CUADERNO PL/SQL

Este cuaderno de PL/SQL tiene como objetivo reforzar la programación PL/SQL con una serie de ejercicios diseñados para ser aplicados en un entorno de desarrollo Oracle. Además de mejorar las habilidades de programación, se busca potenciar la memoria y el desarrollo lógico en la creación y manipulación de datos en bases de datos estructuradas, así como en la programación y el desarrollo dentro del entorno PL/SQL. Este material también proporcionará una oportunidad para consolidar los conocimientos en la gestión de bases de datos en distintos niveles, abarcando desde los conceptos fundamentales hasta niveles más avanzados. El cuaderno debe mantenerse al alcance y completarse a lo largo del curso, el cual se extiende durante 16 semanas según lo establecido en el plan de estudio.

El cuaderno comprende un total de 100 ejercicios diseñados para abordar diversas áreas del aprendizaje. Es esencial que cada estudiante lo lleve de manera individual y que se realice una revisión durante cada clase para verificar el progreso en los ejercicios de la semana. El seguimiento y registro del cuaderno serán elementos fundamentales que contribuirán a la evaluación académica del estudiante. Es importante destacar la relevancia de este ejercicio académico como una herramienta integral para fortalecer las competencias adquiridas en clase y en todo el proceso de aprendizaje relacionado con la temática del curso.

METODOLOGÍA:

- Cada estudiante debe crear un repositorio de GitHub con el nombre de Bases de Datos.
- Se debe crear el link en el Dashboard en la sección de PL/SQL
- Se debe crear un código SQL por cada Ejercicio, realizando la explicación de cómo funciona el código y que resultados se generaron.
- En caso de que una sentencia no genere ningún resultado, explicar la razón del comportamiento de esa sentencia

ESTRUCTURA DE LA BASE DE DATOS:

```
-- Tabla de clientes

CREATE TABLE ClientePLSQL (
```

```
    id_cliente NUMBER PRIMARY KEY,
    nombre VARCHAR2(50),
    direccion VARCHAR2(100),
    telefono VARCHAR2(15)
);

-- Tabla de autos

CREATE TABLE AutoPLSQL (
    id_auto NUMBER PRIMARY KEY,
    marca VARCHAR2(50),
    modelo VARCHAR2(50),
    ano NUMBER
);

-- Tabla de alquileres

CREATE TABLE AlquilerPLSQL (
    id_alquiler NUMBER PRIMARY KEY,
    id_cliente NUMBER,
    id_auto NUMBER,
    fecha_inicio DATE,
    fecha_fin DATE,
    id_reserva NUMBER,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente),
    FOREIGN KEY (id_auto) REFERENCES Auto(id_auto),
    FOREIGN KEY (id_reserva) REFERENCES Reserva(id_reserva)
);

-- Tabla de sucursales

CREATE TABLE SucursalPLSQL (
    id_sucursal NUMBER PRIMARY KEY,
    nombre VARCHAR2(50),
    ciudad VARCHAR2(50),
    pais VARCHAR2(50)
);

-- Tabla de reservas

CREATE TABLE ReservaPLSQL (
    id_reserva NUMBER PRIMARY KEY,
    id_cliente NUMBER,
    id_sucursal NUMBER,
    fecha_reserva DATE,
    FOREIGN KEY (id_cliente) REFERENCES Cliente(id_cliente),
    FOREIGN KEY (id_sucursal) REFERENCES Sucursal(id_sucursal)
);
```

- Cliente: Almacena información sobre los clientes, como su nombre, dirección y número de teléfono.
- Auto: Almacena información sobre los autos, como su marca, modelo y año.
- Alquiler: Almacena información sobre los alquileres, como la fecha de inicio, la fecha de finalización y el auto alquilado.
- Sucursal: Almacena información sobre las sucursales, como su nombre, ciudad y país.
- Reserva: Almacena información sobre las reservas, como la fecha de la reserva y la sucursal en la que se realizó la reserva.

EJERCICIOS PRIMER CICLO (1-30):

1. **Consultas Básicas:**
   - Mostrar todos los clientes en la tabla "Cliente"
     Para mostrar todos los clientes en la tabla Cliente se utiliza el siguiente còdigo: utilizar la instrucción SELECT.
     SELECT * FROM ClientePLSQL;
     Funcionamiento: La sentencia SQL utilizada es una consulta de selección que recupera todos los registros de la tabla "ClientePLSQL." Esta operación es fundamental en la exploración y obtención de datos para análisis.
     Resultado: El resultado de la consulta es una tabla de datos que muestra todos los clientes almacenados en la tabla "ClientPLSQL." Cada fila representa el registro de un cliente, y cada columna corresponde a un atributo específico del cliente, como: ID_ cliente, nombre, direccion, y telefono Esta información se presenta en un formato tabular.

   - Mostrar todos los autos en la tabla "Auto".
     Para mostrar todos los datos de la tabla Auto se utiliza el siguiente código: “SELECT * FROM AutoPLSQL;”
     Funcionamiento: La sentencia SQL utilizada es una consulta de selección que recupera todos los registros de la tabla "AutoPLSQL." Esta operación es fundamental en la exploración y obtención de datos para análisis.
     Resultado: El resultado de la consulta es una tabla de datos que muestra todos los autos almacenados en la tabla "AutoPLSQL." Cada fila representa el registro de un auto, y cada columna corresponde a un atributo específico del auto, como: id_auto, marca, modelo , año.

   - Mostrar todos los alquileres en la tabla "Alquiler".

Para mostrar todos los clientes en la tabla Alquiler se utiliza el siguiente còdigo: utilizar la instrucción SELECT.
SELECT * FROM AlquilerPLSQL;

Funcionamiento: La sentencia SQL utilizada es una consulta de selección que recupera todos los registros de la tabla "AlquilerPLSQL." Esta operación es fundamental en la exploración y obtención de datos para análisis.
Resultado: El resultado de la consulta es una tabla de datos que muestra todos los clientes almacenados en la tabla "AlquilerPLSQL." Cada fila representa el registro de un cliente, y cada columna corresponde a un atributo específico del cliente, como: id_alquiler, id_cliente, id_auto, fecha_inicio DATE, fecha_fin DATE, id_reserva NUMBER.

- Mostrar todas las sucursales en la tabla "Sucursal"
  Se utiliza el siguiente código para llamar la tabla Sucursal: “SELECT * FROM SucursalPLSQL;”
  Funcionamiento: Se utiliza una operación de consulta que recupera datos de una tabla en la base de datos. En este caso, se seleccionan todos los registros de la tabla "SucursalPLSQL," que representan todas las sucursales registradas.
  Resultado: La ejecución de esta consulta es una lista de todas las sucursales almacenadas en la tabla "SucursalPLSQL." Cada fila en el resultado representa una sucursal con sus atributos correspondientes, como el ID de la sucursal, el nombre de la sucursal, la ciudad y el país en columnas separadas. Esta información se presenta en forma de tabla facilitando la visualización y el análisis de los datos de las sucursales.

- Mostrar todas las reservas en la tabla "Reserva"
  Para mostrar todos los clientes en la tabla Reserva se utiliza el siguiente còdigo: utilizar la instrucción SELECT.
  SELECT * FROM ReservaPLSQL;

  Funcionamiento: La sentencia SQL utilizada es una consulta de selección que recupera todos los registros de la tabla "ReservaPLSQL." Esta operación es fundamental en la exploración y obtención de datos para análisis.
  Resultado: El resultado de la consulta es una tabla de datos que muestra todos los clientes almacenados en la tabla "ReservaPLSQL." Cada fila representa el registro de un cliente, y cada columna corresponde a un atributo específico del cliente, como: id_reserva, id_cliente, id_sucursal NUMBER.

  Filtros y Ordenamiento:
- Mostrar los clientes que se llaman "Juan"
  Se utiliza la siguiente consulta: “SELECT * FROM ClientePLSQL WHERE nombre = 'Juan';”
  Funcionamiento: El código utiliza una operación de selección con una cláusula WHERE para recuperar los registros de la tabla "ClientePLSQL" en los que el valor de la

columna "nombre" sea igual a "Juan.". “WHERE” permite filtrar los registros basados en una condición específica, en este caso, se busca a los clientes cuyo nombre sea "Juan".

Resultado: El resultado de la ejecución de esta consulta será una lista de clientes cuyo filtro es el nombre "Juan." Cada fila en el resultado representa un cliente con todos los atributos correspondientes, como el ID del cliente, nombre, dirección y número de teléfono, en columnas separadas. Esta consulta devuelve solo los clientes que cumplen con la condición de tener el nombre "Juan" en la base de datos.

- Mostrar los autos de marca "Toyota".

  Para mostrar todos los clientes en la tabla Reserva se utiliza el siguiente còdigo: utilizar la instrucción SELECT.

  SELECT * FROM AutoPLSQL WHERE marca = 'Toyota';

  Funcionamiento: La sentencia SQL utilizada es una consulta de selección que recupera un registro definido en la tabla "AutoPLSQL." Esta operación es fundamental en la exploración y obtención de datos para análisis.

  Resultado: El resultado de la consulta es ubicar una marca de carro que se encuentra entre el

- Mostrar los alquileres que ocurrieron después de una fecha específica.

  Se utiliza el siguiente código: “SELECT * FROM AlquilerPLSQL WHERE fecha_inicio > TO_DATE('FECHA_ESPECÍFICA', 'YYYY-MM-DD');”

  Funcionamiento: El código utiliza la operación SELECT con una cláusula WHERE para recuperar los registros de la tabla "AlquilerPLSQL" en los que la columna "fecha_inicio" sea posterior (“>”) a una fecha específica. La fecha específica se debe proporcionar en el formato 'YYYY-MM-DD' dentro de la función TO_DATE, y la cláusula WHERE filtra los alquileres que cumplen con esta condición.

  Resultado: El resultado de la consulta es la lista de alquileres que ocurrieron después de la fecha específica proporcionada. El resultado muestra el alquiler con sus atributos correspondientes, como el ID del alquiler, ID del cliente, ID del auto, fecha de inicio y fecha de finalización, en columnas separadas cumpliendo con la condición de haber comenzado después de la fecha especificada.

- Mostrar las sucursales ubicadas en "Madrid".

  Se utuiliza primero SELECT * FROM SucursalPLSQLZ; para identificar los campos a ingresar, despues de esto se realiza la inserción de los datos con este codigo:

  INSERT INTO SucursalPLSQLZ (id_sucursal, nombre, ciudad, pais)
  VALUES (1,'Gran Feria', 'Madrid', 'España');

  Funcionamiento: La sentencia SQL utilizada se INSERT la cual se usa para ingresar los datos de forma manual a la tabla .

  Resultado: El resultado de la insercion en la tabla SucursalPLSQLZ en los campos seleccionados de la tabla: id_sucursal, nombre, ciudad, pais

- Mostrar las reservas realizadas por un cliente específico.

Para realizar esta consulta se utiliza el siguiente código: “SELECT * FROM ReservaPLSQL WHERE id_cliente = 'ID_DEL_CLIENTE_ESPECÍFICO';”

Funcionamiento:Este código utiliza SELECT con una cláusula WHERE para recuperar los registros de la tabla "ReservaPLSQL" en los que el valor de la columna "id_cliente" sea igual al ID del cliente específico que se desea consultar. Se debe reemplazar 'ID_DEL_CLIENTE_ESPECÍFICO' con el ID del cliente deseado.

Resultado: El resultado de la consulta es una lista de reservas realizadas por el cliente específico cuyo ID coincida con el valor proporcionado. Cada fila en el resultado mostrará una reserva con los atributos: ID de la reserva, ID del cliente, ID de la sucursal y la fecha de la reserva, la información se mostrará en columnas separadas y asociadas al cliente específico.

2. Join y Relaciones:

- Mostrar los alquileres con los nombres de los clientes y las marcas de los autos.esto se usa para mostrar los alquileres con los nombres de los clientes y las marcas de los autos en una base de datos, necesitarás realizar una consulta que involucre las tablas relacionadas.

  Funcionamiento: La sentencia SQL utilizada SELECT en los campos a.id_alquiler, c.id_cliente, au.marca_auto y junto con JOIN combina las tablas AlquilerPLSQLZ, ReservaPLSQLZ y SucursalPLSQLZ;El resultado es una tabla que contiene las columnas id_alquiler, id_reserva, y id_auto.

  Resultado: El resultado del JOIN en la tabla SucursalPLSQLZ en los campos seleccionados de la tabla: ID_AUTO, ID_CLIENTE,

- Mostrar los clientes que han realizado reservas en una sucursal específica.

  Se utiliza el siguiente código: SELECT c.ID_CLIENTE, c.NOMBRE_CLIENTE, r.ID_RESERVA, r.FECHA_RESERVA

  FROM Cliente c JOIN ReservaPLSQLZ r ON c.ID_CLIENTE = r.ID_CLIENTE JOIN SucursalPLSQLZ s ON r.ID_SUCURSAL = s.ID_SUCURSAL WHERE s.ID_SUCURSAL = tu_id_sucursal;

  Funcionamiento: El código utiliza una operación JOIN entre las tablas SucursalPLSQLZ (alias "c") y "ReservaPLSQL" (alias "r") basada en la igualdad de la columna "id_cliente" papara combinar las filas de las tablas Cliente, ReservaPLSQLZ y SucursalPLSQLZ en función de las claves foráneas (ID_CLIENTE e ID_SUCURSAL). La cláusula WHERE filtra los resultados para mostrar solo aquellos que corresponden a la sucursal específica.
- Mostrar los autos que han sido alquilados junto con los nombres de los clientes.

  Se utiliza el siguiente codigo SELECT a.ID_ALQUILER, c.ID_CLIENTE,
  c.NOMBRE_CLIENTE, au.ID_AUTO, au.MARCA_AUTO,
  au.MODELO, au.ANO, a.FECHA_INICIO, a.FECHA_FIN

  FROM AlquilerPLSQLZ a JOIN Cliente c ON a.ID_CLIENTE = c.ID_CLIENTE JOIN AutoPLSQLZ au ON a.ID_AUTO = au.ID_AUTO.

  Funcionamiento: El código utilizEsto te dará una lista de autos que han sido alquilados, junto con los nombres de los clientes que realizaron esos alquileres nEsto te dará una lista de autos que han sido alquilados, junto con los nombres de los clientes que realizaron esos alquileres.eración JOIN entre las tablas AlquilerPLSQLZ

(alias "c") y "AutoPLSQLZ" (alias "au") basada en la igualdad de la columna "id_cliente" papara combinar las filas de las tablas Cliente, AlquilerPLSQLZ y AutoPLSQLZ en función de las claves foráneas (ID_CLIENTE e ID_SUCURSAL),Esto te dará una lista de autos que han sido alquilados, junto con los nombres de los clientes que realizaron esos alquileres.

- Mostrar los detalles de las reservas con los nombres de los clientes y las ciudades de las sucursales.
  A continuación se presenta el código utilizado:
  "SELECT r.id_reserva, c.nombre AS nombre_cliente, s.ciudad AS ciudad_sucursal
  FROM ReservaPLSQL r
  JOIN ClientePLSQL c ON r.id_cliente = c.id_cliente
  JOIN SucursalPLSQL s ON r.id_sucursal = s.id_sucursal;"
  Funcionamiento: Se utilizan operaciones JOIN para combinar las tablas "ReservaPLSQL" (alias "r"), "ClientePLSQL" (alias "c") y "SucursalPLSQL" (alias "s") basado en la igualdad de la columna "id_cliente" con el fin de relacionar las reservas con los clientes. Luego, se realiza un segundo JOIN entre "ReservaPLSQL" y "SucursalPLSQL" basado en la igualdad de la columna "id_sucursal" para relacionar las reservas con las sucursales.
  Resultado: La consulta incluye una lista de detalles de reservas que mostrará: el ID de la reserva, el nombre del cliente que realizó la reserva y la ciudad de la sucursal en la que se realizó la reserva. Cada fila en el resultado representa una reserva con esta información combinada de las tres tablas. La utilidad de esta operación es facilitar la visualización de los detalles de las reservas junto con los nombres de los clientes y las ciudades de las sucursales asociadas.

- Mostrar los clientes que no han realizado ninguna reserva.
  Se usa el siguiente codigo para mostrar los clientes que no han realizado ninguna reserva, puedes utilizar una consulta que involucre las tablas Cliente y ReservaPLSQLZ, utilizando la cláusula LEFT JOIN y verificando si hay valores nulos en las columnas de la tabla ReservaPLSQLZ, y se usa el siguiente codigo: SELECT c.ID_CLIENTE, c.NOMBRE_CLIENTE FROM Cliente c LEFT JOIN ReservaPLSQLZ r ON c.ID_CLIENTE = r.ID_CLIENTE WHERE r.ID_RESERVA IS NULL;usando la condición c.ID_CLIENTE = r.ID_CLIENTE, filtra las filas donde no hay coincidencias en la tabla ReservaPLSQLZ, es decir, donde el cliente no ha realizado ninguna reserva.Esto te dará una lista de clientes que no han realizado ninguna reserva.

3. Agregación y Agrupamiento:

- Contar cuántos autos hay de cada marca en la tabla "Auto".
  A continuación se presenta el código:
  "SELECT marca, COUNT(*) AS cantidad_de_autos
  FROM AutoPLSQL
  GROUP BY marca;"

Funcionamiento: Este código utiliza una consulta de selección con la cláusula GROUP BY para agrupar los registros de la tabla "AutoPLSQL" según la columna "marca" de los autos, luego, se utiliza la función de agregación COUNT para contar cuántos autos hay en cada grupo (marca).
Resultado: El resultado de la consulta mostrará una lista de las marcas de autos junto con el conteo de autos de esa marca presentes en la tabla "AutoPLSQL.". Esta información proporciona un resumen de cuántos autos existen para cada marca en la tabla.

- Calcular la duración promedio de los alquileres.Se usa AVG para calcular el promedio de la diferencia entre las fechas de inicio y fin en la tabla AlquilerPLSQLZ, con este codigo: SELECT AVG(FECHA_FIN - FECHA_INICIO) AS DURACION_PROMEDIO FROM AlquilerPLSQLZ;
  Funcionamiento; se calcular el promedio de la diferencia entre las columnas FECHA_FIN y FECHA_INICIO en la tabla AlquilerPLSQLZ. y se resta las fechas para saber el tiempo.

- Mostrar el número total de reservas realizadas en cada sucursal.
  A continuación se presenta el código ejecutado:
  SELECT s.nombre AS nombre_sucursal, COUNT(*) AS total_de_reservas
  FROM SucursalPLSQL s
  JOIN ReservaPLSQL r ON s.id_sucursal = r.id_sucursal
  GROUP BY s.nombre;
  Funcionamiento: Se realiza una operación JOIN entre las tablas "SucursalPLSQL" (alias "s") y "ReservaPLSQL" (alias "r") utilizando como llave de unión la columna "id_sucursal" para relacionar las sucursales con sus respectivas reservas. Luego, se utiliza la función de agregación COUNT junto con la cláusula GROUP BY para contar cuántas reservas se han realizado en cada sucursal.
  Resultado: Se obtiene una tabla de las sucursales junto con el número total de reservas realizadas en cada sucursal proporcionando una visión resumida del número de reservas por sucursal.

- Encontrar el cliente que ha realizado la mayor cantidad de alquileres.
  Se utiliza este codigo para encontrar el cliente que ha realizado la mayor cantidad de alquileres, SELECT c.ID_CLIENTE, c.NOMBRE_CLIENTE,
  COUNT(a.ID_ALQUILER) AS CANTIDAD_ALQUILERES FROM
  Cliente c JOIN AlquilerPLSQLZ a ON c.ID_CLIENTE = a.ID_CLIENTE
  GROUP BY c.ID_CLIENTE, c.NOMBRE_CLIENTE ORDER BY
  CANTIDAD_ALQUILERES DESC FETCH FIRST 1 ROW ONLY; se utiliza la función COUNT para contar el número de alquileres para cada cliente, y luego utiliza GROUP BY para agrupar los resultados por cliente. Con este codigo ORDER BY CANTIDAD_ALQUILERES DESC ordena los resultados en orden descendente según la cantidad de alquileres, y FETCH FIRST 1 ROW ONLY y se seccionales la primera linea (el cliente con la mayor cantidad de alquileres).
- Calcular el promedio de años de los autos en la tabla "Auto".

Se utiliza el siguiente código para dar solución al enunciado: “SELECT AVG(ano) AS promedio_de_anos
FROM AutoPLSQL;”
Funcionamiento: Consiste en utilizar la función de agregación AVG para calcular el promedio de los valores en la columna "ano" de la tabla "AutoPLSQL." La función AVG suma todos los valores de la columna "ano" y luego divide el resultado por el número total de filas en la tabla, lo que proporciona el promedio de años de los autos.
Resultado: Muestra el promedio de años de los autos en la tabla "AutoPLSQL."

4. **Subconsultas**:

- Mostrar los clientes que han realizado al menos una reserva. para realizar esta sentencia se requiere mostrar los clientes que han realizado al menos una reserva, se hace con una consulta que utilice la cláusula EXISTS o IN.
  SELECT c.ID_CLIENTE, c.NOMBRE_CLIENTE FROM Cliente c WHERE
  EXISTS (SELECT 1 FROM ReservaPLSQLZ r WHERE r.ID_CLIENTE = c.ID_CLIENTE);, se selecciona clientes de la tabla Cliente para los cuales existe al menos una reserva en la tabla ReservaPLSQLZ que está asociada a ese cliente. La subconsulta dentro de EXISTS verifica esta condición.

- Mostrar los autos que no han sido alquilados aún.
  Se utiliza el siguiente código:
  “SELECT * FROM AutoPLSQL
  WHERE id_auto NOT IN (SELECT id_auto FROM AlquilerPLSQL);”
  Funcionamiento: Este código realiza una subconsulta en la que se seleccionan los ID de los autos que han sido alquilados en la tabla "AlquilerPLSQL." Luego, se utiliza SELECT para recuperar todos los registros de la tabla "AutoPLSQL" donde la variable ID del auto no esté presente en la lista de ID de autos alquilados.
  Resultado:Se visualiza la tabla de autos que no han sido alquilados, es decir, que no tienen registros en la tabla "AlquilerPLSQL." Cada fila en el resultado representa un auto con sus atributos correspondientes y ayuda a identificar los autos disponibles que aún no han sido alquilados.

- Encontrar los clientes que han alquilado el mismo auto más de una vez.
  Se utiliza el código para encontrar los clientes que han alquilado el mismo auto más de una vez, se usa una consulta para cruzar las tablas de Cliente y AlquilerPLSQLZ, y utilice la cláusula GROUP BY y HAVING, segun este codigo SELECT c.ID_CLIENTE, c.NOMBRE_CLIENTE, a.ID_AUTO,
  COUNT(a.ID_ALQUILER) AS CANTIDAD_ALQUILERES FROM
  Cliente c JOIN AlquilerPLSQLZ a ON c.ID_CLIENTE = a.ID_CLIENTE
  GROUP BY c.ID_CLIENTE, c.NOMBRE_CLIENTE, a.ID_AUTO, se usa
  HAVING COUNT(a.ID_ALQUILER) > 1; y Utilizamos GROUP BY para agrupar los resultados por cliente y auto.
  Funcionamiento: La función COUNT cuenta el número de alquileres para cada combinación de cliente y auto y la funcion HAVING COUNT(a.ID_ALQUILER) > 1 filtra

los resultados para incluir solo aquellas combinaciones donde el cliente ha alquilado el mismo auto más de una vez.
Y nos genera una lista de clientes que han alquilado el mismo auto más de una vez.

- Mostrar los clientes que han realizado alquileres en la misma ciudad en la que viven.
  SELECT c.*
  FROM ClientePLSQL c
  JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente
  JOIN SucursalPLSQL s ON a.id_sucursal = s.id_sucursal
  WHERE c.ciudad = s.ciudad;
  Funcionamiento: El código realiza una operación JOIN entre las tablas "ClientePLSQL" (alias "c"), "AlquilerPLSQL" (alias "a") y "SucursalPLSQL" (alias "s"). Se relaciona a los clientes con los alquileres y luego a los alquileres con las sucursales en las que se han realizado los alquileres. La condición en la cláusula WHERE compara la ciudad de residencia del cliente con la ciudad de la sucursal en la que se ha realizado el alquiler y en caso de que se cumpla la condición se muestra el valor.
  Resultado: El resultado de la consulta mostrará una lista de clientes que han realizado alquileres en la misma ciudad en la que viven.

- Encontrar los autos que han sido alquilados en la misma sucursal donde se realizó una reserva.
  Para este tema usaremos una consulta una consulta que involucre las tablas ReservaPLSQLZ, AlquilerPLSQLZ, AutoPLSQLZ y SucursalPLSQLZ, donde se combine este codigo: SELECT r.ID_RESERVA, a.ID_ALQUILER, au.ID_AUTO, au.MARCA_AUTO, au.MODELO, au.ANO FROM ReservaPLSQLZ r JOIN SucursalPLSQLZ sr ON r.ID_SUCURSAL = sr.ID_SUCURSAL JOIN AlquilerPLSQLZ a ON r.ID_RESERVA = a.ID_RESERVA JOIN AutoPLSQLZ au ON a.ID_AUTO = au.ID_AUTO
  WHERE a.ID_SUCURSAL = r.ID_SUCURSAL; donde se combina las tablas ReservaPLSQLZ y SucursalPLSQLZ para obtener la información de la sucursal donde se realizó la reserva, se combiina la tabla AlquilerPLSQLZ con la tabla AutoPLSQLZ para obtener información sobre el auto alquilado. Asi como e iguala este campoa.ID_SUCURSAL = r.ID_SUCURSAL donde el auto se haya alquilado en la misma sucursal donde se realizó la reserva.

5. Actualizaciones y Eliminaciones:

- Actualizar la dirección de un cliente específico.
  Se utiliza el siguiente código:
  “UPDATE ClientePLSQL
  SET direccion = 'NUEVA_DIRECCIÓN'
  WHERE id_cliente = 'ID_DEL_CLIENTE_ESPECÍFICO';”
  Funcionamiento: Se utiliza la función de actualización UPDATE para modificar los datos en la tabla "ClientePLSQL.", se establece el valor de la columna "direccion" con la nueva dirección que deseas asignar al cliente específico. La cláusula WHERE se utiliza para especificar cuál cliente se va a actualizar, identificado por su ID de cliente ("id_cliente").

Resultado: La ejecución de esta consulta actualiza la dirección del cliente determinado con el ID CLIENTE digitado, en la tabla Cliente.

- Eliminar un auto de la tabla "Auto". Para eliminar un registro de la tabla "Auto" en una base de datos, se usa la instrucción DELETE, adicionalmente con un Where, esto es irreversible si hay restricciones de clave foránea en otras tablas que hacen referencia a este registro, estos permite eliminar esos registros primero o gestionar las restricciones de integridad referencial.
- Marcar una reserva como completada actualizando la fecha de fin.
  Se utiliza el siguiente código como solución:
  " UPDATE ReservaPLSQL
  SET fecha_fin = SYSDATE
  WHERE id_reserva = 'ID_DE_LA_RESERVA';
  Funcionamiento: Este código utiliza la sentencia UPDATE para modificar los datos en la tabla "ReservaPLSQL.", se establece la columna "fecha_fin" con la función SYSDATE, que representa la fecha y hora del sistema en ese momento. La cláusula WHERE se utiliza para especificar cuál reserva se va a marcar como completada, identificada por su ID de reserva ("id_reserva").
  Resultado: La ejecución de esta consulta marca la reserva como completada al actualizar la fecha de fin con la fecha y hora del sistema una vez se confirme el ID de la reserva.

- Eliminar todas las reservas realizadas por un cliente específico. Se eliminar todas las reservas realizadas por un cliente específico en la tabla "ReservaPLSQLZ", puedes utilizar la instrucción DELETE con una cláusula WHERE, DELETE FROM ReservaPLSQLZ WHERE ID_CLIENTE = tu_id_cliente;esta operación eliminará permanentemente todas las filas de la tabla "ReservaPLSQLZ" que cumplan con la condición especificada. Se debe tener en cuenta las restricciones de clave foránea en otras tablas que hacen referencia a estas reservas, la eliminación puede estar sujeta a esas restricciones.

- Actualizar el año de un auto en la tabla "Auto".
  Código:
  "UPDATE AutoPLSQL
  SET ano = 'NUEVO_ANO'
  WHERE id_auto = 'ID_DEL_AUTO_ESPECÍFICO';"
  Funcionamiento: Se UPDATE para modificar los datos en la tabla "AutoPLSQL." Se establece el valor de la columna "ano" con el nuevo año que se desea asignar al auto específico y la cláusula WHERE se utiliza para especificar cuál auto se va a actualizar, identificado por su ID de auto ("id_auto").
  Resultado: La ejecución de este código actualizará el año del auto específico en la tabla.

EJERCICIOS SEGUNDO CICLO (31-80):

- SELECT * FROM ClientePLSQLZ;

Funcionamiento: Esta consulta selecciona y muestra todas las columnas de la tabla llamada "ClientePLSQL". Al usar el asterisco * en lugar de enumerar columnas específicas, la consulta devuelve todas las columnas disponibles en la tabla. Y proporcionará todos los registros y todas las columnas de la tabla "ClientePLSQLZ". Resultado sera un cliente en la base de datos y contendrá los valores de todas las columnas asociadas con ese cliente.

- SELECT * FROM AutoPLSQL;
  Funcionamiento: Esta consulta utiliza la función SELECT para traer todos los registros de la tabla "AutoPLSQL." El asterisco (*) en la sentencia representa todas las columnas de la tabla.
  Resultado: El resultado de la consulta es una lista completa de todos los autos en la tabla "AutoPLSQL." con sus atributos específicos: ID del auto, marca, modelo y año. Esta consulta se utiliza para obtener una vista general de todos los autos disponibles en la base de datos y sus detalles.

- SELECT * FROM AlquilerPLSQL;
  Funcionamiento: selecciona y muestra todas las columnas de la tabla llamada "AlquilerPLSQL". Al utilizar el asterisco * en lugar de enumerar columnas específicas, la consulta devuelve todos los registros y todas las columnas disponibles en la tabla.Proporcionará todos los registros y todas las columnas de la tabla "AlquilerPLSQZL". Cada fila del resultado representará un alquiler en la base de datos y contendrá los valores de todas las columnas asociadas con ese alquiler.

- SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente;
  Funcionamiento: En la consulta se utilizó una operación JOIN para combinar datos de dos tablas: "ClientePLSQL" (alias "c") y "AlquilerPLSQL" (alias "a"). La relación se basa en el dato de la columna "id_cliente" que está en ambas tablas. La consulta selecciona el nombre del cliente (columna "nombre" en la tabla "ClientePLSQL") y los detalles del auto alquilado, que incluyen la marca (columna "marca" en la tabla "AlquilerPLSQL") y el modelo (columna "modelo" en la tabla "AlquilerPLSQL").
  Resultado:El resultado de la consulta detalla una lista de filas donde cada fila representa el nombre del cliente y la información del auto, como la marca y el modelo, puede ser útil para el seguimiento de los clientes y los autos alquilados en la base de datos.

- SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto;
  Funcionamiento: realiza una operación de JOIN entre las tablas "AutoPLSQL" y "AlquilerPLSQL" utilizando la condición ON a.id_auto = al.id_auto. Selecciona las columnas marca, modelo y ano de la tabla "AutoPLSQL" donde hay coincidencias en la columna id_auto entre ambas tablas.En resumen, esta consulta devuelve información sobre los autos que han sido alquilados. Donde:

  - SELECT a.marca, a.modelo, a.ano: Selecciona las columnas marca, modelo y año de la tabla "AutoPLSQL".
  - FROM AutoPLSQL a: Indica que la tabla "AutoPLSQL" se refiere como a.

    - JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto: Realiza una operación de JOIN con la tabla "AlquilerPLSQL" (al es un alias para "AlquilerPLSQL") utilizando la condición de igualdad en las columnas id_auto.
- SELECT * FROM AlquilerPLSQL WHERE id_cliente = 1;
  Funcionamiento: La consulta selecciona todos los registros de la tabla "AlquilerPLSQL." que cumplan la cláusula WHERE que se utiliza para filtrar los resultados y mostrar los registros donde el valor de la columna "id_cliente" es igual a 1.
  Resultado: El resultado de la consulta mostrará una lista de filas donde cada fila representa un alquiler realizado por el cliente con un ID igual a 1. Cada fila incluye detalles del ID del alquiler, el ID del cliente, el ID del auto, la fecha de inicio y la fecha de finalización del alquiler. Esta consulta proporciona una vista de los alquileres específicos asociados a un cliente en particular.

- SELECT * FROM AlquilerPLSQL WHERE id_auto = 1;
  Funcionamiento: se seleccionará todos los registros de la tabla "AlquilerPLSQL" donde la columna id_auto es igual a 1. Y esta consulta filtrara los resultados para mostrar únicamente los alquileres asociados al auto con id_auto igual a 1, cada fila en el resultado representará un alquiler específico para ese auto. Lo importante es que la columna id_auto exista en la tabla "AlquilerPLSQL" y tenga el tipo de datos correcto. Si usas claves foráneas, asegúrate de que el valor 1 en id_auto corresponda a un auto existente en la tabla "AutoPLSQL".

- SELECT * FROM AlquilerPLSQL WHERE id_sucursal = 1;
  Funcionamiento: Esta consulta utiliza SELECT para mostrar todos los registros de la tabla "AlquilerPLSQL." donde se cumpla la cláusula WHERE, la cual se utiliza para filtrar los resultados y mostrar solo los registros donde el valor de la columna "id_sucursal" es igual a 1.
  Resultado:El resultado de la consulta trae la información en una lista de filas donde cada fila representa un alquiler que se ha realizado en la sucursal con un ID igual a 1.

- SELECT * FROM AlquilerPLSQL WHERE fecha_inicio = '2023-09-27'; seleccionará todos los registros de la tabla "AlquilerPLSQL" donde la columna fecha_inicio sea igual a la fecha '2023-09-27'. Esta consulta filtrará los resultados para mostrar únicamente los alquileres que iniciaron exactamente el 27 de septiembre de 2023. Cada fila en el resultado representará un alquiler específico que comenzó en esa fecha.

- SELECT COUNT(*) FROM AlquilerPLSQL;
  Funcionamiento: Esta consulta utiliza la función de agregación COUNT(*) en Oracle SQL para contar el número total de registros en la tabla "AlquilerPLSQL". Esta dunción cuenta todos los registros sin importar el valor de ninguna columna en particular ya que no contiene filtros.
  Resultado: La consulta muestra un solo valor numérico que representa el número total de registros en la tabla "AlquilerPLSQL". Este valor indica cuántos registros de alquiler existen en la base de datos y es útil para obtener una visión general de la cantidad de alquileres almacenados en la tabla.

- SELECT c.nombre FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente JOIN SucursalPLSQL s ON a.id_sucursal = s.id_sucursal WHERE s.nombre = 'Sucursal Central';

Funcionamiento; donde se usa una operación JOIN entre las tablas "ClientePLSQL", "AlquilerPLSQL" y "SucursalPLSQL". Luego, filtra los resultados para incluir solo los clientes cuyos alquileres están asociados a la sucursal con el nombre 'Sucursal Central'. La consulta selecciona el nombre de los clientes que cumplen con estos criterios.

- SELECT a.marca, a.modelo FROM AutoPLSQL a
  JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto WHER al.id_cliente = 1 AND al.fecha_inicio = '2023-09-27';
  Funcionamiento, con esta operación JOIN entre las tablas "AutoPLSQL" y "AlquilerPLSQL" y luego aplica condiciones de filtrado en las columnas id_cliente y fecha_inicio. La consulta selecciona las columnas marca y modelo de la tabla "AutoPLSQL" para aquellos autos que cumplen con las condiciones especificadas.esta es una variable importante JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto: Realiza una operación de JOIN con la tabla "AlquilerPLSQL" (al es un alias para "AlquilerPLSQL") utilizando la condición de igualdad en las columnas id_auto.

- SELECT * FROM AlquilerPLSQL WHERE fecha_fin - fecha_inicio > 7;
  Funcionamiento: Esta consulta selecciona todos los registros de la tabla "AlquilerPLSQL." donde la cláusula WHERE permite aplicar una condición de filtro. En este ejercicio, la condición compara la diferencia entre las columnas "fecha_fin" y "fecha_inicio" para cada registro y verifica si esa diferencia es mayor que 7 días.
  Resultado: La consulta muestra una tabla donde cada fila representa un alquiler. Estos alquileres cumplen con la condición de tener una duración mayor a 7 días entre la fecha de inicio y la fecha de finalización. Cada fila incluye detalles como el ID del alquiler, el ID del cliente, el ID del auto, la fecha de inicio y la fecha de finalización del alquiler y esta información es útil para identificar alquileres que tuvieron una duración prolongada.

- SELECT c.nombre, COUNT(*) AS numero_alquileres FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente GROUP BY c.nombre ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento, se selecciona la columna nombre de la tabla "ClientePLSQL".
  después se usa la función de agregación COUNT(*) para contar el número de filas agrupadas por cliente y por último asigna el alias número alquileres al resultado de la función COUNT(*), Agrupa los resultados por la columna nombre de la tabla "ClientePLSQL".
- SELECT a.marca, a.modelo, COUNT(*) AS numero_alquileres FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto GROUP BY a.marca, a.modelo ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento: El código utiliza una operación JOIN para combinar datos de dos tablas: "AutoPLSQL" (alias "a") y "AlquilerPLSQL" (alias "al"), utilizando cómo llave de unión la columna "id_auto" entre ambas tablas.
  Luego, se aplica una operación de agrupamiento utilizando la cláusula GROUP BY. para agrupar la información por la combinación de su marca y modelo, y realizar el conteo de cada combinación cuándo ha sido alquilada utilizando COUNT(*).
  Los resultados se ordenan en orden descendente (del mayor al menor) en función del número de alquileres, utilizando la cláusula ORDER BY numero_alquileres DESC, y finalmente, se utiliza la cláusula LIMIT 1 para mostrar solo el resultado con el mayor número de alquileres, es decir, el auto más alquilado.

Resultado: La consulta muestra una sola fila que contiene la marca y modelo del auto más alquilado, junto con el número de alquileres que ha tenido, permite identificar cuál es el auto más popular en términos de alquileres en la base de datos.

- SELECT s.nombre, COUNT(*) AS numero_alquileres FROM SucursalPLSQL s JOIN AlquilerPLSQL al ON s.id_sucursal = al.id_sucursal GROUP BY s.nombre ORDER BY numero_alquileres DESC LIMIT 1;
- SELECT EXTRACT(MONTH FROM fecha_inicio) AS mes, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(MONTH FROM fecha_inicio) ORDER BY numero_alquileres DESC LIMIT 1;

  Funcionamiento: Se utiliza la función EXTRACT para obtener el mes a partir de la columna "fecha_inicio" en cada registro de fecha en la tabla "AlquilerPLSQL.". Luego, se aplica la operación de agrupamiento con la cláusula GROUP BY para agrupar los alquileres por mes y se cuenta cuántos alquileres se realizaron utilizando COUNT(*). Los resultados se ordenan de forma descendente (del mayor al menor) en función del número de alquileres, utilizando la cláusula ORDER BY numero_alquileres DESC y se utiliza la cláusula LIMIT 1 para mostrar solo el resultado con el mes que tiene el mayor número de alquileres.

  Resultado:Se muestra una sola fila que contiene el mes con el mayor número de alquileres y la cantidad de alquileres realizados en ese mes para este identificando cuál es el mes más concurrido en términos de alquileres en la base de datos.

- SELECT EXTRACT(DAYOFWEEK FROM fecha_inicio) AS dia_semana, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio) ORDER BY numero_alquileres DESC LIMIT 1;

  Funcionamiento: usa la función EXTRACT para obtener el día de la semana de la columna fecha_inicio en la tabla "AlquilerPLSQL", se Asigna el alias dia_semana al resultado.Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada día de la semana. Agrupa los resultados por el día de la semana obtenido de la columna fecha_inicio

- SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;
  La consulta SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;

  Funcionamiento: La consulta utiliza SELECT para llamar todos los registros de la tabla "AlquilerPLSQL.", después se utiliza la cláusula ORDER BY para ordenar los resultados en orden descendente (del mayor al menor) en función de la columna "precio.", y finalmente, se utiliza la cláusula LIMIT 1 para mostrar solo el primer registro del resultado.

  Resultado: El resultado de la consulta mostrará una sola fila que contiene todos los detalles de un alquiler con el precio de alquiler más alto en la base de datos incluyendo las características correspondientes como el ID del alquiler, el ID del cliente, el ID del auto, las fechas de inicio y finalización del alquiler y el precio.

- SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;

  Funcionamiento selecciona todos los campos de la tabla "AlquilerPLSQL" y los ordena en orden ascendente según la columna "precio". Luego, limita los resultados a una sola fila utilizando LIMIT 1, por lo que se obtiene el alquiler con el precio más bajo, esta consulta te dará la información completa de aquel alquiler que tiene el precio más bajo en la tabla "AlquilerPLSQL

- SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%'; SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000;
  Funcionamiento: La primera consulta “SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%'”, funciona buscando todos los registros de la tabla "ClientePLSQL" donde el nombre del cliente contiene la palabra "Juan" en cualquier parte del nombre utilizando la cláusula LIKE con el patrón '%Juan%' para realizar la búsqueda.
  El resultado es una lista de clientes cuyos nombres cumplen con este criterio.

  La segunda consulta “SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000”, busca los autos en la tabla "AutoPLSQL" cuyo precio es menor(“<”) a 10,000 unidades .
  El resultado muestra una lista de autos que cumplen con el criterio de tener un precio por debajo de 10,000 unidades y presenta la información sobre la marca, modelo y año de cada uno de esos autos.

- SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';
  Funcionamiento; seleccionará todos los registros de la tabla "AlquilerPLSQL" donde la columna "fecha_inicio" esté en el rango entre '2023-09-01' y '2023-09-30'. Recuperará todos los alquileres que comenzaron en el mes de septiembre de 2023.
  Esta consulta te proporcionará todos los alquileres que comenzaron en el mes de septiembre de 2023 en la tabla "AlquilerPLSQL"
- SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente WHERE c.direccion LIKE '%Bogotá%';
  Funcionamiento: Se utiliza una operación JOIN para combinar datos de dos tablas: "ClientePLSQL" (alias "c") y "AlquilerPLSQL" (alias "a"). La relación se establece a través de la igualdad de la columna "id_cliente" en ambas tablas; se aplica una condición de filtro utilizando la cláusula WHERE para verificar si la columna "direccion" en la tabla "ClientePLSQL" contiene la palabra "Bogotá" en cualquier parte de su valor, utilizando el operador LIKE con el patrón '%Bogotá%'. Esta consulta selecciona el nombre del cliente ("c.nombre") y los detalles del alquiler, que incluyen la marca y el modelo del auto ("a.marca" y "a.modelo").
  Resultado: Se obtiene una tabla con filas que cumplen la condición de tener la palabra "Bogotá" en la dirección del cliente y contendrá datos del nombre del cliente, la marca y el modelo del auto alquilado por ese cliente, proporcionado información de los clientes que han realizado alquileres y tienen "Bogotá" en su dirección, junto con sus detalles.

- SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto WHERE al.id_reserva = 1;
- Funcionamiento: realiza una operación JOIN entre las tablas "AutoPLSQL" y "AlquilerPLSQL" utilizando la condición a.id_auto = al.id_auto, y luego filtra los resultados para incluir solo aquellos donde la columna id_reserva de la tabla "AlquilerPLSQL" es igual a 1. La consulta selecciona las columnas marca, modelo y ano de la tabla "AutoPLSQL" para los autos asociados a la reserva con id_reserva igual a 1.esta consulta te proporcionará información sobre los autos asociados a la reserva con id_reserva igual a 1 en la tabla "AlquilerPLSQL".

- SELECT * FROM AlquilerPLSQL WHERE id_cliente IN (1, 2, 3); SELECT * FROM AlquilerPLSQL WHERE id_auto IN (1, 2, 3);

  La primera consulta “SELECT * FROM AlquilerPLSQL WHERE id_cliente IN (1, 2, 3)”, busca todos los registros de la tabla "AlquilerPLSQL" donde el ID del cliente (columna "id_cliente") se encuentra en la lista de valores 1, 2 o 3.

  La segunda consulta “SELECT * FROM AlquilerPLSQL WHERE id_auto IN (1, 2, 3);”, busca todos los registros de la tabla "AlquilerPLSQL" donde el ID del auto (columna "id_auto") se encuentra en la lista de valores 1, 2 o 3.

  En ambas consultas se utiliza la cláusula IN para especificar múltiples valores de búsqueda que se encuentren en una lista y recuperar los registros que coinciden con esos valores, el resultado es una tabla con datos de alquileres asociados a los clientes o autos con los ID especificados en las listas.

- SELECT * FROM AlquilerPLSQL WHERE id_sucursal IN (1, 2, 3);

  Funcionamiento: seleccionará todos los registros de la tabla "AlquilerPLSQL" donde la columna "id_sucursal" esté incluida en el conjunto (1, 2, 3). En otras palabras, recuperará todos los alquileres que pertenecen a las sucursales con ID 1, 2 o 3.

- SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_cliente IN (1, 2, 3);

  Funcionamiento: seleccionará todos los registros de la tabla "AlquilerPLSQL" que cumplen con dos condiciones: la columna "fecha_inicio" debe estar en el rango entre '2023-09-01' y '2023-09-30', y la columna "id_cliente" debe estar incluida en el conjunto (1, 2, 3). En otras palabras, recuperará todos los alquileres que comenzaron en el mes de septiembre de 2023 y que involucran a los clientes con ID 1, 2 o 3. , esta consulta te proporcionará todos los alquileres que comenzaron en el mes de septiembre de 2023 y que involucran a los clientes con ID 1, 2 o 3 en la tabla "AlquilerPLSQL"
- SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_auto IN (1, 2, 3);

  Funcionamiento;seleccionará todos los registros de la tabla "AlquilerPLSQL" que cumplen con dos condiciones: la columna "fecha_inicio" debe estar en el rango entre '2023-09-01' y '2023-09-30', y la columna "id_auto" debe estar incluida en el conjunto (1, 2, 3). En otras palabras, recuperará todos los alquileres que comenzaron en el mes de septiembre de 2023 y que involucran a los autos con ID 1, 2 o 3.

  Consulta te proporcionará todos los alquileres que comenzaron en el mes de septiembre de 2023 y que involucran a los autos con ID 1, 2 o 3 en la tabla "AlquilerPLSQL".
- SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_sucursal IN (1, 2, 3);

  Funcionamiento: seleccionará todos los registros de la tabla "AlquilerPLSQL" que cumplen con tres condiciones: La columna "fecha_inicio" debe estar en el rango entre '2023-09-01' y '2023-09-30' y la columna "id_sucursal" debe estar incluida en el conjunto (1, 2, 3)., En otras palabras, recuperará todos los alquileres que comenzaron en el mes de septiembre de 2023, y que pertenecen a las sucursales con ID 1, 2 o 3.

  Funcionamiento Selecciona la columna nombre de la tabla "ClientePLSQL".

Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada cliente.Asigna el alias numero_alquileres al resultado de la función COUNT(*), la consulta busca encontrar el cliente que ha realizado la mayor cantidad de alquileres, mostrando su nombre y el número total de alquileres realizados.

- SELECT c.nombre, COUNT(*) AS numero_alquileres FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente GROUP BY c.nombre ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento: Selecciona la columna nombre de la tabla "ClientePLSQL".
  Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada cliente. Asigna el alias numero_alquileres al resultado de la función COUNT(*). Asi como Realiza una operación de JOIN entre las tablas "ClientePLSQL" y "AlquilerPLSQL" utilizando la condición de igualdad en las columnas id_cliente. Agrupa Realiza una operación de JOIN entre las tablas "ClientePLSQL" y "AlquilerPLSQL" utilizando la condición de igualdad en las columnas id_cliente.

- SELECT a.marca, a.modelo, COUNT(*) AS numero_alquileres FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto GROUP BY a.marca, a.modelo ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento, Selecciona las columnas marca y modelo de la tabla "AutoPLSQL". Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada combinación de marca y modelo.
  Asigna el alias numero_alquileres al resultado de la función COUNT(*).
  Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada combinación de marca y modelo. Asigna el alias numero_alquileres al resultado de la función COUNT(*).

- SELECT s.nombre, COUNT(*) AS numero_alquileres FROM SucursalPLSQL s JOIN AlquilerPLSQL al ON s.id_sucursal = al.id_sucursal GROUP BY s.nombre ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento: busca identificar la sucursal que ha tenido la mayor cantidad de alquileres. , Selecciona la columna nombre de la tabla "SucursalPLSQL".
  Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada sucursal.
  Asigna el alias numero_alquileres al resultado de la función COUNT(*), Realiza una operación de JOIN entre las tablas "SucursalPLSQL" y "AlquilerPLSQL" utilizando la condición de igualdad en las columnas id_sucursal.Agrupa los resultados por el nombre de la sucursal (GROUP BY s.nombre), de modo que la función de agregación COUNT(*) cuenta el número de alquileres para cada sucursal en lugar de contar todos los alquileres en conjunto

- SELECT EXTRACT(MONTH FROM fecha_inicio) AS mes, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(MONTH FROM fecha_inicio) ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento: Utiliza la función EXTRACT para obtener el mes de la columna fecha_inicio en la tabla "AlquilerPLSQL".

Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada mes. asigna el alias numero_alquileres al resultado de la función COUNT(*).Agrupa los resultados por el mes de la fecha de inicio (GROUP BY EXTRACT(MONTH FROM fecha_inicio)), de modo que la función de agregación COUNT(*) cuenta el número de alquileres para cada mes en lugar de contar todos los alquileres en conjunto.esta consulta te proporcionará el mes con la mayor cantidad de alquileres y el número total de alquileres realizados en ese mes.

- SELECT EXTRACT(DAYOFWEEK FROM fecha_inicio) AS dia_semana, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio) ORDER BY numero_alquileres DESC LIMIT 1;
  Funcionamiento; Utiliza la función EXTRACT para obtener el día de la semana de la columna fecha_inicio en la tabla "AlquilerPLSQL".
  Utiliza la función de agregación COUNT(*) para contar el número de alquileres para cada día de la semana. Asigna el alias numero_alquileres al resultado de la función COUNT(*).Agrupa los resultados por el día de la semana de la fecha de inicio (GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio)), de modo que la función de agregación COUNT(*) cuenta el número de alquileres para cada día de la semana en lugar de contar todos los alquileres en conjunto.Ordena los resultados en orden descendente según el número de alquileres.

- SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;
  Funcionamiento: eleccionará todos los campos de la tabla "AlquilerPLSQL" y los ordenará en orden descendente según la columna "precio". Luego, limitará los resultados a una sola fila, seleccionando así el alquiler con el precio más alto. En resumen, esta consulta te proporcionará toda la información sobre el alquiler con el precio más alto en la tabla "AlquilerPLSQL"
- SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;
  Funcionamiento: seleccionará todos los campos de la tabla "AlquilerPLSQL" y los ordenará en orden ascendente según la columna "precio". Luego, limitará los resultados a una sola fila, seleccionando así el alquiler con el precio más bajo, esta consulta te proporcionará toda la información sobre el alquiler con el precio más bajo en la tabla "AlquilerPLSQL".

- SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%' AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30'; SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000 AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';
  Funcionamiento: Selecciona todas las columnas de la tabla "ClientePLSQL".
  Filtra los resultados para incluir solo aquellos cuyo nombre contiene la cadena "Juan". También filtra por la fecha de inicio, incluyendo solo aquellos con fechas dentro del rango del 1 al 30 de septiembre de 2023.Selecciona las columnas de "marca", "modelo" y "ano" de la tabla "AutoPLSQL".
  Filtra los resultados para incluir solo aquellos autos con un precio inferior a 10000.
  También filtra por la fecha de inicio, incluyendo solo aquellos con fechas dentro del rango del 1 al 30 de septiembre de 2023.Ambas consultas utilizan la cláusula WHERE para aplicar condiciones de filtrado en las filas seleccionadas. La primera está orientada a clientes, mientras que la segunda está orientada a autos. Ambas incluyen una condición de fecha para limitar los resultados al mes de septiembre de 2023.

EJERCICIOS TERCER CICLO (81-90):

- CREATE VIEW vista_clientes_alquilados_sucursal AS SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente JOIN SucursalPLSQL s ON a.id_sucursal = s.id_sucursal WHERE s.nombre = 'Sucursal Central';
  Funcionamiento La vista vista_clientes_alquilados_sucursal que has creado mostrará los nombres de los clientes junto con la marca y el modelo de los autos que han sido alquilados en la "Sucursal Central". Puedes utilizar esta vista en consultas de la misma manera que lo harías con una tabla. Esta vista facilitará la obtención de información específica sobre los clientes y los autos alquilados en la "Sucursal Central" sin tener que repetir la lógica de la consulta JOIN c

- CREATE VIEW vista_autos_alquilados_cliente_fecha AS SELECT a.marca, a.modelo FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto WHERE al.id_cliente = 1 AND al.fecha_inicio = '2023-09-27';
  Funcionamiento La vista vista_autos_alquilados_cliente_fecha que has creado mostrará la marca y el modelo de los autos que han sido alquilados por el cliente con el ID 1 en la fecha de inicio '2023-09-27'. Esta vista facilitará la obtención de información específica sobre los autos alquilados por un cliente en una fecha particular sin tener que repetir la lógica de la consulta JOIN
- CREATE VIEW vista_alquileres_mas_7dias AS SELECT * FROM AlquilerPLSQL WHERE fecha_fin - fecha_inicio > 7;
  FuncionamientoEsta vista selecciona todos los campos de la tabla "AlquilerPLSQL" para los alquileres donde la diferencia entre la fecha de fin y la fecha de inicio es mayor a 7 días
- CREATE VIEW vista_clientes_mas_alquileres AS SELECT c.nombre, COUNT(*) AS numero_alquileres FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente GROUP BY c.nombre ORDER BY numero_alquileres DESC;
  Funcionamiento, esta vista muestra el nombre de los clientes junto con el número total de alquileres realizados por cada cliente, ordenados en orden descendente por la cantidad de alquileres. Utiliza las tablas "ClientePLSQL" y "AlquilerPLSQL", y se agrupa por el nombre del cliente.

- CREATE VIEW vista_autos_mas_alquileres AS SELECT a.marca, a.modelo, COUNT(*) AS numero_alquileres FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto GROUP BY a.marca, a.modelo ORDER BY numero_alquileres DESC;
  Esta vista proporciona la marca y el modelo de los autos junto con el número total de alquileres para cada combinación de marca y modelo. La información está ordenada en orden descendente por el número de alquileres.
- CREATE VIEW vista_sucursales_mas_alquileres AS SELECT s.nombre, COUNT(*) AS numero_alquileres FROM SucursalPLSQL s JOIN AlquilerPLSQL al ON s.id_sucursal = al.id_sucursal GROUP BY s.nombre ORDER BY numero_alquileres DESC;
  Esta vista muestra el nombre de las sucursales junto con el número total de alquileres realizados en cada sucursal. La información está ordenada en orden descendente por el

número de alquileres. Ambas vistas te permitirán analizar fácilmente la relación entre los autos, las sucursales y la frecuencia de alquileres

- CREATE VIEW vista_meses_mas_alquileres AS SELECT EXTRACT(MONTH FROM fecha_inicio) AS mes, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(MONTH FROM fecha_inicio) ORDER BY numero_alquileres DESC;
  Esta vista muestra por mes el numero de alquileres por fecha de inicio. La información está ordenada en orden descendente por el número de alquileres. Ambas vistas te permitirán analizar fácilmente la relación entre los autos, las sucursales y la frecuencia de alquileres
- CREATE VIEW vista_dias_semana_mas_alquileres AS SELECT EXTRACT(DAYOFWEEK FROM fecha_inicio) AS dia_semana, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio) ORDER BY numero_alquileres DESC;
  Crea una vista llamada vista_dias_semana_mas_alquileres. Esta vista se basa en la tabla llamada AlquilerPLSQL. La vista tiene dos columnas:

  dia_semana: Esta columna utiliza la función EXTRACT(DAYOFWEEK FROM fecha_inicio) para extraer el día de la semana de la columna fecha_inicio en la tabla AlquilerPLSQL. Esta función devuelve un número del 1 al 7, donde 1 representa el domingo, 2 el lunes y así sucesivamente.

  numero_alquileres: Esta columna utiliza la función de agregación COUNT(*) para contar el número de registros (alquileres) para cada día de la semana. La vista agrupa los resultados por el día de la semana y utiliza GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio).

  Además, la vista está ordenada en orden descendente por la columna numero_alquileres utilizando la cláusula ORDER BY numero_alquileres DESC.Esta vista proporciona una lista del número de alquileres realizados para cada día de la semana, ordenados de mayor a menor según la cantidad de alquileres.

- CREATE VIEW vista_alquileres_mas_caros AS SELECT * FROM AlquilerPLSQL ORDER BY precio DESC;

  La vista selecciona todas las columnas de la tabla AlquilerPLSQL y las ordena en orden descendente según la columna precio, esto significa que la vista incluirá todas las filas y columnas de la tabla AlquilerPLSQL, pero las ordenará de manera que los registros con el precio más alto aparecerán primero.

- CREATE VIEW vista_alquileres_mas_baratos AS SELECT * FROM AlquilerPLSQL ORDER BY precio ASC;
  Crea una vista llamada vista_alquileres_mas_baratos. Al igual que en el caso anterior, la vista selecciona todas las columnas de la tabla AlquilerPLSQL, pero en lugar de ordenar en orden descendente, ordena los resultados en orden ascendente según la columna precio.

EJERCICIOS TERCER CICLO (91-100):

```
CREATE TRIGGER trg_insert_auto
BEFORE INSERT ON AutoPLSQL
FOR EACH ROW
BEGIN
  -- Actualizar el número de autos disponibles
  UPDATE AutoPLSQL
    SET numero_disponibles = numero_disponibles + 1
  WHERE id_auto = NEW.id_auto;
END;
```

Funcionamiento: define un trigger (desencadenador) llamado trg_insert_auto que se activa antes de realizar una inserción (BEFORE INSERT) en la tabla AutoPLSQL. Este trigger se ejecuta por cada fila (FOR EACH ROW) que se va a insertar.

El propósito de este trigger es actualizar el número de autos disponibles en la tabla AutoPLSQL justo antes de realizar una inserción. La lógica del trigger es la siguiente:

Se utiliza la cláusula BEGIN para indicar el inicio de la sección de código del trigger.

Se realiza una sentencia UPDATE en la misma tabla AutoPLSQL. La sentencia actualiza la columna numero_disponibles incrementándola en 1 (numero_disponibles = numero_disponibles + 1) para el registro que está siendo insertado. La condición WHERE id_auto = NEW.id_auto asegura que solo se actualiza el registro específico que se está insertando.

El trigger finaliza con la cláusula END.

Este trigger se encarga de mantener actualizado el número de autos disponibles en la tabla AutoPLSQL justo antes de que se realice una inserción de un nuevo registro en esa tabla.

```
CREATE TRIGGER trg_delete_auto
BEFORE DELETE ON AutoPLSQL
FOR EACH ROW
BEGIN
  -- Actualizar el número de autos disponibles
  UPDATE AutoPLSQL
    SET numero_disponibles = numero_disponibles - 1
  WHERE id_auto = OLD.id_auto;
END;
```

Funcionamiento: define un trigger (desencadenador) llamado trg_delete_auto que se activa antes de realizar una eliminación (BEFORE DELETE) en la tabla AutoPLSQL. Al igual que el trigger anterior, este trigger se ejecuta por cada fila (FOR EACH ROW) que está a punto de ser eliminada. La función de este trigger es actualizar el número de autos disponibles en la tabla AutoPLSQL justo antes de llevar a cabo la eliminación.

```
CREATE TRIGGER trg_update_auto
BEFORE UPDATE ON AutoPLSQL
FOR EACH ROW
BEGIN
  -- Actualizar el número de autos disponibles
  IF NEW.numero_disponibles != OLD.numero_disponibles THEN
```

```
   UPDATE AutoPLSQL
    SET numero_disponibles = NEW.numero_disponibles
    WHERE id_auto = NEW.id_auto;
 END IF;
END;
```

Funcionamiento: define un trigger llamado trg_update_auto que se activa antes de realizar una actualización (BEFORE UPDATE) en la tabla AutoPLSQL. Este trigger se ejecuta por cada fila (FOR EACH ROW) que está a punto de ser actualizada.

La función de este trigger es verificar si el valor de la columna numero disponibles está siendo modificado en la actualización. Si hay un cambio, se realiza una sentencia UPDATE para reflejar el nuevo valor de numero disponibles, este trigger se encarga de mantener actualizado el número de autos disponibles en la tabla AutoPLSQL justo antes de que se realice una actualización, pero solo si la actualización afecta el valor de la columna numero disponibles.

```
CREATE TRIGGER trg_insert_cliente
BEFORE INSERT ON ClientePLSQL
FOR EACH ROW
BEGIN
 -- Actualizar el número de clientes
 UPDATE ClientePLSQL
   SET numero_clientes = numero_clientes + 1;
END;
```

Funcionamiento: este trigger se encarga de mantener actualizado el número total de clientes en la tabla ClientePLSQL justo antes de que se realice una inserción de un nuevo cliente. Cabe mencionar que este enfoque puede tener limitaciones en entornos de concurrencia, ya que varios intentos de inserción simultánea podrían causar problemas de concurrencia al intentar actualizar el mismo registro.

```
CREATE TRIGGER trg_delete_cliente
BEFORE DELETE ON ClientePLSQL
FOR EACH ROW
BEGIN
 -- Actualizar el número de clientes
 UPDATE ClientePLSQL
   SET numero_clientes = numero_clientes - 1;
END;
```

Funcionamiento: Este código SQL define un trigger llamado trg delete cliente que se activa antes de realizar una eliminación (BEFORE DELETE) en la tabla ClientePLSQL. Al igual que los triggers anteriores, este trigger se ejecuta por cada fila (FOR EACH ROW) que está a punto de ser eliminada.La función de este trigger es actualizar el número total de clientes en la tabla ClientePLSQL justo antes de realizar la eliminación de un cliente

```
CREATE TRIGGER trg_update_cliente
BEFORE UPDATE ON ClientePLSQL
FOR EACH ROW
BEGIN
 -- Actualizar el número de clientes
 IF NEW.numero_alquileres != OLD.numero_alquileres THEN
```

```
   UPDATE ClientePLSQL
    SET numero_alquileres = NEW.numero_alquileres
    WHERE id_cliente = NEW.id_cliente;
  END IF;
END;
```

Funcionamiento: un trigger llamado trg_update_cliente que se activa antes de realizar una actualización (BEFORE UPDATE) en la tabla ClientePLSQL. Al igual que los triggers anteriores, este trigger se ejecuta por cada fila (FOR EACH ROW) que está a punto de ser actualizada. La función de este trigger es verificar si el valor de la columna número alquileres está siendo modificado en la actualización. Si hay un cambio, se realiza una sentencia UPDATE para reflejar el nuevo valor de número alquileres

```
CREATE PROCEDURE proc_calcular_precio_alquiler
(
  IN id_alquiler INT,
  IN id_auto INT,
  IN fecha_inicio DATE,
  IN fecha_fin DATE
)
AS
BEGIN
  -- Calcular el precio del alquiler
  DECLARE
    precio_base NUMERIC(10, 2);
    dias_alquiler INT;
  BEGIN
    precio_base := (SELECT precio FROM AutoPLSQL WHERE id_auto = id_auto);
    dias_alquiler := (fecha_fin - fecha_inicio) + 1;
    SET NEW.precio = precio_base * dias_alquiler;
  END;
END;
```

Funcionamiento: ser un procedimiento almacenado en un entorno que utiliza el lenguaje PL/SQL, como Oracle Database. Sin embargo, hay un problema en el código que debería ser mencionado. El uso de SET NEW es típicamente asociado con triggers (desencadenadores) y no con procedimientos almacenados. En un procedimiento almacenado, generalmente no se usa SET NEW porque esa sintaxis está más relacionada con la manipulación de registros en triggers de eventos como BEFORE INSERT, BEFORE UPDATE, o BEFORE DELETE.

```
CREATE PROCEDURE proc_listar_alquileres_cliente
(
  IN id_cliente INT
)
AS
BEGIN
  -- Listar los alquileres del cliente
  SELECT *
  FROM AlquilerPLSQL
  WHERE id_cliente = id_cliente;
END;
```

Funcionamiento: para recuperar todos los alquileres asociados al cliente con el id_cliente

proporcionado. Ten en cuenta que, dependiendo de cómo se llame a este procedimiento y cómo se manejen los resultados, podrías querer ajustar la forma en que se devuelve o procesa la información recuperada

```
CREATE PROCEDURE proc_listar_autos_sucursal
(
  IN id_sucursal INT
)
AS
BEGIN
  -- Listar los autos de la sucursal
  SELECT *
  FROM AutoPLSQL
  WHERE id_sucursal = id_sucursal;
END;
```

Funcionamiento: para recuperar todos los alquileres asociados a la sucursal con el id_sucursal proporcionado. Ten en cuenta que, dependiendo de cómo se llame a este procedimiento y cómo se manejen los resultados, podrías querer ajustar la forma en que se devuelve o procesa la información recuperada

```
CREATE PROCEDURE proc_agregar_auto
(
  IN marca VARCHAR(255),
  IN modelo VARCHAR(255),
  IN ano INT,
  IN numero_disponibles INT
)
AS
BEGIN
  -- Insertar un nuevo auto
  INSERT INTO AutoPLSQL (marca, modelo, ano, numero_disponibles)
  VALUES (marca, modelo, ano, numero_disponibles);
END;
```

Funcionamiento: simple inserta un nuevo auto en la tabla AutoPLSQL con los valores proporcionados como parámetros. Asume que los parámetros proporcionados son válidos y que no hay restricciones de clave primaria u otras restricciones en la tabla que puedan impedir la inserción

```
CREATE PROCEDURE proc_eliminar_auto
(
  IN id_auto INT
)
AS
BEGIN
  -- Eliminar un auto
  DELETE FROM AutoPLSQL
  WHERE id_auto = id_auto;
END;
```

Funcionamiento: Elimina un auto de la tabla AutoPLSQL basado en el identificador proporcionado. Esta operación eliminará permanentemente el registro de la tabla. Y tener en cuenta cualquier relación o restricción referencial que pueda afectar la eliminación.