USE SistemaEmpleadosProyectos;

/*Para optimizar las estructuras de las vistas, crearé diferentes alias para cada una de las entidades y relaciones:

E -> Empleados

D -> Departamentos

P -> Proyectos

A -> Asignaciones

*/

/* Ejercicio 1: Vista de Empleados por Departamento

-- Crea una vista que muestre los empleados agrupados por departamento, mostrando el nombre del empleado, su salario y el nombre del departamento.

EXPLICACIÓN DE LA VISTA:

--- He creado la vista VistadeEmpleadosporDepartamento para mostrar los empleados junto con su departamento. En esta vista, incluyo los siguientes campos:

- El nombre del empleado.

- Su salario.

- El nombre del departamento al que pertenece.

--- He utilizado la sentencia de unión (JOIN) entre las tablas Empleados y Departamentos, relacionando la clave foránea id_departamento de la tabla de empleados con la tabla de departamentos.

--- Esta vista muestra qué empleados están en qué departamento está y su salario.

*/

CREATE VIEW VistadeEmpleadosporDepartamento AS

SELECT E.nombre, E.salario, D.nombre_departamento

FROM Empleados E

```
JOIN Departamentos D
ON E.id_departamento = D.id_departamento;

/* Realizar la consulta de los datos de la vista (VistadeEmpleadosporDepartamento)

SELECT * FROM VistadeEmpleadosporDepartamento;

*/

-- ----------------------------------------------------------------------------

/*  Ejercicio 2: Vista de Proyectos Activos
-- Crea una vista que muestre los proyectos que aún no han terminado (sin fecha de fin o con fecha de fin futura).

EXPLICACIÓN DE LA VISTA:
--- He creado la vista VistadeProyectosActivos para mostrar los proyectos que aún están activos. En esta vista incluyo:
- El nombre del proyecto.
- La fecha de fin del proyecto.
La vista selecciona los proyectos que aún no han terminado de la tabla Proyectos donde la fecha_fin es nula (IS NULL) o es una fecha de fin futura.

--- Para mostrar cuales son los proyectos que tienen una fecha de fin futura o que aún no tienen fecha de finalización he filtrado por las siguientes condiciones:
   --- Fecha_fin IS NULL: Incluye proyectos sin fecha de fin (sin fecha asignada).
   --- Fecha_fin > CURDATE(): Incluye proyectos donde la fecha de finalización es mayor (posterior) a la fecha actual.

*/

CREATE VIEW VistadeProyectosActivos AS
```

```
SELECT nombre_proyecto, fecha_fin
FROM Proyectos P
WHERE fecha_fin IS NULL
OR fecha_fin > CURDATE();

/* Realizar la consulta de los datos de la vista (VistadeProyectosActivos)

SELECT * FROM VistadeProyectosActivos;

*/

-- ------------------------------------------------------------------------------
```

/* Ejercicio 3: Vista de Empleados Asignados a Proyectos

-- Crea una vista que muestre los empleados asignados a cada proyecto, incluyendo el nombre del proyecto y la fecha de asignación.

EXPLICACIÓN DE LA VISTA:

--- Para poder mostrar quienes son los empleados que están asignados a cada proyecto, he incluido en la vista los siguientes campos:

- El nombre del empleado.
- El nombre del proyecto al que está asignado.
- La fecha de asignación del empleado al proyecto.

--- Primero, he unido la tabla Empleados con la tabla intermedia Asignaciones mediante la clave foránea id_empleado para ver qué empleados están asignados.

- He realizado esta unión para después poder relacionar los empleados que estan asignados a cada proyecto con la tabla Proyectos.

--- Luego, he relacionado la tabla Asignaciones con la tabla Proyectos utilizando la clave foránea id_proyecto para mostrar los proyectos correspondientes.

*/

```
CREATE VIEW VistadeEmpleadosAsignadosaProyectos AS
SELECT E.nombre, P.nombre_proyecto, A.fecha_asignacion
FROM Empleados E
JOIN Asignaciones A
ON E.id_empleado = A.id_empleado
JOIN Proyectos P ON
A.id_proyecto = P.id_proyecto;

/* Realizar la consulta de los datos de la vista (VistadeEmpleadosAsignadosaProyectos)

SELECT * FROM VistadeEmpleadosAsignadosaProyectos;

*/

-- ------------------------------------------------------------------------
```

/* Ejercicio 4: Vista de Salarios por Departamento

-- Crea una vista que muestre el salario promedio y el salario total de los empleados en cada departamento.

EXPLICACIÓN DE LA VISTA:

--- En la creación de esta vista, para mostrar el salario promedio y el salario total de los empleados en cada departamento he incluido:

- El nombre del departamento.

- El salario total de todos los empleados en ese departamento.

- El salario promedio de los empleados en ese departamento.

--- Para obtener estos datos, utilizo un JOIN entre las tablas Empleados y Departamentos, relacionando la clave foránea id_departamento de ambas tablas.

--- Luego, uso la función SUM para sumar todos los salarios y la función AVG para calcular el salario promedio de cada departamento.

--- Por último, he agrupado por el nombre del departamento para ver la información de cada departamento.

*/

CREATE VIEW VistadeSalariosPorDepartamento AS

SELECT D.nombre_departamento, SUM(E.salario) AS salario_total, AVG(E.salario) AS salario_promedio

FROM Empleados E

JOIN Departamentos D ON

E.id_departamento = D.id_departamento

GROUP BY D.nombre_departamento;

/* Realizar la consulta de los datos de la vista (VistadeSalariosPorDepartamento)

SELECT * FROM VistadeSalariosPorDepartamento;

*/

-- ----------

/* Ejercicio 5: Vista de Proyectos por Fecha de Inicio

-- Crea una vista que muestre todos los proyectos ordenados por fecha de inicio.

EXPLICACIÓN DE LA VISTA:

--- En la vista VistadeProyectosporFechadeInicio para saber todos los proyectos ordenados por su fecha de inicio, he incluido estos campos:

- El id del proyecto.

- El nombre del proyecto.

- La fecha de inicio de cada proyecto.

--- Para obtener estos datos, selecciono los campos id_proyecto, nombre_proyecto y fecha_inicio de la tabla Proyectos.

--- Luego, ordeno los resultados de manera ascendente por la fecha de inicio, para ver la información de la fecha de los proyectos desde el más antiguo hasta el más reciente.

*/

CREATE VIEW VistadeProyectosporFechadeInicio AS

SELECT id_proyecto, nombre_proyecto, fecha_inicio

FROM Proyectos P

ORDER BY fecha_inicio ASC;

/* Realizar la consulta de los datos de la vista (VistadeProyectosporFechadeInicio)

SELECT * FROM VistadeProyectosporFechadeInicio;

*/

-- --

/*Ejercicio 6: Vista de Empleados con Mayor Participación en Proyectos

-- Crea una vista que muestre los empleados que han participado en más proyectos, incluyendo el número total de proyectos en los que han participado.

EXPLICACIÓN DE LA VISTA:

--- Para mostrar los empleados que han participado en más proyectos. En esta vista he incluido:

- El nombre del empleado.

- El número total de proyectos en los que ha participado.

--- Para llevar a cabo esta vista y visualizar esta información, utilizo un JOIN entre las tablas Empleados y Asignaciones, uniendo las dos tablas mediante id_empleado.

--- Para saber cuantos los proyectos que tiene cada empleado aplico la función COUNT.

- Esta función COUNT(A.id_proyecto) cuenta en cuántos proyectos ha participado cada empleado.

--- Luego, al haber usado la función COUNT() se tiene que agrupar por el campo que se quiere saber la cantidad.

- En este caso se agrupan los resultados por el nombre del empleado y ordeno el número de proyectos de mayor a menor según la participación.

*/

CREATE VIEW VistadeEmpleadosconMayorParticipacionenProyectos AS

SELECT E.nombre, COUNT(A.id_proyecto) AS Nºtotal_proyectos

FROM Empleados E

JOIN Asignaciones A ON E.id_empleado = A.id_empleado

GROUP BY E.nombre

ORDER BY Nºtotal_proyectos DESC;

/* Realizar la consulta de los datos de la vista (VistadeEmpleadosconMayorParticipacionenProyectos)

SELECT * FROM VistadeEmpleadosconMayorParticipacionenProyectos;

*/

-- --

/* Ejercicio 7: Vista de Proyectos sin Empleados Asignados

-- Crea una vista que muestre los proyectos que no tienen empleados asignados.

EXPLICACIÓN DE LA VISTA:

--- Para mostrar la información sobre los proyectos que no tienen empleados asignados, los campos que he incluido en esta vista son:

- El nombre del proyecto.

- El nombre del empleado encargado del proyecto.

--- Para obtener estos datos, utilizo un JOIN entre la tabla Proyectos y la tabla Asignaciones.

--- Luego, he aplicado la condición WHERE para filtrar los proyectos que no tienen empleados asignados (WHERE A.id_empleado IS NULL).

- La condición WHERE permite filtrar por los proyectos que no tienen ningun empleado asignado.

*/

```
CREATE VIEW VistadeProyectossinEmpleadosAsignados AS
SELECT P.nombre_proyecto, E.nombre
FROM Proyectos P
JOIN Asignaciones A
ON A.id_proyecto = P.id_proyecto
JOIN Empleados E
ON A.id_empleado = E.id_empleado
WHERE A.id_asignacion IS NULL;
```

/* Realizar la consulta de los datos de la vista (VistadeProyectossinEmpleadosAsignados)

```
SELECT * FROM VistadeProyectossinEmpleadosAsignados;
```

*/

/* VERIFICAR SI LA VISTA SE HA REALIZADO CORRECTAMENTE MEDIANTE OTRA CONSULTA

--- Al ejecutar la consulta utilizando la condición IS NULL de la vista creada, esta nos muestra que no existe ningún proyecto que no tenga empleados asignados.

- La consulta solo muestra los nombres de los campos seleccionados (nombre_proyecto y nombre del empleado) sin ningún dato (en blanco).

--- Para comprobar si la información es correcta y no hay proyectos sin empleados asignados, he ejecutado la consulta inicial cambiado la cláusula WHERE con la condición IS NOT NULL para verificar que todos los proyectos tienen al menos un empleado asignado (WHERE A.id_empleado IS NOT NULL).

```
SELECT P.nombre_proyecto, E.nombre
FROM Proyectos P
JOIN Asignaciones A
ON A.id_proyecto = P.id_proyecto
JOIN Empleados E
ON A.id_empleado = E.id_empleado
WHERE A.id_empleado IS NOT NULL;
```

- Al ejecutar esta consulta, obtuve como resultado el mismo número de proyectos que existen en la base de datos (5 proyectos en total).

--- En conclusión, la vista creada de este ejercicio me ha permitido verificar que no existe ningún proyecto sin empleados asignados.

- Dado que un empleado puede estar asignado a varios proyectos y un proyecto puede tener varios empleados mediante la tabla intermedia “Asignaciones” con relación (N) —— (N) muchos a muchos, también podemos ver que todos los empleados (10 en total) tienen asignaciones.

- Esto también demuestra que cada proyecto tiene al menos un empleado asignado.

*/

-- --

/* Ejercicio 8: Vista de Empleados y sus Proyectos

-- Crea una vista que muestre los empleados junto con todos los proyectos en los que han trabajado, ordenados por fecha de asignación.

EXPLICACIÓN DE LA VISTA:

--- He creado la vista VistadeEmpleadosysusProyectos para mostrar los empleados junto con los proyectos en los que han trabajado, ordenados por la fecha de asignación. En esta vista he incluido los siguientes campos:

- El ID del empleado.

- El nombre del empleado.

- El nombre del proyecto en el que ha trabajado.

- La fecha de asignación del empleado al proyecto.

--- Para obtener esta información, he utilizado dos uniones (JOIN) entre las tablas Empleados, Asignaciones y Proyectos.

--- Parar mostrar los empleados con los proyectos en los que han trabajado, he seguido la siguiente estructura:

- El primer JOIN ha sido para relacionar la tabla Empleados a través de la tabla intermedia Asignaciones mediante la columna id_empleado, que es la clave foránea en la tabla Asignaciones.

- A continuación, el segundo JOIN lo he usado para enlazar la tabla Asignaciones con la tabla Proyectos, relacionándolas mediante la columna id_proyecto, que también es la clave foránea en Asignaciones.

--- El uso de los dos JOIN permite relacionar la información de los empleados con los proyectos a través de la tabla intermedia Asignaciones, que contiene las dos claves foráneas id_empleado e id_proyecto.

- Esta estructura permite unir las tres tablas para obtener los datos necesarios y mostrarlos a través de la consulta usando la vista.

--- Por último, para poder visualizar los empleados y los proyectos en los que han trabajado en orden cronológico, he utilizado la cláusula ORDER BY seguida del campo A.fecha_asignacion ASC (ORDER BY A.fecha_asignacion ASC).

- Esta cláusula ordena los resultados por la fecha de asignación, de la más antigua a la más reciente, mostrando en qué proyectos ha trabajado cada empleado y en qué orden han sido asignados a cada proyecto.

*/

CREATE VIEW VistadeEmpleadosysusProyectos AS

SELECT E.id_empleado, E.nombre, P.nombre_proyecto, A.fecha_asignacion

FROM Empleados E

JOIN Asignaciones A ON E.id_empleado = A.id_empleado

JOIN Proyectos P ON A.id_proyecto = P.id_proyecto

ORDER BY A.fecha_asignacion ASC;

/* Realizar la consulta de los datos de la vista (VistadeEmpleadosysusProyectos)

SELECT * FROM VistadeEmpleadosysusProyectos;

*/

-- ---

/* Ejercicio 9: Vista de Departamentos y Cantidad de Empleados

-- Crea una vista que muestre los departamentos y el número total de empleados asignados a cada uno.

EXPLICACIÓN DE LA VISTA:

--- He creado la vista VistadeDepartamentosyCantidaddeEmpleados para mostrar la cantidad de empleados que hay en cada departamento. En esta vista, he incluido los siguientes campos:

- El nombre del departamento.

- El número total de empleados asignados a cada departamento.

--- Para obtener los datos sobre la cantidad de empleados por departamento, he seguido esta estructura:

- Para mostrar el número total de empleados asignados a cada departamento, he utilizado la función COUNT(E.id_empleado).

- Esta función cuenta la cantidad de empleados registrados en la tabla Empleados para cada departamento, mostrando cuántos empleados están asignados a cada uno.

- Luego he utilizado un JOIN entre las tablas Departamentos y Empleados, donde he relacionado las tablas Departamentos y Empleadosmediante la clave foránea id_departamento para asociar los empleados con sus respectivos departamentos

- Por último, al utilizar la función COUNT, es necesario agrupar los resultados para obtener la cantidad requerida en la vista.

- Por ello, he agrupado los datos por el nombre de cada departamento utilizando (GROUP BY D.nombre_departamento) para calcular el total de empleados que hay en cada departamento.

*/

```
CREATE VIEW VistadeDepartamentosyCantidaddeEmpleados AS
SELECT D.nombre_departamento, COUNT(E.id_empleado) AS total_empleados
FROM Departamentos D
JOIN Empleados E ON D.id_departamento = E.id_departamento
GROUP BY D.nombre_departamento;
```

/* Realizar la consulta de los datos de la vista (VistadeDepartamentosyCantidaddeEmpleados)

--- Mostrar los departamentos y el número total de empleados en cada uno.

```
SELECT * FROM VistadeDepartamentosyCantidaddeEmpleados;
```

--- Verificar si coincide la cantidad de empleados por departamento con el total de empleados en la empresa

```
SELECT COUNT(E.id_empleado) AS total_empleados_en_empresa
FROM Empleados E;
```

*/

-- --

/* Ejercicio 10: Vista de Proyectos Finalizados

-- Crea una vista que muestre todos los proyectos que ya han finalizado.

EXPLICACIÓN DE LA VISTA:

--- He creado la vista vistadeaproyectosfinalizados para mostrar todos los proyectos que ya han finalizado, es decir, aquellos proyectos donde fecha de fin es anterior a la fecha actual. En esta vista, incluyo los siguientes campos:

- El nombre del proyecto.

- La fecha de fin del proyecto.

- La fecha actual (opcional)

--- Esta vista no utiliza ninguna unión (JOIN), ya que solo hay que mostrar datos de una única tabla (Proyectos).

--- Para obtener solo los proyectos finalizados, he utilizado la condición WHERE fecha_fin < CURDATE().

- Esta condición filtra y compara aquellos proyectos donde la fecha de finalización es anterior a la fecha actual (CURDATE())

--- Aunque solo se necesita mostrar los nombres de los proyectos donde la fecha de fin ya haya pasado, he añadido un campo adicional llamado fecha_actual usando CURDATE() AS fecha_actual.

- Este campo mostrará una columna con la fecha actual para poder comprobar que las fechas de fin son anteriores/menores a las fechas actuales.

--- La sintaxis realizada para la vista muestra solo los proyectos que ya han finalizado.

*/

CREATE VIEW vistadeaproyectosfinalizados AS

SELECT nombre_proyecto, fecha_fin, CURDATE() AS fecha_actual

FROM Proyectos

WHERE fecha_fin < CURDATE();

```
/* Realizar la consulta de los datos de la vista (vistadeaproyectosfinalizados)

SELECT * FROM vistadeaproyectosfinalizados;

*/
```