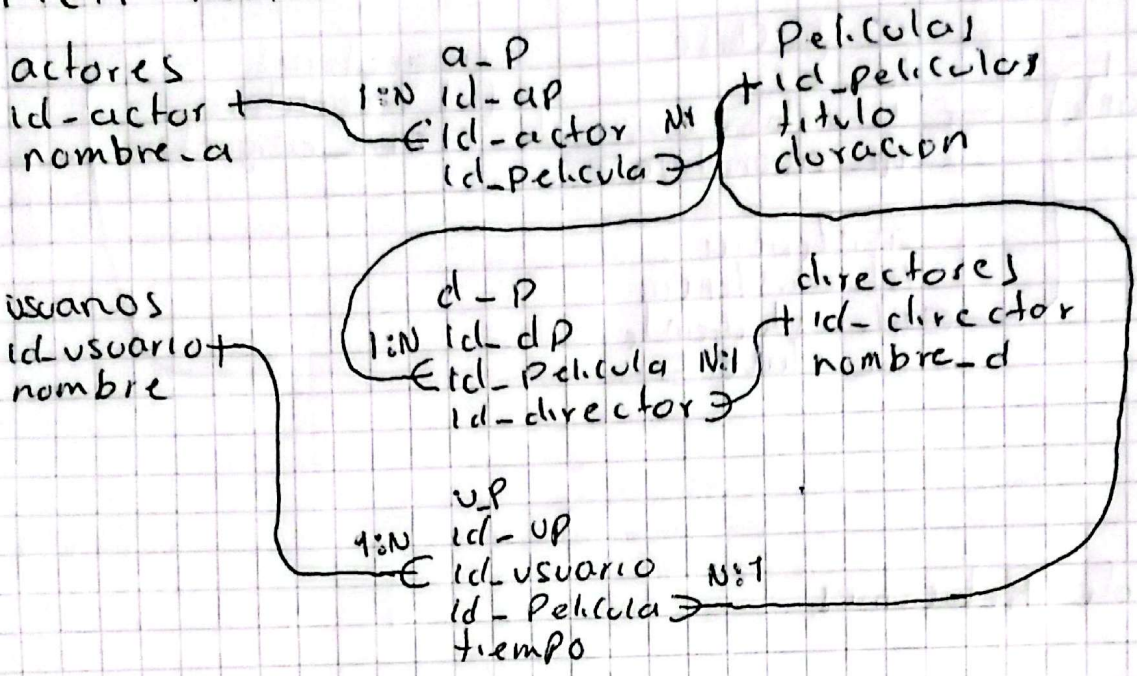
MER netflix



Explicacion de relaciones

1 actores N:M Peliculas

relacion de muchos a muchos por medio de la tabla pivote a_p

- un actor puede estar en muchas peliculas
- una pelicula puede tener muchos actores

2 directores N:M Peliculas

relacion de muchos a muchos por medio de la tabla pivote d_p

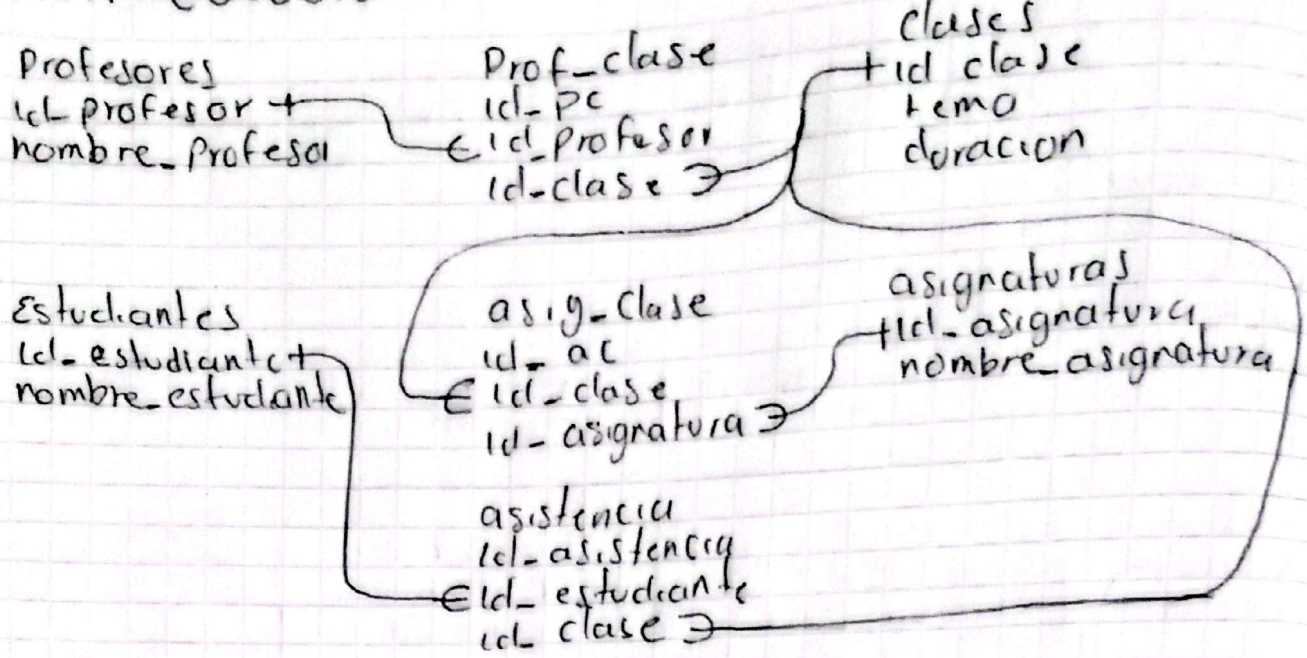
- un director puede dirigir varias peliculas
- una pelicula puede tener varios directores

3 usuarios N:M Peliculas

relacion muchos a muchos mediante la tabla pivote u_p

- un usuario puede ver varias peliculas
- una pelicula puede ser vista por varios usuarios

MER COLEGIO



Explicacion de Relaciones

1. Profesores N:M clases

relacion de muchos a muchos por medio de la tabla Pivote prof_clase

- un profesor puede estar en muchas clases
- una clase puede tener muchos profesores

2. asignaturas N:M Clases
relacion de muchos a muchos por medio de la tabla Pivote asig_clase

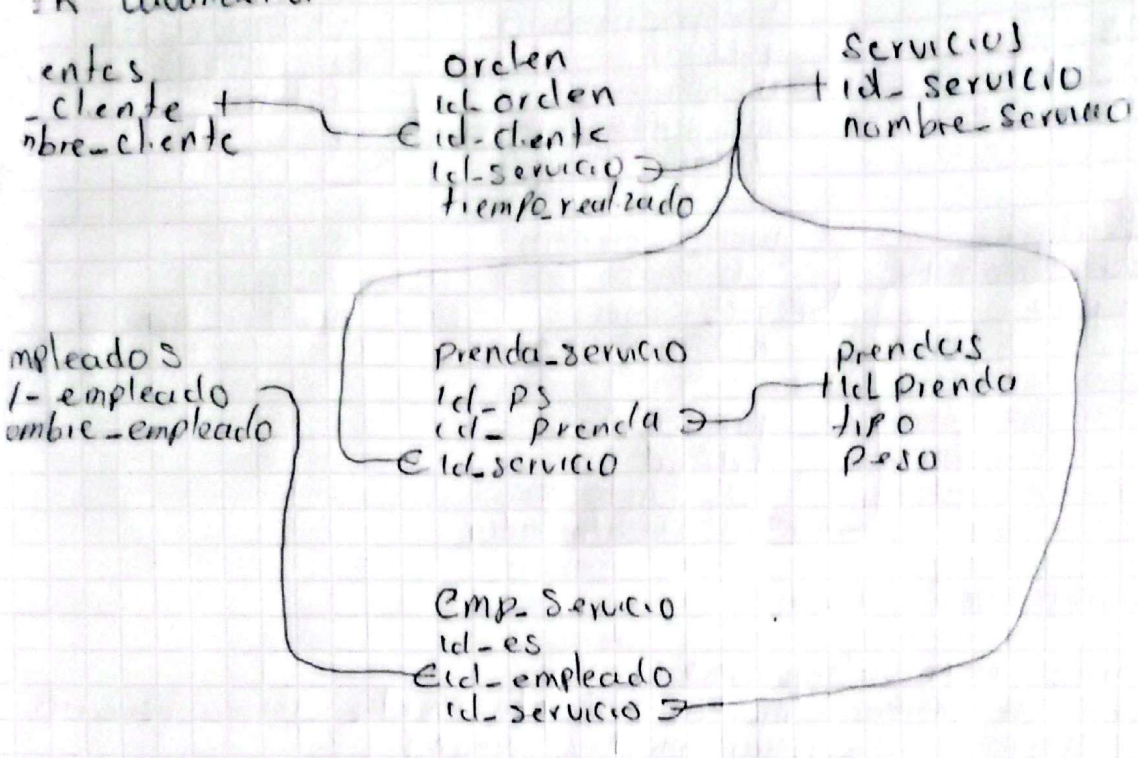
- una asignatura se puede dar en muchas clases
- una clase puede dar muchas asignaturas

3. Estudiantes N:M clases

- relacion de muchos a muchos por medio de la tabla Pivote asistencia

- un estudiante puede asistir a muchas clases
- en una clase pueden asistir muchos estudiantes

:R lavanderia



Explicacion de relaciones - lavanderia

1 empleados N:M servicios

relacion de muchos a muchos a traves de la tabla pivote emp_servicio

- un empleado puede realizar varios servicios
- un servicio puede ser realizado por varios empleados

2 Prendas N:M servicios

relacion de muchos a muchos mediante la tabla pivote prenda_servicio

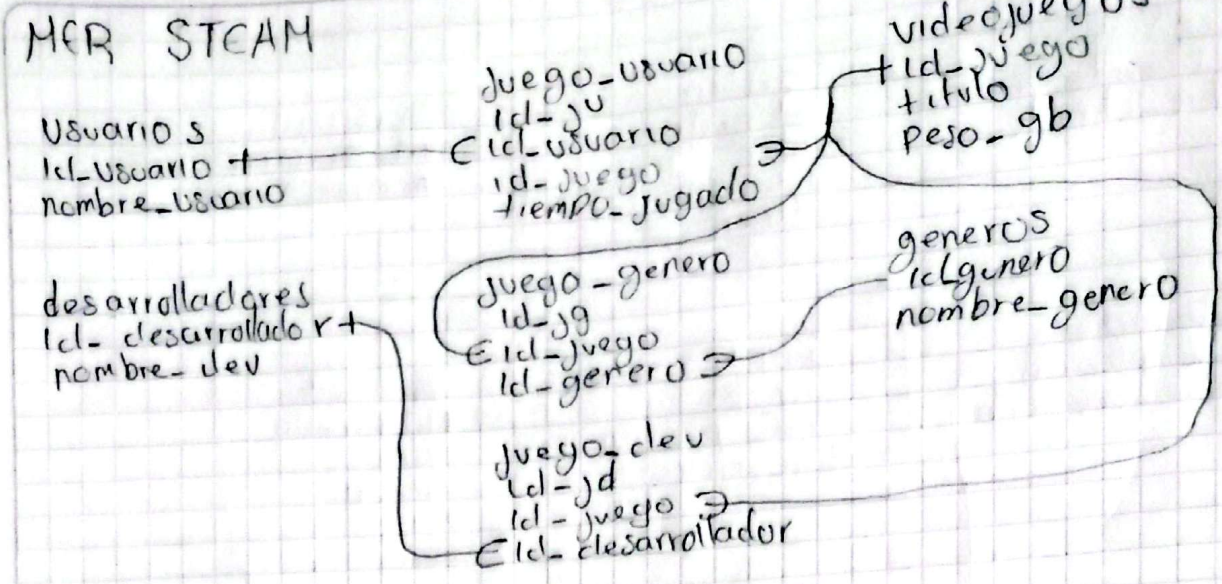
- una prenda puede recibir varios servicios
- un servicio puede aplicarse a varios tipos de prenda

3 clientes N:M servicios

relacion muchos a muchos mediante la tabla pivote orden

- un cliente puede solicitar varios servicios
- un servicio puede ser solicitado por varios clientes

MER STEAM



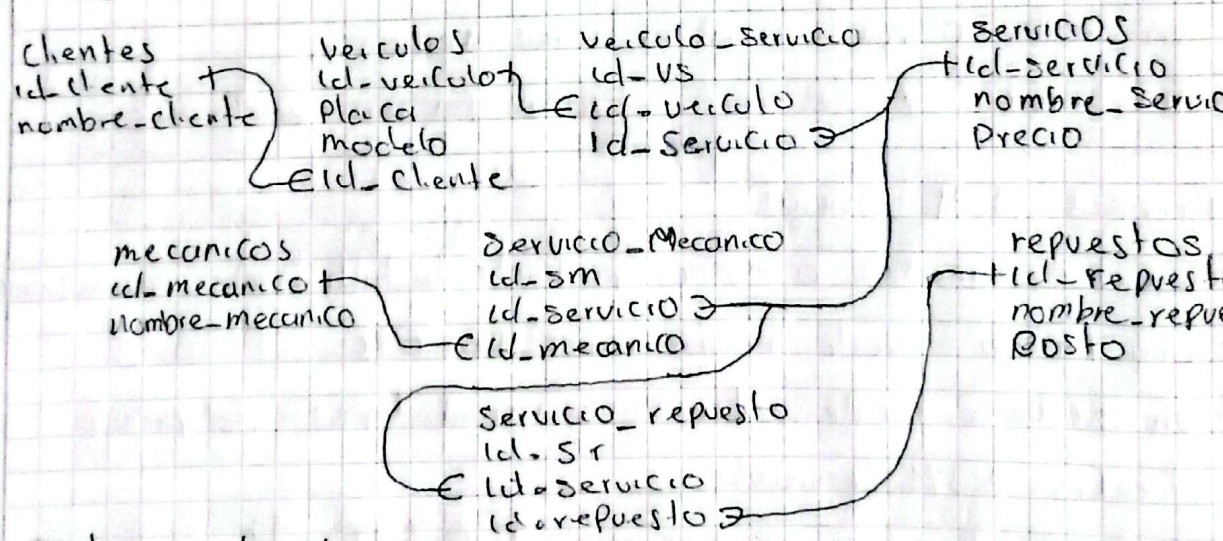
Explicaciones MER Steam

1 usuarios N:M videojuegos
relacion de muchos a muchos mediante Pivote juego_usuario
- un usuario puede jugar muchos juegos
- un juego puede ser jugado por muchos usuarios

2 videojuego N:M generos
relacion de muchos a muchos mediante Pivote juego_genero
- un juego puede pertenecer a varios generos
- un genero puede agrupar muchos juegos

3 videojuego N:M desarrolladores / mediante tiempo_realizado
- un cliente puede solicitar varios
- un servicio puede ser solicitado por varios clientes

MER TALLER



Explicacion de relaciones

1:N → clientes 1:N vehiculos { un cliente puede tener varios vehiculos pero cada vehiculo es de un solo cliente

N:M

- servicios N:M mecanicos / Pivote: servicio_mecanico
un servicios puede ser realizado por varios mecanicos y un mecanico puede realizar muchos servicios
- servicios N:M repuestos / Pivote: servicio_repuesto
un servicio puede requerir varios repuestos y un repuesto puede usarse en muchos servicios
- vehiculos N:M servicios / Pivote: vehiculo_servicio
un vehiculo puede recibir muchos servicios, y un mismo servicio puede aplicarse a distintos vehiculos

Diccionario de datos nexflix

tabla	Campo	Tipo de dato	Descripcion
actores	id_actor	int, PK, AI	identificador unico actor
	nombre_a	varchar(30)	nombre actor
directores	id_director	int, PK, AI	identificador unico del director
	nombre_d	varchar(30)	nombre del director
Peliculas	id_pelicula	int, PK, AI	identificador unico Pelicula
	titulo	varchar(50)	titulo de Pelicula
	duracion	time	duracion de Pelicula
usuarios	id_usuario	int, PK, AI	identificador unico usr
	nombre	varchar(30)	nombre usuario
d_p	id_dp	int, PK, AI	id de relacion dire-Peli
	id_director	int, FK	FK que referencia direc(id_direc)
	id_pelicula	int, FK	FK referencia peli(id_pelicula)
a_p	id_ap	int, PK, AI	identificador relacion act-Peli
	id_pelicula	int, FK	referencia peli(id_pelicula)
	id_actor	int, FK	referencia actor(id_actor)
u_p	id_up	int PK AI	id relacion usuario-Pelicula
	id_usuario	int FK	referencia usuario(id_usuario)
	id_pelicula	int FK	referencia Peli (id_pelicula)
	tiempo	time	tiempo que el usuario ha visto una pelicula

Diccionario de datos Colegio

tabla	campo	tipo de dato	Descripcion
Profesores	idprofesor	int (PK) AI	identificador de profesor
	nombre_profesor	varchar(40)	nombre del profesor
asignaturas	id_asignatura	int (PK) AI	identificador de asignatu
	nombre_asignatura	varchar (40)	nombre de asignatura
clases	id_clase	int (PK) AI	identificador de clase
	tema	varchar (50)	tema que se va ver en
	duracion	time	duracion total clase
estudiantes	id_estudiante	int PK AI	identificador de estudiante
	nombre_estudiante	varchar (30)	nombre del estudiante
prof_clase	id_pc	int PK AI	id de relacion Profe-clase
	id_profesor	int FK	FK hacia Profesores
	id_clase	int FK	FK hacia clases
asig_clase	id_ac	int PK AI	identificador unico regist
	id_clase	int FK	FK hacia clases
	id_asignatura	int FK	FK hacia asignaturas
asistencia	id_asistencia	int PK AI	identificador asistencia
	id_estudiante	int FK	FK hacia estudiantes
	id_clase	int FK	FK hacia clases
	tiempo_presente	time	tiempo que estuvo presente el estudiante en esa clase

Diccionario de Datos - Lavanderia

Tabla	Campo	Tipo de Dato	Descripcion
clientes	id_cliente	int PK AI	identificador de Cl
	nombre_Cliente	varchar(40)	nombre Completo del
Empleados	id_empleado	int PK AI	identificador empleado
	nombre_empleado	varchar(40)	nombre empleado
Prendas	id_prenda	int PK AI	Identificador unico prend
	tipo	varchar(30)	tipo de prenda
	Peso	decimal(5,2)	Peso de prenda (Kg)
Servicios	id_servicio	int PK AI	Identificador de servicio
	nombre_servicio	varchar(50)	nombre del servicio
	duracion	time	Duracion estandar del servi
emp_servicio	id_es	int (PK) AI	identificador del registro
	id_empleado	int FK	FK hacia empleado
	id_servicio	int FK	FK hacia servicios
Prenda_servicio	id_PS	int PK AI	identificador de R= Prenda-ser
	id_prenda	int FK	FK hacia prendas
	id_servicio	int FK	FK hacia servicios
orden	id_orden	int PK AI	identificador de orden
	id_Cliente	int FK	FK hacia Cliente
	id_servicio	int FK	FK hacia servicio
	tiempo_realizado	time	tiempo real que tardo el servicio para el cliente

Diccionario de Datos Steam

Tabla	Campo	Tipo de dato	Descripcion
usuarios	id_usuario	int PK AI	identificador unico de usuario
nombre	nombre_usuario	varchar(30)	nombre de usuario
videojuegos	id_juego	int PK AI	identificador unico del juego
	titulo	varchar(50)	titulo del videojuego
	peso_gb	decimal(5,2)	tamaño del juego en gb
desarrolladores	id_desarrollador	int PK AI	identificador unico develo
	nombre_dev	varchar(40)	Nombre del estudio
generos	id_genero	int PK AI	identificador del genero
	nombre_genero	varchar(30)	nombre del genero
Juego_usuario	id_JU	int PK AI	id del registro de juego por jug
	id_usuario	int FK	FK hacia usuarios
	id_juego	int FK	FK hacia videojuegos
	tiempo_jugado	time	tiempo total jugado po. U
Juego_genero	id_jg	int PK AI	id registro de Pivote
	id_juego	int FK	FK hacia videojuegos
	id_genero	int FK	FK hacia genero
juego_dev	id_jd	int PK AI	id registro de Pivote
	id_juego	int FK	FK hacia juego
	id_desarrollador	int FK	Desarrollador asociado. FK hacia desarrolladores

ccionario de Datos - Taller Mecanico

bla	campo	tipo de dato	Descripcion
ientes	id_Cliente	int PK AI	identificador de cliente
	nombre_Cliente	varchar(50)	nombre del cliente
eículos	id_vehículo	int PK AI	identificador de vehículo
	placa	varchar(10)	placa del vehículo
	modelo	varchar(30)	modelo del vehículo
	id_cliente	int FK	FK hacia clientes
mecanicos	id_mecanico	int PK AI	identificador mecanico
	nombre_mecanico	varchar(50)	nombre del mecanico
servicios	id_servicio	int PK AI	identificador del servicio
	nombre_servicio	varchar(50)	nombre del servicio
	precio	decimal(10,2)	precio del servicio
repuestos	id_repuesto	int PK AI	identificador repuestos
	nombre_repuesto	varchar(50)	Nombre del repuesto
	costo	decimal(10,2)	costo del repuesto
vehículo_servicio	id_vs	int PK AI	identificador de la relacion
	id_vehículo	int FK	FK hacia vehículo
	id_servicio	int FK	servicio que se realizo
	fecha	date	fecha del servicio
servicio_mecanico	id_sm	int PK AI	identificador de relacion
	id_servicio	int FK	servicio realizado
	id_mecanico	int FK	FK mecanico que atiende
servicio_repuesto	id_sr	int PK AI	identificador servicio_repuesto
	id_servicio	int FK	servicio que requiere
	id_repuesto	int FK	repuesto que se compro
	cantidad	int	cantidad de repuestos comprados

```
Create database netflix;
use nexflix;

Create table actores (
 id_actor int Primary key auto_increment,
 nombre_a varchar(30)
);
Create table directores(
 id_director int Primary key auto_increment,
 nombre_d varchar(30)
);
create table peliculas(
 id_pelicula int Primary key auto_increment,
 titulo varchar(50),
 duracion time
);
create table usuarios(
 id_usuario int primary key auto_increment,
 nombre varchar(30)
);
create table d_p(
 id_dp int Primary key auto_increment,
 id_director int,
 id_pelicula int,
 foreign key (id_director) references directores (id_director),
 foreign key (id_pelicula) references peliculas (id_pelicula)
);
create table a_p(
 id_ap int primary key auto_increment,
 id_pelicula int,
 id_actor int,
 foreign key (id_pelicula) references peliculas (id_pelicula),
 foreign key (id_actor) references actores (id_actor)
);
create table u_p(
 id_up int primary key auto_increment,
 id_usuario int,
 id_pelicula int,
 foreign key (id_usuario) references usuarios (id_usuario),
 foreign key (id_pelicula) references peliculas (id_pelicula)
);
-- insertar actores
insert into actores (nombre_a) values
('leonardo dicaprio'),
('scarlett johansson'),
('morgan freeman');
-- insertar datos en directores
insert into directores (nombre_d) values
('christopher nolan'),
('steven spielberg'),
('Quentin Tarantino'),
('neil druckmann');
```

```
-- insertar datos peliculas
insert into peliculas (titulo, duracion) values
('inception', '02:28:00'),
('Pulp Fiction', '02:34:00'),
('jurassic park', '02:07:00'),
('the last of us', '02:07:00');

-- insertar datos en usuarios
insert into usuarios (nombre) values
('Luis'),
('Ana'),
('Carlos');

-- insertar datos en d_p (director - pelicula)
insert into d_p (id_director, id_pelicula) values
(1,1), --nolan -> inception
(3, 2), --tarantino -> pulp
(2,3); -- spielberg -> jurassic

-- insertar datos en a_p (actor pelicula)
insert into a_p (id_pelicula, id_actor) values
(1,1), -- dicaprio -> inception
(2,2), -- johansson -> pulp
(3,3); -- freeman -> jurassic

-- insertar datos en u_p (usuario - pelicula)
insert into u_p (id_usuario, id_pelicula, tiempo) values
(1, 1, '01:00:00'), -- luis vio una hora inception
(2,2, '00:45:00'), -- ana vio 45 min de pulp
(3,3, '01:30:00'); -- carlos vio 1h 30 min de jurasic

-- Elimina el actor que se llama morgan freeman de la bas!
delete from a_p
where id_actor = 3;

delete from actores
where id_actor = 3;

-- Borra el registro de la pelicula pulp fiction

delete from d_p
where id_pelicula = 2;

delete from ap
where id_pelicula = 2;

delete from u_p
where id_pelicula = 2;

delete from peliculas
where id_pelicula = 2;

-- Update

Update directores
set nombre_d = 'chris nolan'
where id_director = 1;

update peliculas
set duracion = '02:30:00'
where id_pelicula = 1;
```

```
ate usuarios
  duracion = 'luis conde'
ere id_usuario = 1;

date u_p
t tiempo = '01:15:00'
ere id_usuario = 2;

date actores
t nombre_a = 'leo dicaprio'
ere id_actor = 1;

elect u.nombre, p.titulo
om usuarios as u
nner join u_p as up on up.id_usuario = u.id_usuario
nner join peliculas as p on p.id_pelicula = up.id_pelicula;

left join
elect p.titulo, d.nombre_d
rom peliculas as p
eft join d_p as dp on dp.id_pelicula = p.id_pelicula
eft join directores as d on d.id_director = dp.id_director;

right join
elect p.titulo, d.nombre_d
s from peliculas as p
right join d_p as dp on dp.id_pelicula = p.id_pelicula
right join directores as d on d.id_director = dp.id_director;

- Count, Contar cuantas peliculas han visto que usuarios
s C select u.nombre, u.id_usuario, count (nombre) as cantidad_pistas
from u_p as up
inner join usuarios as u on u.id_usuario = up.id_usuario
group by u.nombre, u.id_usuario
order by u.nombre;

-- sum
select sum (tiempo)
from up
where tiempo = 2;
-- avg
select avg (duracion)
from peliculas;
-- max con subconsulta
select titulo, max (duracion) as tiempomax
from peliculas
group by titulo
order by tiempomax desc;
-- max con subconsulta
select titulo, duracion
from peliculas
where duracion = (select max (duracion) from peliculas);
-- min
select titulo, min (duracion) as minDura
from pelicula;
group by titulo
order by minDura;
```

--min con subconsulta

```
select titulo, duracion
from Peliculas
where duracion = (select min(duracion) from Peliculas);
```

-- right join con count y group by
```
select u.nombre, count(up.id_up) as cant
from up as up
right join usuarios as u on u.id_usuario = up.id_usuario
group by u.nombre
```

-- uso de having
```
select u.nombre,
sum (up.tiempo) as visto
from up as up
inner join usuarios as u on up.id_usuario = u.id_usuario
group by u.nombre
having visto > '01:00:00';
```

-- avg con subconsulta, usuarios que han visto contenido
-- menos tiempo que el promedio

```
select u.nombre, up.tiempo
from usuarios as u
inner join up as up on u.id_usuario = up.id_usuario
where up.tiempo < (select avg(tiempo) from up);
```

```
-- Crear la base de datos

Create database lavanderia;
use lavanderia;

-- Tablas

Create table clientes (
id_cliente int primary key auto_increment,
nombre_cliente varchar(40)
);

Create table empleados (
id_empleado int primary key auto_increment,
nombre_empleado varchar(40)
);

Create table prendas (
id_prenda int primary key auto_increment,
tipo varchar(30),
peso decimal(5,2) -- kg
);

Create table servicios (
id_servicio int primary key auto_increment,
nombre_servicio varchar(50),
duracion time
);

-- Tablas pivote

Create table emp_servicio (
id_es int primary key auto_increment,
id_empleado int,
id_servicio int,
foreign key (id_empleado) references empleados (id_empleado),
foreign key (id_servicio) references servicios (id_servicio)
);

Create table prenda_servicio (
id_ps int primary key auto_increment,
id_prenda int,
id_servicio int,
foreign key (id_prenda) references prendas (id_prenda),
foreign key (id_servicio) references servicios (id_servicio)
);

Create tabla orden(
id_orden int primary key auto_increment,
id_cliente int,
id_servicio int,
tiempo_realizado time,
foreign key (id_cliente) references clientes (id_cliente),
foreign key (id_servicio) references servicios (id_servicios)
);
```

```
-- INSERTS
insert into clientes (nombre_cliente) values
('Luis Perez'),
('Andreu Gomez'),
('Carlos Lopez');

insert into empleados (nombre_empleado) values
('Juan Martinez'),
('Lucia Torres'),
('Pedro Jimenez');

insert into prenda (tipo, peso) values
('camisa', 0.30),
('pantalon', 0.50),
('cobija', 2.00),
('chaqueta', 1.20);

insert into servicios (nombre_servicio, duracion) values
('lavado normal', '01:00:00'),
('lavado delicado', '01:30:00'),
('secado', '00:45:00'),
('planchado', '00:30:00');

insert into emp_servicio (id_empleado, id_servicio) values
(1, 1),
(2, 2),
(3, 3);

insert into prenda_servicio (id_prenda, id_servicio) values
(1, 1),
(2, 1),
(3, 2);

insert into orden (id_cliente, id_servicio, tiempo_realizado)
values
(1, 1, '00:55:00'),
(2, 2, '01:20:00'),
(3, 3, '00:40:00');

-- DELETE

delete from prenda_servicio where id_prenda = 2;
delete from prendas where id_prenda = 3;

delete from emp_servicio where id_servicio = 2;
delete from orden where id_servicio = 2;
delete from servicios where id_servicio = 2;

-- UPDATE

update empleados set nombre_empleado = 'Juan M.'
where id_empleado = 1;
update servicios set duracion = '01:10:00'
where id_servicio 1;
update clientes set nombre_cliente = 'Luis Conde'
where id_cliente = 1;
update orden set tempo_realizado = '01:00:00'
where id_cliente = 2;
update prendas set tipo = 'camisa Manga Larga'
where id_prenda = 1;
```

INNER JOIN

```
ect c.nombre_cliente, s.nombre_servicio
om clientes as c
ner join order as o on c.id_cliente = o.id_cliente
ner join servicios as s on s.id_servicio = o.id_servicio;
```

LEFT JOIN

```
select s.nombre_servicio, e.nombre_empleado
from servicios as s
left join emp_servicio as es on s.id_servicio = es.id_servicio
left join empleados as e on e.id_empleado = es.id_empleado;
```

-- RIGHT JOIN

```
select s.nombre_servicio, e.nombre_empleado
from servicios as s
right join emp_servicio as es on s.id_servicio = es.id_servicio
right join empleados as e on e.id_empleado = es.id_empleado;
```

-- COUNT

```
select c.nombre_cliente, count(o.id_orden) as ordenes_realizadas
from orden as o
inner join clientes as c on c.id_cliente = o.id_cliente
group by c.nombre_cliente;
```

-- SUM

```
select sum (tiempo_realizado)
from orden
where tiempo_realizado = '00:55:00';
```

-- AVG

```
select avg (peso) from Prendas
```

-- MAX

```
select tipo, max (peso) as max_peso
from prendas
group by tipo
order by max_peso desc;
```

-- MAX con Subconsulta

```
select tipo, peso
from prendas
where peso = (select max (peso) from prendas);
```

-- MIN

```
select tipo min(peso) as min_peso
from prendas
group by tipo
order by min_peso;
```

-- MIN con subconsulta

```
select tipo, peso
from prendas
where peso = (select min(peso) from prendas);
```

-- RIGHT JOIN con COUNT

```
Select c.nombre_cliente, count(o.id_order) as contador
from order as o
right join clientes as c on c.id_cliente = o.id_cliente
group by c.nombre_cliente;
```

-- SUM con HAVING

```
Select c.nombre_cliente, sum(o.tiempo_realizado) as tiempo_total
from order as o
inner join clientes as c on o.id_cliente = c.id_cliente
group by c.nombre_cliente
having tiempo_total > '00:45:00';
```

-- AVG con subconsulta

```
Select c.nombre_cliente, o.tiempo_realizado
from clientes as c
inner join order as o on c.id_cliente = o.id_cliente
where o.tiempo_realizado < (select avg(tiempo_realizado)
from orden);
```

```
-- Crear base de datos
create database colegio;
use colegio;
-- tablas

create table profesores (
  id_profesor int primary key auto_increment,
  nombre_profesor varchar(40)
);
create table asignaturas (
  id_asignatura int primary key auto_increment,
  nombre_asignatura varchar(40)
);
create table clases (
  id_clase int primary key auto_increment,
  tema varchar(50),
  duracion time
);
create table estudiantes (
  id_estudiante int primary key auto_increment,
  nombre_estudiante varchar(30)
);

-- tablas Pivote

create table prof_clase (
  id_pc int primary key auto_increment,
  id_profesor int,
  id_clase int,
  foreign key (id_profesor) references profesores (id_profesor),
  foreign key (id_clase) references clases (id_clase)
);
create table asig_clase (
  id_ac int primary key auto_incvement,
  id_clase int,
  id_asignatura int,
  foreign key (id_asignatura) references asignaturas (id_asignatura),
  foreign key (id_clase) references clases (id_clases
);
create table asistencia (
  id_asistencia int primary key auto_increment,
  id_estudiante int,
  id_clase int,
  tiempo_presente time,
  foreign key (id_estudiante) references estudiantes (id_estudiante),
  foreign key (id_clase) references clases (id_clase)
);
```

-- Insertar datos

```
insert into profesores (nombre_profesor) values
('carlos ramirez'),
('Luisa Fernandez'),
('Jorge peña');
insert into asignaturas (nombre_asignatura) values
('Matematicas'),
('Ciencias'),
('ingles'),
('informatica');
insert into clases (tema, duracion) values
('Ecuaciones de segundo grado', '01:30:00'),
('fotosintesis', '01:15:00'),
('Pasado simple', '01:20:00'),
('Redes LAN', '02:00:00');
insert into estudiantes (nombre_estudiantes) values
('Maria'),
('Santiago'),
('Daniela');
insert into prof_clase (id_profesor, id_clases) values
(1, 1),
(2,2),
(3, 3);
insert into asig_clase (id_clase, id_asignatura) values
(1, 1),
(2, 2),
(3, 3);
insert into asistencia (id_estudiante, id_clase, tiempo_presente)
values
(1, 1, '01:20:00'),
(2, 2, '01:10:00'),
(3, 3, '01:30:00');
```

-- Delete

```
delete from asig_clase where id_asignatura = 2;
delete from asignaturas where id_asignatura = 2;

delete from asistencia where id_clase = 3;
delete from prof_clase where id_clase = 3;
delete from asig_clase where id_clase = 3;
delete from clases where id_clase = 3;
```

-- update

```
update profesores set nombre_profesor = 'Carlos R.' where id_profesor = 1;
update clases set duracion = '01:45:00' where id_clase = 1;
update estudiantes set nombre_estudiante = 'Maria Conde' where id_
estudiante = 1;
update asistencia set tiempo_presente = '01:00:00' where id_estudiante = 2;
update asignaturas set nombre_asignatura = 'Matematicas Avanzadas'
where id_asignatura = 1;
```

```
ner join
t e.nombre_estudiante, c.tema
estudiantes as e
join asistencia as a on e.id_estudiante = a.id_estudiante
join clases as c on c.id_clase = a.id_clase;

eft join
ct c.tema, p.nombre_profesor
m clases as c
t join prof_clase as pc on c.id_clase = pc.id_clase
ft join profesores as p on p.id_profesores = pc.id_profesor;

right join
lect c.tema, p.hombre_profesor
om clases as c
ight join prof_clase as pc on c.id_clase = pc.id_clase
ight join profesores as p on p.id_profesor = pc.id_profesor

-- Count
select e.nombre_estudiante, count(a.id_asistencia)
as clases_asistidas
from asistencia as a
inner join estudiantes as e on e.id_estudiante = a.id_estudiante
group by e.nombre_estudiante;

-- Sum
select sum(tiempo_presente) from asistencia where tiempo_pr-
esente = '01:20:00';

-- Avg
select avg(duracion) from clases;

-- Max
select tema, max(duracion) as duracion_max
from clases
group by tema
order by duracion_max desc;

-- Max con subconsulta
select tema, duracion
from clases
where duracion = (select max(duracion) from clases);

-- Min
select tema, min(duracion) as duracion_min
from clases
group by tema
order by duracion_min;

-- min con subconsulta
select tema, duracion
from clases
where duracion = (select min(duracion) from clases);
```

-- Right join con Count

```
Select e.nombre_estudiante, Count(a.id_asistencia)
as cantidad
from asistencias as a
right join estudiantes as e on e.id_estudiante = a.id_estudiante
group by e.nombre_estudiante;
```

-- Sum con having

```
Select e.nombre_estudiante (sum (a.tiempo_presente)
as tiempo total
from asistencia as a
inner join estudiantes as e on a.id_estudiante=e.id_estud
group by e.nombre_estudiante
having tiempo_total > '01:00:00';
```

-- Avg Con Subconsulta

```
Select e.nombre_estudiante, a.tiempo_presente
from estudiantes as e
inner join asistencia as a on e.id_estudiante = a.id_estudiante
where a.tiempo_presente < (select avg (tiempo_
presente) from asistencia);
```

```
--Crear base de datos
create database steam;
use steam;

--Tablas
create table usuarios (
id_usuario int primary key auto_increment,
nombre_usuario varchar (30)
);

create table videojuegos (
id_juego int primary key auto_increment,
titulo varchar (50),
peso_gb decimal (5,2)
);

create table desarrolladores (
id_desarrollador int primary key auto_increment,
nombre_dev varchar (40)
);

create table generos (
id_genero int primary key auto_increment,
nombre_genero varchar (30)
);

--Tablas

create table juego_usuario (
id_ju int primary key auto_increment,
id_usuario int,
id_juego int,
tiempo_jugado time,
foreign key (id_usuario) references usuarios (id_usuario),
foreign key (id_juego) references videojuegos (id_juego)
);

create table juego_genero (
id_jg int primary key auto_increment,
id_juego int,
id_genero int,
foreign key (id_juego) references videojuegos (id_juego),
foreign key (id_genero) references generos (id_genero)
);

create table juego_dev (
id_jd int primary key auto_increment,
id_juego int,
id_desarrollador int,
foreign key (id_juego) references videojuegos (id_juego),
foreign key (id_desarrollador) references desarrolladores (id_desarrollador)
);
```

```
--Insertar
insert into usuarios (nombre_usuario) values
('David Conde'),
('Laya Garcia'),
('Esteban Diaz');

insert into videojuegos (titulo, peso_gb) values
('Red Dead Redemption 2', 120.5),
('Hollow Knight', 9.7),
('Cyberpunk 2077', 103.3),
('Stardew Valley', 0.8);

insert into desarrolladores (nombre_dev) values
('Rockstar Games'),
('Team Cherry'),
('CD Projekt'),
('Concerned Ape');

insert into genero (nombre_genero) values
('Aventura'),
('Accion'),
('RPG'),
('simulacion');

insert into juego_usuario (id_usuario, id_juego, tiempo_jugado) values
(1,1,'04:20:00'),
(2,2,'01:45:00'),
(3,3,'03:15:00');

insert into juego_genero (id_juego, id_genero) values
(1,1),
(1,2),
(2,1),
(3,2),
(3,3),
(4,4);

insert into juego_dev (id_juego, id_desarrollador) values
(1,1),
(2,2),
(3,3),
(4,4);

--DELETE juego Hollow Knight
Delete from juego_usuario where id_juego=2;
delete from juego_genero where id_juego=2;
delete from juego_dev where id_juego=2;
delete from videojuegos where id_juego=2;

--Delete genero Aventura
delete from juego_genero where id_genero=1;
delete from generos where id_genero=1;

--Update nombre de usuario
update usuarios
set nombre_usuario='David C'.
where id_usuario=1;
--Update peso del juego
update videojuegos
set peso_gb=125.0
where id_juego=1;
```

```
date desarrollador
e desarrolladores
nombre_dev = 'Rockstar'
. id_desarrollador = 1;
date tiempo jugado
te juego_usuario
tiempo_jugado = '05:00:00'
e id_usuario = 1;
date genero
te generos
nombre_genero = 'Aventura Espacial'
re id_genero = 2;

nner join para ver que juegos jugó cada usuario
ct u.nombre_usuario, v.titulo
m usuarios as u
er join juego_usuario as ju on u.id_usuario = ju.id_usuario
ner join videojuegos as v on v.id_juego = ju.id_juego;
eft join para ver todos los juegos y su desarrollador
ct v.titulo, d.nombre_dev
m videojuegos as v
ft join juego_dev as jd on jd.id_juego = v.id_juego
ft join desarrolladores as d on d.id_desarrollador = jd.id_desarrollador

ues
right join para ver todos los desarrolladores y juegos

elect v.titulo, d.nombre_dev
rom videojuegos as v
ight join juego_dev as jd on jd.id_juego = v.id_juego
ight join desarrolladores as d on d.id_desarrollador = jd.id_desarrollador

- count cuantos juegos ha jugado cada jugador
select u.nombre_usuario, count(ju.id_ju) as juegos_jugados
from juego_usuario as ju
inner join usuarios as u on u.id_usuario = ju.id_usuario
group by u.nombre_usuario;

-- sum total de tiempo jugado en juegos que superen 3h
select sum(tiempo_jugado)
from juego_usuario
where tiempo_jugado > '03:00:00';

-- avg promedio de peso de los juegos
select avg(peso_gb)
from videojuegos;

-- max juego con mayor peso
select titulo, max(peso_gb) as peso_maximo
from videojuegos
group by titulo
order by peso_maximo desc;

-- max con subconsulta
select titulo, peso_gb
from video juegos
where peso_gb = (select max(peso_gb) from videojuegos)
```

```
-min juego con menor peso
select titulo, min (peso_gb) as Peso_minimo
from videojuegos
group by titulo
order by Peso_minimo;

-- min con subconsulta
select titulo, peso_gb
from videojuegos
where peso_gb = (select min(peso_gb) from videojuegos);

-- right join con count y group by
select u.nombre_usuario, count (ju.id_ju) as cantidad
from juego_usuario as ju
right join usuarios as u on u.id_usuario = ju.id_usuario
group by u.nombre_usuario;

-- sum con having usuarios que jugaron mas de dos horas
-- en total

select u.nombre_usuario, sum(ju.tiempo_jugado) as totaltiempo
from juego_usuario as ju
inner join usuarios as u on u.id_usuario = ju.id_usuario
group by u.nombre_usuario
having total_tiempo > '02:00:00';

-- avg con subconsulta usuarios que han jugado menos que el prom

select u.nombre_usuario, ju.tiempo_jugado
from usuarios as
inner join juego_usuario as ju on u.id_usuario = ju.id_usuario
where ju.tiempo_jugado < (select avg (tiempo_jugado)
from juego_usuario);
```

```
--Crear base de datos drop database if exists taller_mecanico;
create database taller_mecanico;
use taller_mecanico;
--Tablas

create table clientes (
id_cliente int primary key
auto_increment,
nombre_cliente varchar (50)
);

Create tabla vehiculos (
id_vehiculo int primary key
auto_increment,
placa varchar (10),
modelo varchar (30),
id_cliente int,
Foreign Key (id_cliente)
references clientes (id_cliente)
);

Create tabla mecanicos (
id_mecanico int primary key
auto_increment,
nombre_mecanico varchar (50)
);

Create table servicios (
id_servicio int primary key auto_increment,
nombre_servicio varchar (50),
precio decimal (10,2)
);

Create table repuestos (
id_repuesto int primary key auto_increment,
nombre_repuesto varchar (50),
costo decimal (10,2)
);

--Tablas Pivote

Create table vehiculo_servicio (
id_vs int primary key auto_increment,
id_vehiculo int,
id_servicio int,
fecha date,
foreign key (id_vehiculo) references vehiculos (id_vehiculo),
foreign key (id_servicio) references servicios (id_servicio)
);
```

```
create table servicio_mecanico(
id_sm int primary key auto_increment,
id_servicio int,
id_mecanico int,
foreign key (id_servicio) refaences servicios (id_servicio),
foreign key (id_mecanico) references mecanicos (id_mecanico)
);

Create table servicio_repuesto (
id_sr int primary key auto_increment,
id_servicio int,
id_repuesto int,
cantidad int,
foreign key (id_servicio) references servicios (id_servicio),
foreign key (id_repuesto) references repuestos (id_repuesto).
);

-- Insertar

insert into clientes (nombre_cliente) values
('Luis David Conde'),
('Maria Torres'),
('Carlos Gómez');

insert into vehiculos (placa, modelo, id_cliente) values
('ABC123', 'Mazda 3', 1),
('XYZ987', 'Toyota Hilux', 2),
('LMN456', 'Renault Logan', 3);

insert into mecanicos (nombre_mecanico) values
('Andres Ruiz'),
('Javier Peña'),
('Lucia Parra');

insert into servicios (nombre_servicio, precio) values
('Cambio de aceite', 100000),
('Alineación y balanceo' 80000),
('Revisión de frenos', 120000),
('Cambio de llantas', 300000);

insert into repuestos (nombre_repuesto, costo) values
('Filtro de aceite', 25000),
('Pastillas de freno', 60000),
('Llanta Michelin 16''', 250000),
('Aceite Sintetico 5W30', 45000);

insert into vehiculo_servicio (id_vehiculo, id_servicio, fecha) values
(1,1, '2025-06-01'),
(2,2, '2025-06-03'),
(3,3 '2025-06-05');

insert into servicio_mecanico (id_servicio, id_mecanico) values
(1,1),
(2,2),
(3,3),
(4,2);
```

into servicio_repuesto (id_servicio, id_repuesto, cantidad) values
(1,1), -- cambio de aceite → Filtro de aceite
(4,1), -- Cambio de aceite → Aceite Sintético
(2,1), -- Revisión de Frenos → Postillos de freno
(1,2,4), -- Cambio de llantas → 4 llantas

DELETE: Eliminar datos

-1. Eliminar servicio 2
delete from servicio_mecanico where id_servicio=2;
delete from vehiculo_servicio where id_servicio=2;
delete from servicio_repuesto where id_servicio=2;
delete from servicios where id_servicio=2;

--2. Eliminar Cliente 3 y su vehiculo
delete from vehiculo_servicio where id_vehiculo=3;
delete from vehiculos where id_vehiculo=3;
delete from clientes where id_cliente=3;

--UPDATE

--1. actualizar nombre del cliente
update clientes set nombre_Cliente='Luis D. Conde' where id_Cliente=1;

--2. Actualizar precio del servicio
update servicios set precio=105000 where id_servicio=1;

--3. Actualizar nombre del mecanico
update mecanicos set nombre_mecanico='A. Ruiz' where id_mecanico=1;

--4. Actualizar placa del vehiculo
update vehiculos set placa='DEF456' where id_vehiculo=1;

--5. Actualizar nombre del servicio
update servicios set nombre_servicio='Cambio completo de aceite'
where id_servicio=1;

--INNER JOIN: Ver servicios realizados por cada cliente
select c.nombre_cliente, s.nombre_servicio, vs.fecha
from clientes as c
inner join vehiculos as v on v.id_cliente=c.id_cliente
inner join vehicub_servicio as on vs.id_vehiculo=v.id_vehiculo
inner join servicios as s on s.id_servicio=vs.id_servicio;

--LEFT JOIN: Ver todos los servicios con o sin mecanico asignado
select s.nombre_servicio, m.nombre_mecanico
from servicios as s
left join servicio_mecanico as sm on sm.id_servicio=s.id_servicio
left join mecanicos as m on m.id_mecanico=sm.id_mecanico;

--RIGHT JOIN: Ver todos los mecanicos y los servicios que han realizado
select s.nombre_servicio, m.nombre_mecanico
from servicios as s
right join servicio_mecanico as sm on sm.id_servicio=s.id_servicio
right join mecanicos as m on m.id_mecanico=sm.id_mecanico;

```
-- COUNT: Cuántos servicios ha recibido cada cliente
select c.nombre_cliente, count (vs.id_vs) as servicios_recibidos
from vehiculo_servicio as vs
inner join vehiculos as v on vs.id_vehiculo = v.id_vehiculo
inner join clientes as c on v.id_cliente = c.id_cliente
group by c.nombre_Cliente;

-- SUM: Total gastado por cada cliente
select c.nombre_Cliente, sum (s.precio) as total_gastado
from vehiculo_servicio as vs
inner join servicios as s on s.id_servicio = vs.id_servicio
inner join vehiculos as v on v.id_vehiculo = vs.id_vehiculo
inner join clientes as c on c.id_cliente = v.id_cliente
group by c.nombre_Cliente;

-- AVG: promedio de precios de servicios
select avg (precio) from servicios;

-- Max: Servicio más costoso
select nombre_servicio, max (precio) as max_precio
from servicos
group by nombre_servicio
order by max_precio desc;

-- MAX con subconsulta
select nombre_servicio, precio
from servicios
where precios = (select max (precio) from servicios);

-- MIN: Servicio más barato
select nombre_servicio, min (precio) as min_precio
from servicios
group by nombre_servicio
order by min_precio asc;

-- MIN con subconsulta
select nombre_servicio, precio
from servicios
where precio = (select min (precio) from servicios);
-- RIGHT JOIN con COUNT y GROUP BY: Servicio por mecánica
select m.nombre_mecanico, count (sm.id_sm) as servicios_realizados
from servicio_mecanico as sm
right join mecanicos as m on m.id_mecanico = sm.id_mecanico
group by m.nombre_mecanico;
-- SUM con HAVING: Clientes que han gastado más de 100000
select c.nombre_cliente, sum (s.precio) as total
from vehiculo_servicio as vs
inner join servicios as s on s.id_servicio = vs.id_servicio
inner join cliente as c on c.id_cliente = v.id_cliente
group by c.nombre_cliente
having total > 100000;

-- AVG con Subconsulta; servicios con precio menor al promedio
select nombre_servicio, precio
from servicios
where precio < (select avg (precio) from servicios);.
```