**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;**

“La sentencia trae todos los clientes de la tabla de clientes”

* + Mostrar todos los autos en la tabla "Auto".
* **Select \* from AutoPLSQL;**

“La sentencia trae todos los autos de la tabla de Autos”

* + Mostrar todos los alquileres en la tabla "Alquiler".
* **Select \* from AlquilerPLSQL;**

“La sentencia trae todos los alquileres de la tabla de Alquiler”

* + Mostrar todas las sucursales en la tabla "Sucursal".
* **Select \* from SucursalPLSQL;**

“La sentencia trae todas las sucursales de la tabla de Sucursal”

* + Mostrar todas las reservas en la tabla "Reserva".
* **Select \* from ReservaPLSQL;**

“La sentencia trae todas las reservas de la tabla de Reserva”

1. **Filtros y Ordenamiento:**
   * Mostrar los clientes que se llaman "Juan".

* **select \***

**from ClientePLSQL**

**where NOMBRE like '%Juan%';**

* + Mostrar los autos de marca "Toyota".
* **Select \***

**From AutoPLSQL**

**where MARCA like '%Toyota%';**

* + Mostrar los alquileres que ocurrieron después de una fecha específica.
* **Select \***

**from AlquilerPLSQL**

**where FECHA\_INICIO >= '01/01/23';**

* + Mostrar las sucursales ubicadas en "Madrid".
* **Select \***

**from SucursalPLSQL**

**where PAIS like '%Madrid%';**

“No sale ningun valor ya queno hay sucursales en Madrid”

* + Mostrar las reservas realizadas por un cliente específico.
* **Select \***

**From ReservaPLSQL**

**where ID\_CLIENTE = 700;**

1. **Join y Relaciones:**
   * Mostrar los alquileres con los nombres de los clientes y las marcas de los autos.

* **Select**

**cli.nombre, au.marca**

**From AlquilerPLSQL al**

**Inner join ClientePLSQL cli**

**on cli.ID\_CLIENTE = al.ID\_CLIENTE**

**inner join AutoPLSQL au**

**on au.ID\_AUTO = al.ID\_AUTO;**

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

**cli.nombre, su.\***

**from ReservaPLSQL re**

**inner join SucursalPLSQL su**

**on su.ID\_SUCURSAL = re.ID\_SUCURSAL**

**inner join ClientePLSQL cli**

**on re.ID\_CLIENTE = cli.ID\_CLIENTE**

**where su.ID\_SUCURSAL = 588;**

* + Mostrar los autos que han sido alquilados junto con los nombres de los clientes.
* **select**

**au.\*, cli.nombre**

**from AutoPLSQL au**

**inner join AlquilerPLSQL al**

**on al.id\_auto = au.id\_auto**

**inner join ClientePLSQL cli**

**on cli.id\_cliente = al.id\_cliente;**

* + Mostrar los detalles de las reservas con los nombres de los clientes y las ciudades de las sucursales.
    - * **select**

**re.\*, cli.nombre, su.ciudad**

**from ReservaPLSQL re**

**inner join ClientePLSQL cli on cli.id\_cliente = re.id\_cliente**

**inner join SucursalPLSQL su on su.id\_sucursal = re.id\_sucursal;**

* + Mostrar los clientes que no han realizado ninguna reserva.
    - * **select**

**cli.nombre, cli.id\_cliente**

**from ClientePLSQL cli**

**left join ReservaPLSQL re on re.id\_cliente = cli.id\_cliente**

**where re.id\_cliente is null;**

1. **Agregación y Agrupamiento:**
   * Contar cuántos autos hay de cada marca en la tabla "Auto".
     + - **select**

**marca,count(id\_auto) as no\_autos**

**from AutoPLSQL**

**group by marca;**

* + Calcular la duración promedio de los alquileres.
    - * **select**

**round(**

**avg(**

**TO\_NUMBER(TO\_DATE(fecha\_inicio, 'YYYY-MM-DD') -TO\_DATE(fecha\_fin,'YYYY-MM-DD'))),0)**

**as avg\_duracion\_alquiler**

**from AlquilerPLSQL;**

“La duración promedio de es 9 dias”

* + Mostrar el número total de reservas realizadas en cada sucursal.
    - * **select**

**id\_sucursal,count(id\_reserva) as no\_reservas**

**from ReservaPLSQL**

**group by id\_sucursal;**

* + Encontrar el cliente que ha realizado la mayor cantidad de alquileres.
    - * **select**

**cli.nombre, count(al.id\_alquiler) as no\_alquileres**

**from AlquilerPLSQL al**

**inner join ClientePLSQL cli on cli.id\_cliente = al.id\_cliente**

**group by cli.nombre**

**order by count(al.id\_alquiler) desc**

**fetch first 1 row only;**

“El cliente con mayor número de alquileres es John Greene”

* + Calcular el promedio de años de los autos en la tabla "Auto".
    - * **select**

**round(avg(ano),0) as promedio\_año**

**from AutoPLSQL;**

**“**El año promedio de los autos es 2006”

1. **Subconsultas**:
   * Mostrar los clientes que han realizado al menos una reserva.
     + - **select**

**cli.id\_cliente,**

**cli.nombre,count(re.id\_reserva) as no\_reservas**

**from ReservaPLSQL re**

**inner join ClientePLSQL cli on cli.id\_cliente = re.id\_cliente**

**group by cli.id\_cliente, cli.nombre**

**order by count(re.id\_reserva) asc;**

* + Mostrar los autos que no han sido alquilados aún.
    - * **SELECT au.ID\_AUTO, au.marca**

**FROM AutoPLSQL au**

**LEFT JOIN AlquilerPLSQL al ON au.ID\_AUTO = al.ID\_AUTO**

**WHERE al.ID\_AUTO IS NULL;**

* + Encontrar los clientes que han alquilado el mismo auto más de una vez.
    - * **SELECT**

**cli.ID\_CLIENTE,**

**cli.nombre,**

**al.ID\_AUTO,**

**COUNT(\*) AS veces\_alquilado**

**FROM AlquilerPLSQL al**

**INNER JOIN ClientePLSQL cli ON al.ID\_CLIENTE = cli.ID\_CLIENTE**

**GROUP BY cli.ID\_CLIENTE, cli.nombre, al.ID\_AUTO**

**HAVING COUNT(\*) > 1;**

* + Mostrar los clientes que han realizado alquileres en la misma ciudad en la que viven.
    - * **SELECT**

**c.id\_cliente, c.nombre,**

**SUBSTR(c.direccion, INSTR(c.direccion, ', ') + 2, INSTR(c.direccion, ', ', 1, 2) - INSTR(c.direccion, ', ') - 2) AS ciudad\_residencia,**

**a.ciudad AS ciudad\_alquila**

**FROM ClientePLSQL c**

**INNER JOIN AlquilerPLSQL al ON c.id\_cliente = al.id\_cliente**

**INNER JOIN ReservaPLSQL re ON re.id\_reserva = al.id\_reserva**

**INNER JOIN SucursalPLSQL a ON re.id\_sucursal = a.id\_sucursal**

**WHERE SUBSTR(c.direccion, INSTR(c.direccion, ', ') + 2, INSTR(c.direccion, ', ', 1, 2) - INSTR(c.direccion, ', ') - 2) = a.ciudad;**

* + Encontrar los autos que han sido alquilados en la misma sucursal donde se realizó una reserva.

1. **Actualizaciones y Eliminaciones:**
   * Actualizar la dirección de un cliente específico.
     + - **UPDATE ClientePLSQL**

**SET direccion =**

**'9409 Alvarado Ramp, Grossbury, Kentucky 37569'**

**WHERE id\_cliente = 625;**

* + Eliminar un auto de la tabla "Auto".
    - * **DELETE FROM AutoPLSQL**

**WHERE id\_auto = 2;**

* + Marcar una reserva como completada actualizando la fecha de fin.
    - * **UPDATE AlquilerPLSQL**

**SET fecha\_fin = TO\_DATE('2023-05-01', 'YYYY-MM-DD')**

**WHERE id\_reserva = 585;**

* + Eliminar todas las reservas realizadas por un cliente específico.
    - * **DELETE FROM ReservaPLSQL**

**WHERE id\_cliente = 863;**

* + Actualizar el año de un auto en la tabla "Auto".
    - * **UPDATE AutoPLSQL**

**SET ano = 2020**

**WHERE id\_auto =3;**

**EJERCICIOS SEGUNDO CICLO (31-80):**

* SELECT \* FROM ClientePLSQL;

**Trae toda la información de los clientes.**

* SELECT \* FROM AutoPLSQL;

**Trae toda la información de los autos.**

* SELECT \* FROM AlquilerPLSQL;

**Trae toda la información de los alquilados.**

* SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id\_cliente = a.id\_cliente;

**Trae los datos de nombre, marca, modelo encontrada de los clientes qu realizaron un alquiler.**

* SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a JOIN AlquilerPLSQL al ON a.id\_auto = al.id\_auto;

**Trae toda la información de marca, modelo y año de todos los autos alquilados.**

* SELECT \* FROM AlquilerPLSQL WHERE id\_cliente = 1;

**Trae la información de los autos alquilados al cliente 1.**

* SELECT \* FROM AlquilerPLSQL WHERE id\_auto = 1;

**Trae toda la información del alquiler del auto 1.**

* SELECT \* FROM AlquilerPLSQL WHERE id\_sucursal = 1;

**Trae toda la información de los alquileres de la sucursal 1.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_inicio = '2023-09-27';

**Traer todos los alquileres de la fecha 27/09/2023.**

* SELECT COUNT(\*) FROM AlquilerPLSQL;

**Trae la cantidad de alquileres que se han realizado en el historico.**

* 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';

**Trae los clientes que han alquilado 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';

**Trae la información de marca y modelo que ha alquilado el cliente 1 para el dia 27/09/2023.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_fin - fecha\_inicio > 7;

**Trae los alquileres que hayan durado más de 7 días.**

* 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;

**Trae el cliente que más veces a alquilado autos en la compañía.**

* 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;

**Trae la marca y el modelo que más veces se ha alquilado en el negocio.**

* 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;

**Trae el nombre del cliente que más veces ha alquilado.**

* 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;

**Traer el mes en el que mas alquileres se han hecho.**

* 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;

**Trae el dia de la semana en el que más veces se ha alquilado.**

* SELECT \* FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;

**Trae el valor más alto por el cual se haya alquilado un auto.**

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

**Trae el valor más bajo por el cual se haya alquilado un auto.**

* SELECT \* FROM ClientePLSQL WHERE nombre LIKE '%Juan%';

**Trae todos los clientes que se llamen juan**

* SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000;

**Trae la marca, el modelo y el año del auto cuyo precio haya sido inferior a 10000.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30';

**Trae todos los alquileres que se hayan hecho en septiembre del 2023.**

* 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%';

**Trae todos los clientes que han alquilado junto con la informacion de marca y modelo, que viven en bogotá.**

* 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;

**Trae la información de marca, modelo y año para la primera reserva que se realizó.**

* SELECT \* FROM AlquilerPLSQL WHERE id\_cliente IN (1, 2, 3); SELECT \* FROM AlquilerPLSQL WHERE id\_auto IN (1, 2, 3);

**Trae todos los alquileres que se le hicieron a los carros con id 1,2,3.**

* SELECT \* FROM AlquilerPLSQL WHERE id\_sucursal IN (1, 2, 3);

**Trae todos los alquileres que se realizaron en la sucursal 1,2,3.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id\_cliente IN (1, 2, 3);

**Trae todos los alquileres que se realizaron en septiembre para los clientes 1,2,3.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id\_auto IN (1, 2, 3);

**Trae todos los alquileres en el mes de septiembre para los autos 1,2,3.**

* SELECT \* FROM AlquilerPLSQL WHERE fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND id\_sucursal IN (1, 2, 3);

**Trae todos los alquileres que se realizaron en septiembre para las sucursales 1,2,3.**

* 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;

**Trae el nombre del cliente con mayor número 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;

**Trae la marca, el modelo y el número de veces que se alquiló el carro más frecuentado por los usuarios.**

* 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;

**Trae el cliente con mayor número del alquiler.**

* 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;

**Trae el mes con mayor número de 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;

**Trae el dia de la semana en el que más alquileres se hace.**

* SELECT \* FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;

**Trae la información del alquiler más costoso.**

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

**Trae la información del alquiler más económico.**

* SELECT \* FROM ClientePLSQL WHERE nombre LIKE '%Juan%' AND fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30';

**Trae la información de los clientes que se llaman juan, los cuales tomaron en alquiler un auto en septiembre.**

* SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000 AND fecha\_inicio BETWEEN '2023-09-01' AND '2023-09-30';

**Trae la información de marca, modelo y año para los autos que se alquilaron por un valor inferior a 10000 y se alquilaron en el mes de septiembre del año 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';

**Crea una vista llamada vista\_clientes\_alquilados\_sucursal, donde se trae el nombre, la marca y el modelo del auto alquilado, esto desde la tabla pincipal ClientePLSQL generando una conexión con las tablas AlquilerPLSQL y SucursalPLSQL. De estas conexiones solo trae los datos en común con llaves como el id\_cliente y el id\_sucursal. Ya para finalizar filtra estos datos solo para la sucursal principal.**

* 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';

**Se crea una vista llamada vista\_autos\_alquilados\_cliente\_fecha, donde trae los datos como la marca y el modelo del auto alquilado, desde la tabla principal AutoPLSQL haciendo una conexión con AlquilerPLSQL con la llave id\_auto, filtrando no más la información para el cliente 1 y el día 27/09/2023.**

* CREATE VIEW vista\_alquileres\_mas\_7dias AS

SELECT \*

FROM AlquilerPLSQL

WHERE fecha\_fin - fecha\_inicio > 7;

**Se crea una vista denominada vista\_alquileres\_mas\_7\_dias, donde se trae toda la información de la tabla AlquilerPLSQL donde la fecha de alquiler sea mayor a 7 días (esto se refleja en la diferencia que calcula entre la fecha fin y la fecha de inicio.**

* 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;

**Se crea la visa clientes\_mas\_alquileres, donde trae el nombre del cliente y desde esta base genera un conteo del número de alquileres que tiene cada cliente, la agrupación se genera por cliente y va a ordenar los datos de manera descendente, lo que indica que en primer lugar tendremos al cliente que más alquileres ha tenido.**

* 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;

**Se crea una vista denominada vistas\_autos\_mas\_alquileres, donde se trae la información de marca, modelo y el número de alquileres (determinada por el conteo de registros en la base). Esta información proviene de la tabla AutoPLSQL, generando una conexión con la tabla AlquilerPLSQL donde la llave es el id\_auto. La información vendrá agrupada por marca y modelo y se organizará de manera 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;

**Se crea una vista denominada sucursales\_mas\_alquileres, donde viene información de las sucursales tales como el nombre y el conteo de alquileres realizados por cada sucursal, vendrá organizada en orden descendente por el número de alquileres realizados.**

* 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;

**Se crea una vista denomindad meses\_mas\_alquileres, donde se trae información como el mes y el número de alquileres realizados por cada mes, la información proviene de la tabla AlquilerPLSQL, agrupada por meses y ordenada de mayor a menor por número 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;

**Se crea una vista denominada dias\_semana\_mas\_alquileres, donde se discrimina por día de la semana el número de alquileres que se hace en cada día, la información proviene de la tabla AlquilerPLSQL, agrupado por día de la semana y organizadola de mayor a menor número de alquileres.**

* CREATE VIEW vista\_alquileres\_mas\_caros AS

SELECT \*

FROM AlquilerPLSQL

ORDER BY precio DESC;

**Se crea una vista denominada alquileres\_mas\_caros, donde me trae toda la información de la tabla de AlquileresPLSQL, ordenada de mayor a menor precio del alquiler.**

* CREATE VIEW vista\_alquileres\_mas\_baratos AS

SELECT \*

FROM AlquilerPLSQL

ORDER BY precio ASC;

**Se crea una vista denominada alquileres\_mas\_baratos, donde se trae toda la información de la tabla AlquilerPLSQL, ordenada de menor a mayor precio de alquiler.**

**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;  **El objetivo de este trigger es mantener actualizado el número de autos disponibles en la tabla "AutoPLSQL." Cuando alguien intenta agregar información sobre un nuevo automóvil en la tabla, queremos asegurarnos de que el número de autos disponibles se ajuste correctamente**  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;  **Este trigger se asegura de que cuando se elimina un registro que representa un automóvil de la base de datos, la cantidad de autos disponibles en esa categoría se ajusta automáticamente. Esto es útil para mantener actualizado el inventario de automóviles disponibles después de eliminar registros en la base de datos.**  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;  **Este trigger se asegura de que cuando se modifica la cantidad de autos disponibles al actualizar un registro en la base de datos, se refleje correctamente en la tabla. Esto ayuda a mantener actualizado el inventario de automóviles disponibles cuando se realizan cambios en la información de los registros.**  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;  **Este trigger se asegura de que cuando se registra un nuevo cliente en la base de datos, el número total de clientes se actualice automáticamente. Esto es útil para llevar un seguimiento de la cantidad de clientes registrados en la tabla "ClientePLSQL."**  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;  **Este trigger se asegura de que cuando se elimina un cliente de la base de datos, el número total de clientes se actualice automáticamente. Esto es útil para llevar un seguimiento preciso de la cantidad de clientes registrados en la tabla "ClientePLSQL."**  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;  **Este trigger se asegura de que cuando se actualiza el número de alquileres realizados por un cliente en la base de datos, la información se refleje correctamente en la tabla. Esto es útil para llevar un seguimiento preciso de la cantidad de alquileres realizados por cada cliente en la tabla "ClientePLSQL."** |
| 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;  **Obtiene el precio base del automóvil a partir de la tabla "AutoPLSQL" usando el "id\_auto" proporcionado.**  **Calcula cuántos días dura el alquiler restando la "fecha\_inicio" de la "fecha\_fin" y luego sumando 1 para incluir el último día.**  **Multiplica el precio base por el número de días de alquiler para determinar el precio total del alquiler.**  **El resultado se almacena en una columna llamada "precio."**  **Este procedimiento hace los cálculos necesarios para determinar cuánto costará alquilar un automóvil durante un período específico, basándose en el precio base del automóvil y las fechas de inicio y fin 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;  **Realiza una consulta (SELECT) en la tabla "AlquilerPLSQL."**  **La consulta busca todas las filas en la tabla "AlquilerPLSQL" donde el valor de la columna "id\_cliente" coincide con el valor proporcionado en el parámetro "id\_cliente."**  **La consulta selecciona y muestra todas las columnas y registros que cumplen con la condición.**  **En resumen, este procedimiento se utiliza para obtener una lista de todos los alquileres asociados a un cliente específico en la base de datos. Al proporcionar el "id\_cliente" como entrada, se obtendrán todos los registros de alquiler relacionados con 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;  **Ejecuta una consulta (SELECT) en la tabla "AutoPLSQL."**  **La consulta busca todas las filas en la tabla "AutoPLSQL" donde el valor de la columna "id\_sucursal" coincide con el valor proporcionado en el parámetro "id\_sucursal."**  **La consulta selecciona y muestra todas las columnas y registros que cumplen con esta condición.**  **Este procedimiento permite obtener una lista de todos los automóviles asociados a una sucursal específica en la base de datos. Al proporcionar el "id\_sucursal" como entrada, se obtendrán todos los registros de automóviles relacionados con 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;  **Utiliza una instrucción "INSERT INTO" para agregar una nueva fila (registro) en la tabla "AutoPLSQL."**  **Los valores proporcionados en los parámetros se insertan en las columnas correspondientes de la tabla ("marca," "modelo," "ano," "numero\_disponibles").**  **Este procedimiento permite agregar un nuevo automóvil a la base de datos. Al proporcionar los detalles del automóvil a través de los parámetros de entrada, se crea un nuevo registro en la tabla "AutoPLSQL" con esa información.**  CREATE PROCEDURE proc\_eliminar\_auto  (  IN id\_auto INT  )  AS  BEGIN  -- Eliminar un auto  DELETE FROM AutoPLSQL  WHERE id\_auto = id\_auto;  END;  **Utiliza una instrucción "DELETE FROM" para eliminar una fila (registro) de la tabla "AutoPLSQL."**  **La condición "WHERE" en la instrucción busca el registro en la tabla donde el valor de la columna "id\_auto" coincide con el valor proporcionado en el parámetro "id\_auto."**  **Este procedimiento permite eliminar un automóvil específico de la base de datos. Al proporcionar el "id\_auto" como entrada, se elimina el registro correspondiente en la tabla "AutoPLSQL."** |