ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
CARRERA DE SOFTWARE

(Bases de Datos**) (**1**)**

Informe de tarea No. 7

1. DATOS GENERALES:

TEMA: Consultas SQL-ejercicios
NOMBRE: [redacted] **CODIGO(S)** [redacted]
GRUPO No.: N/A
FECHA DE ENTREGA: 02/06/2024

2. OBJETIVO:

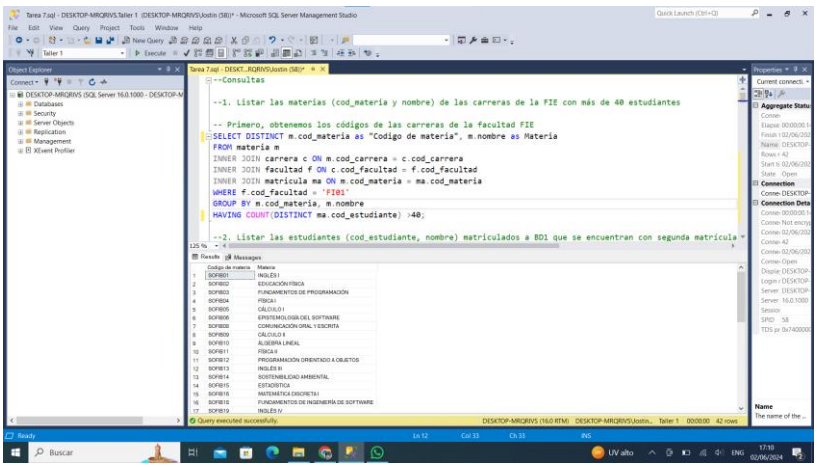
El objetivo general de este trabajo individual es realizar consultas SQL sobre la base de datos del curso "TAREA 7: Consultas SQL-ejercicios" centrándose en aspectos específicos de la gestión académica de la carrera de Software el propósito es analizar la dinámica académica, identificar áreas de mejora y tomar decisiones informadas para mejorar la calidad educativa y el rendimiento académico de los estudiantes.

3. ACTIVIDADES DESARROLLADAS

Durante el desarrollo de esta tarea se llevaron a cabo una serie de actividades centradas en el análisis de datos académicos mediante consultas SQL en la base de datos proporcionada por el curso estas actividades se enfocaron en explorar y comprender diferentes aspectos de la gestión académica de la carrera de Software tales como la matriculación de estudiantes el rendimiento en diferentes materias y la tasa de repitencia. A continuación, se presenta los ejercicios planteados:

Conforme a la base de datos de trabajo de curso, realizar las siguientes consultas:

1. **Listar las materias (cod_materia y nombre) de las carreras de la FIE con más de 40 estudiantes**

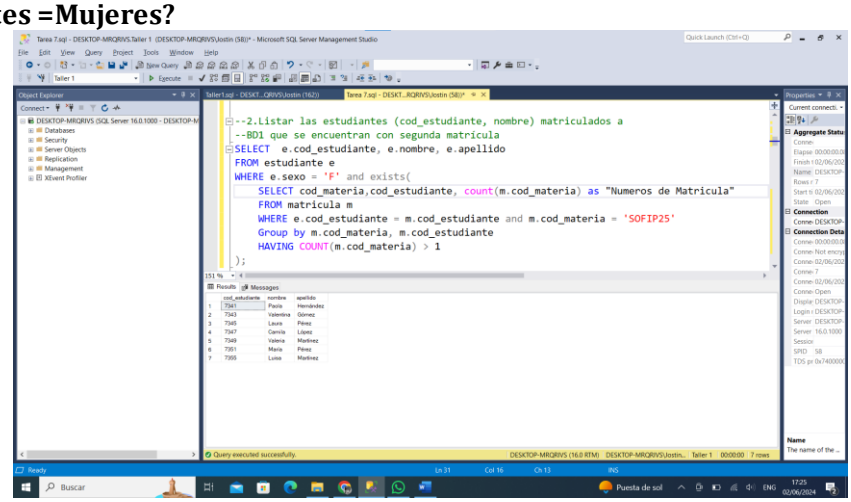


EXPLICACION

Este código SQL tiene como objetivo listar las materias especificadas por su código y nombre pertenecientes a las carreras de la Facultad de Informática y Electrónica (FI01) que cuentan con más de 40 estudiantes matriculados para ello se realiza una unión interna entre las tablas materia carrera, facultad y matricula filtrando los resultados para incluir únicamente aquellas materias de la facultad mencionada posteriormente se agrupan los resultados por código y nombre de la materia aplicando un filtro adicional para seleccionar solo aquellas materias que superen el umbral de 40 estudiantes matriculados.

2. **Listar las estudiantes (cod_estudiante, nombre) matriculados a BD1 que se encuentran con segunda matrícula.**

¿Las estudiantes =Mujeres?

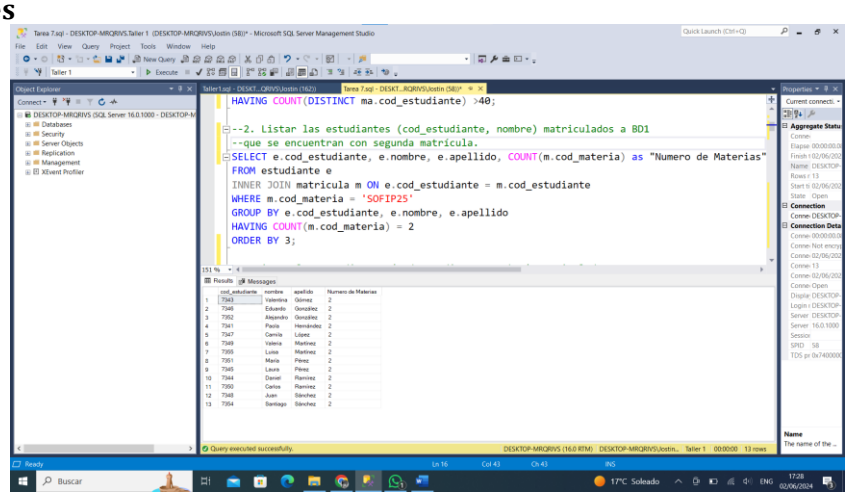


```
--2.Listar las estudiantes (cod_estudiante, nombre) matriculados a
--BD1 que se encuentran con segunda matricula
SELECT  e.cod_estudiante, e.nombre, e.apellido
FROM estudiante e
WHERE e.sexo = 'F' and exists(
    SELECT cod_materia,cod_estudiante, count(m.cod_materia) as "Numeros de Matricula"
    FROM matricula m
    WHERE e.cod_estudiante = m.cod_estudiante and m.cod_materia = 'SOFIP25'
    Group by m.cod_materia, m.cod_estudiante
    HAVING COUNT(m.cod_materia) > 1
);
```

EXPLICACION

Este código SQL busca listar las estudiantes femeninas que están matriculadas en la materia con el código 'SOFIP25' más de una vez la consulta principal selecciona el código del estudiante su nombre y apellido de la tabla estudiante y filtra para incluir solo a las estudiantes de sexo femenino (e.sexo = 'F') luego utiliza una subconsulta en la cláusula EXISTS para verificar si existe al menos un registro en la tabla matricula para cada estudiante en el que el estudiante esté matriculado en la materia 'SOFIP25' más de una vez la subconsulta agrupa los resultados por el código de la materia y el código del estudiante y utiliza HAVING COUNT(m.cod_materia) > 1 para filtrar aquellos estudiantes que han estado matriculados en la materia más de una vez. En resumen, el código identifica y lista a las estudiantes femeninas que están cursando la materia 'SOFIP25' repetidamente.

Los estudiantes



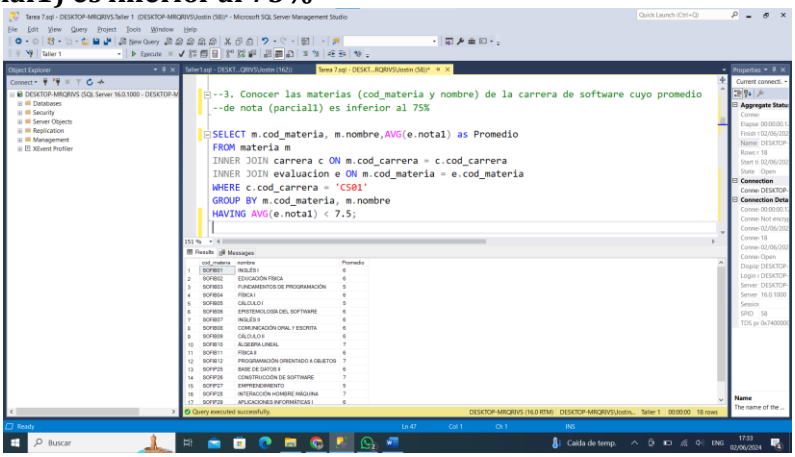
```
HAVING COUNT(DISTINCT ma.cod_estudiante) >40;

--2. Listar las estudiantes (cod_estudiante, nombre) matriculados a BD1
--que se encuentran con segunda matrícula.
SELECT e.cod_estudiante, e.nombre, e.apellido, COUNT(m.cod_materia) as "Numero de Materias"
FROM estudiante e
INNER JOIN matricula m ON e.cod_estudiante = m.cod_estudiante
WHERE m.cod_materia = 'SOFIP25'
GROUP BY e.cod_estudiante, e.nombre, e.apellido
HAVING COUNT(m.cod_materia) = 2
ORDER BY 3;
```

EXPLICACION

Este código SQL tiene como objetivo identificar a los estudiantes que están matriculados en la materia con el código 'SOFIP25' por segunda vez la consulta selecciona el código del estudiante su

nombre y apellido y cuenta el número de veces que cada estudiante ha estado matriculado en esa materia para ello se realiza una unión entre las tablas estudiante y matricula filtrando los registros para la materia específica luego los resultados se agrupan por estudiante y se filtran para incluir únicamente aquellos que han cursado la materia dos veces. Finalmente, los resultados se ordenan alfabéticamente por el apellido del estudiante.

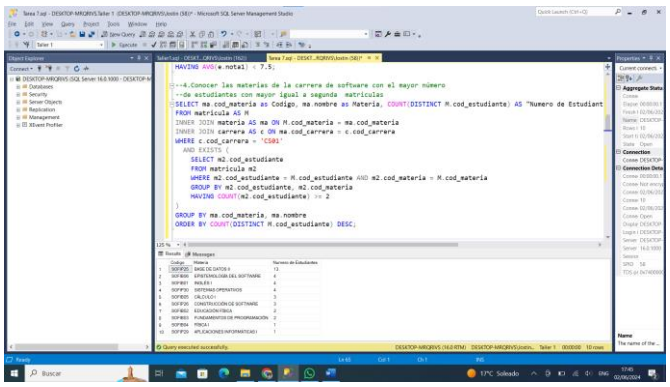
3. **Conocer las materias (cod_materia y nombre) de la carrera de software cuyo promedio de nota (parcial1) es inferior al 75%**



EXPLICACION

Este código SQL tiene como objetivo identificar las materias de la carrera de Software (código de carrera 'CS01') cuyo promedio de la primera nota parcial (nota1) es inferior a 7.5 la consulta selecciona el código y nombre de la materia y calcula el promedio de la primera nota parcial para cada materia se realizan uniones internas entre las tablas materia, carrera y evaluación para vincular las materias con sus respectivas carreras y evaluaciones. Luego, los resultados se agrupan por código y nombre de la materia. Finalmente, la cláusula HAVING filtra los grupos para incluir solo aquellas materias cuyo promedio de la primera nota parcial es menor a 7.5. Esta consulta proporciona una visión clara de las materias que podrían necesitar atención adicional para mejorar el rendimiento académico en la carrera de Software.

4. **Conocer las materias de la carrera de software con el mayor número de estudiantes con mayor igual a segunda matriculas.**

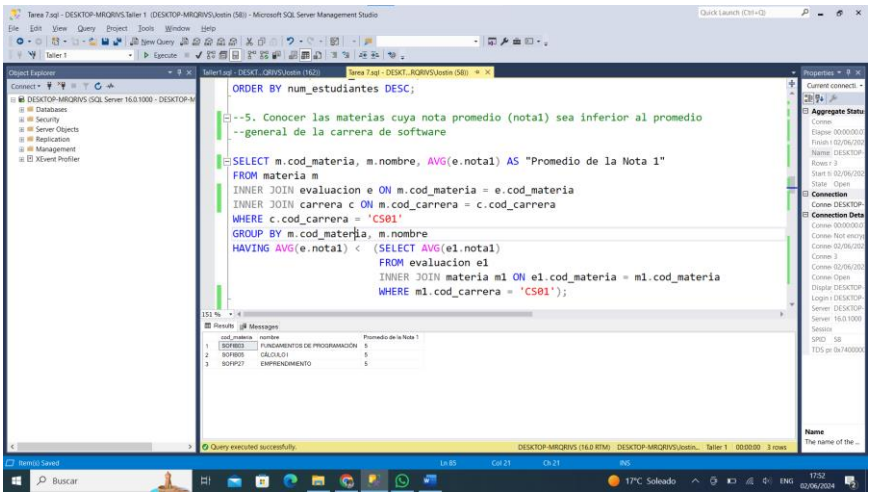


EXPLICACION

Este código SQL busca identificar las materias de la carrera de Software (código de carrera 'CS01') que tienen el mayor número de estudiantes con al menos una segunda matrícula la consulta realiza una unión interna entre las tablas matricula y materia y una unión con la tabla carrera para asociar las materias con su respectiva carrera luego se filtran los resultados para incluir únicamente aquellos estudiantes que tienen al menos dos matrículas en la misma materia utilizando una subconsulta en la cláusula EXISTS posteriormente los resultados se agrupan por código y nombre

de la materia y se ordenan en orden descendente según el número de estudiantes con segundas matrículas este enfoque proporciona información valiosa sobre las materias que pueden presentar mayores desafíos para los estudiantes lo que permite tomar medidas específicas para mejorar su rendimiento académico.

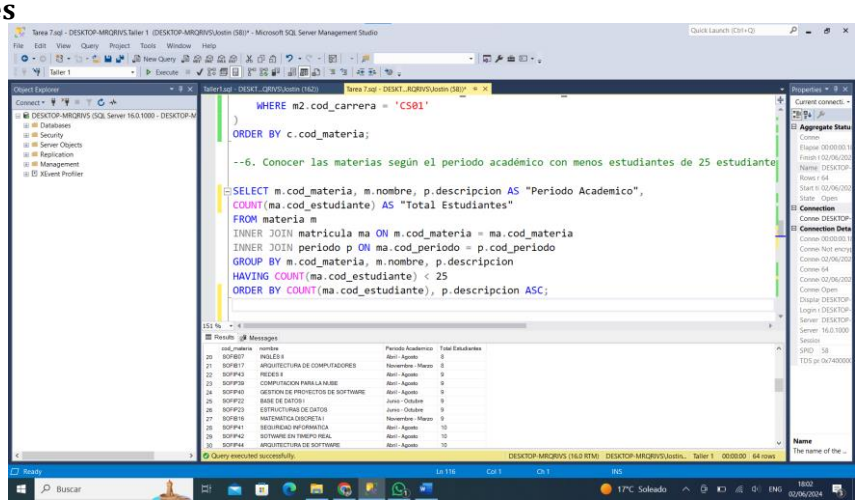
5. **Conocer las materias cuya nota promedio (nota1) sea inferior al promedio general de la carrera de software**



EXPLICACION

Este código SQL busca identificar las materias de la carrera de Software (código de carrera 'CS01') cuyo promedio de la primera nota parcial es inferior al promedio general de la primera nota parcial de todas las materias de la misma carrera la consulta principal realiza una unión interna entre las tablas materia, evaluación y carrera y filtra los resultados para incluir únicamente aquellas materias de la carrera de Software luego se agrupan los resultados por código y nombre de la materia y se aplica una condición en la cláusula HAVING para seleccionar solo aquellas materias cuyo promedio de la primera nota parcial es menor que el promedio general de la primera nota parcial calculado mediante una subconsulta este enfoque permite identificar materias que podrían necesitar una atención adicional para mejorar el rendimiento académico de los estudiantes en la carrera de Software.

6. **Conocer las materias según el periodo académico con menos estudiantes de 25 estudiantes**

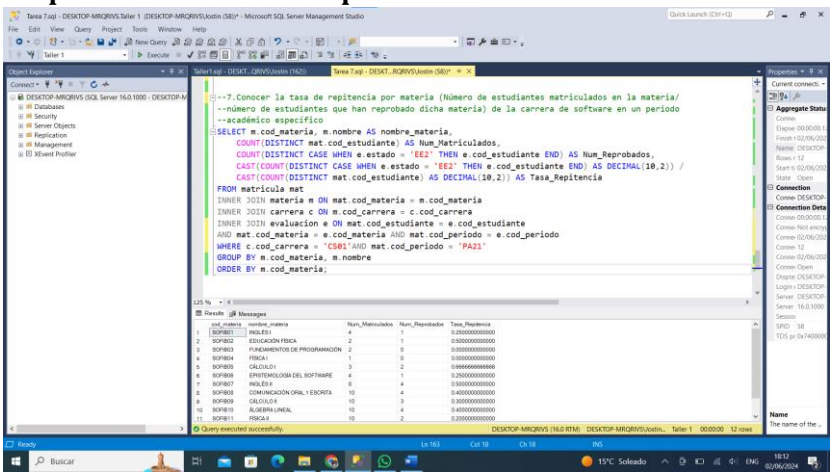


EXPLICACION

Este código SQL busca identificar las materias y los periodos académicos en los que el número total de estudiantes matriculados es inferior a 25 la consulta realiza una unión interna entre las tablas materia, matricula y periodo relacionando las materias con sus respectivas matrículas y períodos

académicos luego los resultados se agrupan por código de materia, nombre de materia y descripción del periodo académico calculando el total de estudiantes matriculados en cada combinación de materia y periodo posteriormente se aplica una condición en la cláusula HAVING para seleccionar solo aquellas combinaciones de materia y periodo en las que el número total de estudiantes es menor que 25. Finalmente, los resultados se ordenan en orden ascendente según el número total de estudiantes matriculados lo que proporciona una visión clara de las materias con baja matriculación en los diferentes periodos académicos.

7. **Conocer la tasa de repitencia por materia (Número de estudiantes matriculados en la materia / número de estudiantes que han reprobado dicha materia) de la carrera de software en un periodo académico específico.**



```
--7.Conocer la tasa de repitencia por materia (Número de estudiantes matriculados en la materia/
--número de estudiantes que han reprobado dicha materia) de la carrera de software en un periodo
--académico específico
SELECT m.cod_materia, m.nombre AS nombre_materia,
    COUNT(DISTINCT mat.cod_estudiante) AS Num_Matriculados,
    COUNT(DISTINCT CASE WHEN e.estado = 'EE2' THEN e.cod_estudiante END) AS Num_Reprobados,
    CAST(COUNT(DISTINCT CASE WHEN e.estado = 'EE2' THEN e.cod_estudiante END) AS DECIMAL(10,2)) /
    CAST(COUNT(DISTINCT mat.cod_estudiante) AS DECIMAL(10,2)) AS Tasa_Repitencia
FROM matricula mat
INNER JOIN materia m ON mat.cod_materia = m.cod_materia
INNER JOIN carrera c ON m.cod_carrera = c.cod_carrera
INNER JOIN evaluacion e ON mat.cod_estudiante = e.cod_estudiante
AND mat.cod_materia = e.cod_materia AND mat.cod_periodo = e.cod_periodo
WHERE c.cod_carrera = 'CS01'AND mat.cod_periodo = 'PA21'
GROUP BY m.cod_materia, m.nombre
ORDER BY m.cod_materia;
```

EXPLICACION

El código SQL proporciona un análisis detallado de la tasa de repitencia por materia en la carrera de Software durante un periodo académico específico al unir las tablas de matrícula, materia, carrera y evaluación se obtiene información integral sobre los estudiantes, las materias, las carreras y las evaluaciones luego mediante la aplicación de filtros se limitan los resultados a las materias de la carrera de Software y al periodo académico designado ('CS01' y 'PA21' respectivamente). El código calcula el número total de estudiantes matriculados en cada materia y el número de estudiantes que han reprobado la misma posteriormente se determina la tasa de repitencia dividiendo el número de estudiantes reprobados entre el total de estudiantes matriculados en la materia presentado como un valor decimal esta consulta proporciona una visión exhaustiva de la tasa de repitencia en la carrera de Software durante el periodo académico específico establecido.

4. RESULTADOS OBTENIDOS

En el contexto de la tarea se llevaron a cabo consultas SQL para analizar datos académicos de la carrera de Software los resultados obtenidos ofrecen información valiosa sobre la matriculación de estudiantes y su rendimiento estos hallazgos son relevantes para identificar tendencias y áreas de mejora en el ámbito académico.

- Listar las materias (cod_materia y nombre) de las carreras de la FIE con más de 40 estudiantes

```
--1. Listar las materias (cod_materia y nombre) de las carreras de la FIE con más de 40
estudiantes
-- Primero, obtenemos los códigos de las carreras de la facultad FIE
SELECT DISTINCT m.cod_materia as "Codigo de materia", m.nombre as Materia
FROM materia m
INNER JOIN carrera c ON m.cod_carrera = c.cod_carrera
INNER JOIN facultad f ON c.cod_facultad = f.cod_facultad
```

```
INNER JOIN matricula ma ON m.cod_materia = ma.cod_materia
WHERE f.cod_facultad = 'FI01'
GROUP BY m.cod_materia, m.nombre
HAVING COUNT(DISTINCT ma.cod_estudiante) >40;
```

- Listar las estudiantes (cod_estudiante, nombre) matriculados a BD1 que se encuentran con segunda matrícula.

```
LAS ESTUDIANTES = MUJERES
--2.Listar las estudiantes (cod_estudiante, nombre) matriculados a
--BD1 que se encuentran con segunda matrícula
SELECT  e.cod_estudiante, e.nombre, e.apellido
FROM estudiante e
WHERE e.sexo = 'F' and exists(
       SELECT cod_materia,cod_estudiante, count(m.cod_materia) as "Numeros de Matricula"
       FROM matricula m
       WHERE e.cod_estudiante = m.cod_estudiante and m.cod_materia = 'SOFIP25'
       Group by m.cod_materia, m.cod_estudiante
       HAVING COUNT(m.cod_materia) > 1
);
```

```
TODOS
--2. Listar las estudiantes (cod_estudiante, nombre) matriculados a BD1
--que se encuentran con segunda matrícula.
SELECT e.cod_estudiante, e.nombre, e.apellido, COUNT(m.cod_materia) as "Numero de
Materias"
FROM estudiante e
INNER JOIN matricula m ON e.cod_estudiante = m.cod_estudiante
WHERE m.cod_materia = 'SOFIP25'
GROUP BY e.cod_estudiante, e.nombre, e.apellido
HAVING COUNT(m.cod_materia) = 2
ORDER BY 3;
```

- Conocer las materias (cod_materia y nombre) de la carrera de software cuyo promedio de nota (parcial1) es inferior al 75%

```
--3. Conocer las materias (cod_materia y nombre) de la carrera de software cuyo promedio
--de nota (parcial1) es inferior al 75%
SELECT m.cod_materia, m.nombre,AVG(e.nota1) as Promedio
FROM materia m
INNER JOIN carrera c ON m.cod_carrera = c.cod_carrera
INNER JOIN evaluacion e ON m.cod_materia = e.cod_materia
WHERE c.cod_carrera = 'CS01'
GROUP BY m.cod_materia, m.nombre
HAVING AVG(e.nota1) < 7.5;
```

- Conocer las materias de la carrera de software con el mayor número de estudiantes con mayor igual a segunda matriculas.

```
--4.Conocer las materias de la carrera de software con el mayor número
--de estudiantes con mayor igual a segunda  matriculas
SELECT ma.cod_materia as Codigo, ma.nombre as Materia, COUNT(DISTINCT M.cod_estudiante)
AS "Numero de Estudiantes"
FROM matricula AS M
INNER JOIN materia AS ma ON M.cod_materia = ma.cod_materia
INNER JOIN carrera AS c ON ma.cod_carrera = c.cod_carrera
WHERE c.cod_carrera = 'CS01'
  AND EXISTS (
    SELECT m2.cod_estudiante
    FROM matricula m2
```

```
    WHERE m2.cod_estudiante = M.cod_estudiante AND m2.cod_materia = M.cod_materia
    GROUP BY m2.cod_estudiante, m2.cod_materia
    HAVING COUNT(m2.cod_estudiante) >= 2
)
GROUP BY ma.cod_materia, ma.nombre
ORDER BY COUNT(DISTINCT M.cod_estudiante) DESC;
```

- Conocer las materias cuya nota promedio (nota1) sea inferior al promedio general de la carrera de software

```
--5. Conocer las materias cuya nota promedio (nota1) sea inferior al promedio
--general de la carrera de software
SELECT m.cod_materia, m.nombre, AVG(e.nota1) AS "Promedio de la Nota 1"
FROM materia m
INNER JOIN evaluacion e ON m.cod_materia = e.cod_materia
INNER JOIN carrera c ON m.cod_carrera = c.cod_carrera
WHERE c.cod_carrera = 'CS01'
GROUP BY m.cod_materia, m.nombre
HAVING AVG(e.nota1) <  (SELECT AVG(e1.nota1)
                        FROM evaluacion e1
                        INNER JOIN materia m1 ON e1.cod_materia = m1.cod_materia
                        WHERE m1.cod_carrera = 'CS01');
```

- Conocer las materias según el periodo académico con menos estudiantes de 25 estudiantes

```
--6. Conocer las materias según el periodo académico con menos estudiantes de 25
estudiantes
SELECT m.cod_materia, m.nombre, p.descripcion AS "Periodo Academico",
COUNT(ma.cod_estudiante) AS "Total Estudiantes"
FROM materia m
INNER JOIN matricula ma ON m.cod_materia = ma.cod_materia
INNER JOIN periodo p ON ma.cod_periodo = p.cod_periodo
GROUP BY m.cod_materia, m.nombre, p.descripcion
HAVING COUNT(ma.cod_estudiante) < 25
ORDER BY COUNT(ma.cod_estudiante), p.descripcion ASC;
```

- Conocer la tasa de repitencia por materia (Número de estudiantes matriculados en la materia / número de estudiantes que han reprobado dicha materia) de la carrera de software en un periodo académico específico.

```
--7.Conocer la tasa de repitencia por materia (Número de estudiantes matriculados en la
materia/
--número de estudiantes que han reprobado dicha materia) de la carrera de software en un
periodo
--académico específico
SELECT m.cod_materia, m.nombre AS nombre_materia,
    COUNT(DISTINCT mat.cod_estudiante) AS Num_Matriculados,
    COUNT(DISTINCT CASE WHEN e.estado = 'EE2' THEN e.cod_estudiante END) AS
Num_Reprobados,
    CAST(COUNT(DISTINCT CASE WHEN e.estado = 'EE2' THEN e.cod_estudiante END) AS
DECIMAL(10,2)) /
    CAST(COUNT(DISTINCT mat.cod_estudiante) AS DECIMAL(10,2)) AS Tasa_Repitencia
FROM matricula mat
INNER JOIN materia m ON mat.cod_materia = m.cod_materia
INNER JOIN carrera c ON m.cod_carrera = c.cod_carrera
INNER JOIN evaluacion e ON mat.cod_estudiante = e.cod_estudiante
AND mat.cod_materia = e.cod_materia AND mat.cod_periodo = e.cod_periodo
WHERE c.cod_carrera = 'CS01'AND mat.cod_periodo = 'PA21'
GROUP BY m.cod_materia, m.nombre
ORDER BY m.cod_materia;
```

5. CONCLUSIONES

El análisis exhaustivo realizado mediante las consultas SQL en la base de datos del curso ha proporcionado una comprensión profunda de la gestión académica de la carrera de Software a través de la identificación de tendencias y patrones significativos estas consultas han destacado áreas específicas que podrían beneficiarse de mejoras en el proceso educativo particularmente al examinar la tasa de repitencia por materia se han identificado posibles áreas de dificultad que requieren atención adicional esta información es crucial para la revisión de enfoques de enseñanza y la implementación de estrategias destinadas a mejorar el rendimiento de los estudiantes en esas áreas específicas. Además, al comparar el promedio de notas de una materia con el promedio general de la carrera se han identificado posibles disparidades en el rendimiento académico que pueden requerir intervenciones específicas para mejorar el proceso de aprendizaje.

El análisis de la matriculación de estudiantes en diferentes períodos académicos ha revelado patrones de demanda de cursos lo que puede guiar la planificación estratégica para la distribución de recursos y la oferta de cursos en momentos específicos del año.
En conclusión, estas consultas SQL no solo han proporcionado una instantánea de la situación académica actual, sino que también han servido como una herramienta valiosa para la identificación de áreas de mejora y la planificación estratégica orientada a la mejora continua del programa educativo en la carrera de Software.

6. RECOMENDACIONES

Tras el análisis exhaustivo de los datos académicos de la carrera de Software mediante consultas SQL en la base de datos del curso se derivan las siguientes recomendaciones:

Revisión de Enfoques de Enseñanza: Se sugiere realizar una revisión detallada de los enfoques de enseñanza utilizados en las materias con altas tasas de repitencia esto puede incluir la implementación de métodos pedagógicos alternativos o el desarrollo de recursos educativos adicionales para abordar las áreas de dificultad identificadas.

Apoyo Académico Personalizado: Considerando las disparidades en el rendimiento académico entre materias se recomienda proporcionar apoyo académico personalizado a los estudiantes que enfrentan dificultades esto puede incluir tutorías individuales sesiones de refuerzo o recursos de aprendizaje adicionales adaptados a las necesidades específicas de cada estudiante.

Estrategias de Intervención Temprana: Es fundamental implementar estrategias de intervención temprana para identificar y apoyar a los estudiantes en riesgo desde el principio del semestre esto podría implicar la realización de evaluaciones diagnósticas al inicio del curso para identificar las necesidades individuales de los estudiantes y proporcionarles el apoyo necesario para tener éxito académico.

Planificación de Oferta de Cursos: Basándose en los patrones de demanda de cursos identificados en diferentes períodos académicos se recomienda realizar una planificación estratégica de la oferta de cursos esto puede ayudar a garantizar una distribución equitativa de recursos y una oferta de cursos que satisfaga las necesidades académicas de los estudiantes en diferentes momentos del año.

Monitoreo Continuo y Evaluación: Finalmente, se sugiere establecer un proceso de monitoreo continuo y evaluación para seguir de cerca el impacto de las intervenciones implementadas y realizar ajustes según sea necesario esto garantizará que las estrategias adoptadas sean efectivas y estén alineadas con los objetivos de mejora académica de la carrera de Software