CONSULTAS

-- Número de clases que realiza cada aprendiz y si no tiene clases, mostrar "SIN CLASES"

```
SELECT A.ID_MIEMBRO, A.NIVEL,
    IF(COUNT(R.CLASES_ID_CLASES) = 0, 'SIN CLASES',
COUNT(R.CLASES_ID_CLASES)) AS TOTAL_CLASES
FROM APRENDICES A LEFT JOIN REALIZAN R
ON A.ID_MIEMBRO = R.APRENDICES_ID_MIEMBRO
GROUP BY A.ID_MIEMBRO, A.NIVEL;
```

-- Aprendices que han realizado más clases que el aprendiz con ID 1

```
SELECT A.ID_MIEMBRO, A.NIVEL, COUNT(R.CLASES_ID_CLASES) AS
TOTAL_CLASES
FROM APRENDICES A INNER JOIN REALIZAN R
ON A.ID_MIEMBRO = R.APRENDICES_ID_MIEMBRO
GROUP BY A.ID_MIEMBRO, A.NIVEL
HAVING COUNT(R.CLASES_ID_CLASES) > (
  SELECT COUNT(R2.CLASES_ID_CLASES)
  FROM APRENDICES A2 INNER JOIN REALIZAN R2
  ON A2.ID_MIEMBRO = R2.APRENDICES_ID_MIEMBRO
  WHERE A2.ID_MIEMBRO = 1
);
```

-- Entrenadores con la última clase dada

```
SELECT E.ID_MIEMBRO, E.RANGO, C.HORARIO AS ULTIMA_CLASE, C.NOMBRE AS
NOMBRE_CLASE
FROM ENTRENADORES E LEFT JOIN CLASES C
ON E.CLASES_ID_CLASES = C.ID_CLASES
WHERE C.HORARIO = (
    SELECT MAX(C2.HORARIO)
    FROM CLASES C2
    WHERE C2.ID_CLASES = E.CLASES_ID_CLASES
  )
LIMIT 8;
```

-- Mostrar los pagos realizados con tarjetas activas y la matrícula asociada

```
SELECT P.NUMERO_TARJETA, P.FECHA_VENCIMIENTO, M.NOMBRE AS
NOMBRE_MATRICULA, P.ESTADO_TARJETA
FROM PAGO P INNER JOIN MATRICULA M
ON P.MATRICULA_ID_MATRICULA = M.ID_MATRICULA
WHERE P.ESTADO_TARJETA = 'ACTIVA';
```

-- Entrenadores con el mayor rango de cada clase

```
SELECT E.ID_MIEMBRO, E.RANGO, C.NOMBRE AS NOMBRE_CLASE
FROM ENTRENADORES E INNER JOIN CLASES C
ON E.CLASES_ID_CLASES = C.ID_CLASES
WHERE E.RANGO = (
  SELECT MAX(E2.RANGO)
  FROM ENTRENADORES E2
  WHERE E2.RANGO  = 'AVANZADO'
);
```

VISTAS

-- Vista 1: Aprendices por Nivel

```
CREATE VIEW vista_aprendices_por_nivel AS
SELECT
  nivel,
  COUNT(*) AS total_aprendices
FROM
  aprendices
GROUP BY
  nivel;
```

-- Vista 2: clases aprendices y si no han dado ninguna (sin clases)

```
CREATE VIEW clases_aprendiz AS
SELECT A.ID_MIEMBRO, A.NIVEL,
   IF(COUNT(R.CLASES_ID_CLASES) = 0, 'SIN CLASES',
COUNT(R.CLASES_ID_CLASES)) AS TOTAL_CLASES
FROM APRENDICES A LEFT JOIN REALIZAN R
ON A.ID_MIEMBRO = R.APRENDICES_ID_MIEMBRO
GROUP BY A.ID_MIEMBRO, A.NIVEL;
```

FUNCIONES

```
-- Función 1: Contar clases dependiendo del objetivo
DELIMITER &&

DROP FUNCTION IF EXISTS contar_aprendices_objetivo&&
CREATE FUNCTION contar_aprendices_objetivo(objetivo_buscar VARCHAR(45))
RETURNS INT UNSIGNED
DETERMINISTIC
BEGIN
   DECLARE total INT UNSIGNED;

   SELECT COUNT(*)
   INTO total
   FROM aprendices
   WHERE objetivo = objetivo_buscar;

   IF total IS NULL THEN
      SET total = 0;
   END IF;

   RETURN total;
END &&

DELIMITER ;
SELECT contar_aprendices_objetivo('Mejorar flexibilidad');
-- Función 2: Contar clases dependiendo del horario que queramos buscar

DELIMITER &&

DROP FUNCTION IF EXISTS contar_clases_horario&&
CREATE FUNCTION contar_clases_horario(horario_buscar VARCHAR(45))
RETURNS INT UNSIGNED
DETERMINISTIC
BEGIN
   DECLARE total INT UNSIGNED;
      SELECT COUNT(*)
   INTO total
   FROM clases
   WHERE horario = horario_buscar;
      IF total IS NULL THEN
      SET total = 0;
   END IF;
      RETURN total;
END &&
DELIMITER ;
SELECT contar_clases_horario('Mañana');
```

PROCEDIMIENTOS

-- Procedimiento 1: Contar el total y las tarifas activas dependiendo de la calidad de la tarifa

```
DELIMITER &&

DROP PROCEDURE IF EXISTS calcular_tarifas_calidad&&
CREATE PROCEDURE calcular_tarifas_calidad(
   IN calidad_buscar VARCHAR(45),
   OUT total_tarifas INT,
   OUT total_activas INT
)
BEGIN
   -- Inicializar las variables
   SET total_tarifas = 0;
   SET total_activas = 0;

   -- Contar todas las tarifas para la calidad que elijamos
   SELECT COUNT(*)
   INTO total_tarifas
   FROM tarifas
   WHERE calidad = calidad_buscar;

   -- Contar solo las tarifas activas para la calidad que queramos
   SELECT COUNT(*)
   INTO total_activas
   FROM tarifas
   WHERE calidad = calidad_buscar
   AND estado = 'Activo';
END &&

DELIMITER ;

-- ver los distintos tipos de tarifas
SELECT DISTINCT calidad
FROM tarifas;

-- para ver las tarifas premium
CALL calcular_tarifas_calidad('Premium', @total_premium, @activas_premium);
SELECT @total_premium AS Total_Premium, @activas_premium AS Activas_Premium;
-- para ver las tarifas vip
CALL calcular_tarifas_calidad('VIP', @total_vip, @activas_vip);
SELECT @total_vip AS Total_VIP, @activas_vip AS Activas_VIP;
-- para ver las tarifas estandar
CALL calcular_tarifas_calidad('Estándar', @total_estandar, @activas_estandar);
SELECT @total_estandar AS Total_Estandar, @activas_estandar AS Activas_Estandar;
```

-- Procedimiento 2: Lista los aprendices del nivel que queramos y nos muestra su objetivo

```
DELIMITER &&

DROP PROCEDURE IF EXISTS listar_aprendices_nivel&&
CREATE PROCEDURE listar_aprendices_nivel(IN nivel_buscar VARCHAR(45))
BEGIN
   SELECT
      objetivo,
      fecha_finalizacion
   FROM
      aprendices
   WHERE
      nivel = nivel_buscar
   ORDER BY
      fecha_finalizacion;
END &&

DELIMITER ;
CALL listar_aprendices_nivel('Basico'');
CALL listar_aprendices_nivel('Intermedio');
CALL listar_aprendices_nivel('Avanzado');
```

TRIGGERS

-- Creacion de tabla donde se van a guardar los datos que actualicemos.

```
CREATE TABLE auditoria_cambios (
   id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
   tabla_afectada VARCHAR(45),
   accion VARCHAR(45),
   dato_antiguo VARCHAR(45),
   dato_nuevo VARCHAR(45),
   fecha_cambio DATETIME
);
```

-- Trigger 1: Cambiar objetivo y registrarlo en la tabla auditoria_cambios

```
DELIMITER &&

DROP TRIGGER IF EXISTS trigger_registrar_cambio_objetivo_aprendices&&
CREATE TRIGGER trigger_registrar_cambio_objetivo_aprendices
AFTER UPDATE
ON aprendices FOR EACH ROW
BEGIN
   DECLARE objetivo_antiguo VARCHAR(45);
   DECLARE objetivo_nuevo VARCHAR(45);

   -- Almacenar los valores antiguo y nuevo en variables
   SET objetivo_antiguo = OLD.objetivo;
   SET objetivo_nuevo = NEW.objetivo;

   -- Si el objetivo ha cambiado, registrar en la tabla de auditoría
   IF objetivo_antiguo != objetivo_nuevo THEN
      INSERT INTO auditoria_cambios (tabla_afectada, accion, dato_antiguo, dato_nuevo,
fecha_cambio)
      VALUES ('aprendices', 'UPDATE', objetivo_antiguo, objetivo_nuevo, NOW());
   END IF;
END &&
DELIMITER ;
```

UPDATE aprendices
SET objetivo = 'Perder Peso'
WHERE id_miembro = 1;