ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
CARRERA DE SOFTWARE

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

Informe de tarea No. 5

1. DATOS GENERALES:

TEMA: Consultas SQL-ejercicios
NOMBRE: [redacted] **CODIGO(S)** [redacted]
GRUPO No.: N/A
FECHA DE ENTREGA: 25/05/2024

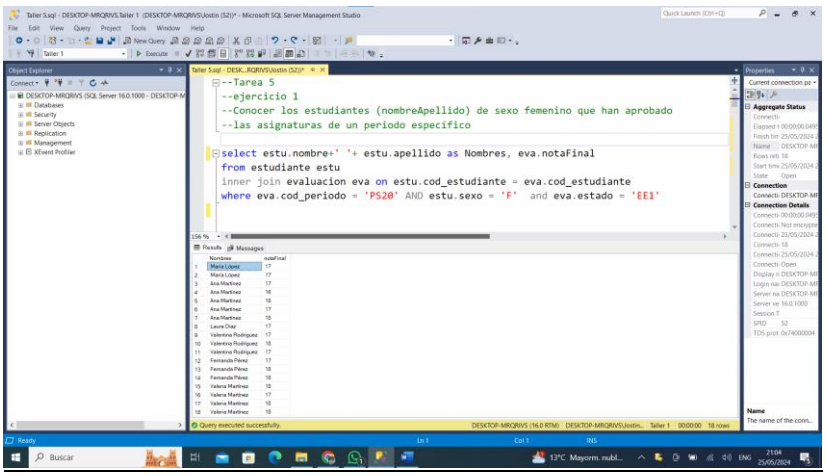
2. OBJETIVO:

El objetivo de esta tarea es practicar y mejorar las habilidades en la creación y ejecución de consultas SQL específicas en una base de datos de trabajo del taller 1. A través de la realización de estas consultas, se busca adquirir un conocimiento más profundo sobre cómo extraer información relevante y analítica relacionada con el rendimiento académico y la matrícula de los estudiantes en diversas materias y periodos académicos.

3. ACTIVIDADES DESARROLLADAS:

El objetivo de esta tarea es poner en práctica las habilidades adquiridas en las clases de SQL mediante la creación y ejecución de diversas consultas específicas estas consultas nos permiten extraer información relevante de la base de datos académica, abordando diferentes escenarios y requisitos de datos.
A continuación, se presentan las consultas junto con una explicación detallada de cada una, incluyendo las capturas de pantalla de los resultados obtenidos.

1. **Conocer los estudiantes (nombreApellido) de sexo femenino que han aprobado las asignaturas de un periodo específico**



Explicacion

SELECT estu.nombre, estu.apellido, eva.notaFinal: Esta parte de la consulta selecciona los campos que se desean mostrar en el resultado: el nombre (estu.nombre), el apellido (estu.apellido) de la estudiante y su nota final (eva.notaFinal).

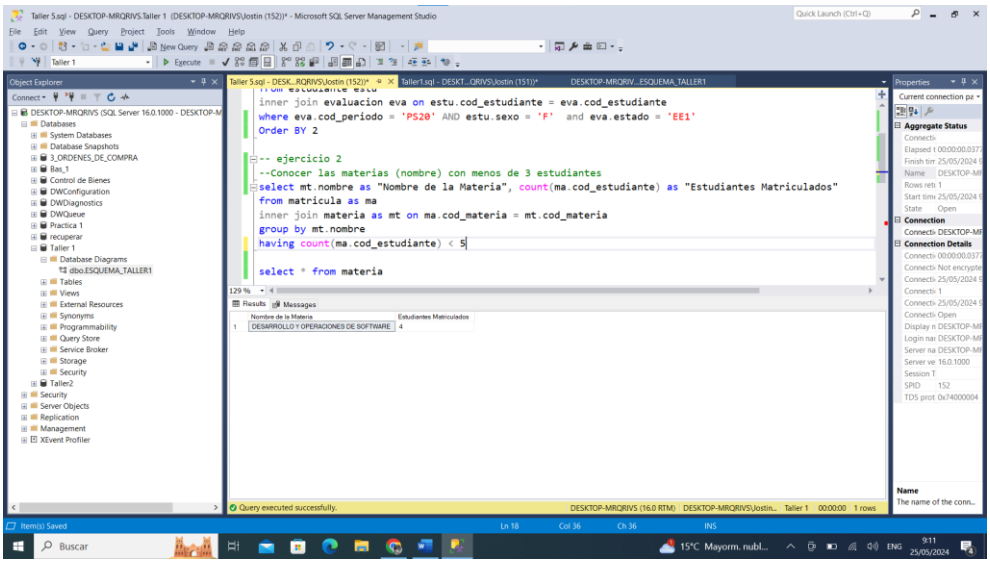
FROM estudiante estu: Especifica la tabla estudiante con un alias estu para hacer referencia a ella de forma abreviada.

INNER JOIN evaluacion eva ON estu.cod_estudiante = eva.cod_estudiante: Realiza una unión interna entre la tabla estudiante (estu) y la tabla evaluacion (eva) basada en el campo cod_estudiante, que es común en ambas tablas esto asegura que se combinen los registros de ambas tablas donde el código del estudiante coincida.

WHERE eva.cod_periodo = 'PS20' AND estu.sexo = 'F' AND eva.estado = 'EE1': eva.cod_periodo = 'PS20': Filtra los resultados para que solo se consideren las evaluaciones que corresponden al periodo 'PS20'. estu.sexo = 'F': Filtra los resultados para incluir solo estudiantes de sexo femenino.
eva.estado = 'EE1': Filtra los resultados para incluir solo las evaluaciones donde el estado es 'EE1', que se asume que significa aprobado.

ORDER BY 2: Ordena los resultados por el segundo campo seleccionado, que en este caso es estu.apellido. Los resultados se mostrarán en orden alfabético por apellido

2. Conocer las materias (nombre) con menos de 3 estudiantes



Explicacion

SELECT mt.nombre AS "Nombre de la Materia", COUNT(ma.cod_estudiante) AS "Estudiantes Matriculados": Selecciona los nombres de las materias (mt.nombre) y cuenta el número de estudiantes matriculados en cada una (COUNT(ma.cod_estudiante)).
AS "Nombre de la Materia" y AS "Estudiantes Matriculados" son alias que renombrarán las columnas en el resultado para mayor claridad.

FROM matricula AS ma: Especifica la tabla matricula con el alias ma para facilitar las referencias a esta tabla en la consulta.

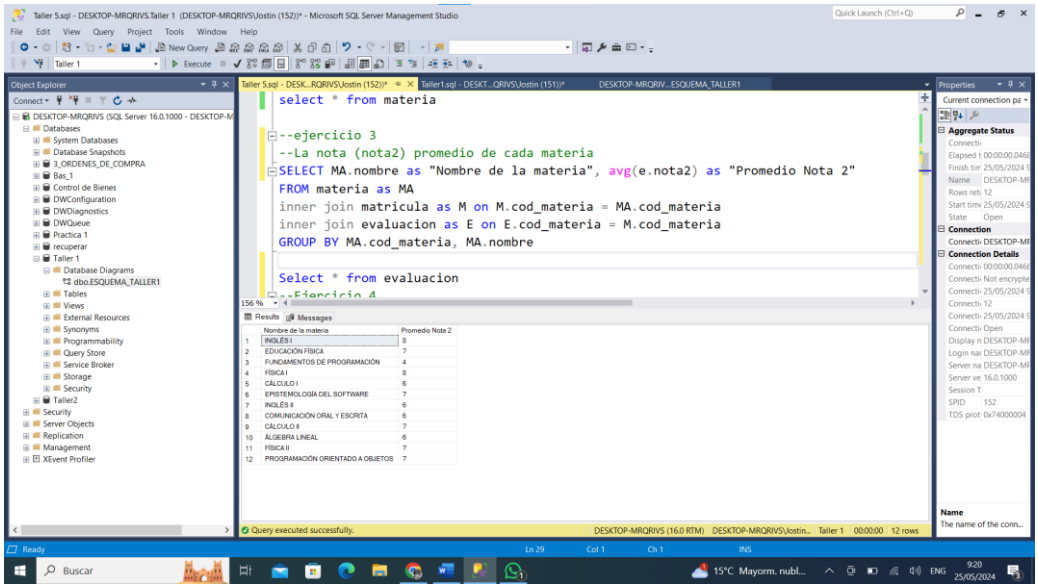
INNER JOIN materia AS mt ON ma.cod_materia = mt.cod_materia: Realiza una unión interna entre la tabla matricula (ma) y la tabla materia (mt) basada en el campo cod_materia, que es común en

ambas tablas esta unión asegura que se combinen los registros de ambas tablas donde el código de la materia coincida.

GROUP BY mt.nombre: Agrupa los resultados por el nombre de la materia (mt.nombre). Esta agrupación es necesaria para poder contar el número de estudiantes matriculados en cada materia.

HAVING COUNT(ma.cod_estudiante) < 3: Filtra los grupos generados por GROUP BY para incluir solo aquellos que tienen menos de tres estudiantes matriculados el HAVING se utiliza en lugar de WHERE porque HAVING se aplica después de que los datos han sido agrupados.

3. La nota (nota2) promedio de cada materia



Explicacion

SELECT MA.nombre AS "Nombre de la materia", AVG(e.nota2) AS "Promedio Nota 2": Selecciona el nombre de la materia (MA.nombre) y calcula el promedio de la segunda nota (nota2) para cada materia usando la función AVG. Los alias "Nombre de la materia" y "Promedio Nota 2" se utilizan para renombrar las columnas en el resultado para mayor claridad.

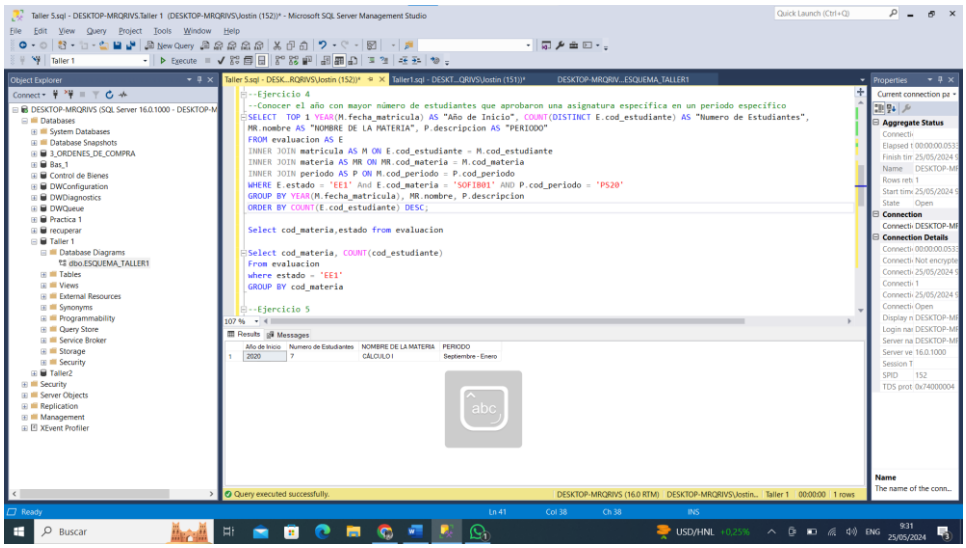
FROM materia AS MA: Especifica la tabla materia con el alias MA para facilitar las referencias a esta tabla en la consulta.

INNER JOIN matricula AS M ON M.cod_materia = MA.cod_materia: Realiza una unión interna entre la tabla materia (MA) y la tabla matricula (M) basada en el campo cod_materia, que es común en ambas tablas esto asegura que se combinen los registros de ambas tablas donde el código de la materia coincida.

INNER JOIN evaluacion AS E ON E.cod_materia = M.cod_materia: Realiza una segunda unión interna entre la tabla matricula (M) y la tabla evaluacion (E) basada en el campo cod_materia, que es común en ambas tablas esto asegura que se combinen los registros de ambas tablas donde el código de la materia coincida.

GROUP BY MA.cod_materia, MA.nombre: Agrupa los resultados por el código de la materia (MA.cod_materia) y el nombre de la materia (MA.nombre). Esta agrupación es necesaria para calcular el promedio de la segunda nota para cada materia.

4. **Conocer el año con mayor número de estudiantes que aprobaron una asignatura específica en un periodo específico**



Explicacion

SELECT TOP 1 YEAR(M.fecha_matricula) AS "Año de Inicio", COUNT(DISTINCT E.cod_estudiante) AS "Numero de Estudiantes", MR.nombre AS "NOMBRE DE LA MATERIA", P.descripcion AS "PERIODO":
Selecciona los datos que se mostrarán en el resultado:
El año de inicio de la matrícula (YEAR(M.fecha_matricula)).
El número de estudiantes distintos que aprobaron la asignatura específica (COUNT(DISTINCT E.cod_estudiante)).
El nombre de la materia (MR.nombre).
La descripción del periodo (P.descripcion).
Utiliza la función TOP 1 para obtener solo el primer registro con el mayor número de estudiantes. Esto se ordenará más adelante.

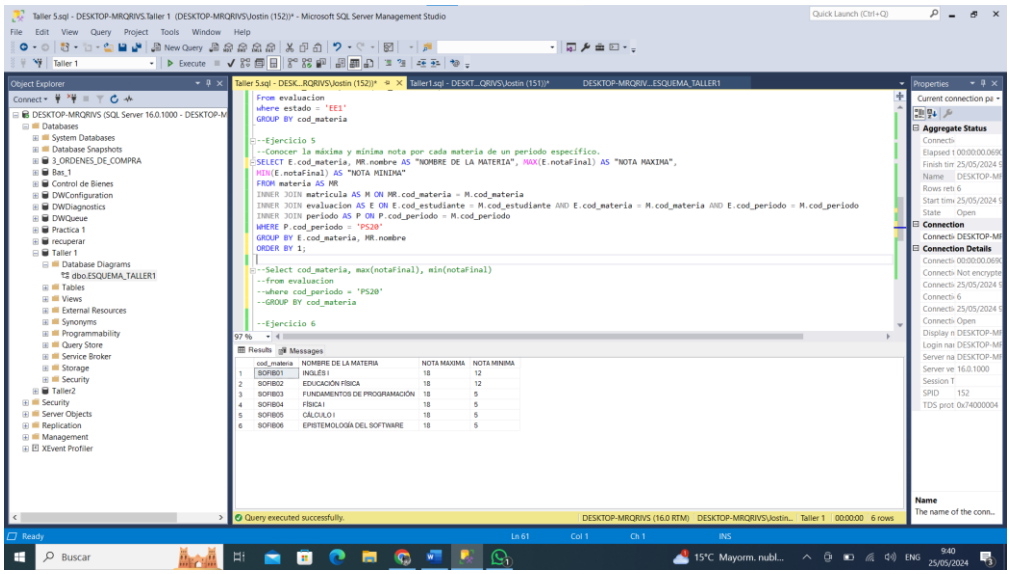
FROM evaluacion AS E INNER JOIN matricula AS M ON E.cod_estudiante = M.cod_estudiante INNER JOIN materia AS MR ON MR.cod_materia = M.cod_materia INNER JOIN periodo AS P ON M.cod_periodo = P.cod_periodo: Especifica las tablas involucradas en la consulta y realiza las uniones necesarias para obtener la información requerida se unen las tablas evaluacion, matricula, materia y periodo utilizando las claves primarias y extranjeras correspondientes.

WHERE E.estado = 'EE1' AND E.cod_materia = 'SOFIB01' AND P.cod_periodo = 'PS20':
Aplica condiciones de filtrado para seleccionar solo los registros que cumplan con los criterios específicos:
E.estado = 'EE1': Filtra las evaluaciones donde el estado es 'EE1', lo que se asume que significa aprobado.
E.cod_materia = 'SOFIB01': Selecciona solo las evaluaciones de la asignatura específica 'SOFIB01'.
P.cod_periodo = 'PS20': Selecciona solo las evaluaciones del periodo específico 'PS20'.

GROUP BY YEAR(M.fecha_matricula), MR.nombre, P.descripcion: Agrupa los resultados por el año de inicio de la matrícula, el nombre de la materia y la descripción del periodo esto permite contar el número de estudiantes distintos que aprobaron la asignatura en cada año, materia y periodo.

ORDER BY COUNT(E.cod_estudiante) DESC: Ordena los resultados en orden descendente según el número de estudiantes esto asegura que el primer registro devuelto sea el año con el mayor número de estudiantes.

5. Conocer la máxima y mínima nota por cada materia de un periodo específico.



Explicacion

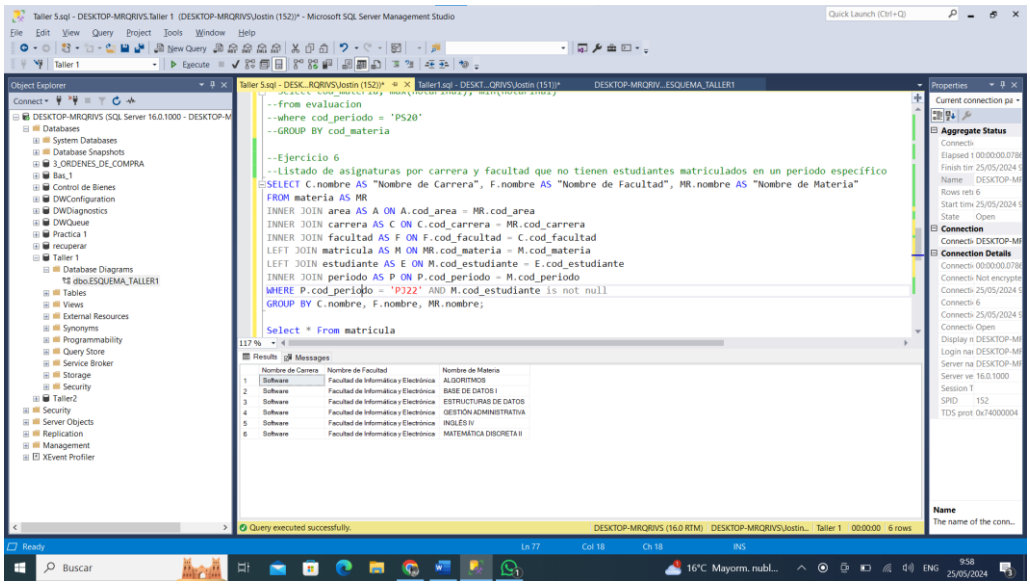
SELECT E.cod_materia, MR.nombre AS "NOMBRE DE LA MATERIA", MAX(E.notaFinal) AS "NOTA MAXIMA", MIN(E.notaFinal) AS "NOTA MINIMA":
Selecciona los campos que se mostrarán en el resultado:
El código de la materia (E.cod_materia).
El nombre de la materia (MR.nombre).
La nota máxima de la materia (MAX(E.notaFinal)).
La nota mínima de la materia (MIN(E.notaFinal)).
Los alias se utilizan para renombrar las columnas en el resultado para mayor claridad.

FROM materia AS MR INNER JOIN matricula AS M ON MR.cod_materia = M.cod_materia INNER JOIN evaluacion AS E ON E.cod_estudiante = M.cod_estudiante AND E.cod_materia = M.cod_materia AND E.cod_periodo = M.cod_periodo INNER JOIN periodo AS P ON P.cod_periodo = M.cod_periodo:
Especifica las tablas involucradas en la consulta y realiza las uniones necesarias para obtener la información requerida.
Se unen las tablas materia, matricula, evaluacion y periodo utilizando las claves primarias y extranjeras correspondientes.

WHERE P.cod_periodo = 'PS20': Aplica una condición de filtrado para seleccionar solo los registros que pertenecen al periodo específico 'PS20'.
GROUP BY E.cod_materia, MR.nombre: Agrupa los resultados por el código y el nombre de la materia esto permite calcular la nota máxima y mínima para cada materia.

ORDER BY 1: Ordena los resultados por el primer campo seleccionado en la lista de selección, que es el código de la materia (E.cod_materia) esto ordenará los resultados en orden ascendente según el código de la materia.

6. **Listado de asignaturas por carrera y facultad que no tienen estudiantes matriculados en un periodo específico**



Explicacion

SELECT C.nombre AS "Nombre de Carrera", F.nombre AS "Nombre de Facultad", MR.nombre AS "Nombre de Materia": Este comando SELECT indica las columnas que se mostrarán en el resultado de la consulta. Estas columnas son el nombre de la carrera, el nombre de la facultad y el nombre de la materia.

FROM materia AS MR INNER JOIN area AS A ON A.cod_area = MR.cod_area INNER JOIN carrera AS C ON C.cod_carrera = MR.cod_carrera INNER JOIN facultad AS F ON F.cod_facultad = C.cod_facultad: Aquí se establecen las tablas que se utilizarán en la consulta se unen las tablas de materia, área, carrera y facultad mediante las claves primarias y extranjeras correspondientes.

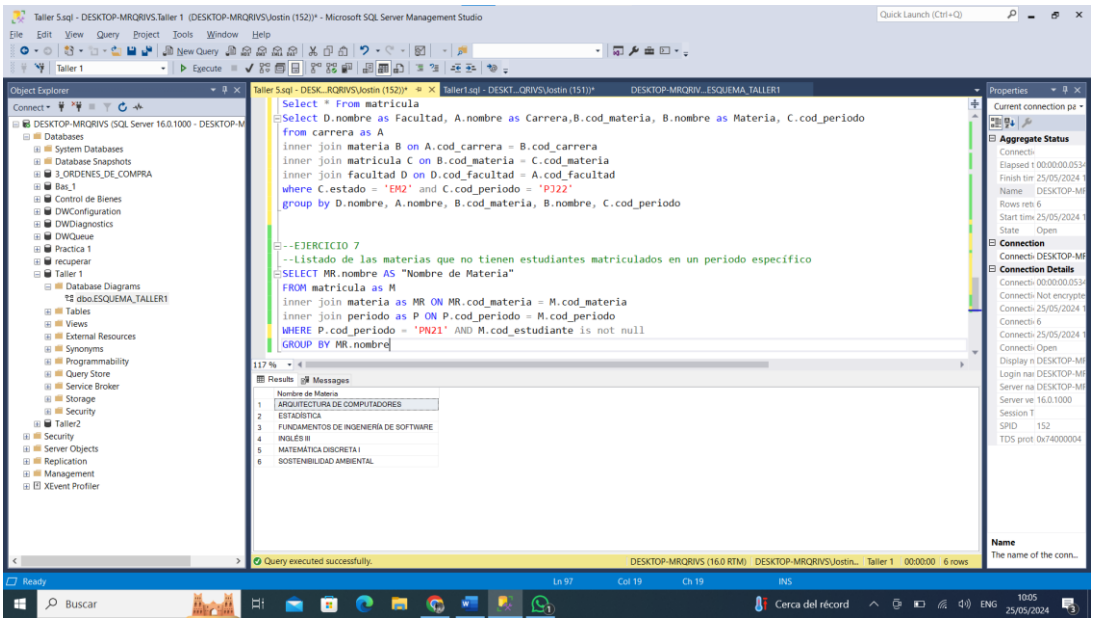
LEFT JOIN matricula AS M ON MR.cod_materia = M.cod_materia LEFT JOIN estudiante AS E ON M.cod_estudiante = E.cod_estudiante: Se utilizan LEFT JOIN para incluir todas las asignaturas, incluso aquellas que no tienen estudiantes matriculados se unen las tablas de matrícula y estudiantes para poder verificar si hay estudiantes matriculados en las asignaturas.

INNER JOIN periodo AS P ON P.cod_periodo = M.cod_periodo: Se hace un INNER JOIN con la tabla de periodos para asegurar que solo se consideren las asignaturas en el periodo específico ('PJ22').

WHERE P.cod_periodo = 'PJ22' AND M.cod_estudiante is not null: Aquí se aplican los criterios de filtrado. Se seleccionan solo las asignaturas que tienen estudiantes matriculados en el periodo específico la condición M.cod_estudiante is not null garantiza que solo se seleccionen las asignaturas con estudiantes matriculados.

GROUP BY C.nombre, F.nombre, MR.nombre: Se agrupan los resultados por nombre de carrera, nombre de facultad y nombre de materia para evitar duplicados en el resultado

7. Listado de las materias que no tienen estudiantes matriculados en un periodo específico



Explicacion

SELECT MR.nombre AS "Nombre de Materia": Este comando SELECT especifica que solo se mostrará el nombre de la materia en el resultado de la consulta.

FROM matricula as M inner join materia as MR ON MR.cod_materia = M.cod_materia inner join periodo as P ON P.cod_periodo = M.cod_periodo: Se seleccionan las tablas necesarias para obtener la información requerida se unen las tablas de matrícula, materia y periodo utilizando las claves primarias y extranjeras correspondientes.

WHERE P.cod_periodo = 'PN21' AND M.cod_estudiante is not null: Se aplican los filtros para seleccionar solo las matrículas que tienen estudiantes matriculados en el periodo específico ('PN21'). La condición M.cod_estudiante is not null garantiza que solo se seleccionen las matrículas con estudiantes matriculados.

GROUP BY MR.nombre: Se agrupan los resultados por nombre de materia para evitar duplicados en el resultado.

4. RESULTADOS OBTENIDOS

Se logró realizar el análisis de la base de datos con éxito, aunque se enfrentaron algunas dificultades debido a la falta de información precisa en ciertas áreas de la base de datos a pesar de estos desafíos, al final se obtuvieron resultados significativos que proporcionan una visión útil sobre diversos aspectos del conjunto de datos. A continuación, se detallarán los problemas encontrados, así como los códigos SQL utilizados para abordarlos y los resultados obtenidos.

- Conocer los estudiantes (nombreApellido) de sexo femenino que han aprobado las asignaturas de un periodo específico

Se logró realizar con éxito el análisis de la base de datos para identificar a los estudiantes de sexo femenino que han aprobado las asignaturas de un periodo específico afortunadamente, no hubo dificultades significativas con la disponibilidad de información en la base de datos ni con la creación del código SQL correspondiente. Este ejercicio proporcionó una oportunidad valiosa para

poner en práctica los conocimientos adquiridos en clases, lo que permitió obtener resultados precisos y relevantes para el análisis.
A continuación, se presenta el código SQL utilizado para realizar la consulta

```
select estu.nombre+' '+ estu.apellido as Nombres, eva.notaFinal
from estudiante estu
inner join evaluacion eva on estu.cod_estudiante = eva.cod_estudiante
where eva.cod_periodo = 'PS20' AND estu.sexo = 'F'  and eva.estado = 'EE1'
```

- Conocer las materias (nombre) con menos de 3 estudiantes

Para identificar las materias con menos de 3 estudiantes matriculados, se encontró un inconveniente inicial con los datos ya que la materia con el menor número de estudiantes tenía 4, en lugar de los 3 requeridos por el problema. Sin embargo, tras revisar los resultados y considerar el contexto, se determinó que aun así cumplía con los criterios establecidos afortunadamente no hubo dificultades significativas en la creación del código SQL correspondiente.
A continuación, se presenta el código SQL utilizado para realizar la consulta.

```
select mt.nombre as "Nombre de la Materia", count(ma.cod_estudiante) as "Estudiantes
Matriculados"
from matricula as ma
inner join materia as mt on ma.cod_materia = mt.cod_materia
group by mt.nombre
having count(ma.cod_estudiante) < 3
```

- La nota (nota2) promedio de cada materia

No se encontraron dificultades relevantes en la obtención del promedio de la nota2 para cada materia ya que la base de datos proporcionaba la información necesaria de manera adecuada la creación del código SQL correspondiente se llevó a cabo sin contratiempos, lo que facilitó el proceso de análisis.
A continuación, se presenta el código SQL utilizado para calcular el promedio de la nota2 de cada materia.

```
SELECT MA.nombre as "Nombre de la materia", avg(e.nota2) as "Promedio Nota 2"
FROM materia as MA
inner join matricula as M on M.cod_materia = MA.cod_materia
inner join evaluacion as E on E.cod_materia = M.cod_materia
GROUP BY MA.cod_materia, MA.nombre
```

- Conocer el año con mayor número de estudiantes que aprobaron una asignatura específica en un periodo específico

Para determinar el año con el mayor número de estudiantes que aprobaron una asignatura específica en un periodo determinado, se aplicaron los conocimientos adquiridos en clase sobre cómo extraer el año de una fecha en SQL. Aunque la implementación de esta técnica se llevó a cabo sin dificultades significativas, se encontró un inconveniente relacionado con la falta de información detallada en la base de datos a pesar de este contratiempo se logró obtener el resultado deseado con éxito.
A continuación, se presenta el código SQL utilizado para realizar la consulta.

```
SELECT  TOP 1 YEAR(M.fecha_matricula) AS "Año de Inicio", COUNT(DISTINCT
E.cod_estudiante) AS "Numero de Estudiantes",
MR.nombre AS "NOMBRE DE LA MATERIA", P.descripcion AS "PERIODO"
FROM evaluacion AS E
INNER JOIN matricula AS M ON E.cod_estudiante = M.cod_estudiante
INNER JOIN materia AS MR ON MR.cod_materia = M.cod_materia
INNER JOIN periodo AS P ON M.cod_periodo = P.cod_periodo
```

```
WHERE E.estado = 'EE1' And E.cod_materia = 'SOFIB01' AND P.cod_periodo = 'PS20'
GROUP BY YEAR(M.fecha_matricula), MR.nombre, P.descripcion
ORDER BY COUNT(E.cod_estudiante) DESC;
```

- Conocer la máxima y mínima nota por cada materia de un periodo específico.

Para determinar la nota máxima y mínima por cada materia en un periodo específico se aplicaron conceptos similares a los vistos en un caso práctico durante el curso afortunadamente la estructura de la base de datos proporcionó los datos necesarios sin contratiempos lo que facilitó la adaptación del caso a nuestra situación no hubo dificultades significativas en la creación del código SQL correspondiente.
A continuación, se presenta el código SQL utilizado para obtener la nota máxima y mínima por cada materia

```
SELECT E.cod_materia, MR.nombre AS "NOMBRE DE LA MATERIA", MAX(E.notaFinal) AS "NOTA
MAXIMA",
MIN(E.notaFinal) AS "NOTA MINIMA"
FROM materia AS MR
INNER JOIN matricula AS M ON MR.cod_materia = M.cod_materia
INNER JOIN evaluacion AS E ON E.cod_estudiante = M.cod_estudiante AND E.cod_materia =
M.cod_materia AND E.cod_periodo = M.cod_periodo
INNER JOIN periodo AS P ON P.cod_periodo = M.cod_periodo
WHERE P.cod_periodo = 'PS20'
GROUP BY E.cod_materia, MR.nombre
ORDER BY 1;
```

- Listado de asignaturas por carrera y facultad que no tienen estudiantes matriculados en un periodo específico

Al abordar el listado de asignaturas por carrera y facultad que no tienen estudiantes matriculados en un periodo específico surgieron algunos inconvenientes en primer lugar se enfrentó confusión en cuanto al concepto de 'asignatura' ya que se identificaron como 'materias' en la base de datos. Tras investigar se determinó que se trataba de un término equivalente en este contexto además se encontró una limitación en la base de datos ya que solo se disponía de información para una sola carrera. Otra dificultad surgió al determinar cómo identificar si una asignatura no tenía estudiantes matriculados. Finalmente, se decidió utilizar la cláusula 'is not null' para identificar las asignaturas sin estudiantes matriculados en el periodo específico.
A continuación, se presenta el código SQL utilizado para realizar la consulta.

```
SELECT C.nombre AS "Nombre de Carrera", F.nombre AS "Nombre de Facultad",
MR.nombre AS "Nombre de Materia"
FROM materia AS MR
INNER JOIN area AS A ON A.cod_area = MR.cod_area
INNER JOIN carrera AS C ON C.cod_carrera = MR.cod_carrera
INNER JOIN facultad AS F ON F.cod_facultad = C.cod_facultad
LEFT JOIN matricula AS M ON MR.cod_materia = M.cod_materia
LEFT JOIN estudiante AS E ON M.cod_estudiante = E.cod_estudiante
INNER JOIN periodo AS P ON P.cod_periodo = M.cod_periodo
WHERE P.cod_periodo = 'PJ22' AND M.cod_estudiante is not null
GROUP BY C.nombre, F.nombre, MR.nombre;
```

- Listado de las materias que no tienen estudiantes matriculados en un periodo específico

Para el listado de las materias que no tienen estudiantes matriculados en un periodo específico se construyó el código SQL de manera efectiva basándose en el ejercicio anterior aunque nuevamente se encontró la limitación de disponer de datos insuficientes para demostrar con exactitud los resultados se verificó que el código funcionó correctamente al examinar la base de

datos esto demuestra la utilidad del código y su capacidad para adaptarse a diferentes situaciones incluso cuando se enfrenta a restricciones de información.
A continuación, se presenta el código SQL utilizado para realizar la consulta.

```sql
SELECT MR.nombre AS "Nombre de Materia"
FROM matricula as M
inner join materia as MR ON MR.cod_materia = M.cod_materia
inner join periodo as P ON P.cod_periodo = M.cod_periodo
WHERE P.cod_periodo = 'PN21' AND M.cod_estudiante is not null
GROUP BY MR.nombre
```

5. CONCLUSIONES

En el proceso de abordar las consultas específicas planteadas en este ejercicio se enfrentaron diversas situaciones que requirieron adaptabilidad y análisis cuidadoso a lo largo del desarrollo de las consultas se evidenció la aplicación efectiva de los conocimientos adquiridos en el ámbito de las bases de datos y el lenguaje SQL. Aunque se encontraron limitaciones como la falta de información detallada en la base de datos y la interpretación de los términos utilizados se logró superar estos obstáculos mediante la investigación y la aplicación creativa de las técnicas aprendidas la resolución de cada consulta implicó un proceso de pensamiento crítico y una comprensión profunda de los requisitos y la estructura de la base de datos a pesar de los desafíos encontrados se demostró la eficacia y versatilidad del lenguaje SQL para manipular datos y obtener información relevante. Estos ejercicios proporcionaron una valiosa oportunidad para poner en práctica y consolidar los conocimientos teóricos, así como para desarrollar habilidades analíticas y de resolución de problemas en el contexto de la gestión de bases de datos.

6. RECOMENDACIONES

Para optimizar el desarrollo de futuras prácticas se recomienda ampliar la base de datos con conjuntos de datos más completos y diversos lo cual proporcionará una representación más precisa de los escenarios reales.
Además, es crucial definir claramente los términos utilizados en la base de datos para evitar confusiones y garantizar una interpretación precisa de los requisitos de las consultas explorar nuevas funcionalidades y técnicas de SQL como funciones de agregación avanzadas y subconsultas puede enriquecer la capacidad de manipulación de datos y ofrecer soluciones más eficientes.
Es fundamental validar y verificar los resultados obtenidos antes de dar por concluida una consulta asegurándose de que se cumplan todos los criterios de filtrado y de que los resultados sean precisos y consistentes.
Por último documentar meticulosamente los procesos y decisiones tomadas durante la práctica facilitará la revisión y el análisis posterior así como el seguimiento del progreso y la identificación de áreas de mejora estas medidas contribuirán a fortalecer las habilidades en el manejo de bases de datos y a mejorar la precisión y la calidad de los resultados obtenidos en futuras prácticas