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".
     SELECT * FROM ClientePLSQL;
   - Mostrar todos los autos en la tabla "Auto".
     SELECT * FROM AutoPLSQL;
   - Mostrar todos los alquileres en la tabla "Alquiler".
     SELECT * FROM AlquilerPLSQL;
   - Mostrar todas las sucursales en la tabla "Sucursal".
     SELECT * FROM SucursalPLSQL;
   - Mostrar todas las reservas en la tabla "Reserva".
     SELECT * FROM ReservaPLSQL;

2. **Filtros y Ordenamiento:**

   - Mostrar los clientes que se llaman "Juan".
     SELECT * FROM ClientePLSQL WHERE nombre = 'Juan';
   - Mostrar los autos de marca "Toyota".
     SELECT * FROM AutoPLSQL WHERE marca = 'Toyota';
   - Mostrar los alquileres que ocurrieron después de una fecha específica.
     SELECT * FROM AlquilerPLSQL WHERE fecha_inicio > TO_DATE('2023-10-31', 'YYYY-MM-DD');
   - Mostrar las sucursales ubicadas en "Madrid".
     SELECT * FROM SucursalPLSQL WHERE ciudad = 'Madrid';
   - Mostrar las reservas realizadas por un cliente específico.
     SELECT * FROM ReservaPLSQL WHERE id_cliente = 1;

3. **Join y Relaciones:**

   "Al.id_alquiler, C.nombre AS nombre_cliente, AU.marca AS marca_auto" son las columnas seleccionadas.
   "JOIN ClientePLSQL C ON Al.id_cliente = C.id_cliente" combina AlquilerPLSQL con ClientePLSQL usando la condición de igualdad en IDs de clientes.

- Muestra los alquileres con los nombres de los clientes y las marcas de los autos.
  SELECT Al.id_alquiler, C.nombre AS nombre_cliente, AU.marca AS marca_auto
  FROM AlquilerPLSQL Al
  JOIN ClientePLSQL C ON Al.id_cliente = C.id_cliente
  JOIN AutoPLSQL AU ON Al.id_auto = AU.id_auto;
- Mostrar los clientes que han realizado reservas en una sucursal específica.
  SELECT C.id_cliente, C.nombre AS nombre_cliente
  FROM ClientePLSQL C
  JOIN ReservaPLSQL R ON C.id_cliente = R.id_cliente
  WHERE R.id_sucursal = 1;
- Mostrar los autos que han sido alquilados junto con los nombres de los clientes.
  SELECT AU.id_auto, AU.marca AS marca_auto, C.nombre AS nombre_cliente
  FROM AutoPLSQL AU
  JOIN AlquilerPLSQL Al ON AU.id_auto = Al.id_auto
  JOIN ClientePLSQL C ON Al.id_cliente = C.id_cliente;
- Muestra los detalles de las reservas con los nombres de los clientes y las ciudades de las sucursales.
  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;
- Mostrar los clientes que no han realizado ninguna reserva.
  SELECT C.id_cliente, C.nombre AS nombre_cliente
  FROM ClientePLSQL C
  LEFT JOIN ReservaPLSQL R ON C.id_cliente = R.id_cliente
  WHERE R.id_cliente IS NULL;

4. **Agregación y Agrupamiento:**

.

- Contar cuántos autos hay de cada marca en la tabla "Auto".
  SELECT marca, COUNT(*) AS cantidad
  FROM AutoPLSQL
  GROUP BY marca;
- Calcular la duración promedio de los alquileres.
  SELECT AVG(DATEDIFF(DAY, fecha_inicio, fecha_fin)) AS duracion_promedio
  FROM AlquilerPLSQL;
- Mostrar el número total de reservas realizadas en cada sucursal.
  SELECT S.id_sucursal, S.nombre AS nombre_sucursal, COUNT(*) AS cantidad_reservas
  FROM SucursalPLSQL S
  LEFT JOIN ReservaPLSQL R ON S.id_sucursal = R.id_sucursal
  GROUP BY S.id_sucursal, S.nombre;

- Encontrar el cliente que ha realizado la mayor cantidad de alquileres.
  SELECT C.id_cliente, C.nombre AS nombre_cliente, COUNT(*) AS cantidad_alquileres
  FROM ClientePLSQL C
  JOIN AlquilerPLSQL Al ON C.id_cliente = Al.id_cliente
  GROUP BY C.id_cliente, C.nombre
  ORDER BY cantidad_alquileres DESC
  LIMIT 1;
- Calcular el promedio de años de los autos en la tabla "Auto".
  SELECT AVG(ano) AS promedio_anos
  FROM AutoPLSQL;

5. **Subconsultas**:
   - Mostrar los clientes que han realizado al menos una reserva.
     SELECT id_cliente, nombre
     FROM ClientePLSQL
     WHERE id_cliente IN (SELECT DISTINCT id_cliente FROM ReservaPLSQL);
   - Mostrar los autos que no han sido alquilados aún.
     SELECT id_auto, marca, modelo, ano
     FROM AutoPLSQL
     WHERE id_auto NOT IN (SELECT DISTINCT id_auto FROM AlquilerPLSQL);
   - Encontrar los clientes que han alquilado el mismo auto más de una vez.
     SELECT id_cliente, nombre
     FROM ClientePLSQL
     WHERE id_cliente IN (
     SELECT id_cliente
     FROM AlquilerPLSQL
     GROUP BY id_cliente, id_auto
     HAVING COUNT(*) > 1);
   - Mostrar los clientes que han realizado alquileres en la misma ciudad en la que viven.
     SELECT id_cliente, nombre
     FROM ClientePLSQL
     WHERE direccion IN (
     SELECT ciudad
     FROM AlquilerPLSQL
     JOIN SucursalPLSQL ON AlquilerPLSQL.id_sucursal = SucursalPLSQL.id_sucursal
     WHERE id_cliente = ClientePLSQL.id_cliente);
   - Encontrar los autos que han sido alquilados en la misma sucursal donde se realizó una reserva.
     SELECT id_auto, marca, modelo, ano
     FROM AutoPLSQL
     WHERE id_auto IN (
     SELECT DISTINCT AlquilerPLSQL.id_auto
     FROM AlquilerPLSQL

JOIN ReservaPLSQL ON AlquilerPLSQL.id_reserva = ReservaPLSQL.id_reserva);

6. **Actualizaciones y Eliminaciones:**
   - Actualizar la dirección de un cliente específico.
     UPDATE ClientePLSQL
     SET direccion = 'Calle 1 # 2 - 3'
     WHERE id_cliente = 1;
   - Eliminar un auto de la tabla "Auto".
     DELETE FROM AutoPLSQL
     WHERE id_auto = 1;
   - Marcar una reserva como completada actualizando la fecha de fin.
     UPDATE ReservaPLSQL
     SET fecha_fin = SYSDATE
     WHERE id_reserva = 1;
   - Eliminar todas las reservas realizadas por un cliente específico.
     DELETE FROM ReservaPLSQL
     WHERE id_cliente = 1;
   - Actualizar el año de un auto en la tabla "Auto".
     UPDATE AutoPLSQL
     SET ano = 2022
     WHERE id_auto = 1;

EJERCICIOS SEGUNDO CICLO (31-80):

- `SELECT * FROM ClientePLSQL;` Muestra una lista completa de información sobre todos los clientes en la base de datos.
- `SELECT * FROM AutoPLSQL;` Muestra una lista completa de información sobre todos los autos en la base de datos.
- `SELECT * FROM AlquilerPLSQL;` Muestra una lista completa de información sobre todos los alquileres en la base de datos.

- `SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente;` Muestra los nombres de los clientes junto con los autos que han alquilado.

- `SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id_auto = al.id_auto;` Muestra detalles de los autos que han sido parte de alquileres.

- `SELECT * FROM AlquilerPLSQL WHERE id_cliente = 1;` Muestra los alquileres específicos asociados al cliente número 1 en la base de datos.

- `SELECT * FROM AlquilerPLSQL WHERE id_auto = 1;` Muestra los alquileres relacionados con el auto número 1 en la base de datos.

- `SELECT * FROM AlquilerPLSQL WHERE id_sucursal = 1;` Muestra los alquileres específicamente relacionados con la sucursal número 1 en la base de datos.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_inicio = '2023-09-27';`
  Muestra los alquileres que comenzaron en la fecha especificada en la base de datos.

- `SELECT COUNT(*) FROM AlquilerPLSQL;`
  Proporciona el recuento de todos los alquileres almacenados en la base de datos.

- `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';` Muestra los nombres de los clientes que alquilaron en la 'Sucursal Central

- `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';` Muestra qué autos alquiló un cliente en un día específico, el 27 de septiembre de 2023, mostrando solo la marca y el modelo de los autos.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_fin - fecha_inicio > 7;`
  Muestra los alquileres que tuvieron una duración de más de una semana.

- `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;` Muestra el nombre del cliente que ha realizado la mayor cantidad de alquileres.

- `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;`
  Muestra el automóvil más alquilado, indicando su marca y modelo, al contar cuántas veces se ha alquilado y ordenarlos por popularidad.

- `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;` Muestra la sucursal con la mayor cantidad de alquileres, contando cuántos alquileres se han hecho en cada sucursal y mostrando la más popular.

- `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;` Muestra el mes con la mayor cantidad de alquileres, contando cuántos alquileres se hicieron en cada mes y resaltando el mes más concurrido.

- `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;` Muestra el día de la semana con más alquileres, contando cuántos alquileres se realizaron en cada día y destacando el día más concurrido.

- `SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;` Muestra el alquiler más caro, ordenando los registros de alquiler por precio de forma descendente y mostrando el de mayor costo.

- `SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;` Muestra el alquiler más barato al ordenar los registros de alquiler por precio en orden ascendente y mostrar el de menor costo.

- `SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%';` Selecciona los clientes cuyo nombre contiene "Juan".

- `SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000;` Selecciona los autos con un precio inferior a 10,000.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';` Muestra los alquileres realizados durante ese período específico.

- `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%';` Muestra los clientes que alquilaron autos y tienen "Bogotá" en su dirección.

- `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;` Muestra los detalles de los autos asociados a una reserva particular.

- `SELECT * FROM AlquilerPLSQL WHERE id_cliente IN (1, 2, 3);` Selecciona los alquileres relacionados con esos clientes específicos.
- `SELECT * FROM AlquilerPLSQL WHERE id_auto IN (1, 2, 3);` Selecciona los alquileres relacionados con esos autos específicos.

- `SELECT * FROM AlquilerPLSQL WHERE id_sucursal IN (1, 2, 3);` Selecciona los alquileres relacionados con esas sucursales específicas.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_cliente IN (1, 2, 3);` Selecciona alquileres dentro de un período específico y relacionados con esos clientes.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_auto IN (1, 2, 3);` Selecciona alquileres dentro de un período específico y relacionados con esos autos.

- `SELECT * FROM AlquilerPLSQL WHERE fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id_sucursal IN (1, 2, 3);` Selecciona alquileres dentro de un período específico y relacionados con esas sucursales.

- `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;` Muestra el nombre del cliente que ha realizado la mayor cantidad de alquileres.

- `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;` Muestra la marca y modelo del automóvil que ha sido alquilado con mayor frecuencia.

- `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;` Muestra el nombre de la sucursal que ha tenido la mayor cantidad de alquileres.

- `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;` Muestra el mes en el que se realizaron la mayoría de los alquileres.

- `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;` Muestra el día de la semana en el que se han realizado la mayoría de los alquileres.

- `SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;` Muestra el alquiler más costoso al ordenar los registros de alquiler en orden descendente según el precio y mostrar el alquiler con el precio más alto.

- `SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;` Muestra el alquiler más económico al ordenar los registros de alquiler en orden ascendente según el precio y mostrar el alquiler con el precio más bajo.

- `SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%' AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';` Muestra clientes con el nombre "Juan" que alquilaron durante ese período.

- `SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000 AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';` Muestra autos asequibles que estuvieron involucrados en alquileres durante ese período.

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';` Crea una vista que muestra los nombres de los clientes, junto con la marca y modelo de los autos que han alquilado en la "Sucursal Central".

- `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';` En este se crea una vista que muestra la marca y modelo de los autos alquilados por un cliente específico (ID 1) en una fecha particular (27 de septiembre de 2023).

- `CREATE VIEW vista_alquileres_mas_7dias AS SELECT * FROM AlquilerPLSQL WHERE fecha_fin - fecha_inicio > 7;` Crea una vista que muestra todos los alquileres que tienen una duración de más de 7 días al calcular la diferencia entre las fechas de inicio y finalización.

- `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;` Crea una vista que muestra los nombres de los clientes junto con la cantidad de alquileres que han realizado. Los resultados se ordenan de mayor a menor número de alquileres.

- `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;` Crea una vista que muestra las marcas y modelos de autos junto con la cantidad de alquileres que han tenido. Los resultados se ordenan de mayor a menor número de alquileres, destacando los autos más alquilados.

- `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;` Crea una vista que muestra el nombre de las sucursales junto con la cantidad de alquileres que han tenido. Los resultados se ordenan de mayor a menor número de alquileres, destacando las sucursales más concurridas en términos 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;` Crea una vista que muestra los meses con la cantidad de alquileres realizados en cada uno, ordenados de mayor a menor número de alquileres. Destaca los meses más activos en términos 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 que muestra los días de la semana con la cantidad de alquileres realizados en cada día, ordenados de mayor a menor número de alquileres. Resalta los días más activos en términos de alquileres.

- `CREATE VIEW vista_alquileres_mas_caros AS SELECT * FROM AlquilerPLSQL ORDER BY precio DESC;` Crea una vista que muestra todos los alquileres ordenados en orden descendente según el precio. Esta vista destaca los alquileres más costosos.

- `CREATE VIEW vista_alquileres_mas_baratos AS SELECT * FROM AlquilerPLSQL ORDER BY precio ASC;` Crea una vista que muestra todos los alquileres ordenados en orden ascendente según el precio. Esta vista resalta los alquileres más económicos.

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;
```

Su función es agregar un nuevo auto a la tabla "AutoPLSQL", el número de autos disponibles se incrementa en uno, asegurando que la disponibilidad se mantenga actualizada automáticamente.

```
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;
```

Su función es reducir en uno el número de autos disponibles cuando se elimina un auto específico. Asegura que la disponibilidad se actualice automáticamente al eliminar autos.

```
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;
```

Su función es actualizar el número de autos disponibles si se cambia el valor de "numero_disponibles" en el registro que se está modificando. Asegura que la disponibilidad se mantenga actualizada automáticamente cuando se realizan cambios en el número de autos 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;
```
Su función es incrementar el número total de clientes en la tabla automáticamente en uno cada vez que se agrega un nuevo cliente. Asegura que el contador de clientes se mantenga actualizado automáticamente.

```
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;
```
Su propósito es disminuir en uno el número total de clientes en la tabla automáticamente cada vez que se elimina un cliente. Esto asegura que el contador de clientes se mantenga actualizado automáticamente cuando se eliminan clientes.

```
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;
```
Su objetivo es mantener actualizado el número de alquileres de un cliente. Si el nuevo número de alquileres es distinto al valor anterior, se efectúa una actualización en la tabla para reflejar el cambio, asegurando que el contador de alquileres del cliente se mantenga actualizado.

```
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;
```
Esta función calcula el precio de un alquiler en función de un auto, la fecha de inicio y la fecha de finalización del alquiler. Toma en cuenta el precio base del auto y la duración del alquiler. Luego, asigna el resultado al campo de precio del alquiler.

```
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;
```

Esta función se encarga de listar todos los alquileres asociados a un cliente específico. Toma el ID del cliente como entrada y recupera todos los registros de alquiler en la tabla "AlquilerPLSQL" que corresponden a ese cliente.

```
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;
```

Esta función tiene como propósito listar todos los autos asociados a una sucursal específica. Recibe el ID de la sucursal como entrada y selecciona todos los registros de autos en la tabla "AutoPLSQL" que pertenecen a esa sucursal.

```
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;
```

Esta función se utiliza para agregar un nuevo registro de auto en la tabla "AutoPLSQL". Requiere la entrada de la marca, modelo, año y número disponibles para el nuevo auto. Luego, ejecuta una sentencia SQL de inserción para agregar el nuevo registro a la tabla.

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

La función se utiliza para eliminar un registro de auto de la tabla "AutoPLSQL". Requiere la entrada del ID del auto que se desea eliminar. Luego, ejecuta una sentencia SQL de eliminación para eliminar el registro correspondiente de la tabla.