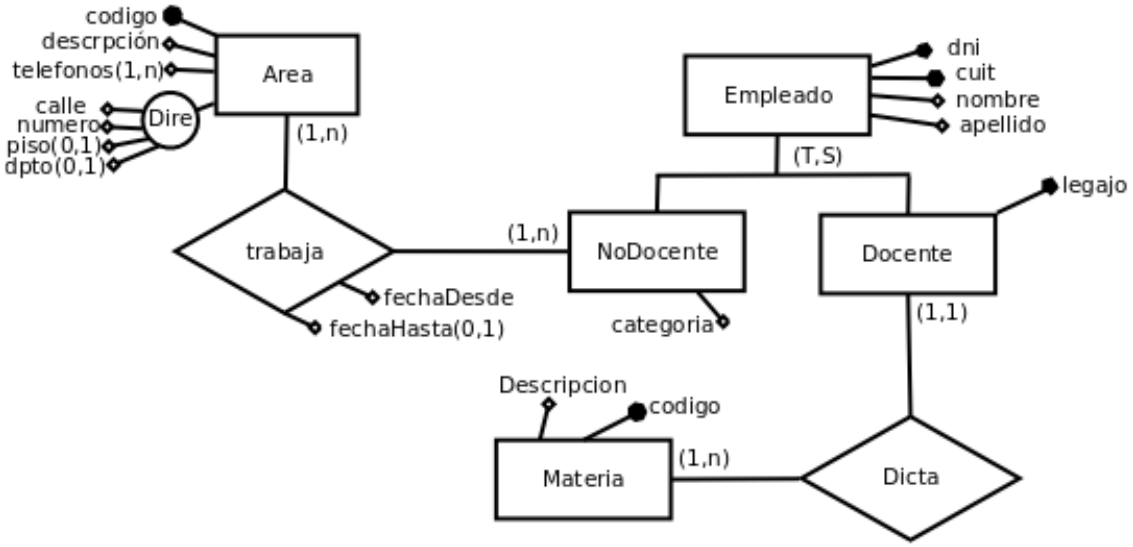
Ejercicio 1.



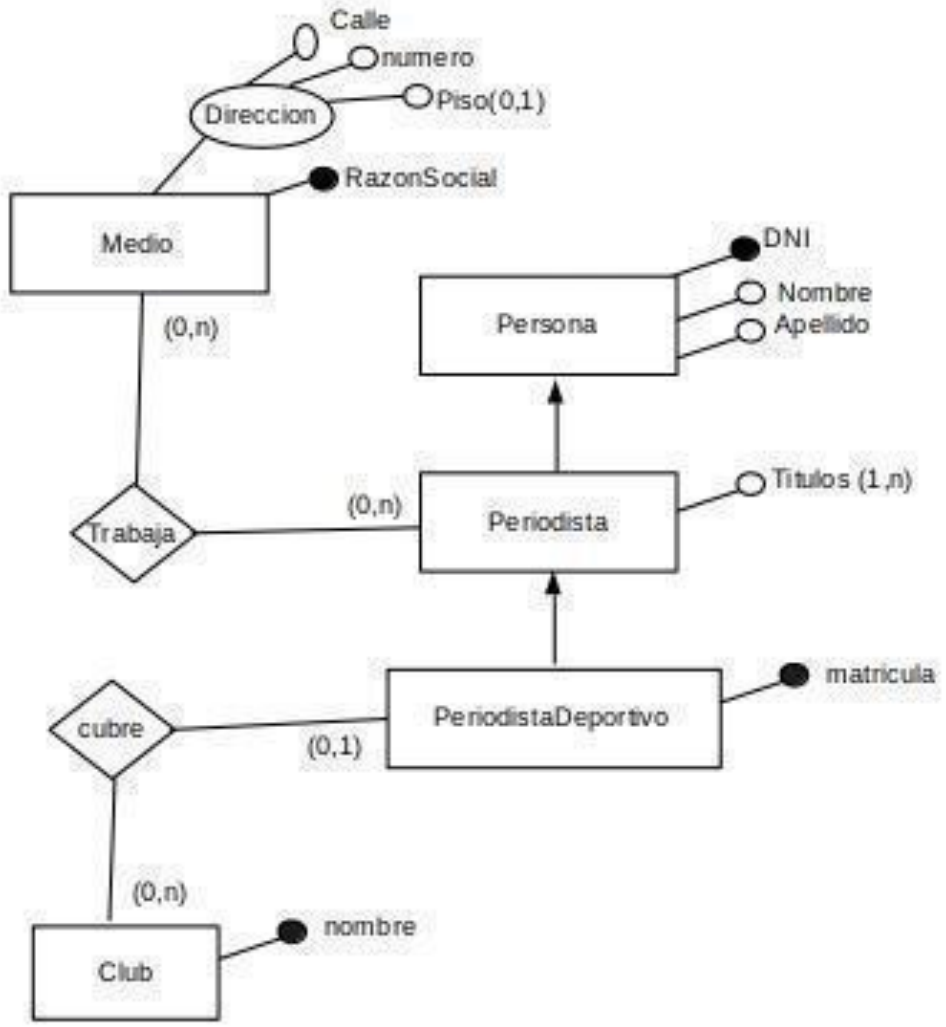
Ejercicio 2.


razonSocial
cantidadSocios
colores(1,n)
Club
calle
numero
piso(0,1)
dpto(0,1)
Dire
(0,n)
entrena
(1,1)
dni
cuit
Deportista
nombre
apellido
Futbolista
apodo
cantidadGoles
posiciones(1,n)
(1,n)
(1,n)
juegaCon

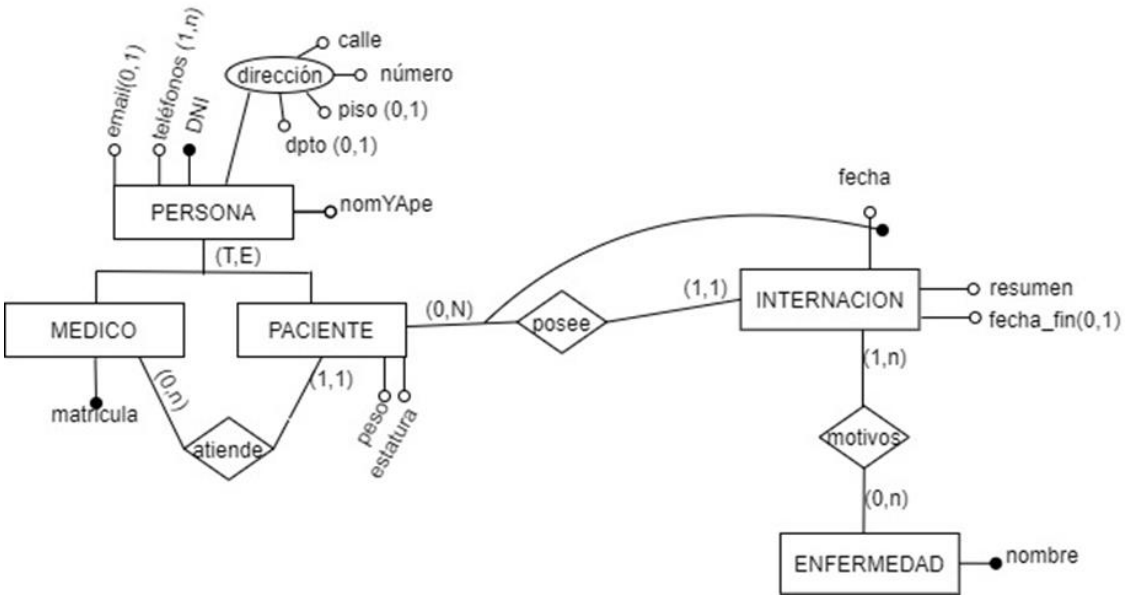
Ejercicio 3.


codigo
descrpción
telefonos(1,n)
Area
calle
numero
piso(0,1)
dpto(0,1)
Dire
(1,n)
trabaja
fechaDesde
fechaHasta(0,1)
(1,n)
NoDocente
categoria
Empleado
dni
cuit
nombre
apellido
(T,S)
Docente
legajo
(1,1)
Descripcion
codigo
Materia
(1,n)
Dicta

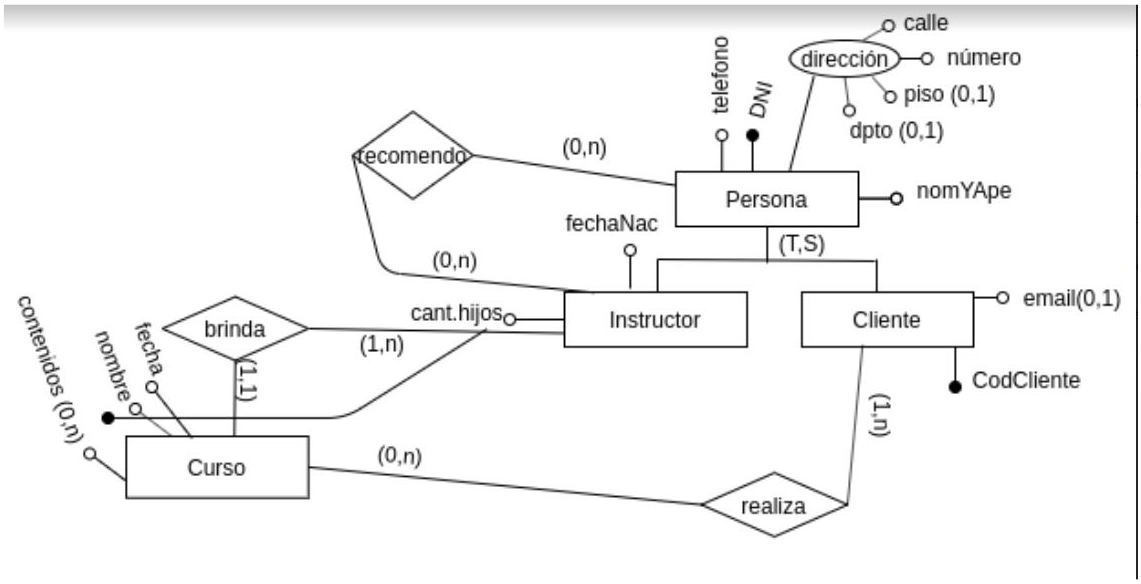
Ejercicio 4.


Calle
numero
Piso(0,1)
Direccion
RazonSocial
Medio
DNI
Nombre
Apellido
Persona
(0,n)
Titulos (1,n)
Trabaja
(0,n)
Periodista
matricula
cubre
(0,1)
PeriodistaDeportivo
(0,n)
nombre
Club

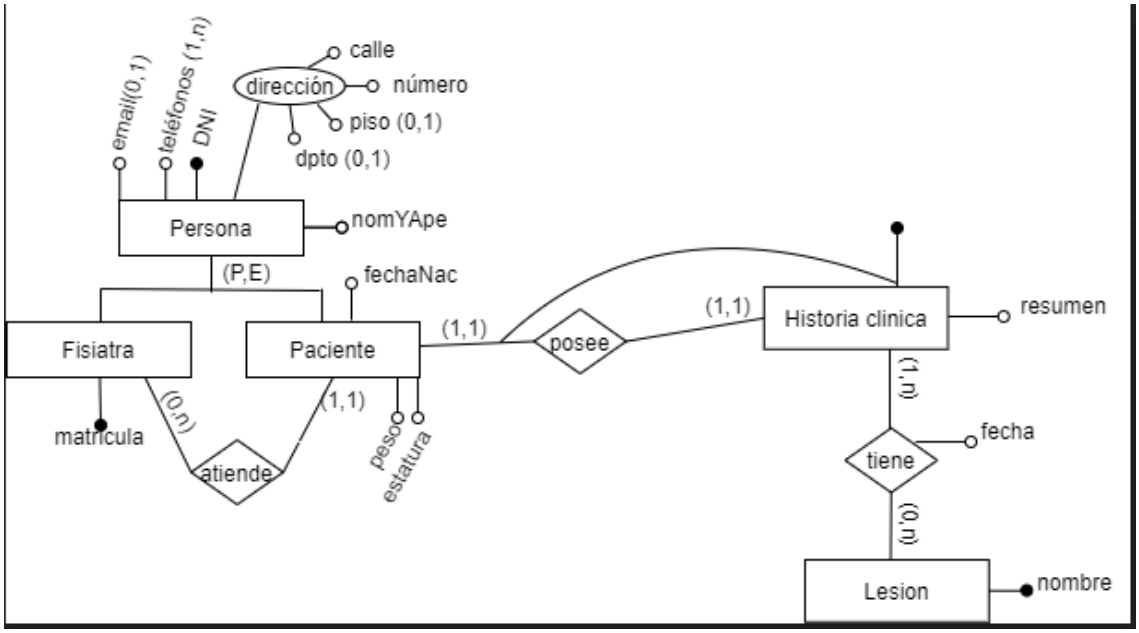
Ejercicio 5.


email(0,1)
teléfonos (1,n)
DNI
dirección
calle
número
piso (0,1)
dpto (0,1)
PERSONA
nomYApe
(T,E)
MEDICO
PACIENTE
matrícula
(0,n)
atiende
(1,1)
peso
estatura
(0,N)
posee
(1,1)
fecha
INTERNACION
resumen
fecha_fin(0,1)
(1,n)
motivos
(0,n)
ENFERMEDAD
nombre

Ejercicio 6.


calle
dirección
número
piso (0,1)
dpto (0,1)
telefono
DNI
recomendo
(0,n)
Persona
nomYApe
fechaNac
(T,S)
(0,n)
Instructor
Cliente
email(0,1)
CodCliente
brinda
cant.hijos
(1,n)
(1,1)
contenidos (0,n)
nombre
fecha
Curso
(0,n)
realiza
(1,n)

Ejercicio 7.


email(0,1)
teléfonos (1,n)
DNI
calle
dirección
número
piso (0,1)
dpto (0,1)
Persona
nomYApe
(P,E)
fechaNac
Fisiatra
Paciente
(1,1)
posee
(1,1)
Historia clinica
resumen
matrícula
(0,n)
atiende
(1,1)
peso
estatura
(1,n)
fecha
tiene
(0,n)
Lesion
nombre

Juan Menduiña

Trabajo Práctico N° 3:
Álgebra Relacional.

Ejercicio 1.

Cliente= ($\underline{idCliente}$*, nombre, apellido, DNI, telefono, direccion).*
Factura= ($\underline{nroTicket}$*, total, fecha, hora, idCliente(Fk)).*
Detalle= ($\underline{nroTicket(Fk), idProducto(Fk)}$*, cantidad, precioUnitario).*
Producto= ($\underline{idProducto}$*, nombreP, descripcion, precio, stock).*

(1) *Listar nombre, apellido, DNI, teléfono y dirección de clientes con DNI superior a 22.222.222.*

ClientesCumplen $\Leftarrow \sigma_{DNI>22222222}(Cliente)$.

$\pi_{nombre,apellido,DNI,telefono,direccion}(ClientesCumplen)$.

(2) *Listar nombre, apellido, DNI, teléfono y dirección de clientes con DNI superior a 22.222.222 y que tengan facturas cuyo total no supere los $100.000.*

FacturasCumplen $\Leftarrow \pi_{idCliente}(\sigma_{total\leq 100000}(Factura))$
ClientesCumplen $\Leftarrow \sigma_{DNI>22222222}(Cliente)$

$\pi_{nombre,apellido,DNI,telefono,direccion}(FacturasCumplen|X|ClientesCumplen)$.

(3) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que realizaron compras durante 2020.*

Facturas2020 $\Leftarrow \pi_{idCliente}(\sigma_{(fecha\geq '01/01/2020')\wedge(fecha\leq '31/12/2020')}(Factura))$.

$\pi_{nombre,apellido,DNI,telefono,direccion}(Facturas2020|X|Cliente)$.

(4) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que no realizaron compras durante 2020.*

Facturas2020 $\Leftarrow \pi_{idCliente}(\sigma_{(fecha\geq '01/01/2020')\wedge(fecha\leq '31/12/2020')}(Factura))$.
Clientes2020 $\Leftarrow$ Facturas2020 $|X|$ Cliente.

$\pi_{nombre,apellido,DNI,telefono,direccion}(Clientes - Clientes2020)$.

(5) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que sólo tengan compras durante 2020.*

FacturasFuera2020 $\Leftarrow \pi_{idCliente}(\sigma_{(fecha<'01/01/2020')\vee(fecha>'31/12/2020')}(Factura))$.
ClientesFuera2020 $\Leftarrow$ FacturasFuera2020 $|X|$ Cliente.

$\pi_{nombre,apellido,DNI,telefono,direccion}(Cliente - ClientesFuera2020)$.

(6) *Listar nombre, descripcion, precio y stock de productos no vendidos.*

ProductosVendidos $\Leftarrow$
$\pi_{idProducto,nombreP,descripcion,precio,stock}(Detalle|X|Producto)$.

$\pi_{nombreP,descripcion,precio,stock}(Producto - ProductosVendidos)$.

(7) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que no compraron el producto con nombre 'ProductoX' durante 2020.*

Facturas2020 $\Leftarrow$
$\pi_{nroTicket,idCliente}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(Factura))$.
ProductoX $\Leftarrow \pi_{idProducto}(\sigma_{nombreP='ProductoX'}(Producto))$.
ClientesNoCumplen $\Leftarrow$
$\pi_{idCliente,nombre,apellido,DNI,telefono,direccion}(Detalle|X|Facturas2020|X|ProductoX|X|Cliente)$
.

$\pi_{nombre,apellido,DNI,telefono,direccion}(Cliente - ClientesNoCumplen)$.

(8) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que compraron el producto con nombre 'Producto A' y no compraron el producto con nombre 'Producto B'.*

ProductoA $\Leftarrow \pi_{idProducto}(\sigma_{nombreP='Producto\ A'}(Producto))$.
ProductoB $\Leftarrow \pi_{idProducto}(\sigma_{nombreP='Producto\ B'}(Producto))$.
ClientesA $\Leftarrow$
$\pi_{nombre,apellido,DNI,telefono,direccion}(Detalle|X|Factura|X|ProductoA|X|Cliente)$.
ClientesB $\Leftarrow$
$\pi_{nombre,apellido,DNI,telefono,direccion}(Detalle|X|Factura|X|ProductoB|X|Cliente)$.

ClientesA - ClientesB.

(9) *Listar nroTicket, total, fecha, hora y DNI del cliente de aquellas facturas donde se haya comprado el producto 'Producto C'.*

ProductoC $\Longleftarrow \pi_{idProducto}(\sigma_{nombreP='Producto\ C'}(Producto))$.

$\pi_{nroTicket,total,fecha,hora,DNI}(Detalle|X|Factura|X|ProductoC|X|Cliente)$.

(10) *Agregar un producto con id de producto 1.000, nombre "Producto Z", descripción "mi producto", precio $10.000 y stock 1.000. Se supone que el idProducto 1.000 no existe.*

Producto $\Longleftarrow$ Producto $\cup$ {(1000, 'Producto Z', 'mi producto', 10000, 1000)}.

Ejercicio 2.

Banda= ($\underline{codigoB}$*, nombreBanda, genero_musical, año_creacion).*
Integrante= ($\underline{DNI}$*, nombre, apellido, direccion, email, fecha_nacimiento, codigoB(Fk)).*
Escenario= ($\underline{nroEscenario}$*, nombre_escenario, ubicacion, cubierto, m2, descripcion).*
Recital= ($\underline{fecha, hora, nroEscenario(Fk)}$*, codigoB(Fk)).*

(1) *Listar datos personales de integrantes con apellido 'García' o fecha de nacimiento anterior a 2005, que toquen en bandas de rock and roll.*

IntegrantesCumplen ⇐
$\sigma_{(apellido='García')\vee(fecha\_nacimiento<'01/01/2005')}(Integrantes)$.
BandasCumplen ⇐ $\pi_{codigoB}(\sigma_{genero\_musical='rock\ and\ roll'}(Banda))$.

$\pi_{DNI,nombre,apellido,direccion,email,fecha\_nacimiento}(IntegrantesCumplen|X|BandasCumplen)$
.

(2) *Listar nombre de escenario, ubicación y descripción de escenarios que no tuvieron recitales durante 2019.*

Recitales2019 ⇐ $\pi_{nroEscenario}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(Recital))$.
Escenarios2019 ⇐
$\pi_{nroEscenario,nombre\_escenario,ubicacion,cubierto,m2,descripcion}(Recitales2019|X|Escenario)$
.

$\pi_{nombre\_escenario,ubicacion,descripcion}(Escenario - Escenarios2019)$.

(3) *Listar nombre de escenario, ubicación y descripción de escenarios que tuvieron recitales con género musical 'rock and roll' o tuvieron recitales durante 2020.*

$\pi_{nombre\_escenario,ubicacion,descripcion}$
$((\sigma_{((fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020'))\vee(genero\_musical='rock\ and\ roll')}(Recital|X|Banda))|X|Escenario)$
.

(4) *Listar nombre, género musical y año de creación de bandas que hayan realizado recitales en escenarios cubiertos durante 2019. // cubierto es true, false según corresponda.*

Recitales2019 ⇐
$\pi_{nroEscenario,codigoB}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(Recital))$.
EscenariosCumplen ⇐ $\pi_{nroEscenario}(\sigma_{cubierto=true}(Escenario))$.

$\pi_{nombreBanda,genero\_musical,año\_creacion}(Recitales2019|X|EscenariosCumplen|X|Banda)$
.

(5) *Listar DNI, nombre, apellido, dirección y email de integrantes nacidos entre 2000 y 2005 y que toquen en bandas con género 'pop' que hayan tenido recitales durante 2020.*

Recitales2020 ⇐ $\pi_{codigoB}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(Recital))$
BandasCumplen ⇐ $\pi_{codigoB}(\sigma_{genero\_musical='pop'}(Banda))$.
IntegrantesCumplen ⇐
$\sigma_{(fecha\_nacimiento\geq'01/01/2000')\wedge(fecha\_nacimiento\leq'31/12/2005')}(Integrantes)$.

$\pi_{DNI,nombre,apellido,direccion,email}(Recitales2020|X|BandasCumplen|X|IntegrantesCumplen)$
.

(6) *Listar DNI, nombre, apellido, email de integrantes que hayan tocado en el escenario con nombre 'Gustavo Cerati' y no hayan tocado en el escenario con nombre 'Carlos Gardel'.*

EscenarioGC ⇐ $\pi_{nroEscenario}(\sigma_{nombre\_escenario='Gustavo\ Cerati'}(Escenario))$.
EscenarioCG ⇐ $\pi_{nroEscenario}(\sigma_{nombre\_escenario='Carlos\ Gardel'}(Escenario))$.
IntegrantesGC ⇐
$\pi_{DNI,nombre,apellido,direccion,email}(Recital|X|EscenarioGC|X|Integrante)$.
IntegrantesCG ⇐
$\pi_{DNI,nombre,apellido,direccion,email}(Recital|X|EscenarioCG|X|Integrante)$.

IntegrantesGC - IntegrantesCG.

(7) *Modificar el año de creación de la banda de nombre 'Ratones Paranoicos' a 1983.*

δ año_creacion ⇐ 1983 ($\sigma_{nombreBanda='Ratones\ Paranoicos'}(Banda)$).

(8) *Reportar nombre, género musical y año de creación de bandas que hayan realizado recitales durante 2019 y, además, hayan tocado durante 2020.*

Recitales2019 ⇐ $\pi_{codigoB}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(Recital))$.
Recitales2020 ⇐ $\pi_{codigoB}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(Recital))$.
Bandas2019 ⇐ Recitales2019 $|X|$ Banda.
Bandas2020 ⇐ Recitales2020 $|X|$ Banda.

$\pi_{nombreBanda,genero\_musical,año\_creacion}(Bandas2019 \cap Bandas2020)$.

(9) *Listar el cronograma de recitales del día 04/12/2019. Se deberá listar: nombre de la banda que ejecutará el recital, fecha, hora, y el nombre y ubicación del escenario correspondiente.*

RecitalesCumplen $\Leftarrow \sigma_{fecha='04/12/2019'}(Recital)$.

$\pi_{nombreBanda,fecha,hora,nombre\_escenario,ubicacion}(RecitalesCumplen|X|Escenario|X|Banda)$
.

Ejercicio 3.

Agencia= ($\underline{razon\_social}$*, direccion, telef, e-mail).*
Ciudad= ($\underline{codigoPostal}$*,nombreCiudad, añoCreacion).*
Cliente= ($\underline{DNI}$*, nombre, apellido, telefono, direccion).*
Viaje= ($\underline{fecha, hora, DNI(Fk)}$*, cpOrigen(Fk), cpDestino(Fk), razon_social(Fk), descripcion). // cpOrigen y cpDestino corresponden a las ciudades origen y destino del viaje, respectivamente.*

(1) *Eliminar el cliente con DNI 25.326.992.*

ClienteEliminar $\Leftarrow \sigma_{DNI=25326992}(Cliente)$.
ViajesEliminar $\Leftarrow$
$\pi_{fecha,hora,DNI,cpOrigen,cpDestino,razon\_social,descripcion}(Viaje|X|ClienteEliminar)$.

Viaje $\Leftarrow$ Viaje - ViajesEliminar.
Cliente $\Leftarrow$ Cliente - ClienteEliminar.

(2) *Listar datos personales de clientes que sólo realizaron viajes locales. Se consideran viajes locales aquellos que tienen la misma ciudad como origen y destino.*

ViajesNoCumplen $\Leftarrow \pi_{DNI}(\sigma_{cpOrigen<>cpDestino}(Viaje))$.
ClientesNoCumplen $\Leftarrow$ ViajesNoCumplen $|X|$ Cliente.

Cliente - ClientesNoCumplen.

(3) *Listar información de agencias que no tengan viajes para el cliente con DNI 22.222.222 durante el primer semestre de 2020.*

ViajesNoCumplen $\Leftarrow$
$\pi_{DNI,razon\_social}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'30/06/2020')}(Viaje))$.
ClienteNoCumple $\Leftarrow \pi_{DNI}(\sigma_{DNI=22222222}(Cliente))$.
AgenciasNoCumplen $\Leftarrow$
$\pi_{razon\_social,direccion,telef,e-mail}(ViajesNoCumplen|X|ClienteNoCumple|X|Agencia)$
.

Agencia - AgenciasNoCumplen.

(4) *Listar información de agencias que realizaron viajes durante 2019 y no realizaron viajes durante 2020.*

Viajes2019 ⇐ $\pi_{razon\_social}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(Viaje))$.
Viajes2020 ⇐ $\pi_{razon\_social}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(Viaje))$.
Agencias2019 ⇐ Viajes2019 $|X|$ Agencia.
Agencias2020 ⇐ Viajes2020 $|X|$ Agencia.

Agencias2019 - Agencias2020.

(5) *Agregar una agencia de viajes con los datos que desee.*

Agencia ⇐ Agencia ∪ {('JIM', '13 22', '2211234567', 'jim@gmail.com')}.

(6) *Listar datos personales de clientes que viajaron con destino a la ciudad de 'Lincoln', pero no realizaron viajes con origen en 'La Plata'.*

Lincoln ⇐ $\pi_{codigoPostal}(\sigma_{nombreCiudad='Lincoln'}(Ciudad))$.
LaPlata ⇐ $\pi_{codigoPostal}(\sigma_{nombreCiudad='La\ Plata'}(Ciudad))$.
ViajesDestinoLincoln ⇐ $\pi_{DNI}(\sigma_{cpDestino=codigoPostal}(Viaje\ X\ Lincoln))$.
ViajesOrigenLaPlata ⇐ $\pi_{DNI}(\sigma_{cpOrigen=codigoPostal}(Viaje\ X\ LaPlata))$.
ClientesDestinoLincoln ⇐ ViajesDestinoLincoln $|X|$ Cliente.
ClientesOrigenLaPlata ⇐ ViajesOrigenLaPlata $|X|$ Cliente.

ClientesDestinoLincoln - ClientesOrigenLaPlata.

(7) *Listar nombre, apellido, dirección y teléfono de clientes que viajaron con todas las agencias.*

$(\pi_{razon\_social,nombre,apellido,direccion,telefono}(Viaje|X|Cliente))\%(\pi_{razon\_social}(Agencia))$
.

(8) *Listar código postal, nombre y año de creación de ciudades que no recibieron viajes durante 2020.*

Viajes2020 ⇐ $\pi_{cpDestino}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(Viaje))$.
Ciudades2020 ⇐
$\pi_{codigoPostal,nombreCiudad,añoCreacion}(\sigma_{cpDestino=codigoPostal}(Viajes2020\ X\ Ciudad))$
.

Ciudad - Ciudades2020.

(9) *Reportar información de agencias que realizaron viajes durante 2019 o que tengan dirección igual a 'General Pinto 1234'.*

$\pi_{razon\_social,direccion,telef,e-mail}(\sigma_{((fecha \geq '01/01/2019') \wedge (fecha \leq '31/12/2019')) \vee (direccion='General\ Pinto\ 1234')}(Viaje|X|Agencia))$
.

(10) *Actualizar el teléfono del cliente con DNI 2.789.655 al siguiente número de teléfono: 221-4400345.*

δ telefono $\Leftarrow$ '221-4400345' ($\sigma_{DNI=2789655}(Cliente)$).

Ejercicio 4.

Equipo= ($\underline{codigoE}$*, nombreE, descripcionE).*
Integrante= ($\underline{DNI}$*, nombre, apellido, ciudad, email, telefono, codigoE(Fk)).*
Laguna= ($\underline{nroLaguna}$*, nombreL, ubicacion, extension, descripcion).*
TorneoPesca= ($\underline{codTorneo}$*, fecha, hora, nroLaguna(Fk), descripcion).*
Inscripcion= ($\underline{codTorneo(Fk), codigoE(Fk)}$*, asistio, gano). // asistio y gano son true/false.*

(1) *Listar DNI, nombre, apellido y email de integrantes que sean de la ciudad 'La Plata' y estén inscriptos en torneos que se disputaron durante 2019.*

IntegrantesLP $\Leftarrow \pi_{DNI,nombre,apellido,email,codigoE}(\sigma_{ciudad='La\ Plata'}(Integrante))$.
Torneos2019 $\Leftarrow \pi_{codTorneo}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(TorneoPesca))$.

$\pi_{DNI,nombre,apellido,email}(Inscripcion|X|Torneos2019|X|IntegrantesLP)$.

(2) *Reportar nombre y descripción de equipos que sólo se hayan inscripto en torneos de 2019.*

TorneosFuera2019 $\Leftarrow \sigma_{(fecha<'01/01/2019')\vee(fecha>'31/12/2019')}(TorneoPesca)$.
InscripcionesFuera2019 $\Leftarrow \pi_{codigoE}(Inscripciones|X|TorneosFuera2019)$.
Torneos2019 $\Leftarrow \sigma_{(fecha\geq'01/01/2019')\vee(fecha\leq'31/12/2019')}(TorneoPesca)$.
Inscripciones2019 $\Leftarrow \pi_{codigoE}(Inscripciones|X|Torneos2019)$.
InscripcionesSolo2019 $\Leftarrow$ Inscripciones2019 - InscripcionesFuera2019.

$\pi_{nombreE,descripcionE}(InscripcionesSolo2019|X|Equipo)$.

(3) *Listar nombre, ubicación, extensión y descripción de lagunas que hayan tenido torneos durante 2019 y no hayan tenido torneos durante 2020.*

Torneos2019 $\Leftarrow \pi_{nroLaguna}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(TorneoPesca))$.
Torneos2020 $\Leftarrow \pi_{nroLaguna}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(TorneoPesca))$.
TorneosCumplen $\Leftarrow$ Torneos2019 - Torneos2020.

$\pi_{nombreL,ubicacion,extension,descripcion}(TorneosCumplen|X|Laguna)$.

(4) *Listar, para la laguna con nombre 'laguna x', nombre y descripción de equipos ganadores de torneos que se disputaron durante 2019 en la mencionada laguna.*

InscripcionesCumplen $\Leftarrow \pi_{codTorneo,codigoE}(\sigma_{gano=true}(Inscripcion))$.
Torneos2019 $\Leftarrow$
$\pi_{codTorneo,nroLaguna}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(TorneoPesca))$.
LagunaX $\Leftarrow \pi_{nroLaguna}(\sigma_{nombreL='laguna\ x'}(Laguna))$.

$\pi_{nombreE,descripcionE}(InscripcionesCumplen|X|Torneos2019|X|Equipo|X|LagunaX)$
.

(5) *Reportar nombre y descripción de equipos que tengan inscripciones en todas las lagunas.*

TorneoPescaAux $\Leftarrow \pi_{codTorneo,nroLaguna}(TorneoPesca)$.

$(\pi_{nombreE,descripcionE,nroLaguna}(Inscripcion|X|TorneoPescaAux|X|Equipo))$
$\%(\pi_{nroLaguna}(Laguna))$.

(6) *Eliminar el equipo con código 10.000.*

EquipoEliminar $\Leftarrow \sigma_{codigoE=10000}(Equipo)$.
InscripcionesEliminar $\Leftarrow$
$\pi_{codTorneo,codigoE,asistio,gano}(Inscripcion|X|EquipoEliminar)$.
IntegrantesEliminar $\Leftarrow$
$\pi_{DNI,nombre,apellido,ciudad,email,telefono,codigoE}(Integrante|X|EquipoEliminar)$.

Inscripcion $\Leftarrow$ Inscripcion - InscripcionesEliminar.
Integrante $\Leftarrow$ Integrante - IntegrantesEliminar.
Equipo $\Leftarrow$ Equipo - EquipoEliminar.

(7) *Listar nombre, ubicación, extensión y descripción de lagunas que no tuvieron torneos.*

TorneoPescaAux $\Leftarrow \pi_{nroLaguna}(TorneoPesca)$.
LagunasConTorneo $\Leftarrow$ TorneoPescaAux $|X|$ Laguna.

$\pi_{nombreL,ubicacion,extension,descripcion}(Laguna - LagunasConTorneo)$.

(8) *Reportar nombre y descripción de equipos que tengan inscripciones a torneos a disputarse durante 2019, pero no tienen inscripciones a torneos de 2020.*

Torneos2019 $\Leftarrow \pi_{codTorneo}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(TorneoPesca))$.
Torneos2020 $\Leftarrow \pi_{codTorneo}(\sigma_{(fecha\geq'01/01/2020')\wedge(fecha\leq'31/12/2020')}(TorneoPesca))$.

Equipos2019 $\Leftarrow$
$\pi_{codigoE,nombreE,descripcionE}(Inscripcion|X|Torneos2019|X|Equipo)$.
Equipos2020 $\Leftarrow$
$\pi_{codigoE,nombreE,descripcionE}(Inscripcion|X|Torneos2020|X|Equipo)$.

$\pi_{nombreE,descripcionE}(Equipos2019 - Equipos2020)$.

(9) *Listar DNI, nombre, apellido, ciudad y email de integrantes que asistieron o ganaron algún torneo que se disputó en la laguna con nombre 'Laguna Brava'.*

TorneoPescaAux $\Leftarrow \pi_{codTorneo,nroLaguna}(TorneoPesca)$.
LagunaBrava $\Leftarrow \pi_{nroLaguna}(\sigma_{nombreL='Laguna\ Brava'}(Laguna))$.
TorneosBrava $\Leftarrow \pi_{codTorneo}(TorneoPescaAux|X|LagunaBrava)$.
InscripcionesCumplen $\Leftarrow$
$\pi_{codigoE}((\sigma_{(asistio=true)\vee(gano=true)}(Inscripcion))|X|TorneosBrava)$.

$\pi_{DNI,nombre,apellido,ciudad,email}(InscripcionesCumplen|X|Integrante)$.

Ejercicio 5.

Club= ($\underline{codigoClub}$*, nombre, anioFundacion, codigoCiudad(Fk)).*
Ciudad= ($\underline{codigoCiudad}$*, nombre).*
Estadio= ($\underline{codigoEstadio}$*, codigoClub(Fk), nombre, direccion).*
Jugador= ($\underline{DNI}$*, nombre, apellido, edad, codigoCiudad(Fk)).*
ClubJugador= ($\underline{codigoClub(Fk), DNI(Fk), desde}$*, hasta).*

(1) *Reportar nombre y año de fundación de clubes de la ciudad de La Plata, además del nombre y dirección del estadio del mismo.*

LaPlata $\Leftarrow \pi_{codigoCiudad}(\sigma_{nombre='La\ Plata'}(Ciudad))$.

$\pi_{club.nombre,anioFundacion,estadio.nombre,direccion}$
$(\sigma_{club.codigoClub=estadio.codigoClub}((Club|X|LaPlata)\ X\ Estadio))$.

(2) *Listar datos personales de jugadores actuales del club River Plate que hayan jugado en el club Boca Juniors.*

River $\Leftarrow \pi_{codigoClub}(\sigma_{nombre='River'}(Club))$.
Boca $\Leftarrow \pi_{codigoClub}(\sigma_{nombre='Boca'}(Club))$.

JugadoresActualesRiver $\Leftarrow$
$\pi_{DNI,nombre,apellido,edad,codigoCiudad}(Jugador|X|(\sigma_{hasta=null}(ClubJugador))|X|River)$
.
JugadoresViejosBoca $\Leftarrow$
$\pi_{DNI,nombre,apellido,edad,codigoCiudad}(Jugador|X|(\sigma_{hasta<>null}(ClubJugador))|X|Boca)$
.

JugadoresActualesRiver ∩ JugadoresViejosBoca.

(3) *Listar información de todos los clubes donde se desempeñó el jugador Marcelo Gallardo. Indicar nombre, año de fundación y ciudad del club.*

JugadorMG $\Leftarrow \pi_{DNI}(\sigma_{(nombre='Marcelo')\wedge(apellido='Gallardo')}(Jugador))$
ClubesMG $\Leftarrow \pi_{codigoClub}(JugadorMG|X|ClubJugador))$.

$\pi_{club.nombre,anioFundacion,ciudad.nombre}$
$(ClubesMG|X|(\sigma_{club.codigoCiudad=ciudad.codigoCiudad}(Club\ X\ Ciudad)))$.

(4) *Reportar DNI, nombre y apellido de aquellos jugadores que no tengan más de 25 años y jueguen en algún club de la ciudad de Junín.*

Jugadores25 $\Leftarrow \pi_{DNI,nombre,apellido}(\sigma_{edad\leq 25}(Jugador))$.
Junin $\Leftarrow \pi_{codigoCiudad}(\sigma_{nombre='Junin'}(Ciudad))$.
ClubesJunin $\Leftarrow \pi_{codigoClub}(Club|X|Junin)$.

$\pi_{DNI,nombre,apellido}(Jugadores25|X|ClubJugador|X|ClubesJunin)$.

(5) *Mostrar el nombre de los clubes que tengan jugadores de la ciudad de Chivilcoy mayores de 25 años.*

Chivilcoy $\Leftarrow \pi_{codigoCiudad}(\sigma_{nombre='Chivilcoy'}(Ciudad))$.
Jugadores25 $\Leftarrow \pi_{DNI,codigoCiudad}(\sigma_{edad>25}(Jugador))$.
JugadoresCumplen $\Leftarrow \pi_{DNI}(Jugadores25|X|Chivilcoy)$.

$\pi_{nombre}(JugadoresCumplen|X|ClubJugador|X|Club)$.

(6) *Reportar el nombre y apellido de aquellos jugadores que hayan jugado en todos los clubes.*

$(\pi_{nombre,apellido,codigoClub}(Jugador|X|ClubJugador))\%(\pi_{codigoClub}(Club))$.

(7) *Listar nombre de los clubes que no hayan tenido ni tengan jugadores de la ciudad de La Plata.*

LaPlata $\Leftarrow \pi_{codigoCiudad}(\sigma_{nombre='La\ Plata'}(Ciudad))$.
JugadoresLP $\Leftarrow \pi_{DNI}(Jugador|X|LaPlata)$.
ClubesLP $\Leftarrow$
$\pi_{codigoClub,nombre,anioFundacion,codigoCiudad}(JugadoresLP|X|ClubJugador|X|Club)$.

$\pi_{nombre}(Club - ClubesLP)$.

(8) *Mostrar DNI, nombre y apellido de aquellos jugadores que jugaron o juegan en el club 'Club Atlético Rosario Central'.*

RosarioCentral $\Leftarrow \pi_{codigoClub}(\sigma_{nombre='Club\ Atlético\ Rosario\ Central'}(Club))$.

$\pi_{DNI,nombre,apellido}(Jugador|X|ClubJugador|X|RosarioCentral)$.

(9) *Eliminar al jugador cuyo DNI es 24.242.424.*

JugadorEliminar $\Leftarrow \sigma_{DNI=24242424}(Jugador)$.
ClubesJugadorEliminar $\Leftarrow$
$\pi_{codigoClub,DNI,desde,hasta}(ClubJugador|X|JugadorEliminar)$.

ClubJugador $\Leftarrow$ ClubJugador - ClubesJugadorEliminar.
Jugador $\Leftarrow$ Jugador - JugadorEliminar.

Ejercicio 6.

Proyecto= ($\underline{codProyecto}$*, nombreP, descripcion, fechaInicioP, fechaFinP, fechaFinEstimada, DNIResponsable(Fk), equipoBackend(Fk), equipoFrontend(Fk)). // DNIResponsable corresponde a un empleado, equipoBackend y equipoFrontend corresponden a un equipo.*
Equipo= ($\underline{codEquipo}$*, nombreE, descripcionTecnologias, DNILider(Fk)). // DNILider corresponde a un empleado.*
Empleado= ($\underline{DNI}$*, nombre, apellido, telefono, direccion, fechaIngreso).*
Empleado_Equipo= ($\underline{codEquipo(Fk), DNI(Fk), fechaInicio}$*, fechaFin, descripcionRol).*

(1) *Listar nombre, descripción, fecha de inicio y fecha de fin de proyectos ya finalizados que no fueron terminados antes de la fecha de fin estimada.*

ProyectosCumplen ⇐ $\sigma_{(fechaFinP<>null)\wedge(fechaFinP\geq fechaFinEstimada)}(Proyecto)$.

$\pi_{nombreP,descripcion,fechaInicioP,fechaFinP}(ProyectosCumplen)$.

(2) *Listar DNI, nombre, apellido, teléfono, dirección y fecha de ingreso de empleados que no hayan sido responsables de proyectos.*

EmpleadosResponsables ⇐
$\pi_{DNI,nombre,apellido,telefono,direccion,fechaIngreso}$
$(\sigma_{(fechaFinP<>null)\wedge(DNIResponsable=DNI)}(Proyecto\ X\ Empleado))$.

Empleado - EmpleadosResponsables.

(3) *Listar DNI, nombre, apellido, teléfono y dirección de todos los empleados que trabajan en el proyecto con nombre 'Proyecto X'. No es necesario informar responsable y líderes.*

EquiposBackend ⇐
$\pi_{equipoBackend}(\sigma_{(fechaFinP=null)\wedge(nombreP='Proyecto\ X')}(Proyecto))$.
EquiposFrontend ⇐
$\pi_{equipoFrontend}(\sigma_{(fechaFinP=null)\wedge(nombreP='Proyecto\ X')}(Proyecto))$.
EmpleadosBackend ⇐
$\pi_{DNI,nombre,apellido,telefono,direccion}(Empleado|X|(\sigma_{codEquipo=equipoBackend}(Empleado\_Equipo\ X\ EquiposBackend)))$
.
EmpleadosFrontend ⇐
$\pi_{DNI,nombre,apellido,telefono,direccion}(Empleado|X|(\sigma_{codEquipo=equipoFrontend}(Empleado\_Equipo\ X\ EquiposFrontend)))$
.

EmpleadosBackend ∪ EmpleadosFrontend.

(4) *Listar nombre de equipo y datos personales de líderes de equipos que no tengan empleados asignados y trabajen con tecnología 'Java'.*

EquiposConEmpleados ⇐
$\pi_{codEquipo,nombreE,descripcionTecnologias,DNILider}(Empleado\_Equipo|X|Equipo)$.
EquiposSinEmpleados ⇐ Equipo - EquiposConEmpleados
EquiposCumplen ⇐
$\pi_{nombreE,DNILider}(\sigma_{descripcionTecnologias='Java'}(EquiposSinEmpleados))$.

$\pi_{nombreE,DNI,nombre,apellido,telefono,direccion,fechaIngreso}$
$(\sigma_{DNILider=DNI}(EquiposCumplen\ X\ Empleado))$.

(5) *Modificar nombre, apellido y dirección del empleado con DNI 40.568.965 con los datos que desee.*

δ nombre ⇐ 'Juan' $(\sigma_{DNI=40568965}(Empleado))$.
δ apellido ⇐ 'Menduiña' $(\sigma_{DNI=40568965}(Empleado))$.
δ direccion ⇐ '13 Nro. 22' $(\sigma_{DNI=40568965}(Empleado))$.

(6) *Listar DNI, nombre, apellido, teléfono y dirección de empleados que son responsables de proyectos, pero no han sido líderes de equipo.*

EmpleadosResponsables ⇐
$\pi_{DNI,nombre,apellido,telefono,direccion}(\sigma_{(fechaFinP=null)\wedge(DNIResponsable=DNI)}(Proyecto\ X\ Empleado))$
.
EmpleadosLideres ⇐
$\pi_{DNI,nombre,apellido,telefono,direccion}(\sigma_{DNILider=DNI}(Equipo\ X\ Empleado))$.

EmpleadosResponsables - EmpleadosLideres.

(7) *Listar nombre de equipo y descripción de tecnologías de equipos que hayan sido asignados como equipos frontend y backend.*

EquiposBackend ⇐
$\pi_{codEquipo,nombreE,descripcionTecnologias}(\sigma_{equipoBackend=codEquipo}(Proyecto\ X\ Equipo))$
.
EquiposFrontend ⇐
$\pi_{codEquipo,nombreE,descripcionTecnologias}(\sigma_{equipoFrontend=codEquipo}(Proyecto\ X\ Equipo))$
.

$\pi_{nombreE,descripcionTecnologias}(EquiposBackend \cap EquiposFrontend)$.

(8) *Listar nombre, descripción, fecha de inicio, nombre y apellido de responsables de proyectos a finalizar durante 2019.*

Proyectos2019 ⇐
$\pi_{nombreP,descripcion,fechaInicioP,DNIResponsable}$
$(\sigma_{(fechaFinP=null)\wedge((fechaFinEstimada\geq'01/01/2019')\wedge(fechaFinEstimada\leq'31/12/2019'))}(Proyecto))$
.

$\pi_{nombreP,descripcion,fechaInicioP,nombre,apellido}(\sigma_{DNIResponsable=DNI}(Proyectos2019\ X\ Empleado))$
.

(9) *Listar nombre de equipo, descripción de tecnología y la información personal del líder, de equipos que no estén asignados a ningún proyecto aún.*

EquiposBackend ⇐
$\pi_{codEquipo,nombreE,descripcionTecnologias,DNILider}$
$(\sigma_{(fechaFinP=null)\wedge(equipoBackend=codEquipo)}(Proyecto\ X\ Equipo))$.
EquiposFrontend ⇐
$\pi_{codEquipo,nombreE,descripcionTecnologias,DNILider}$
$(\sigma_{(fechaFinP=null)\wedge(equipoFrontend=codEquipo)}(Proyecto\ X\ Equipo))$.
EquiposSinProyecto ⇐
$\pi_{nombreE,descripcionTecnologias,DNILider}$
$(Equipo - (EquiposBackend \cup EquiposFrontend))$.

$\pi_{nombreE,descripcionTecnologias,DNI,nombre,apellido,telefono,direccion,fechaIngreso}$
$(\sigma_{DNILider=DNI}(EquiposSinProyecto\ X\ Empleado))$.

Ejercicio 7.

Vehiculo= ($\underline{patente}$*, modelo, marca, peso, km).*
Camion= ($\underline{patente(Fk)}$*, largo, max_toneladas, cant_ruedas, tiene_acoplado).*
Auto= ($\underline{patente(Fk)}$*, es_electrico, tipo_motor).*
Service= ($\underline{fecha, patente(Fk)}$*, km_service, observaciones, monto).*
Parte= ($\underline{cod\_parte}$*, nombre, precio_parte).*
Service_Parte= ($\underline{[fecha, patente](Fk), cod\_parte(Fk)}$*, precio, cantidad).*

(1) *Listar todos los datos de aquellos camiones que tengan entre 8 y 12 ruedas, y que hayan realizado algún service antes de los 10.000 km.*

CamionesCumplen ⇐
$\pi_{patente,largo,max\_toneladas,cant\_ruedas,tiene\_acoplado}(\sigma_{(cant\_ruedas\geq 8)\wedge(cant\_ruedas\leq 12)}(Camion))$
.
ServicesCumplen ⇐ $\pi_{patente}(\sigma_{km\_service<10000}(Service))$.

$\pi_{patente,modelo,marca,peso,km,largo,max\_toneladas,cant\_ruedas,tiene\_acoplado}$
$(ServicesCumplen|X|Vehiculo|X|CamionesCumplen)$.

(2) *Listar los autos que hayan realizado el service 'cambio de aceite' antes de los 13.000 km o hayan realizado el service 'inspección general' que incluya la parte 'filtro de combustible'.*

ServicesCumplen1 ⇐
$\pi_{patente}(\sigma_{(observaciones='cambio\ de\ aceite')\wedge(km\_service<13000)}(Service))$.
ServicesCumplen2 ⇐
$\pi_{patente}(\sigma_{observaciones='inspección\ general'}(Service))$.
AutosCumplen1 ⇐
$\pi_{patente,modelo,marca,peso,km,es\_electrico,tipo\_motor}(ServicesCumplen1|X|Auto)$.
AutosCumplen2 ⇐
$\pi_{patente,modelo,marca,peso,km,es\_electrico,tipo\_motor}$
$(Service\_Parte|X|ServicesCumplen2|X|(\sigma_{nombre='filtro\ de\ combustible'}(Parte))|X|Auto)$
.

AutosCumplen1 ∪ AutosCumplen2.

(3) *Dar de baja todos los camiones con más de 350.000 km.*

CamionesBaja ⇐
$\pi_{patente,largo,max\_toneladas,cant\_ruedas,tiene\_acoplado}((\sigma_{km>350000}(Vehiculo))|X|Camion)$
.

ServicesParteBaja ⇐
$\pi_{fecha,patente,cod\_parte,precio,cantidad}(Service\_Parte|X|CamionesBaja)$.
ServicesBaja ⇐
$\pi_{fecha,patente,km\_service,observaciones,monto}(Service|X|CamionesBaja)$.
VehiculosBaja ⇐ $\pi_{patente,modelo,marca,peso,km}(Vehiculo|X|CamionesBaja)$.

Service_Parte ⇐ Service_Parte - ServicesParteBaja.
Service ⇐ Service - ServicesBaja.
Camion ⇐ Camion - CamionesBaja.
Vehiculo ⇐ Vehiculo - VehiculosBaja.

(4) *Listar el nombre y precio de aquellas partes que figuren en todos los services realizados durante 2019.*

Services2019 ⇐ $\pi_{fecha,patente}(\sigma_{(fecha\geq'01/01/2019')(fecha\leq'31/12/2019')}(Service))$.

$\pi_{fecha,patente,nombre,precio\_parte}(Service\_Parte|X|Parte))\%(\pi_{fecha,patente}(Services2019))$
.

(5) *Listar todos los autos que sean eléctricos. Mostrar información de patente, modelo, marca y peso.*

AutosCumplen ⇐ $\pi_{patente}(\sigma_{es\_electrio=true}(Auto))$.

$\pi_{patente,modelo,marca,peso}(Vehiculo|X|AutosCumplen)$.

(6) *Dar de alta una parte, cuyo nombre sea 'Aleron' y precio $3.400.*

Parte ⇐ Parte ∪ {(100, 'Aleron', 3400)}. (*)

(*) Se supone que el cod_parte 100 no existe.

(7) *Dar de baja todos los services que se realizaron al auto con patente 'AAA564'.*

ServicesParteBaja ⇐ $\sigma_{patente='AAA564'}(Service\_Parte)$.
ServicesBaja ⇐
$\pi_{fecha,patente,km\_service,observaciones,monto}(Service|X|ServicesParteBaja)$.

Service_Parte ⇐ Service_Parte - ServicesParteBaja.
Service ⇐ Service - ServicesBaja.

(8) *Modificar el precio de las partes incrementando un 15% dicho valor.*

δ precio_parte ⟸ precio_parte * 1,15 (Parte).

(9) *Listar todos los vehículos que hayan tenido services durante el 2019.*

Services2019 ⟸ $\pi_{patente}(\sigma_{(fecha\geq'01/01/2019')\wedge(fecha\leq'31/12/2019')}(Service))$.

Services2019 $|X|$ Vehiculo.

Ejercicio 8.

Barberia= ($\underline{codBarberia}$*, razon_social, direccion, telefono).*
Cliente= ($\underline{nroCliente}$*, DNI, nombreApellidoC, direccionC, fechaNacimiento, celular).*
Barbero= ($\underline{codEmpleado}$*, DNIB, nombreApellidoB, direccionB, telefonoContacto, mail).*
Atencion= ($\underline{codEmpleado(Fk), fecha, hora}$*, codBarberia(Fk), nroCliente(Fk), descTratamiento, valor).*

(1) *Listar DNI, nombre completo, dirección, teléfono de contacto y email de barberos que tengan atenciones con valor superior a 5.000.*

AtencionesCumplen $\Leftarrow \pi_{codEmpleado}(\sigma_{valor>5000}(Atencion))$.

$\pi_{DNIB,nombreApellidoB,direccionB,telefonoContacto,mail}(AtencionesCumplen|X|Barbero)$
.

(2) *Listar DNI, nombre y apellido, dirección, fecha de nacimiento y celular de clientes que tengan atenciones en la barbería con razón social 'Corta barba' y que también se hayan atendido en la barbería con razón social 'Barberia Barbara'.*

BarberiaCB $\Leftarrow \pi_{codBarberia}(\sigma_{razon\_social='Corta\ Barba'}(Barberia))$.
BarberiaBC $\Leftarrow \pi_{codBarberia}(\sigma_{razon\_social='Barberia\ Barbara'}(Barberia))$.
ClientesCB $\Leftarrow$
$\pi_{nroCliente,DNI,nombreApellidoC,direccionC,fechaNacimiento,celular}(Atencion|X|BarberiaCB|X|Cliente)$
.
ClientesBC $\Leftarrow$
$\pi_{nroCliente,DNI,nombreApellidoC,direccionC,fechaNacimiento,celular}(Atencion|X|BarberiaBC|X|Cliente)$
.

$\pi_{DNI,nombreApellidoC,direccionC,fechaNacimiento,celular}(ClientesCB \cap ClientesBC)$.

(3) *Eliminar el cliente con DNI 22.222.222.*

ClienteEliminar $\Leftarrow \sigma_{DNI=22222222}(Cliente)$.
AtencionesEliminar $\Leftarrow$
$\pi_{codEmpleado,fecha,hora,codBarberia,nroCliente,descTratamiento,valor}(Atenciones|X|ClienteEliminar)$
.

Atencion $\Leftarrow$ Atencion - AtencionesEliminar.
Cliente $\Leftarrow$ Cliente - ClienteEliminar.

(4) *Listar los clientes más jóvenes que el cliente con nombre y apellido 'Juan Perez'.*

ClienteJP $\Leftarrow \pi_{fechaNacimiento}(\sigma_{nombreApellidoC='Juan\ Perez'}(Cliente))$. (*)

$\pi_{DNI,nombreApellidoC,direccionC,fechaNacimiento,celular}$
$(\sigma_{cliente.fechaNacimiento>clientejp.fechaNacimiento}(Cliente\ X\ ClienteJP))$.

(*) Se supone que existe un solo cliente 'Juan Perez', de lo contrario habrá más de una fecha de nacimiento en la tabla ClienteJP. Si no se cumple esto, en la consulta final, aparecerán los clientes 'Juan Perez' que sean más jóvenes que el cliente 'Juan Perez' más viejo (excepto que todos los 'Juan Perez' tengan la misma fecha de nacimiento, caso en el cual ningún 'Juan Perez' será más joven que otro 'Juan Perez', y no aparecerá ninguno en la consulta final).

(5) *Listar los clientes que han tenido atenciones con todos los barberos que han trabajado en la barbería con razón social 'Corta Barba'.*

BarberiaCB $\Leftarrow \pi_{codBarberia}(\sigma_{razon\_social='Corta\ Barba'}(Barberia))$.
BarberosCB $\Leftarrow \pi_{codEmpleado}(Atencion|X|BarberiaCB|X|Barbero)$.

$(\pi_{codEmpleado,DNI,nombreApellidoC,direccionC,fechaNacimiento,celular}(Atencion|X|Cliente))$
$\%(\pi_{codEmpleado}(BarberosCB))$.

(6) *Listar DNI y nombre completo de los barberos que sólo tengan atenciones a partir de 2024.*

AtencionesAntes2024 $\Leftarrow \pi_{codEmpleado}(\sigma_{fecha<'01/01/2024'}(Atenciones))$.
AtencionesDespues2024 $\Leftarrow \pi_{codEmpleado}(\sigma_{fecha\geq'01/01/2024'}(Atenciones))$.
AtencionesSoloDespues2024 $\Leftarrow$ AtencionesDespues2024 - AtencionesAntes2024.

$\pi_{DNIB,nombreApellidoB}(AtencionesSoloDespues2024|X|Barbero)$.

(7) *Modificar la dirección de la barbería con razón social 'Pelo & Barba' con su nueva dirección: '13 N° 1234 La Plata'.*

δ direccion $\Leftarrow$ '13 N° 1234 La Plata' $(\sigma_{razon\_social='Pelo\ \&\ Barba'}(Barberia))$.

(8) *Listar los datos de las atenciones realizadas por las barberías durante el mes de Septiembre de 2024, indicando, por cada atención, la razón social de la barbería, el nombre completo del empleado que realizó la atención, el nombre completo del cliente*

que recibió la atención, la fecha y hora, la descripción de los tratamientos aplicados y el valor de la atención.

Atenciones2024 $\Leftarrow$
$\pi_{codEmpleado,codBarberia,nroCliente}(\sigma_{(fecha\geq'01/01/2024')\wedge(fecha\leq'31/12/2024')}(Atencion))$
.

$\pi_{razon\_social,nombreApellidoB,nombreApellidoC,fecha,hora,descTratamiento,valor}$
$(Atenciones2024|X|Barbero|X|Barberia|X|Cliente)$.

Ejercicio 9.

Club= (IdClub, nombre, ciudad).
Complejo= (IdComplejo, nombre, IdClub(Fk)).
Cancha= (IdCancha, nombre, IdComplejo(Fk)).
Entrenador= (IdEntrenador, nombreEntrenador, fechaNacimiento, direccion).
Entrenamiento= (IdEntrenamiento, fecha, IdEntrenador(Fk), IdCancha(Fk)).

(1) *Listar nombre, fecha de nacimiento y dirección de entrenadores que hayan entrenado en la canchas denominadas 'Cancha 1' y 'Cancha 2' del complejo con nombre 'Norte' del club 'Deportivo La Plata'.*

CanchasCumplen ⇐
$\pi_{idCancha,idComplejo}(\sigma_{(nombre='Cancha\ 1')\vee(nombre='Cancha\ 2')}(Cancha))$.
ComplejosCumplen ⇐ $\pi_{idComplejo,idClub}(\sigma_{nombre='Norte'}(Complejo))$.
ClubesCumplen ⇐ $\pi_{idClub}(\sigma_{nombre='Deportivo\ La\ Plata'}(Club))$.

$\pi_{nombreEntrenador,fechaNacimiento,direccion}$
$(Entrenamiento|X|Entrenador|X|CanchasCumplen|X|ComplejosCumplen|X|ClubesCumplen)$
.

(2) *Listar nombre y ciudad de todos los clubes en los que entrena el entrenador 'Marcos Perez'.*

EntrenadorMP ⇐ $\pi_{idEntrenador}(\sigma_{nombreEntrenador='Marcos\ Perez'}(Entrenador))$.
CanchaAux ⇐ $\pi_{idCancha,idComplejo}(Cancha)$.
ComplejoAux ⇐ $\pi_{idComplejo,idClub}(Complejo)$.

$\pi_{nombre,ciudad}(Entrenamiento|X|EntrenadorMP|X|CanchaAux|X|ComplejoAux|X|Club)$
.

(3) *Eliminar los entrenamientos del entrenador 'Hansi Flick'.*

EntrenadorEliminar ⇐ $\sigma_{nombreEntrenador='Hansi\ Flick'}(Entrenador)$.
EntrenamientosEliminar ⇐
$\pi_{idEntrenamiento,fecha,idEntrenador,idCancha}(Entrenamiento|X|EntrenadorEliminar)$
.

Entrenamiento ⇐ Entrenamiento - EntrenamientosEliminar.
Entrenador ⇐ Entrenador - EntrenadorEliminar.

(4) *Listar los nombres de los clubes que se ubican en la misma ciudad que el club con nombre 'Crucero del Sur'.*

ClubCS $\Leftarrow \pi_{ciudad}(\sigma_{nombre='Crucero\ del\ Sur'}(Club))$.

$\pi_{nombre}(Club|X|ClubCS)$.

(5) *Listar nombre y fecha de nacimiento de los entrenadores que hayan realizado, en una misma fecha, entrenamientos en todas las canchas del complejo con nombre 'Centro' del club 'Centro Fomento LH'.*

ComplejosCumplen $\Leftarrow \pi_{idComplejo,idClub}(\sigma_{nombre='Centro'}(Complejo))$.
ClubesCumple $\Leftarrow \pi_{idClub}(\sigma_{nombre='Centro\ Fomento\ LH'}(Club))$.
CanchasCumplen $\Leftarrow \pi_{idCancha}(Cancha|X|ComplejosCumplen|X|ClubesCumplen)$.

$\pi_{nombreEntrenador,fechaNacimiento}$
$((\pi_{idCancha,fecha,nombreEntrenador,fechaNacimiento}(Entrenamiento|X|Entrenador|X|Cancha))$
$\%(\pi_{idCancha}(CanchasCumplen)))$.

(6) *Listar nombre, fecha de nacimiento y dirección de aquellos entrenadores que entrenan clubes de la ciudad 'La Plata', pero que no entrenan clubes de la ciudad 'Berisso'.*

CanchaAux $\Leftarrow \pi_{idCancha,idComplejo}(Cancha)$.
ComplejoAux $\Leftarrow \pi_{idComplejo,idClub}(Complejo)$.
ClubesLaPlata $\Leftarrow \pi_{idClub}(\sigma_{ciudad='La\ Plata'}(Club))$.
ClubesBerisso $\Leftarrow \pi_{idClub}(\sigma_{ciudad='Berisso'}(Club))$.
EntrenadoresLaPlata $\Leftarrow$
$\pi_{idEntrenador,nombreEntrenador,fechaNacimiento,direccion}$
$(Entrenamiento|X|Entrenador|X|CanchaAux|X|ComplejoAux|X|ClubesLaPlata)$
.
EntrenadoresBerisso $\Leftarrow$
$\pi_{idEntrenador,nombreEntrenador,fechaNacimiento,direccion}$
$(Entrenamiento|X|Entrenador|X|CanchaAux|X|ComplejoAux|X|ClubesBerisso)$
.

$\pi_{nombreEntrenador,fechaNacimiento,direccion}(EntrenadoresLaPlata -$
$EntrenadoresBerisso)$.

(7) *Listar la información de las canchas que disponen los clubes de la ciudad 'La Plata'. Por cada resultado, se debe informar el nombre del club, el nombre del complejo y el nombre de la cancha.*

ClubesLP $\Leftarrow \pi_{idClub,nombre}(\sigma_{ciudad='La\ Plata'}(Club))$.
ComplejosLP $\Leftarrow$
$\pi_{idComplejo,complejoslp.nombre,clubeslp.nombre}(\sigma_{complejo.idClub=clubeslp.idClub}(Complejo\ X\ ClubesLP)$
.
CanchasLP $\Leftarrow$
$\pi_{nombre,complejoslp.nombre,clubeslp.nombre}(\sigma_{cancha.idComplejo=complejoslp.idComplejo}(Cancha\ X\ ComplejosLP))$
.

$\pi_{clubeslp.nombre,complejoslp.nombre,nombre}(CanchasLP)$.

Juan Menduiña

Trabajo Práctico N° 4:
SQL.

Ejercicio 1.

Cliente= (idCliente, nombre, apellido, DNI, telefono, direccion).
Factura= (nroTicket, total, fecha, hora, idCliente(Fk)).
Detalle= (nroTicket(Fk), idProducto(Fk), cantidad, precioUnitario).
Producto= (idProducto, nombreP, descripcion, precio, stock).

(1) *Listar datos personales de clientes cuyo apellido comience con el string 'Pe'. Ordenar por DNI.*

```
SELECT nombre, apellido, DNI, telefono, direccion
FROM Cliente
WHERE (apellido LIKE 'Pe%')
ORDER BY DNI
```

(2) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que realizaron compras sólo durante 2024.*

```
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Factura f
NATURAL JOIN Cliente c
WHERE (YEAR(f.fecha) = 2024)
EXCEPT
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Factura f
NATURAL JOIN Cliente c
WHERE (YEAR(f.fecha) <> 2024)
```

(3) *Listar nombre, descripción, precio y stock de productos vendidos al cliente con DNI 45.789.456, pero que no fueron vendidos a clientes de apellido 'Garcia'.*

```
SELECT DISTINCT p.nombreP, p.descripcion, p.precio, p.stock
FROM Detalle d
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
INNER JOIN Producto p ON (d.idProducto = p.idProducto)
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
WHERE (c.DNI = 45789456 AND p.idProducto NOT IN (
        SELECT d2.idProducto
        FROM Detalle d2
        INNER JOIN Factura f2 ON (d2.nroTicket = f2.nroTicket)
        INNER JOIN Cliente c2 ON (f2.idCliente = c2.idCliente)
```

```
        WHERE (c2.apellido = 'Garcia')
))
```

(4) *Listar nombre, descripción, precio y stock de productos no vendidos a clientes que tengan teléfono con característica 221 (la característica está al comienzo del teléfono). Ordenar por nombre.*

```
SELECT p.nombreP, p.descripcion, p.precio, p.stock
FROM Producto p
WHERE (p.idProducto NOT IN (
        SELECT p.idProducto
        FROM Detalle d
        INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
        INNER JOIN Producto p ON (d.idProducto = p.idProducto)
        INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
        WHERE (c.telefono LIKE '221%')
))
ORDER BY p.nombreP
```

(5) *Listar, para cada producto, nombre, descripción, precio y cuantas veces fue vendido. Tener en cuenta que puede no haberse vendido nunca el producto.*

```
SELECT p.nombreP, p.descripcion, p.precio, SUM(COALESCE(d.cantidad, 0)) AS cantVendida
FROM Detalle d
RIGHT JOIN Producto p ON (d.idProducto = p.idProducto)
GROUP BY p.idProducto, p.nombreP, p.descripcion, p.precio
```

(6) *Listar nombre, apellido, DNI, teléfono y dirección de clientes que compraron los productos con nombre 'prod1' y 'prod2', pero nunca compraron el producto con nombre 'prod3'.*

```
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Detalle d
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
INNER JOIN Producto p ON (d.idProducto = p.idProducto)
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
WHERE (p.nombreP IN ('prod1', 'prod2'))
EXCEPT
SELECT c.nombre, c.apellido, c.DNI, c.telefono, c.direccion
FROM Detalle d
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
INNER JOIN Producto p ON (d.idProducto = p.idProducto)
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
```

```
WHERE (p.nombreP = 'prod3')
```

(7) *Listar nroTicket, total, fecha, hora y DNI del cliente de aquellas facturas donde se haya comprado el producto 'prod38' o la factura tenga fecha de 2023.*

```
SELECT DISTINCT f.nroTicket, f.total, f.fecha, f.hora, c.DNI
FROM Detalle d
INNER JOIN Factura f ON (d.nroTicket = f.nroTicket)
INNER JOIN Producto p ON (d.idProducto = p.idProducto)
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
WHERE (YEAR(f.fecha) = 2023 OR p.nombreP = 'prod38')
```

(8) *Agregar un cliente con los siguientes datos: nombre 'Jorge Luis', apellido 'Castor', DNI 40.578.999, teléfono '221-4400789', dirección '11 entre 500 y 501 Nro. 2587' e id de cliente 500002. Se supone que el idCliente 500002 no existe.*

```
INSERT INTO Cliente (idCliente, nombre, apellido, DNI, telefono, direccion)
VALUES (500002, 'Jorge Luis', 'Castor', 40578999, '221-4400789', '11 entre 500 y 501 Nro. 2587')
```

(9) *Listar nroTicket, total, fecha, hora para las facturas del cliente 'Jorge Pérez' donde no haya comprado el producto 'Z'.*

```
SELECT f.nroTicket, f.total, f.fecha, f.hora
FROM Factura f
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
WHERE (c.nombre = 'Jorge' AND c.apellido = 'Pérez' AND f.nroTicket NOT IN (
        SELECT d.nroTicket
        FROM Detalle d
        INNER JOIN Producto p ON (d.idProducto = p.idProducto)
        WHERE (p.nombreP = 'Z')
))
```

(10) *Listar DNI, apellido y nombre de clientes donde el monto total comprado, teniendo en cuenta todas sus facturas, supere $100.000.*

```
SELECT c.DNI, c.apellido, c.nombre
FROM Factura f
INNER JOIN Cliente c ON (f.idCliente = c.idCliente)
GROUP BY c.DNI, c.apellido, c.nombre
HAVING (SUM(f.total) > 100000)
```

Ejercicio 2.

Localidad= (codigoPostal, nombreL, descripcion, nroHabitantes).
Arbol= (nroArbol, especie, anios, calle, nro, codigoPostal(Fk)).
Podador= (DNI, nombre, apellido, telefono, fnac, codigoPostalVive(Fk)).
Poda= (codPoda, fecha, DNI(Fk), nroArbol(Fk)).

(1) *Listar especie, años, calle, nro. y localidad de árboles podados por el podador 'Juan Perez' y por el podador 'Jose Garcia'.*

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (EXISTS (
        SELECT 1
        FROM Poda p
        INNER JOIN Podador pod ON (p.DNI = pod.DNI)
        WHERE (a.nroArbol = p.nroArbol AND pod.nombre = 'Juan' AND pod.apellido
= 'Perez')
)
AND EXISTS (
        SELECT 1
        FROM Poda p
        INNER JOIN Podador pod ON (p.DNI = pod.DNI)
        WHERE (a.nroArbol = p.nroArbol AND pod.nombre = 'Jose' AND pod.apellido
= 'Garcia')
)
)
```

(2) *Reportar DNI, nombre, apellido, fecha de nacimiento y localidad donde viven de aquellos podadores que tengan podas realizadas durante 2023.*

```
SELECT DISTINCT pod.DNI, pod.nombre, pod.apellido, pod.telefono, pod.fnac,
l.nombreL
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Localidad l ON (pod.codigoPostalVive = l.codigoPostal)
WHERE (YEAR(p.fecha) = 2023)
```

(3) *Listar especie, años, calle, nro. y localidad de árboles que no fueron podados nunca.*

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (a.nroArbol NOT IN (
```

```
        SELECT p.nroArbol
        FROM Poda p
))
```

(4) *Reportar especie, años, calle, nro. y localidad de árboles que fueron podados durante 2022 y no fueron podados durante 2023.*

```
SELECT a.especie, a.anios, a.calle, a.nro, l.nombreL
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (a.nroArbol IN (
        SELECT p.nroArbol
        FROM Poda p
        WHERE (YEAR(p.fecha) = 2022)
)
AND a.nroArbol NOT IN (
        SELECT p.nroArbol
        FROM Poda p
        WHERE (YEAR(p.fecha) = 2023)
)
)
```

(5) *Reportar DNI, nombre, apellido, fecha de nacimiento y localidad donde viven de aquellos podadores con apellido terminado con el string 'ata' y que tengan, al menos, una poda durante 2024. Ordenar por apellido y nombre.*

```
SELECT pod.DNI, pod.nombre, pod.apellido, pod.fnac, l.nombreL
FROM Podador pod
INNER JOIN Localidad l ON (pod.codigoPostalVive = l.codigoPostal)
WHERE (pod.apellido LIKE '%ata' AND pod.DNI IN (
        SELECT p.DNI
        FROM Poda p
        WHERE (YEAR(p.fecha) = 2024)
))
ORDER BY pod.apellido, pod.nombre
```

(6) *Listar DNI, apellido, nombre, teléfono y fecha de nacimiento de podadores que sólo podaron árboles de especie 'Conífera'.*

```
SELECT pod.DNI, pod.apellido, pod.nombre, pod.telefono, pod.fnac
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Arbol a ON (p.nroArbol = a.nroArbol)
WHERE (a.especie = 'Conífera')
```

```
EXCEPT
SELECT pod.DNI, pod.apellido, pod.nombre, pod.telefono, pod.fnac
FROM Poda p
INNER JOIN Podador pod ON (p.DNI = pod.DNI)
INNER JOIN Arbol a ON (p.nroArbol = a.nroArbol)
WHERE (a.especie <> 'Conífera')
```

(7) *Listar especies de árboles que se encuentren en la localidad de 'La Plata' y también en la localidad de 'Salta'.*

```
SELECT a.especie
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (l.nombreL = 'La Plata')
INTERSECT
SELECT a.especie
FROM Arbol a
INNER JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
WHERE (l.nombreL = 'Salta')
```

(8) *Eliminar el podador con DNI 22.234.566.*

```
DELETE FROM Poda WHERE (DNI = 22234566)
DELETE FROM Podador WHERE (DNI = 22234566)
```

(9) *Reportar nombre, descripción y cantidad de habitantes de localidades que tengan menos de 5 árboles.*

```
SELECT l.nombreL, l.descripcion, l.nroHabitantes
FROM Arbol a
RIGHT JOIN Localidad l ON (a.codigoPostal = l.codigoPostal)
GROUP BY l.codigoPostal, l.nombreL, l.descripcion, l.nroHabitantes
HAVING (COUNT(*) < 5)
```

Ejercicio 3.

Banda= (codigoB, nombreBanda, genero_musical, anio_creacion).
Integrante= (DNI, nombre, apellido, direccion, email, fecha_nacimiento, codigoB(Fk)).
Escenario= (nroEscenario, nombre_escenario, ubicacion, cubierto, m2, descripcion).
Recital= (fecha, hora, nroEscenario(Fk), codigoB(Fk)).

(1) *Listar DNI, nombre, apellido, dirección y email de integrantes nacidos entre 1980 y 1990, y que hayan realizado algún recital durante 2023.*

```
SELECT i.DNI, i.nombre, i.apellido, i.direccion, i.email
FROM Integrante i
WHERE (i.fecha_nacimiento BETWEEN '1980-01-01' AND '1990-12-31' AND i.DNI
IN (
        SELECT i.DNI
        FROM Recital r
        INNER JOIN Banda b ON (r.codigoB = b.codigoB)
        INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
        WHERE (YEAR(r.fecha) = 2023)
))
```

(2) *Reportar nombre, género musical y año de creación de bandas que hayan realizado recitales durante 2023, pero no hayan tocado durante 2022.*

```
SELECT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Banda b
WHERE (b.codigoB IN (
        SELECT r.codigoB
        FROM Recital r
        WHERE (YEAR(r.fecha) = 2023)
)
AND b.codigoB NOT IN (
        SELECT r.codigoB
        FROM Recital r
        WHERE (YEAR(r.fecha) = 2022)
)
)
```

(3) *Listar el cronograma de recitales del día 04/12/2023. Se deberá listar nombre de la banda que ejecutará el recital, fecha, hora, y el nombre y ubicación del escenario correspondiente.*

```
SELECT DISTINCT b.nombreBanda, r.fecha, r.hora, e.nombre_escenario, e.ubicacion
FROM Recital r
```

```
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
WHERE (r.fecha = '2023-12-04')
```

(4) *Listar DNI, nombre, apellido, email de integrantes que hayan tocado en el escenario con nombre 'Gustavo Cerati' y en el escenario con nombre 'Carlos Gardel'.*

```
SELECT i.DNI, i.nombre, i.apellido, i.email
FROM Recital r
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
WHERE (e.nombre_escenario = 'Gustavo Cerati')
INTERSECT
SELECT i.DNI, i.nombre, i.apellido, i.email
FROM Recital r
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
WHERE (e.nombre_escenario = 'Carlos Gardel')
```

(5) *Reportar nombre, género musical y año de creación de bandas que tengan más de 5 integrantes.*

```
SELECT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Banda b
INNER JOIN Integrante i ON (b.codigoB = i.codigoB)
GROUP BY b.codigoB, b.nombreBanda, b.genero_musical, b.anio_creacion
HAVING (COUNT(*) > 5)
```

(6) *Listar nombre de escenario, ubicación y descripción de escenarios que sólo tuvieron recitales con el género musical rock and roll. Ordenar por nombre de escenario.*

```
SELECT e.nombre_escenario, e.ubicacion, e.descripcion
FROM Escenario e
WHERE (e.nroEscenario IN (
        SELECT r.nroEscenario
        FROM Recital r
        INNER JOIN Banda b ON (r.codigoB = b.codigoB)
        WHERE (b.genero_musical IN ('Rock', 'Rock Alternativo', 'Rock Nacional'))
)
AND e.nroEscenario NOT IN (
        SELECT r.nroEscenario
        FROM Recital r
```

```
        INNER JOIN Banda b ON (r.codigoB = b.codigoB)
        WHERE (b.genero_musical NOT IN ('Rock', 'Rock Alternativo', 'Rock Nacional'))
)
)
ORDER BY e.nombre_escenario
```

(*) No hay ningún género musical que se llame ‘rock and rol’, por lo cual se usa ‘Rock’, ‘Rock Alternativo’ y ‘Rock Nacional’.

(7) *Listar nombre, género musical y año de creación de bandas que hayan realizado recitales en escenarios cubiertos durante 2023. // cubierto es true, false según corresponda.*

```
SELECT DISTINCT b.nombreBanda, b.genero_musical, b.anio_creacion
FROM Recital r
INNER JOIN Banda b ON (r.codigoB = b.codigoB)
INNER JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
WHERE (YEAR(r.fecha) = 2023 AND e.cubierto = true)
```

(8) *Reportar, para cada escenario, nombre del escenario y cantidad de recitales durante 2024.*

```
SELECT e.nombre_escenario, COUNT(r.nroEscenario) AS cantRecitales
FROM Recital r
RIGHT JOIN Escenario e ON (r.nroEscenario = e.nroEscenario)
WHERE (YEAR(r.fecha) = 2024)
GROUP BY e.nroEscenario, e.nombre_escenario
```

(9) *Modificar el nombre de la banda ‘Mempis la Blusera’ a ‘Memphis la Blusera’.*

```
UPDATE Banda
SET nombreBanda = 'Memphis la Blusera'
WHERE (nombreBanda = 'Mempis la Blusera')
```

Ejercicio 4.

Persona= (DNI, Apellido, Nombre, Fecha_Nacimiento, Estado_Civil, Genero).
Alumno= (DNI(Fk), Legajo, Anio_Ingreso).
Profesor= (DNI(Fk), Matricula, Nro_Expediente).
Titulo= (Cod_Titulo, Nombre, Descripcion).
Titulo_Profesor= (Cod_Titulo(Fk), DNI(Fk), Fecha).
Curso= (Cod_Curso, Nombre, Descripcion, Fecha_Creacion, Duracion).
Alumno_Curso= (DNI(Fk), Cod_Curso(Fk), Anio, Desempenio, Calificacion).
Profesor_Curso= (DNI(Fk), Cod_Curso(Fk), Fecha, Desde, Fecha_Hasta?).

(1) *Listar DNI, legajo y apellido y nombre de todos los alumnos que tengan año de ingreso inferior a 2014.*

```
SELECT a.DNI, a.Legajo, p.Apellido, p.Nombre
FROM Alumno a
INNER JOIN Persona p ON (a.DNI = p.DNI)
WHERE (a.Anio_Ingreso < 2014)
```

(2) *Listar DNI, matrícula, apellido y nombre de los profesores que dictan cursos que tengan más de 100 horas de duración. Ordenar por DNI.*

```
SELECT DISTINCT p.DNI, p.Matricula, per.Apellido, per.Nombre
FROM Profesor_Curso pc
INNER JOIN Profesor p ON (pc.DNI = p.DNI)
INNER JOIN Persona per ON (pc.DNI = per.DNI)
INNER JOIN Curso c ON (pc.Cod_Curso = c.Cod_Curso)
WHERE (c.Duracion > 100)
ORDER BY p.DNI
```

(3) *Listar DNI, Apellido, Nombre, Género y Fecha de nacimiento de los alumnos inscriptos al curso con nombre 'Diseño de Bases de Datos' en 2023.*

```
SELECT per.DNI, per.Apellido, per.Nombre, per.Genero, per.Fecha_Nacimiento
FROM Persona per
WHERE (per.DNI IN (
        SELECT ac.DNI
        FROM Alumno_Curso ac
        INNER JOIN Curso c ON (ac.Cod_Curso = c.Cod_Curso)
        WHERE (ac.anio = 2023 AND c.Nombre = 'Diseño de Bases de Datos')
))
```

(4) *Listar DNI, Apellido, Nombre y Calificación de aquellos alumnos que obtuvieron una calificación superior a 8 en algún curso que dicta el profesor 'Juan Garcia'. Dicho listado deberá estar ordenado por Apellido y nombre.*

```
SELECT perA.DNI, perA.Apellido, perA.Nombre, ac.Calificacion
FROM Alumno_Curso ac
INNER JOIN Persona perA ON (ac.DNI = perA.DNI)
INNER JOIN Profesor_Curso pc ON (ac.Cod_Curso = pc.Cod_Curso)
INNER JOIN Persona perP ON (pc.DNI = perP.DNI)
WHERE (ac.Calificacion > 8 AND perP.Nombre = 'Juan' AND perP.Apellido = 'Garcia')
ORDER BY perA.Apellido, perA.Nombre
```

(5) *Listar DNI, Apellido, Nombre y Matrícula de aquellos profesores que posean más de 3 títulos. Dicho listado deberá estar ordenado por Apellido y Nombre.*

```
SELECT p.DNI, per.Apellido, per.Nombre, p.Matricula
FROM Profesor p
INNER JOIN Persona per ON (p.DNI = per.DNI)
INNER JOIN Titulo_Profesor tp ON (p.DNI = tp.DNI)
GROUP BY p.DNI, per.Apellido, per.Nombre, p.Matricula
HAVING (COUNT(*) > 3)
ORDER BY per.Apellido, per.Nombre
```

(6) *Listar DNI, Apellido, Nombre, Cantidad de horas y Promedio de horas que dicta cada profesor. La cantidad de horas se calcula como la suma de la duración de todos los cursos que dicta.*

```
SELECT p.DNI, per.Apellido, per.Nombre, SUM(c.Duracion), AVG(c.Duracion)
FROM Profesor p
INNER JOIN Persona per ON (p.DNI = per.DNI)
LEFT JOIN Profesor_Curso pc ON (p.DNI = pc.DNI)
LEFT JOIN Curso c ON (pc.Cod_Curso = c.Cod_Curso)
GROUP BY p.DNI, per.Apellido, per.Nombre
```

(7) *Listar Nombre y Descripción del curso que posea más alumnos inscriptos y del que posea menos alumnos inscriptos durante 2024.*

```
SELECT c.Nombre, c.Descripcion
FROM Curso c
WHERE (c.Cod_Curso IN (
        SELECT ac.Cod_Curso
        FROM Alumno_Curso ac
        WHERE (ac.anio = 2024)
```

```
        GROUP BY ac.Cod_Curso
        HAVING (COUNT(*) = (
                SELECT MAX(cant)
                FROM (
                        SELECT COUNT(*) AS cant
                        FROM Alumno_Curso
                        WHERE (Anio = 2024)
                        GROUP BY Cod_Curso
                ) AS Max
        )
        OR COUNT(*) = (
                SELECT MIN(cant)
                FROM (
                        SELECT COUNT(*) AS cant
                        FROM Alumno_Curso
                        WHERE (Anio = 2024)
                        GROUP BY Cod_Curso
                ) AS Min
        )
        )
))
```

(8) *Listar el DNI, Apellido, Nombre y Legajo de alumnos que realizaron cursos con nombre conteniendo el string 'BD' durante 2022, pero no realizaron ningún curso durante 2023.*

```
SELECT a.DNI, per.Apellido, per.Nombre, a.Legajo
FROM Alumno_Curso ac
INNER JOIN Alumno a ON (ac.DNI = a.DNI)
INNER JOIN Persona per ON (ac.DNI = per.DNI)
INNER JOIN Curso c on (ac.Cod_Curso = c.Cod_Curso)
WHERE (ac.Anio = 2022 AND c.Nombre LIKE '%BD%')
EXCEPT
SELECT a.DNI, per.Apellido, per.Nombre, a.Legajo
FROM Alumno_Curso ac
INNER JOIN Alumno a ON (ac.DNI = a.DNI)
INNER JOIN Persona per ON (ac.DNI = per.DNI)
WHERE (ac.Anio = 2023)
```

(9) *Agregar un profesor con los datos que se prefiera y agregarle el título con código 25.*

```
INSERT INTO Persona (DNI, Apellido, Nombre, Fecha_Nacimiento, Estado_Civil, Genero)
VALUES (37102205, 'Menduiña', 'Juan', '1992-09-08', 'Soltero', 'M')

INSERT INTO Profesor (DNI, Matricula, Nro_Expediente)
```

```
VALUES (37102205, 'MAT 1000', 10000)

INSERT INTO Titulo_Profesor (Cod_Titulo, DNI, Fecha)
VALUES (25, 37102205, '2025-01-01')
```

(10) *Modificar el estado civil del alumno cuyo legajo es '2020/09'; el nuevo estado civil es divorciado.*

```
UPDATE Persona
SET Estado_Civil = 'Divorciado'
WHERE (DNI IN (SELECT DNI FROM Alumno WHERE (Legajo = '2020/09')))
```

(11) *Dar de baja el alumno con DNI 30.568.989. Realizar todas las bajas necesarias para no dejar el conjunto de relaciones en un estado inconsistente.*

```
DELETE FROM Alumno_Curso WHERE (DNI = 30568989)
DELETE FROM Alumno WHERE (DNI = 30568989)
DELETE FROM Persona WHERE (DNI = 30568989)
```

Ejercicio 5.

Agencia= (razon_social, direccion, telef, email).
Ciudad= (codigo_postal,nombreCiudad, anioCreacion).
Cliente= (dni, nombre, apellido, telefono, direccion).
Viaje= (fecha,hora,dni(Fk), cpOrigen(Fk), cpDestino(Fk), razon_social(Fk), descripcion). // cpOrigen y cpDestino corresponden a las ciudades origen y destino del viaje, respectivamente.

(1) *Listar razón social, dirección y teléfono de agencias que realizaron viajes desde la ciudad de 'La Plata' (ciudad origen) y que el cliente tenga apellido 'Roma'. Ordenar por razón social y, luego, por teléfono.*

```
SELECT DISTINCT a.razon_social, a.direccion, a.telef
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cOrigen ON (v.cpOrigen = cOrigen.codigo_postal)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (c.apellido = 'Roma' AND cOrigen.nombreCiudad = 'La Plata')
ORDER BY a.razon_social, a.telef
```

(2) *Listar fecha, hora, datos personales del cliente, nombres de ciudades origen y destino de viajes realizados en enero de 2019 donde la descripción del viaje contenga el String 'demorado'.*

```
SELECT v.fecha, v.hora, c.dni, c.nombre, c.apellido, c.telefono, c.direccion,
cOrigen.nombreCiudad, cDestino.nombreCiudad
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cOrigen ON (v.cpOrigen = cOrigen.codigo_postal)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (YEAR(v.fecha) = 2019 AND v.descripcion LIKE '%demorado%')
```

(3) *Reportar información de agencias que realizaron viajes durante 2019 o que tengan dirección de mail que termine con '@jmail.com'.*

```
SELECT DISTINCT a.razon_social, a.direccion, a.telef, a.email
FROM Viaje v
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (YEAR(v.fecha) = 2019 OR a.email LIKE '%@jmail.com')
```

(4) *Listar datos personales de clientes que viajaron sólo con destino a la ciudad de 'Coronel Brandsen'.*

```
SELECT c.dni, c.nombre, c.apellido, c.telefono, c.direccion
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (cDestino.nombreCiudad = 'Coronel Brandsen')
EXCEPT
SELECT c.dni, c.nombre, c.apellido, c.telefono, c.direccion
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
WHERE (cDestino.nombreCiudad <> 'Coronel Brandsen')
```

(5) *Informar cantidad de viajes de la agencia con razón social 'TAXI Y' realizados a 'Villa Elisa'.*

```
SELECT COUNT(*) AS cantViajes
FROM Viaje v
INNER JOIN Ciudad cDestino ON (v.cpDestino = cDestino.codigo_postal)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
WHERE (cDestino.nombreCiudad = 'Villa Elisa' AND a.razon_social = 'TAXI Y')
```

(6) *Listar nombre, apellido, dirección y teléfono de clientes que viajaron con todas las agencias.*

```
SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Cliente c
WHERE (NOT EXISTS (
        SELECT *
        FROM Agencia a
        WHERE (NOT EXISTS (
                SELECT *
                FROM Viaje v
                WHERE (a.razon_social = v.razon_social AND c.dni = v.dni)
        ))
))

SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
GROUP BY c.dni, c.nombre, c.apellido, c.direccion, c.telefono
HAVING (COUNT(DISTINCT a.razon_social) = (
        SELECT COUNT(*)
        FROM Agencia
```

))

(7) *Modificar el cliente con DNI 38.495.444 actualizando el teléfono a '221-4400897'.*

```
UPDATE Cliente
SET telefono = '221-4400897'
WHERE (dni = 38495444)
```

(8) *Listar razón social, dirección y teléfono de la/s agencias que tengan mayor cantidad de viajes realizados.*

```
SELECT a.razon_social, a.direccion, a.telef
FROM Viaje v
INNER JOIN Agencia a ON (v.razon_social = a.razon_social)
GROUP BY a.razon_social, a.direccion, a.telef
HAVING (COUNT(*) >= ALL (
        SELECT COUNT(*)
        FROM Viaje v
        GROUP BY v.razon_social
))
```

(9) *Reportar nombre, apellido, dirección y teléfono de clientes con, al menos, 5 viajes.*

```
SELECT c.nombre, c.apellido, c.direccion, c.telefono
FROM Viaje v
INNER JOIN Cliente c ON (v.dni = c.dni)
GROUP BY c.dni, c.nombre, c.apellido, c.telefono, c.direccion
HAVING (COUNT(*) >= 5)
```

(10) *Borrar al cliente con DNI 40.325.692.*

```
DELETE FROM Viaje WHERE (dni = 40325692)
DELETE FROM Cliente WHERE (dni = 40325692)
```

Ejercicio 6.

Tecnico= (codTec, nombre, especialidad). // técnicos.
Repuesto= (codRep, nombre, stock, precio). // repuestos.
RepuestoReparacion= (nroReparac(Fk), codRep(Fk), cantidad, precio). // repuestos utilizados en reparaciones.
Reparacion= (nroReparac, codTec(Fk), precio_total, fecha). // reparaciones realizadas.

(1) *Listar los repuestos, informando nombre, stock y precio. Ordenar el resultado por precio.*

```
SELECT nombre, stock, precio
FROM Repuesto
ORDER BY Precio
```

(2) *Listar nombre, stock y precio de repuestos que se usaron en reparaciones durante 2023 y que no se usaron en reparaciones del técnico 'José Gonzalez'.*

```
SELECT rep.nombre, rep.stock, rep.precio
FROM Repuesto rep
WHERE (rep.codRep IN (
        SELECT rr.codRep
        FROM RepuestoReparacion rr
        INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
        WHERE (YEAR(r.fecha) = 2023)
)
AND rep.codRep NOT IN (
        SELECT rr.codRep
        FROM RepuestoReparacion rr
        INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
        INNER JOIN Tecnico t ON (r.codTec = t.codTec)
        WHERE (t.nombre = 'José Gonzalez')
)
)
```

(3) *Listar nombre y especialidad de técnicos que no participaron en ninguna reparación. Ordenar por nombre ascendentemente.*

```
SELECT t.nombre, t.especialidad
FROM Tecnico t
WHERE (t.codTec NOT IN (
        SELECT r.codTec
        FROM Reparacion r
```

```
))
ORDER BY t.nombre
```

(4) *Listar nombre y especialidad de los técnicos que sólo participaron en reparaciones durante 2022.*

```
SELECT t.nombre, t.especialidad
FROM Tecnico t
WHERE (t.codTec IN (
        SELECT r.codTec
        FROM Reparacion r
        WHERE (YEAR(r.fecha) = 2022)
)
AND t.codTec NOT IN (
        SELECT r.codTec
        FROM Reparacion r
        WHERE (YEAR(r.fecha) <> 2022)
)
)
```

(5) *Listar, para cada repuesto, nombre, stock y cantidad de técnicos distintos que lo utilizaron. Si un repuesto no participó en alguna reparación, igual debe aparecer en dicho listado.*

```
SELECT rep.nombre, rep.stock, COUNT(DISTINCT r.codTec) AS cantUsados
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
RIGHT JOIN Repuesto rep ON (rr.codRep = rep.codRep)
GROUP BY rep.codRep, rep.nombre, rep.stock
```

(6) *Listar nombre y especialidad del técnico con mayor cantidad de reparaciones realizadas y el técnico con menor cantidad de reparaciones.*

```
SELECT t.nombre, t.especialidad
FROM Reparacion r
INNER JOIN Tecnico t ON (r.codTec = t.codTec)
GROUP BY t.codTec, t.nombre, t.especialidad
HAVING (COUNT(*) >= ALL (
        SELECT COUNT(*)
        FROM Reparacion r
        GROUP BY r.codTec
)
OR COUNT(*) <= ALL (
        SELECT COUNT(*)
```

```
        FROM Reparacion r
        GROUP BY r.codTec
)
)
```

(7) *Listar nombre, stock y precio de todos los repuestos con stock mayor a 0 y que dicho repuesto no haya estado en reparaciones con un precio total superior a $10.000.*

```
SELECT rep.nombre, rep.stock, rep.precio
FROM Repuesto rep
WHERE (rep.stock > 0 AND rep.codRep NOT IN (
        SELECT rr.codRep
        FROM RepuestoReparacion rr
        INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
        WHERE (r.precio_total > 10000)
))
```

(8) *Proyectar número, fecha y precio total de aquellas reparaciones donde se utilizó algún repuesto con precio en el momento de la reparación mayor a $10.000 y menor a $15.000.*

```
SELECT r.nroReparac, r.fecha, r.precio_total
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
WHERE (rr.precio BETWEEN 10000 AND 15000)
```

(9) *Listar nombre, stock y precio de repuestos que hayan sido utilizados por todos los técnicos.*

```
SELECT rep.nombre, rep.stock, rep.precio
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
INNER JOIN Repuesto rep ON (rr.codRep = rep.codRep)
GROUP BY rep.codRep, rep.nombre, rep.stock, rep.precio
HAVING (COUNT(DISTINCT r.codTec) = (
        SELECT COUNT(*)
        FROM Tecnico
))
```

(10) *Listar fecha, técnico y precio total de aquellas reparaciones que necesitaron, al menos, 4 repuestos distintos.*

```
SELECT r.fecha, t.nombre, r.precio_total
FROM RepuestoReparacion rr
INNER JOIN Reparacion r ON (rr.nroReparac = r.nroReparac)
INNER JOIN Tecnico t ON (r.codTec = t.codTec)
GROUP BY r.nroReparac, r.fecha, t.nombre, r.precio_total
HAVING (COUNT(*) >= 4)
```

Ejercicio 7.

Club= (codigoClub, nombre, anioFundacion, codigoCiudad(Fk)).
Ciudad= (codigoCiudad, nombre).
Estadio= (codigoEstadio, codigoClub(Fk), nombre, direccion).
Jugador= (DNI, nombre, apellido, edad, codigoCiudad(Fk)).
ClubJugador= (codigoClub(Fk), DNI(Fk), desde, hasta).

(1) *Reportar nombre y año de fundación de aquellos clubes de la ciudad de La Plata que no poseen estadio.*

```
SELECT c.nombre, c.anioFundacion
FROM Club c
INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (ciu.nombre = 'La Plata' AND c.codigoClub NOT IN (
        SELECT e.codigoClub
        FROM Estadio e
))
```

(2) *Listar nombre de los clubes que no hayan tenido ni tengan jugadores de la ciudad de Berisso.*

```
SELECT c.nombre
FROM Club c
WHERE (c.codigoClub NOT IN (
        SELECT DISTINCT cj.codigoClub
        FROM Jugador j
        INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
        INNER JOIN Ciudad ciu ON (j.codigoCiudad = ciu.codigoCiudad)
        WHERE (ciu.nombre = 'Berisso')
))
```

(3) *Mostrar DNI, nombre y apellido de aquellos jugadores que jugaron o juegan en el club 'Gimnasia y Esgrima La Plata'.*

```
SELECT j.DNI, j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
WHERE (c.nombre = 'Gimnasia y Esgrima La Plata')
```

(4) *Mostrar DNI, nombre y apellido de aquellos jugadores que tengan más de 29 años y hayan jugado o juegan en algún club de la ciudad de Córdoba.*

```
SELECT DISTINCT j.DNI, j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
INNER JOIN Ciudad ciU ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (j.edad > 29 AND ciu.nombre = 'Córdoba')
```

(5) *Mostrar, para cada club, nombre de club y la edad promedio de los jugadores que juegan, actualmente, en cada uno.*

```
SELECT c.nombre, AVG(j.edad) AS edadProm
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
WHERE (cj.hasta IS NULL)
GROUP BY c.codigoClub, c.nombre
```

(6) *Listar, para cada jugador, nombre, apellido, edad y cantidad de clubes diferentes en los que jugó (incluído el actual).*

```
SELECT j.nombre, j.apellido, j.edad, COUNT(DISTINCT cj.codigoClub) AS cantClubes
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
GROUP BY j.DNI, j.nombre, j.apellido, j.edad
```

(7) *Mostrar el nombre de los clubes que nunca hayan tenido jugadores de la ciudad de Mar del Plata.*

```
SELECT c.nombre
FROM Club c
WHERE (c.codigoClub NOT IN (
        SELECT DISTINCT cj.codigoClub
        FROM Jugador j
        INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
        INNER JOIN Ciudad ciu ON (j.codigoCiudad = ciu.codigoCiudad)
        WHERE (ciu.nombre = 'Mar del Plata')
))
```

(8) *Reportar el nombre y apellido de aquellos jugadores que hayan jugado en todos los clubes de la ciudad de Córdoba.*

```
SELECT j.nombre, j.apellido
FROM Jugador j
INNER JOIN ClubJugador cj ON (j.DNI = cj.DNI)
INNER JOIN Club c ON (cj.codigoClub = c.codigoClub)
INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
WHERE (ciu.nombre = 'Córdoba')
GROUP BY j.DNI, j.nombre, j.apellido
HAVING (COUNT(DISTINCT c.codigoClub) = (
        SELECT COUNT(*)
        FROM Club c
        INNER JOIN Ciudad ciu ON (c.codigoCiudad = ciu.codigoCiudad)
        WHERE (ciu.nombre = 'Córdoba')
))
```

(9) *Agregar el club "Estrella de Berisso", con código 1234, que se fundó en 1921 y que pertenece a la ciudad de Berisso. Puede asumirse que el codigoClub 1234 no existe en la tabla Club.*

```
INSERT INTO Club (codigoClub, nombre, anioFundacion, codigoCiudad)
VALUES (1234, 'Estrella de Berisso', 1921, (
        SELECT codigoCiudad
        FROM Ciudad
        WHERE (nombre = 'Berisso')
))
```

Ejercicio 8.

Equipo= (codigoE, nombreE, descripcionE).
Integrante= (DNI, nombre, apellido, ciudad, email, telefono, codigoE(Fk)).
Laguna= (nroLaguna, nombreL, ubicacion, extension, descripcion).
TorneoPesca= (codTorneo, fecha, hora, nroLaguna(Fk), descripcion).
Inscripcion= (codTorneo(Fk), codigoE(Fk), asistio, gano). // asistio y gano son true/false según corresponda.

(1) *Listar DNI, nombre, apellido y email de integrantes que sean de la ciudad 'La Plata' y estén inscriptos en torneos disputados en 2023.*

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.email
FROM Integrante i
INNER JOIN Inscripcion ins ON (i.codigoE = ins.codigoE)
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
WHERE (i.ciudad = 'La Plata' AND YEAR(t.fecha) = 2023)
```

(2) *Reportar nombre y descripción de equipos que sólo se hayan inscripto en torneos de 2020.*

```
SELECT e.nombreE, e.descripcionE
FROM Equipo e
WHERE (e.codigoE IN (
        SELECT ins.codigoE
        FROM Inscripcion ins
        INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
        WHERE (YEAR(t.fecha) = 2020)
)
AND e.codigoE NOT IN (
        SELECT ins.codigoE
        FROM Inscripcion ins
        INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
        WHERE (YEAR(t.fecha) <> 2020)
)
)
```

(3) *Listar DNI, nombre, apellido, email y ciudad de integrantes que asistieron a torneos en la laguna con nombre 'La Salada, Coronel Granada' y su equipo no tenga inscripciones a torneos disputados en 2023.*

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.email, i.ciudad
FROM Integrante i
INNER JOIN Inscripcion ins ON (i.codigoE = ins.codigoE)
```

```
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
WHERE (ins.asistio = TRUE AND l.nombreL = 'La Salada, Coronel Granada' AND
i.codigoE NOT IN (
        SELECT ins.codigoE
        FROM Inscripcion ins
        INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
        WHERE (YEAR(t.fecha) = 2023)
))
```

(4) *Reportar nombre y descripción de equipos que tengan, al menos 5, integrantes. Ordenar por nombre.*

```
SELECT e.nombreE, e.descripcionE
FROM Integrante i
INNER JOIN Equipo e ON (i.codigoE = e.codigoE)
GROUP BY e.codigoE, e.nombreE, e.descripcionE
HAVING (COUNT(*) >= 5)
ORDER BY e.nombreE
```

(5) *Reportar nombre y descripción de equipos que tengan inscripciones en todas las lagunas.*

```
SELECT e.nombreE, e.descripcionE
FROM Inscripcion ins
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Equipo e ON (ins.codigoE = e.codigoE)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
GROUP BY e.codigoE, e.nombreE, e.descripcionE
HAVING (COUNT(DISTINCT l.nroLaguna) = (
        SELECT COUNT(*)
        FROM Laguna
))
```

(6) *Eliminar el equipo con código 10.000.*

```
DELETE FROM Integrante WHERE (codigoE = 10000)
DELETE FROM Inscripcion WHERE (codigoE = 10000)
DELETE FROM Equipo WHERE (codigoE = 10000)
```

(7) *Listar nombre, ubicación, extensión y descripción de lagunas que no tuvieron torneos.*

```
SELECT l.nombreL, l.ubicacion, l.extension, l.descripcion
FROM Laguna l
WHERE (l.nroLaguna NOT IN (
        SELECT t.nroLaguna
        FROM TorneoPesca t
))
```

(8) *Reportar nombre y descripción de equipos que tengan inscripciones a torneos a disputarse durante 2024, pero no tienen inscripciones a torneos de 2023.*

```
SELECT e.nombreE, e.descripcionE
FROM Equipo e
WHERE (e.codigoE IN (
        SELECT ins.codigoE
        FROM Inscripcion ins
        INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
        WHERE (YEAR(t.fecha) = 2024)
)
AND e.codigoE NOT IN (
        SELECT ins.codigoE
        FROM Inscripcion ins
        INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
        WHERE (YEAR(t.fecha) = 2023)
)
)
```

(9) *Listar DNI, nombre, apellido, ciudad y email de integrantes que ganaron algún torneo que se disputó en la laguna con nombre 'Laguna de Chascomús'.*

```
SELECT DISTINCT i.DNI, i.nombre, i.apellido, i.ciudad, i.email
FROM Inscripcion ins
INNER JOIN TorneoPesca t ON (ins.codTorneo = t.codTorneo)
INNER JOIN Equipo e ON (ins.codigoE = e.codigoE)
INNER JOIN Laguna l ON (t.nroLaguna = l.nroLaguna)
INNER JOIN Integrante i ON (e.codigoE = i.codigoE)
WHERE (ins.gano = TRUE AND l.nombreL = 'Laguna de Chascomús')
```

Ejercicio 9.

Proyecto= (codProyecto, nombrP, descripcion, fechaInicioP, fechaFinP?, fechaFinEstimada, DNIResponsable(Fk), equipoBackend(Fk), equipoFrontend(Fk)). // DNIResponsable corresponde a un empleado, equipoBackend y equipoFrontend corresponden a un equipo.
Equipo= (codEquipo, nombreE, descTecnologias, DNILider(Fk)). // DNILider corresponde a un empleado.
Empleado= (DNI, nombre, apellido, telefono, direccion, fechaIngreso).
Empleado_Equipo= (codEquipo(Fk), DNI(Fk), fechaInicio, fechaFin, descripcionRol).

(1) *Listar nombre, descripción, fecha de inicio y fecha de fin de proyectos ya finalizados que no fueron terminados antes de la fecha de fin estimada.*

```
SELECT nombreP, descripcion, fechaInicioP, fechaFinP
FROM Proyecto
WHERE (fechaFinP IS NOT NULL AND fechaFinP >= fechaFinEstimada)
```

(2) *Listar DNI, nombre, apellido, teléfono, dirección y fecha de ingreso de empleados que no son ni fueron responsables de proyectos. Ordenar por apellido y nombre.*

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion, emp.fechaIngreso
FROM Empleado emp
WHERE (emp.DNI NOT IN (
        SELECT DISTINCT p.DNIResponsable
        FROM Proyecto p
))
ORDER BY emp.apellido, emp.nombre
```

(3) *Listar DNI, nombre, apellido, teléfono y dirección de líderes de equipo que tenga más de un equipo a cargo.*

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Equipo e
INNER JOIN Empleado emp ON (e.DNILider = emp.DNI)
GROUP BY emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
HAVING (COUNT(*) > 1)
```

(4) *Listar DNI, nombre, apellido, teléfono y dirección de todos los empleados que trabajan en el proyecto con nombre 'Proyecto X'. No es necesario informar responsable y líderes.*

```
SELECT DISTINCT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Proyecto p
INNER JOIN Empleado_Equipo ee ON (p.equipoBackend = ee.codEquipo OR p.equipoFrontend = ee.codEquipo)
INNER JOIN Empleado emp ON (ee.DNI = emp.DNI)
WHERE (p.fechaFinP IS NULL AND p.nombrP = 'Proyecto X')
```

(5) *Listar nombre de equipo y datos personales de líderes de equipos que no tengan empleados asignados y trabajen con tecnología 'Java'.*

```
SELECT e.nombreE, emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Empleado_Equipo ee
RIGHT JOIN Equipo e ON (ee.codEquipo = e.codEquipo)
INNER JOIN Empleado emp ON (e.DNILider = emp.DNI)
WHERE (ee.codEquipo IS NULL AND eqdescTecnologias LIKE '%Java%')
```

(6) *Modificar nombre, apellido y dirección del empleado con DNI 40.568.965 con los datos que se desee.*

```
UPDATE Empleado
SET nombre = 'Juan', apellido = 'Menduiña', direccion = '13 Nro. 22'
WHERE (DNI = 40568965)
```

(7) *Listar DNI, nombre, apellido, teléfono y dirección de empleados que son responsables de proyectos, pero no han sido líderes de equipo.*

```
SELECT emp.DNI, emp.nombre, emp.apellido, emp.telefono, emp.direccion
FROM Empleado emp
WHERE (emp.DNI IN (
        SELECT p.DNIResponsable
        FROM Proyecto p
        WHERE (p.fechaFinP IS NULL)
)
AND emp.DNI NOT IN (
        SELECT e.DNILider
        FROM Equipo e
)
)
```

(8) *Listar nombre de equipo y descripción de tecnologías de equipos que hayan sido asignados como equipos frontend y backend.*

```
SELECT e.nombreE, e.descTecnologias
FROM Equipo e
WHERE (e.codEquipo IN (
        SELECT p.equipoFrontend
        FROM Proyecto p
)
AND e.codEquipo IN (
        SELECT p.equipoBackend
        FROM Proyecto p
)
)
```

(9) *Listar nombre, descripción, fecha de inicio, nombre y apellido de responsables de proyectos que se estiman finalizar durante 2025.*

```
SELECT p.nombrP, p.descripcion, p.fechaInicioP, emp.nombre, emp.apellido
FROM Proyecto p
INNER JOIN Empleado emp ON (p.DNIResponsable = emp.DNI)
WHERE (p.fechaFinP IS NULL AND YEAR(p.fechaFinEstimada) = 2025)
```

Ejercicio 10.

Vehiculo= (patente, modelo, marca, peso, km).
Camion= (patente(Fk), largo, max_toneladas, cant_ruedas, tiene_acoplado).
Auto= (patente(Fk), es_electrico, tipo_motor).
Service= (fecha, patente(Fk), km_service, descripcion, monto).
Parte= (cod_parte, nombre, precio_parte).
Service_Parte= ([fecha, patente](Fk), cod_parte(Fk), precio).

(1) *Listar todos los datos de aquellos camiones que tengan entre 4 y 8 ruedas, y que hayan realizado algún service en los últimos 365 días. Ordenar por marca, modelo y patente.*

```
SELECT v.patente, v.modelo, v.marca, v.peso, v.km, c.largo, c.max_toneladas, c.cant_ruedas, c.tiene_acoplado
FROM Service s
INNER JOIN Vehiculo v ON (s.patente = v.patente)
INNER JOIN Camion c ON (s.patente = c.patente)
WHERE (c.cant_ruedas BETWEEN 4 AND 8 AND s.fecha >= CURRENT_DATE - INTERVAL '365' DAY)
ORDER BY v.marca, v.modelo, v.patente
```

(2) *Listar los autos que hayan realizado el service "cambio de aceite" antes de los 13.000 km o hayan realizado el service "inspección general" que incluya la parte "Filtro de combustible".*

```
SELECT v.patente, v.modelo, v.marca, v.peso, v.km, a.es_electrico, a.tipo_motor
FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
INNER JOIN Vehiculo v ON (sp.patente = v.patente)
INNER JOIN Auto a ON (sp.patente = a.patente)
WHERE ((s.descripcion = 'cambio de aceite' AND s.km_service < 13000) OR (s.descripcion = 'inspección general' AND p.nombre = 'Filtro de combustible'))
```

(3) *Listar nombre y precio de todas las partes que aparezcan en más de 30 services que hayan salido (partes) más de $4.000.*

```
SELECT p.nombre, p.precio_parte
FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
WHERE (p.precio_parte > 4000)
```

```
GROUP BY p.cod_parte, p.nombre, p.precio_parte
HAVING (COUNT(*) > 30)
```

(4) *Dar de baja todos los camiones con más de 250.000 km.*

```
DELETE FROM Service_Parte sp
INNER JOIN Service s ON (sp.fecha = s.fecha AND sp.patente = s.patente)
WHERE (s.patente IN (
        SELECT v.patente
        FROM Vehiculo v
        WHERE (v.km > 250000)
))

DELETE FROM Service s
WHERE (s.patente IN (
        SELECT v.patente
        FROM Vehiculo v
        WHERE (v.km > 250000)
))

DELETE FROM Camion c
WHERE (c.patente IN (
        SELECT v.patente
        FROM Vehiculo v
        WHERE (v.km > 250000)
))

DELETE FROM Vehiculo WHERE (km > 250000)
```

(5) *Listar el nombre y precio de aquellas partes que figuren en todos los services realizados en el año actual.*

```
SELECT p.nombre, p.precio_parte
FROM Service_Parte sp
INNER JOIN Parte p ON (sp.cod_parte = p.cod_parte)
WHERE (YEAR(sp.fecha) = YEAR(CURRENT_DATE))
GROUP BY p.cod_parte, p.nombre, p.precio_parte
HAVING (COUNT(DISTINCT sp.fecha, sp.patente) = (
        SELECT COUNT(*)
        FROM Service s
        WHERE (YEAR(s.fecha) = YEAR(CURRENT_DATE))
))
```

(6) *Listar todos los autos que sean eléctricos. Mostrar información de patente, modelo, marca y peso.*

```
SELECT v.patente, v.modelo, v.marca, v.peso
FROM Vehiculo v
INNER JOIN Auto a ON (v.patente = a.patente)
WHERE (a.es_electrico = 1)
```

(7) *Dar de alta una parte, cuyo nombre sea "Aleron" y precio $5.000.*

```
INSERT INTO Parte (cod_parte, nombre, precio_parte)
VALUES (100, 'Aleron', 5000)
```

(8) *Dar de baja todos los services que se realizaron al auto con patente 'AWA564'.*

```
DELETE FROM Service_Parte WHERE (patente = 'AWA564')
DELETE FROM Service WHERE (patente = 'AWA564')
```

(9) *Listar todos los vehículos que hayan tenido services durante el 2024.*

```
SELECT DISTINCT v.patente, v.modelo, v.marca, v.peso, v.km
FROM Service s
INNER JOIN Vehiculo v ON (s.patente = v.patente)
WHERE (YEAR(s.fecha) = 2024)
```

Ejercicio 11.

Box= (nroBox, m2, ubicacion, capacidad, ocupacion). // ocupación es un numérico indicando cantidad de mascotas en el box actualmente, capacidad es una descripción.
Mascota= (codMascota, nombre, edad, raza, peso, telefonoContacto).
Veterinario= (matricula, CUIT, nombYAp, direccion, telefono).
Supervision= (codMascota(Fk), nroBox(Fk), fechaEntra, fechaSale?, matricula(Fk), descripcionEstadia). // fechaSale tiene valor null si la mascota está, actualmente, en el box.

(1) *Listar, para cada veterinario, cantidad de supervisiones realizadas con fecha de salida (fechaSale) durante enero de 2024. Indicar matrícula, CUIT, nombre y apellido, dirección, teléfono y cantidad de supervisiones.*

```
SELECT v.matricula, v.CUIT, v.nombYAp, v.direccion, v.telefono,
COUNT(s.matricula) AS cantSupervisiones
FROM Supervision s
RIGHT JOIN Veterinario v ON (s.matricula = v.matricula)
WHERE (s.fechaSale IS NOT NULL AND YEAR(s.fechaSale) = 2024 AND
MONTH(s.fechaSale) = 01)
GROUP BY v.matricula, v.CUIT, v.nombYAp, v.direccion, v.telefono
```

(2) *Listar CUIT, matrícula, nombre, apellido, dirección y teléfono de veterinarios que no tengan mascotas bajo supervisión actualmente.*

```
SELECT v.CUIT, v.matricula, v.nombYAp, v.direccion, v.telefono
FROM Veterinario v
WHERE (v.matricula NOT IN (
        SELECT s.matricula
        FROM Supervision s
        WHERE (s.fechaSale IS NULL)
))
```

(3) *Listar nombre, edad, raza, peso y teléfono de contacto de mascotas que fueron atendidas por el veterinario 'Oscar Lopez'. Ordenar por nombre y raza de manera ascendente.*

```
SELECT m.nombre, m.edad, m.raza, m.peso, m.telefonoContacto
FROM Supervision s
INNER JOIN Mascota m ON (s.codMascota = m.codMascota)
INNER JOIN Veterinario v ON (s.matricula = v.matricula)
WHERE (s.fechaSale IS NOT NULL AND v.nombYAp = 'Oscar Lopez')
ORDER BY m.nombre, m.raza
```

(4) *Modificar nombre y apellido al veterinario con matrícula 'MP 10000'; deberá llamarse 'Pablo Lopez'.*

```
UPDATE Veterinario
SET nombYAp = 'Pablo Lopez'
WHERE (matricula = 'MP 10000')
```

(5) *Listar nombre, edad, raza y peso de mascotas que tengan supervisiones con el veterinario con matrícula 'MP 1000' y con el veterinario con matrícula 'MN 4545'.*

```
SELECT m.nombre, m.edad, m.raza, m.peso
FROM Mascota m
WHERE (EXISTS (
        SELECT 1
        FROM Supervision s
        WHERE (m.codMascota = s.codMascota AND s.matricula = 'MP 1000')
)
AND EXISTS (
        SELECT 1
        FROM Supervision s
        WHERE (m.codMascota = s.codMascota AND s.matricula = 'MP 4545')
)
)
```

(6) *Listar número de box, m2, ubicación, capacidad y nombre de mascota para supervisiones con fecha de entrada durante 2024.*

```
SELECT b.nroBox, b.m2, b.ubicacion, b.capacidad, m.nombre
FROM Supervision s
INNER JOIN Mascota m ON (s.codMascota = m.codMascota)
INNER JOIN Box b ON (s.nroBox = b.nroBox)
WHERE (YEAR(s.fechaEntra) = 2024)
```

Ejercicio 12.

Barberia= (codBarberia, razon_social, direccion, telefono).
Cliente= (nroCliente, DNI, nombYAp, direccionC, fechaNacimiento, celular).
Barbero= (codEmpleado, DNIB, nombYApB, direccionB, telefonoContacto, mail).
Atencion= (codEmpleado(Fk), fecha, hora, codBarberia(Fk), nroCliente(Fk), descTratamiento, valor).

(1) *Listar DNI, nombre y apellido, dirección, fecha de nacimiento y celular de clientes que no tengan atenciones durante 2024.*

```
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Cliente c
WHERE (c.nroCliente NOT IN (
        SELECT a.nroCliente
        FROM Atencion a
        WHERE (YEAR(a.fecha) = 2024)
))
```

(2) *Listar, para cada barbero, cantidad de atenciones que realizaron durante 2023. Listar DNI, nombre y apellido, dirección, teléfono de contacto, mail y cantidad de atenciones.*

```
SELECT b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail,
COUNT(a.codEmpleado) AS cantAtenciones
FROM Atencion a
RIGHT JOIN Barbero b ON (a.codEmpleado = b.codEmpleado)
WHERE (YEAR(a.fecha) = 2023)
GROUP BY b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail
```

(3) *Listar razón social, dirección y teléfono de barberías que tengan atenciones para el cliente con DNI 22.283.566. Ordenar por razón social y dirección ascendente.*

```
SELECT DISTINCT bar.razon_social, bar.direccion, bar.telefono
FROM Atencion a
INNER JOIN Barberia bar ON (a.codBarberia = bar.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (c.DNI = 22283566)
ORDER BY bar.razon_social, bar.direccion
```

(4) *Listar DNI, nombre y apellido, dirección, teléfono de contacto y mail de barberos que tengan atenciones con valor superior a $5.000.*

```
SELECT DISTINCT b.DNIB, b.nombYApB, b.direccionB, b.telefonoContacto, b.mail
FROM Atencion a
INNER JOIN Barbero b ON (a.codEmpleado = b.codEmpleado)
WHERE (a.valor > 5000)
```

(5) *Listar DNI, nombYAp, direccionC, fechaNacimiento y celular de clientes que tengan atenciones en la barbería con razón social 'Corta barba' y también se hayan atendido en la barbería con razón social 'Barberia Barbara'.*

```
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Atencion a
INNER JOIN Barberia b ON (a.codBarberia = b.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (b.razon_social = 'Corta barba')
INTERSECT
SELECT c.DNI, c.nombYAp, c.direccionC, c.fechaNacimiento, c.celular
FROM Atencion a
INNER JOIN Barberia b ON (a.codBarberia = b.codBarberia)
INNER JOIN Cliente c ON (a.nroCliente = c.nroCliente)
WHERE (b.razon_social = 'Barberia Barbara')
```

(6) *Eliminar el cliente con DNI 22.222.222.*

```
DELETE FROM Atencion a
WHERE (a.nroCliente IN (
        SELECT c.nroCliente
        FROM Cliente c
        WHERE (c.DNI = 22222222)
))

DELETE FROM Cliente WHERE (DNI = 22222222)
```

Ejercicio 13.

Club= (IdClub, nombreClub, ciudad).
Complejo= (IdComplejo, nombreComplejo, IdClub(Fk)).
Cancha= (IdCancha, nombreCancha, IdComplejo(Fk)).
Entrenador= (IdEntrenador, nombreEntrenador, fechaNacimiento, direccion).
Entrenamiento= (IdEntrenamiento, fecha, IdEntrenador(Fk), IdCancha(Fk)).

(1) *Listar nombre, fecha de nacimiento y dirección de entrenadores que hayan tenido entrenamientos durante 2023.*

```
SELECT DISTINCT ent.nombreEntrenador, ent.fechaNacimiento, ent.direccion
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
WHERE (YEAR(e.fecha) = 2023)
```

(2) *Listar, para cada cancha del complejo "Complejo 1", la cantidad de entrenamientos que se realizaron durante el 2022. Informar nombre de la cancha y cantidad de entrenamientos.*

```
SELECT c.idCancha, c.nombreCancha, COUNT(e.IdCancha) AS cantEntrenamientos
FROM Cancha c
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
LEFT JOIN Entrenamiento e ON (c.IdCancha = e.IdCancha)
WHERE (com.nombreComplejo = 'Complejo 1' AND YEAR(e.fecha) = 2022)
GROUP BY c.idCancha, c.nombreCancha
```

(3) *Listar los complejos donde haya realizado entrenamientos el entrenador 'Jorge Gonzalez'. Informar nombre de complejo. Ordenar el resultado de manera ascendente.*

```
SELECT DISTINCT com.nombreComplejo
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
WHERE (ent.nombreEntrenador = 'Jorge Gonzalez')
ORDER BY com.nombreComplejo
```

(4) *Listar nombre, fecha de nacimiento y dirección de entrenadores que hayan entrenado en los clubes con nombre 'Everton' y 'Estrella de Berisso'.*

```
SELECT DISTINCT ent.nombreEntrenador, ent.fechaNacimiento, ent.direccion
```

```
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
INNER JOIN Club clu ON (com.IdClub = clu.IdClub)
WHERE (clu.nombreClub IN ('Everton', 'Estrella de Berisso'))
```

(5) *Listar todos los clubes en los que entrena el entrenador 'Marcos Perez'. Informar nombre del club y ciudad.*

```
SELECT DISTINCT clu.IdClub, clu.nombreClub, clu.ciudad
FROM Entrenamiento e
INNER JOIN Entrenador ent ON (e.IdEntrenador = ent.IdEntrenador)
INNER JOIN Cancha c ON (e.IdCancha = c.IdCancha)
INNER JOIN Complejo com ON (c.IdComplejo = com.IdComplejo)
INNER JOIN Club clu ON (com.IdClub = clu.IdClub)
WHERE (ent.nombreEntrenador = 'Marcos Perez')
```

(6) *Eliminar los entrenamientos del entrenador 'Juan Perez'.*

```
DELETE FROM Entrenamiento e
WHERE (e.IdEntrenador IN (
        SELECT ent.IdEntrenador
        FROM Entrenador ent
        WHERE (ent.nombreEntrenador = 'Juan Perez')
))
```