

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)
);

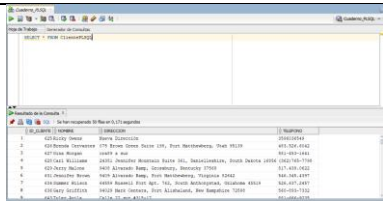
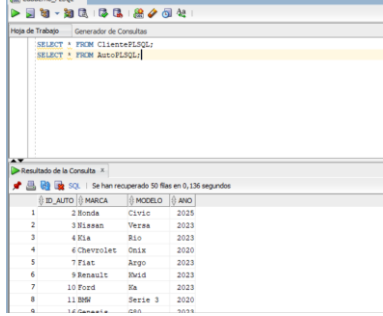
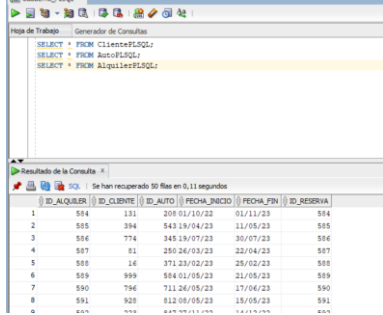
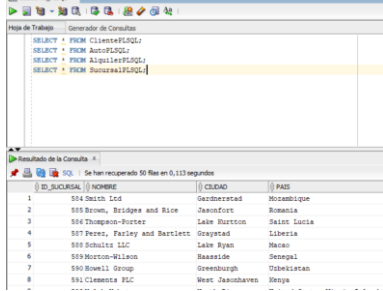
```

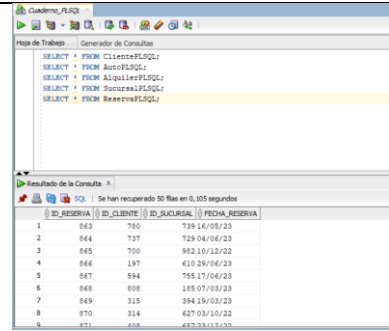
- Cliente: Almacena información sobre los clientes, como su nombre, dirección y número de teléfono.
- Auto: Almacena información sobre los autos, como su marca, modelo y año.
- Alquiler: Almacena información sobre los alquileres, como la fecha de inicio, la fecha de finalización y el auto alquilado.
- 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.

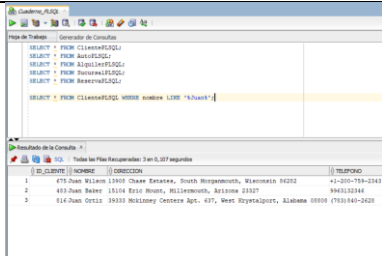
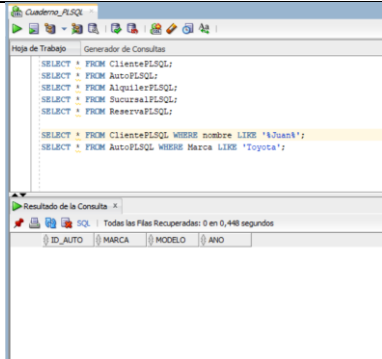
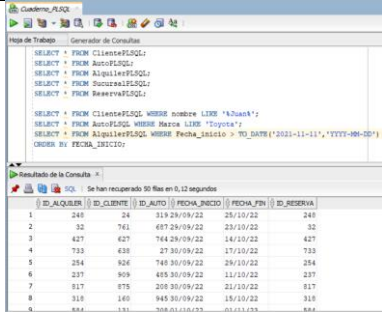
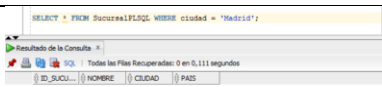
EJERCICIOS PRIMER CICLO (1-30):

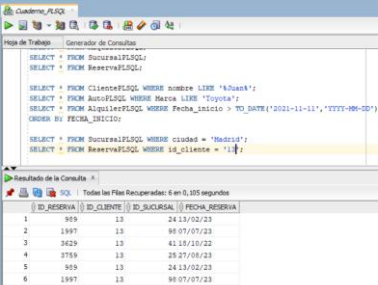
1. Consultas Básicas:

Consulta	Sentencia	Resultado	Descripción
Mostrar todos los clientes en la tabla "Cliente".	SELECT * FROM ClientePLSQL;		La consulta trae todos los registros de la tabla clientePLSQL
Mostrar todos los autos en la tabla "Auto".	SELECT * FROM AutoPLSQL;		La consulta trae todos los registros de la tabla AutoPLSQL
Mostrar todos los alquileres en la tabla "Alquiler".	SELECT * FROM AlquilerPLSQL;		La consulta trae todos los registros de la tabla AlquilerPLSQL
Mostrar todas las sucursales en la tabla "Sucursal".	SELECT * FROM SucursalPLSQL;		La consulta trae todos los registros de la tabla SucursalPLSQL

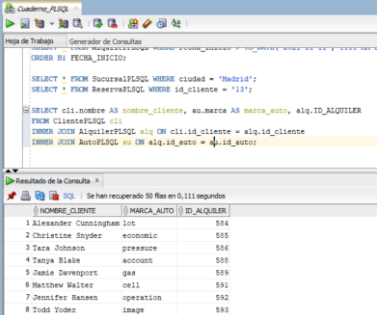
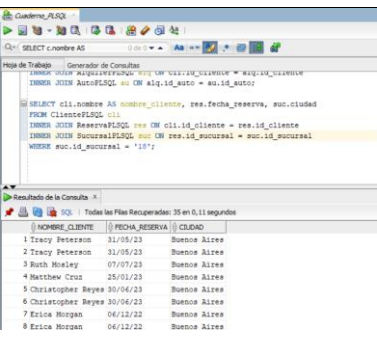
Mostrar todas las reservas en la tabla "Reserva".	<pre>SELECT * FROM ReservaPLSQL;</pre>		La consulta trae todos los registros de la tabla ReservaPLSQL
---	--	--	---

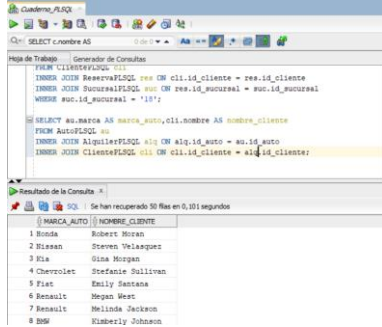
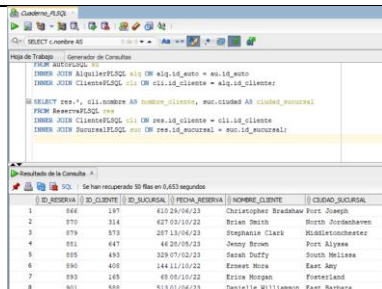
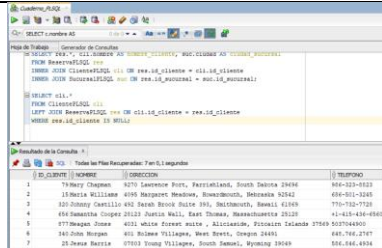
2. Filtros y Ordenamiento:

Consulta	Sentencia	Resultado	Descripción
Mostrar los clientes que se llaman "Juan".	<pre>SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%';</pre>		Con la expresión where y like, se logra filtrar todos los clientes que en el campo nombre contengan la palabra Juan.
Mostrar los autos de marca "Toyota".	<pre>SELECT * FROM AutoPLSQL WHERE Marca LIKE 'Toyota';</pre>		Con la expresión where y like, se filtra todos los autos cuya marca contenga la palabra Toyota. La respuesta no trae resultados ya que ningún auto tiene esa marca.
Mostrar los alquileres que ocurrieron después de una fecha específica.	<pre>SELECT * FROM AlquilerPLSQL WHERE Fecha_inicio > TO_DATE('2021-11-11','YYYY-MM-DD') ORDER BY FECHA_INICIO;</pre>		La consulta muestra todos los alquileres que hayan sido realizados luego del 11 de noviembre del 2021 y los ordena por su fecha de inicio.
Mostrar las sucursales ubicadas en "Madrid".	<pre>SELECT * FROM SucursalPLSQL WHERE ciudad = 'Madrid';</pre>		Con la expresión where, e buscan todas las sucursales que contengan la palabra Madrid

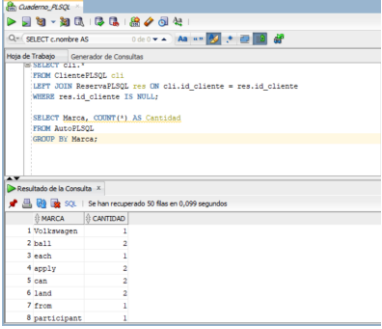
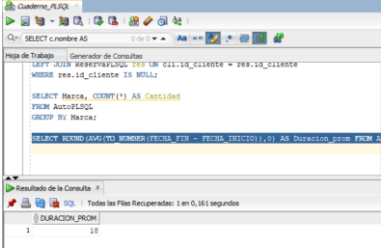
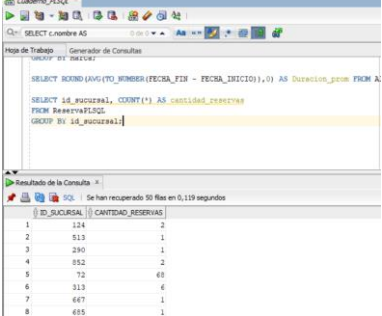
			en el campo ciudad. La respuesta no trae resultados ya que ninguna sucursal tiene esa ciudad.
Mostrar las reservas realizadas por un cliente específico.	SELECT * FROM ReservaPLSQL WHERE id_cliente = '13';		La consulta muestra todas las reservas realizadas por el id_cliente '13'

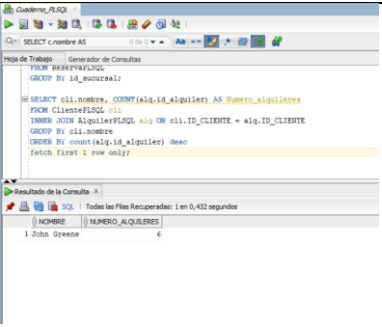
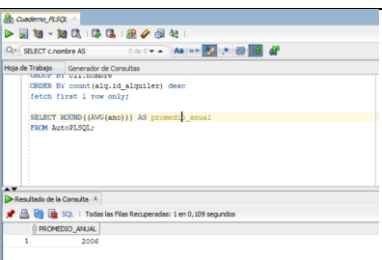
3. Join y Relaciones:

Consulta	Sentencia	Resultado	Descripción
Mostrar los alquileres con los nombres de los clientes y las marcas de los autos.	SELECT cli.nombre AS nombre_cliente, au.marca AS marca_auto, alq.ID_ALQUILER FROM ClientePLSQL cli INNER JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.id_cliente INNER JOIN AutoPLSQL au ON alq.id_auto = au.id_auto;		Esta consulta combina las tablas alquiler, auto y cliente a través de las expresiones inner join. Se trae como respuesta el nombre del cliente, la marca del auto y el id del alquiler.
Mostrar los clientes que han realizado reservas en una sucursal específica.	SELECT cli.nombre AS nombre_cliente, res.fecha_reserva, suc.ciudad FROM ClientePLSQL cli INNER JOIN ReservaPLSQL res ON cli.id_cliente = res.id_cliente INNER JOIN SucursalPLSQL suc ON res.id_sucursal = suc.id_sucursal WHERE suc.id_sucursal = '18';		Esta consulta combina las tablas cliente, sucursal y reserva a través de la expresión inner join. Se adiciona un filtro con la expresión where donde especifica la sucursal de búsqueda.

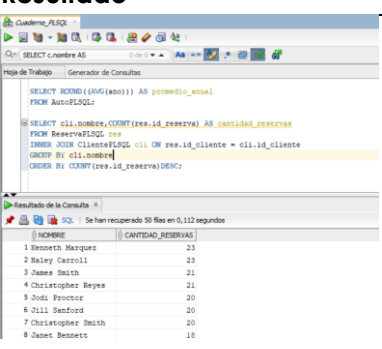
Mostrar los autos que han sido alquilados junto con los nombres de los clientes.	<pre>SELECT au.marca AS marca_auto,cli.nombre AS nombre_cliente FROM AutoPLSQL au INNER JOIN AlquilerPLSQL alq ON alq.id_auto = au.id_auto INNER JOIN ClientePLSQL cli ON cli.id_cliente = alq.id_cliente;</pre>	 <table><thead><tr><th></th><th>MARCA_AUTO</th><th>NOMBRE_CLIENTE</th></tr></thead><tbody><tr><td>1</td><td>Honda</td><td>Robert Moran</td></tr><tr><td>2</td><td>Blaaaan</td><td>Steven Velasquez</td></tr><tr><td>3</td><td>Fila</td><td>Gina Morgan</td></tr><tr><td>4</td><td>Chevrolet</td><td>Stedanie Sullivan</td></tr><tr><td>5</td><td>Fiat</td><td>Emily Santana</td></tr><tr><td>6</td><td>Renault</td><td>Negan West</td></tr><tr><td>7</td><td>Renault</td><td>Nelinda Jackson</td></tr><tr><td>8</td><td>Bmw</td><td>Kimberly Johnson</td></tr></tbody></table>		MARCA_AUTO	NOMBRE_CLIENTE	1	Honda	Robert Moran	2	Blaaaan	Steven Velasquez	3	Fila	Gina Morgan	4	Chevrolet	Stedanie Sullivan	5	Fiat	Emily Santana	6	Renault	Negan West	7	Renault	Nelinda Jackson	8	Bmw	Kimberly Johnson	Esta consulta une la tabla auto, alquiler y cliente a través de la expresión inner join con los id de auto y id de cliente. Da como respuesta el nombre del cliente y la marca del auto alquilado.																											
	MARCA_AUTO	NOMBRE_CLIENTE																																																							
1	Honda	Robert Moran																																																							
2	Blaaaan	Steven Velasquez																																																							
3	Fila	Gina Morgan																																																							
4	Chevrolet	Stedanie Sullivan																																																							
5	Fiat	Emily Santana																																																							
6	Renault	Negan West																																																							
7	Renault	Nelinda Jackson																																																							
8	Bmw	Kimberly Johnson																																																							
Mostrar los detalles de las reservas con los nombres de los clientes y las ciudades de las sucursales.	<pre>SELECT res.*, cli.nombre AS nombre_cliente, suc.ciudad AS ciudad_sucursal FROM ReservaPLSQL res INNER JOIN ClientePLSQL cli ON res.id_cliente = cli.id_cliente INNER JOIN SucursalPLSQL suc ON res.id_sucursal = suc.id_sucursal;</pre>	 <table><thead><tr><th></th><th>ID_RESERVA</th><th>ID_CLIENTE</th><th>ID_SUCURSAL</th><th>NOMBRE_CLIENTE</th><th>Ciudad_SUCURSAL</th></tr></thead><tbody><tr><td>1</td><td>568</td><td>137</td><td>412/26/04/23</td><td>Christopher Bradshaw</td><td>Port Joseph</td></tr><tr><td>2</td><td>870</td><td>314</td><td>427/03/10/22</td><td>Brian Smith</td><td>North Jordanhaven</td></tr><tr><td>3</td><td>879</td><td>879</td><td>287/10/04/23</td><td>Stephane Clark</td><td>MaldenWestminster</td></tr><tr><td>4</td><td>881</td><td>447</td><td>44/28/08/23</td><td>Jenny Brown</td><td>Port Alayne</td></tr><tr><td>5</td><td>885</td><td>493</td><td>329/07/02/23</td><td>Sarah Duffy</td><td>South Melrose</td></tr><tr><td>6</td><td>890</td><td>400</td><td>144/11/10/22</td><td>Ernest More</td><td>East Jay</td></tr><tr><td>7</td><td>893</td><td>148</td><td>40/06/10/22</td><td>Ernie Morgan</td><td>Porterland</td></tr><tr><td>8</td><td>901</td><td>588</td><td>512/01/04/23</td><td>Danielle Williamson</td><td>East Bathurst</td></tr></tbody></table>		ID_RESERVA	ID_CLIENTE	ID_SUCURSAL	NOMBRE_CLIENTE	Ciudad_SUCURSAL	1	568	137	412/26/04/23	Christopher Bradshaw	Port Joseph	2	870	314	427/03/10/22	Brian Smith	North Jordanhaven	3	879	879	287/10/04/23	Stephane Clark	MaldenWestminster	4	881	447	44/28/08/23	Jenny Brown	Port Alayne	5	885	493	329/07/02/23	Sarah Duffy	South Melrose	6	890	400	144/11/10/22	Ernest More	East Jay	7	893	148	40/06/10/22	Ernie Morgan	Porterland	8	901	588	512/01/04/23	Danielle Williamson	East Bathurst	Esta consulta muestra todos los datos de la tabla reserva, y añade el nombre del cliente y la ciudad de la sucursal usando inner join de las tablas cliente, reserva y sucursal.
	ID_RESERVA	ID_CLIENTE	ID_SUCURSAL	NOMBRE_CLIENTE	Ciudad_SUCURSAL																																																				
1	568	137	412/26/04/23	Christopher Bradshaw	Port Joseph																																																				
2	870	314	427/03/10/22	Brian Smith	North Jordanhaven																																																				
3	879	879	287/10/04/23	Stephane Clark	MaldenWestminster																																																				
4	881	447	44/28/08/23	Jenny Brown	Port Alayne																																																				
5	885	493	329/07/02/23	Sarah Duffy	South Melrose																																																				
6	890	400	144/11/10/22	Ernest More	East Jay																																																				
7	893	148	40/06/10/22	Ernie Morgan	Porterland																																																				
8	901	588	512/01/04/23	Danielle Williamson	East Bathurst																																																				
Mostrar los clientes que no han realizado ninguna reserva.	<pre>SELECT cli.* FROM ClientePLSQL cli LEFT JOIN ReservaPLSQL res ON cli.id_cliente = res.id_cliente WHERE res.id_cliente IS NULL;</pre>	 <table><thead><tr><th></th><th>ID_CLIENTE</th><th>NOMBRE</th><th>DIRECCION</th><th>TELEFONO</th></tr></thead><tbody><tr><td>1</td><td>137</td><td>Vibrey Chapman</td><td>4070 Lawrence Pkwy, Fayetteville, South Dakota 58706</td><td>605-333-0323</td></tr><tr><td>2</td><td>15</td><td>Maria Williams</td><td>4095 Margaret Meadows, Roundabout, Nebraska 68142</td><td>408-911-5245</td></tr><tr><td>3</td><td>314</td><td>Sherry Cavallaro</td><td>401 South Brook Ridge WY, Westminster, Hawaii 61069</td><td>408-911-5245</td></tr><tr><td>4</td><td>493</td><td>Emmanuel Cooper</td><td>21033 Justice Hall, East Omaha, Massachusetts 01108</td><td>770-510-7738</td></tr><tr><td>5</td><td>447</td><td>William Jones</td><td>4011 White Street Mills - Alcatraz, Picoletto Island 37049</td><td>415-428-4845</td></tr><tr><td>6</td><td>148</td><td>John Morgan</td><td>401 Melrose Village, West River, Oregon 97465</td><td>503-366-2747</td></tr><tr><td>7</td><td>28</td><td>Debra Burtis</td><td>07013 Trump Village, South Dakota, Wyoming 39049</td><td>505-366-4534</td></tr></tbody></table>		ID_CLIENTE	NOMBRE	DIRECCION	TELEFONO	1	137	Vibrey Chapman	4070 Lawrence Pkwy, Fayetteville, South Dakota 58706	605-333-0323	2	15	Maria Williams	4095 Margaret Meadows, Roundabout, Nebraska 68142	408-911-5245	3	314	Sherry Cavallaro	401 South Brook Ridge WY, Westminster, Hawaii 61069	408-911-5245	4	493	Emmanuel Cooper	21033 Justice Hall, East Omaha, Massachusetts 01108	770-510-7738	5	447	William Jones	4011 White Street Mills - Alcatraz, Picoletto Island 37049	415-428-4845	6	148	John Morgan	401 Melrose Village, West River, Oregon 97465	503-366-2747	7	28	Debra Burtis	07013 Trump Village, South Dakota, Wyoming 39049	505-366-4534	Esta consulta trae las columnas de la tabla clientepsql, a través del left join con la tabla reserva, y con el where puede identificar aquellos clientes que están solo en la tabla cliente y no en la tabla reserva.														
	ID_CLIENTE	NOMBRE	DIRECCION	TELEFONO																																																					
1	137	Vibrey Chapman	4070 Lawrence Pkwy, Fayetteville, South Dakota 58706	605-333-0323																																																					
2	15	Maria Williams	4095 Margaret Meadows, Roundabout, Nebraska 68142	408-911-5245																																																					
3	314	Sherry Cavallaro	401 South Brook Ridge WY, Westminster, Hawaii 61069	408-911-5245																																																					
4	493	Emmanuel Cooper	21033 Justice Hall, East Omaha, Massachusetts 01108	770-510-7738																																																					
5	447	William Jones	4011 White Street Mills - Alcatraz, Picoletto Island 37049	415-428-4845																																																					
6	148	John Morgan	401 Melrose Village, West River, Oregon 97465	503-366-2747																																																					
7	28	Debra Burtis	07013 Trump Village, South Dakota, Wyoming 39049	505-366-4534																																																					

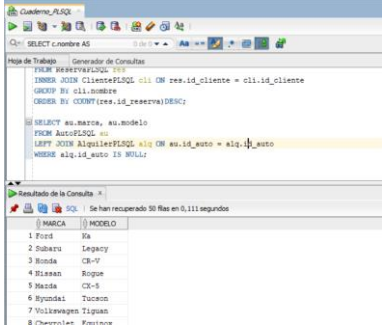
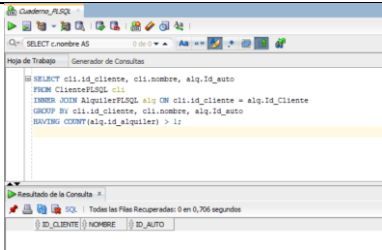
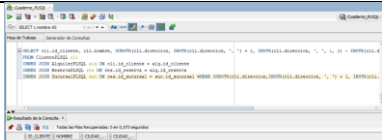
4. Agregación y Agrupamiento:

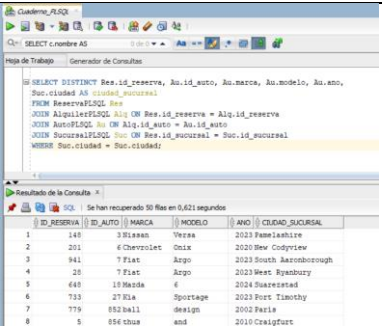
Consulta	Sentencia	Resultado	Descripción
Contar cuántos autos hay de cada marca en la tabla "Auto".	SELECT Marca, COUNT(*) AS Cantidad FROM AutoPLSQL GROUP BY Marca;		Esta consulta selecciona la columna marca de la tabla auto, cuenta cuantos autos hay para cada marca utilizando la función count y finalmente agrupa los resultados por la marca mediante group by.
Calcular la duración promedio de los alquileres.	SELECT ROUND(AVG(TO_NUMBER(FECHA_FIN - FECHA_INICIO)),0) AS Duracion_prom FROM AlquilerPLSQL;		Esta consulta usa la tabla alquiler y calcula la duración de cada alquiler restando la fecha fin de la fecha de inicio. Luego convierte esa diferencia en numero y calcula el promedio total.
Mostrar el número total de reservas realizadas en cada sucursal.	SELECT id_sucursal, COUNT(*) AS cantidad_reservas FROM ReservaPLSQL GROUP BY id_sucursal;		Esta consulta toma la tabla reserva y extrae el dato id_sucursal, además se hace un conteo de las reservas y las agrupa por id_sucursal en la columna cantidad_reserv a

Encontrar el cliente que ha realizado la mayor cantidad de alquileres.	<pre> SELECT cli.nombre, COUNT(alq.id_alquiler) AS Numero_alquileres FROM ClientePLSQL cli INNER JOIN AlquilerPLSQL alq ON cli.ID_CLIENTE = alq.ID_CLIENTE GROUP BY cli.nombre ORDER BY count(alq.id_alquiler) desc fetch first 1 row only; </pre>		Esta consulta trae las columnas nombre y numero de alquiler que se crea a partir del conteo del id_alquiler. Luego realiza un inner join entre las tablas cliente y alquiler. Luego, agrupa los resultados por el nombre, los ordena por la cantidad de alquileres, y selecciona solo la primera fila, obteniendo así el cliente con el mayor número de alquileres.
Calcular el promedio de años de los autos en la tabla "Auto".	<pre> SELECT ROUND((AVG(ano))) AS promedio_anual FROM AutoPLSQL; </pre>		Esta consulta calcula el promedio del año de los autos a través de la función avg.

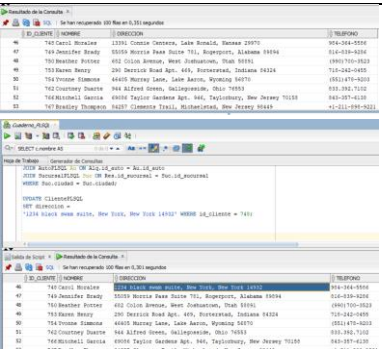

5. Subconsultas:

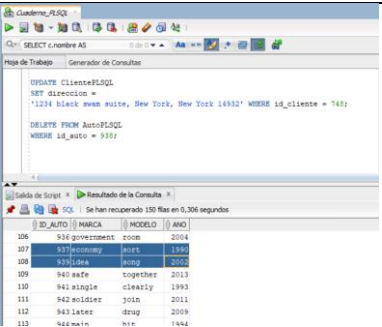
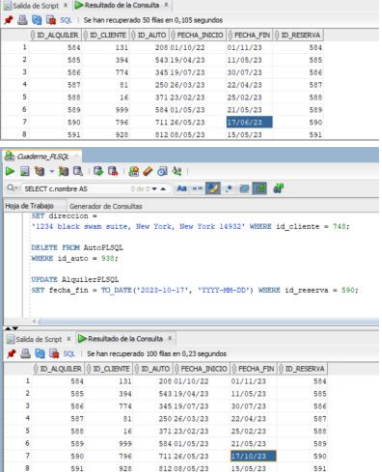
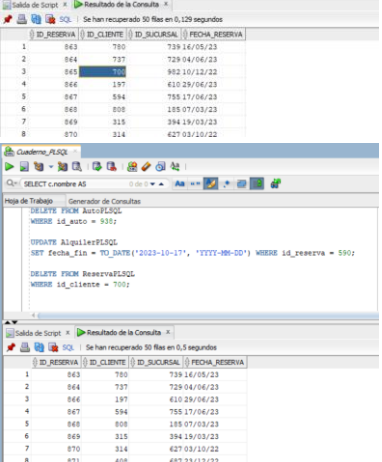
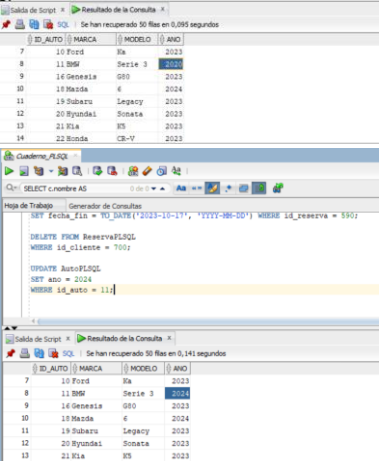
Consulta	Sentencia	Resultado	Descripción
Mostrar los clientes que han realizado al menos una reserva.	<pre> SELECT cli.nombre,COUNT(res.id_reserva) AS cantidad_reservas FROM ReservaPLSQL res INNER JOIN ClientePLSQL cli ON res.id_cliente = cli.id_cliente GROUP BY cli.nombre ORDER BY COUNT(res.id_reserva)DESC C; </pre>		Esta consulta selecciona el nombre del cliente y la cuenta de reservas realizadas por cada cliente. Utiliza un INNER JOIN entre las tablas reserva y cliente, agrupa los resultados por

			el nombre del cliente y los ordena por la cantidad de reservas
Mostrar los autos que no han sido alquilados aún.	<pre>SELECT au.marca, au.modelo FROM AutoPLSQL au LEFT JOIN AlquilerPLSQL alq ON au.id_auto = alq.id_auto WHERE alq.id_auto IS NULL;</pre>		Esta consulta selecciona las columnas marca y modelo de la tabla auto luego se realiza left join en las tablas alquiler y auto donde el alquiler id_auto este vacío
Encontrar los clientes que han alquilado el mismo auto más de una vez.	<pre>SELECT cli.id_cliente, cli.nombre, alq.Id_auto FROM ClientePLSQL cli INNER JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.Id_Cliente GROUP BY cli.id_cliente, cli.nombre, alq.Id_auto HAVING COUNT(alq.id_alquiler) > 1;</pre>		Esta consulta relaciona las tablas clientes y alquiler. Se utiliza la expresión having count para filtrar los resultados y seleccionar solo aquellos grupos donde el id_reserva es mayor que 1.
Mostrar los clientes que han realizado alquileres en la misma ciudad en la que viven.	<pre>SELECT cli.id_cliente, cli.nombre, SUBSTR(cli.direccion, INSTR(cli.direccion, ',') + 2, INSTR(cli.direccion, ', ', 1, 2) - INSTR(cli.direccion, ',') - 2) AS ciudad_residencia, suc.ciudad AS ciudad_alquila FROM ClientePLSQL cli INNER JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.id_cliente INNER JOIN ReservaPLSQL res ON res.id_reserva = alq.id_reserva INNER JOIN SucursalPLSQL suc ON res.id_sucursal = suc.id_sucursal WHERE SUBSTR(cli.direccion,INSTR(cli.direccion, ',') + 2, INSTR(cli.direccion, ', ', 1, 2)</pre>		Esta consulta muestra el id_cliente, nombre y la ciudad de residencia de un cliente con la ciudad de la sucursal del alquiler. Se realizan inner join en las tablas clientes, reserva, alquiler y sucursal. Luego, se aplica un filtro WHERE donde se extrae el nombre de la ciudad de la dirección del

	- INSTR(cli.direccion, ',') - 2) = suc.ciudad;		cliente por posición. El resultado es vacío porque no hay coincidencia entre clientes cuyo id_sucursal coincida con la ciudad extraída de sus direcciones.
Encontrar los autos que han sido alquilados en la misma sucursal donde se realizó una reserva.	<pre> SELECT DISTINCT Res.id_reserva, Au.id_auto, Au.marca, Au.modelo, Au.ano, Suc.ciudad AS ciudad_sucursal FROM ReservaPLSQL Res JOIN AlquilerPLSQL Alq ON Res.id_reserva = Alq.id_reserva JOIN AutoPLSQL Au ON Alq.id_auto = Au.id_auto JOIN SucursalPLSQL Suc ON Res.id_sucursal = Suc.id_sucursal WHERE Suc.ciudad = Suc.ciudad;</pre>		

6. Actualizaciones y Eliminaciones:

Consulta	Sentencia	Resultado	Descripción
Actualizar la dirección de un cliente específico.	<pre> UPDATE ClientePLSQL SET direccion = '1234 black swam suite, New York, New York 14932' WHERE id_cliente = 748;</pre>		Con esta actualización se modificó el campo dirección del registro id_cliente 748
Eliminar un auto de la tabla "Auto".	<pre> DELETE FROM AutoPLSQL WHERE id_auto = 938;</pre>		Con este delete se eliminó de la tabla auto el Id_auto 938

			
<p>Marcar una reserva como completa da actualizan do la fecha de fin.</p>	<p>UPDATE AlquilerPLSQL SET fecha_fin = TO_DATE('2023-10-17', 'YYYY-MM-DD') WHERE id_reserva = 590;</p>		<p>Con esta actualización, se modificó la fecha fin del id_reserva 590</p>
<p>Eliminar todas las reservas realizadas por un cliente específico</p>	<p>DELETE FROM ReservaPLSQL WHERE id_cliente = 700;</p>		<p>Con este delete, se eliminó de la tabla las reservas del cliente 700</p>
<p>Actualizar el año de un auto en la tabla "Auto".</p>	<p>UPDATE AutoPLSQL SET ano = 2024 WHERE id_auto = 11;</p>		<p>Con este update, se modificó el año del id_auto 11</p>

EJERCICIOS SEGUNDO CICLO (31-80):

Sentencia SELECT * FROM ClientePLSQL;	Resultado 	Explicación Esta consulta muestra las columnas id_cliente, nombre, dirección y teléfono de la tabla clientes.
SELECT * FROM AutoPLSQL;		Esta consulta muestra las columnas id_auto, marca, modelo y año de la tabla auto.
SELECT * FROM AlquilerPLSQL;		Esta consulta muestra las columnas id_alquiler, id_cliente, id_auto, fecha inicio, fecha fin y id_reserva de la tabla alquiler.
SELECT c.nombre, a.marca, a.modelo FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente;		Esta consulta muestra datos del nombre del cliente y datos del auto de alquileres realizados, presenta error debido a que no esta relacionado con la tabla auto, se ajusta el código de la siguiente forma SELECT cli.nombre, au.marca, au.modelo FROM ClientePLSQL cli JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.id_cliente JOIN AutoPLSQL au ON alq.id_auto = au.id_auto;

SELECT a.marca, a.modelo,
a.ano FROM AutoPLSQL a
JOIN AlquilerPLSQL al ON
a.id_auto = al.id_auto;

ID	MARCA	MODELO	ANO
1	Buick	Civica	2020
2	Buick	Thera	2020
3	Buick	Thera	2020
4	Fiat	Rio	2020
5	Fiat	Rio	2020
6	Chrysler	Dacia	2020
7	Chrysler	Dacia	2020
8	Fiat	Rio	2020

La siguiente consulta muestra la información de la tabla "auto" correspondiente a los vehículos que han sido objeto de alquiler. Esta relación se obtiene mediante un Join con la tabla "alquiler" a través del identificador "id_auto".

SELECT * FROM AlquilerPLSQL
WHERE id_cliente = 1;

ID_ALQUIL	ID_CLIENTE	ID_AUTO	FECHA_I...	FECHA_FIN	ID_RESER
-----------	------------	---------	------------	-----------	----------

La consulta busca todos los alquileres asociados al ID_Cliente 1. El resultado es nulo debido a que el id_cliente 1 no existe.

SELECT * FROM AlquilerPLSQL
WHERE id_auto = 1;

ID_ALQUIL	ID_CLIENTE	ID_AUTO	FECHA_I...	FECHA_FIN	ID_RESER
-----------	------------	---------	------------	-----------	----------

La consulta busca todos los alquileres asociados al ID_Auto 1. Aunque existe un auto con ese id, la tabla de alquiler no contiene registros con ese ID.

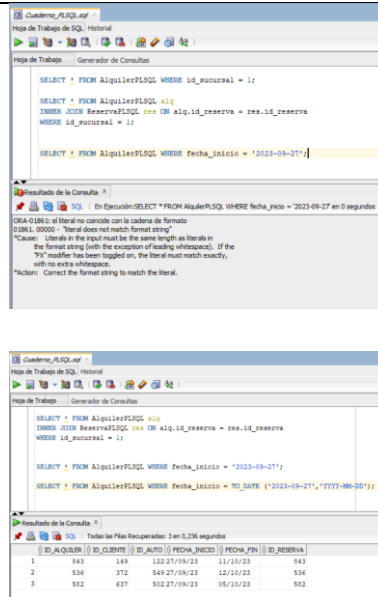
SELECT * FROM
AlquilerPLSQL WHERE
id_sucursal = 1;

ORA-00904: "ID_SUCURSAL": identificador no válido
ORA-00904: "ID_SUCURSAL": "No es un identificador"
"Causa":
Factor:
Error en la línea: 18, columna: 35

La consulta actual presenta un error ya que no existe una relación con el campo "id_sucursal" en la tabla de alquiler. El código para que funcione sería:

SELECT * FROM AlquilerPLSQL
alq
INNER JOIN ReservaPLSQL
res ON alq.id_reserva =
res.id_reserva
WHERE id_sucursal = 1;

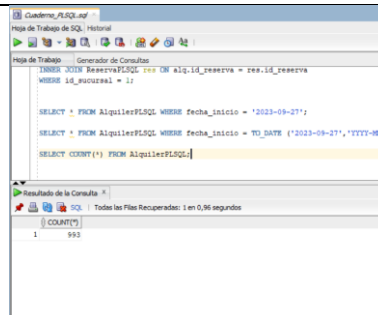
SELECT * FROM AlquilerPLSQL
WHERE fecha_inicio = '2023-09-27';



La consulta presenta un error debido a que no reconoce el texto dentro de las comillas para aplicar el filtro.

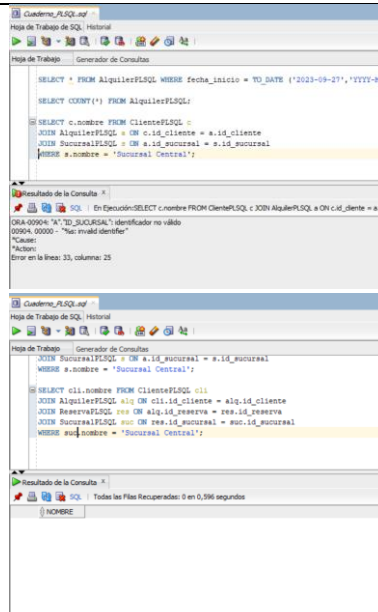
El código para que funcione sería:
SELECT * FROM AlquilerPLSQL
WHERE fecha_inicio = TO_DATE ('2023-09-27','YYYY-MM-DD');

SELECT COUNT(*) FROM
AlquilerPLSQL;



Esta consulta muestra el conteo de filas en la tabla alquiler.

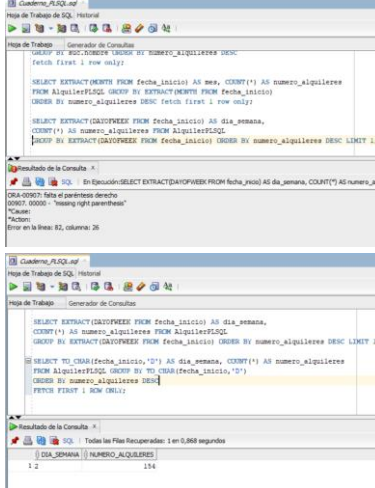
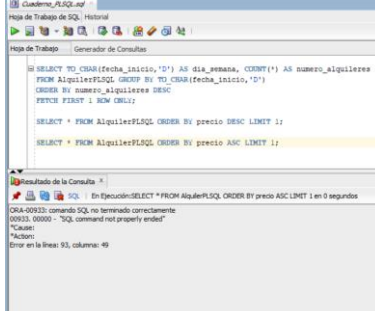
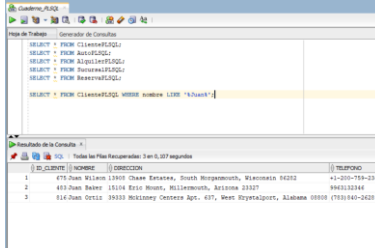
SELECT c.nombre FROM
ClientePLSQL c JOIN
AlquilerPLSQL a ON
c.id_cliente = a.id_cliente
JOIN SucursalPLSQL s ON
a.id_sucursal = s.id_sucursal
WHERE s.nombre = 'Sucursal
Central';



La consulta actual contiene un error debido a la falta de relación entre las tablas alquiler y sucursal, por lo cual se hará un join entre estas tablas:

SELECT cli.nombre FROM
ClientePLSQL cli
JOIN AlquilerPLSQL alq ON
cli.id_cliente = alq.id_cliente
JOIN ReservaPLSQL res ON
alq.id_reserva =
res.id_reserva
JOIN SucursalPLSQL suc ON
res.id_sucursal =
suc.id_sucursal
WHERE suc.nombre =
'Sucursal Central';

		
<p>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;</p>		<p>Se corrige el error de limit y se debe hacer un join adicional entre sucursal y alquiler, para unirlos se debe relacionar con la tabla reserva:</p> <pre> SELECT suc.nombre, COUNT(*) AS numero_alquileres FROM SucursalPLSQL suc JOIN ReservaPLSQL res ON suc.id_sucursal = res.id_sucursal JOIN AlquilerPLSQL alq ON res.id_reserva = alq.id_reserva GROUP BY suc.nombre ORDER BY numero_alquileres DESC fetch first 1 row only; </pre>
<p>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;</p>		<p>La consulta extrae el mes a partir de la fecha de inicio, realiza un conteo de los alquileres, agrupa por mes y ordena según la cantidad de alquileres en orden descendente. El resultado revela el mes con la mayor cantidad de alquileres en el año.</p> <pre> 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 fetch first 1 row only; </pre>

<p>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;</p>		<p>Esta consulta, además del error de limit requiere un ajuste en el formato de fecha para obtener el día de la semana con mas alquileres.</p> <p>El código corregido sería: SELECT TO_CHAR(fecha_inicio,'D') AS dia_semana, COUNT(*) AS numero_alquileres FROM AlquilerPLSQL GROUP BY TO_CHAR(fecha_inicio,'D') ORDER BY numero_alquileres DESC FETCH FIRST 1 ROW ONLY;</p>
<p>SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;</p>		<p>Se corrige el error de limit para mostrar el alquiler con el precio más alto. Sin embargo, la corrección genera error ya que ninguna tabla tiene la columna precio.</p>
<p>SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;</p>		<p>Se corrige el error de limit para mostrar el alquiler con el precio más bajo. Sin embargo, la corrección genera error ya que ninguna tabla tiene la columna precio.</p>
<p>SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%';</p>		<p>La consulta trae todos los clientes que contengan el nombre Juan en la columna Nombre.</p>

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

La consulta muestra todos los autos cuyo precio sea menor a 10.000.

El código presenta error debido a que ninguna tabla tiene la columna precio.

SELECT * FROM AlquilerPLSQL
WHERE fecha_inicio
BETWEEN '2023-09-01' AND
'2023-09-30';

Se corrige el formato de las fechas, para que pueda mostrar todos los alquileres que se realizaron en el mes de septiembre.

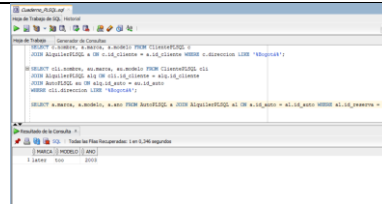
SELECT * FROM AlquilerPLSQL
WHERE
TO_DATE(fecha_inicio, 'DD-MM-YYYY')
BETWEEN TO_DATE('01-09-23', 'YYYY-MM-DD') AND
TO_DATE('30-09-23', 'YYYY-MM-DD');

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%';

La consulta busca el nombre del cliente, la marca y el modelo del automóvil, filtrando por la dirección del cliente que incluya "Bogotá". Se corrige el código uniendo la tabla "auto" para extraer los datos de marca y modelo.

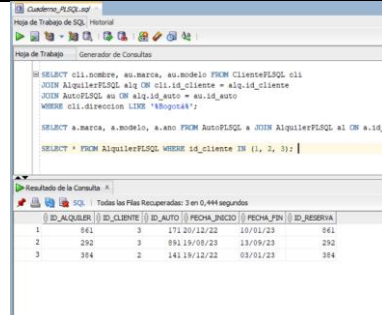
SELECT cli.nombre,
au.marca, au.modelo FROM
ClientePLSQL cli
JOIN AlquilerPLSQL alq ON
cli.id_cliente = alq.id_cliente
JOIN AutoPLSQL au ON
alq.id_auto = au.id_auto
WHERE cli.direccion LIKE
'%Bogotá%';

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;



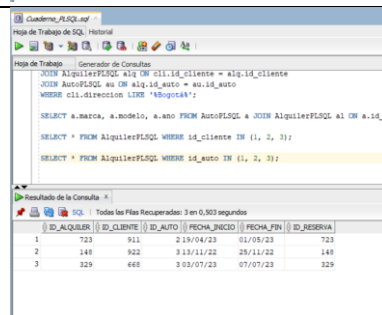
La consulta muestra la información del auto, que fue utilizado en la primera reserva registrada.

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



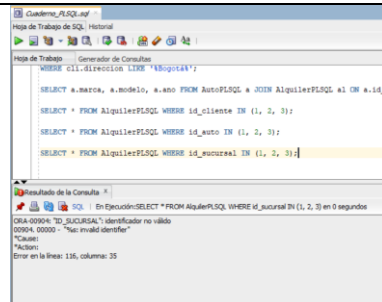
La consulta muestra todas las columnas de la tabla alquiler asociado a los id clientes 1,2 y 3

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

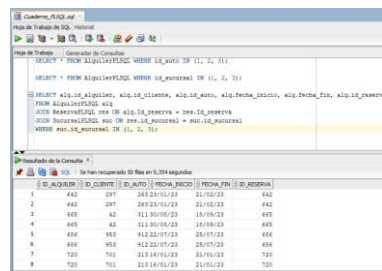


La consulta muestra todas las columnas de la tabla alquiler asociado a los id auto 1,2 y 3

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



La consulta muestra toda la información de la tabla alquiler correspondiente al id sucursal 1,2,3. Se corrige el código haciendo un join con las tablas reserva y sucursal.



SELECT alq.id_alquiler,
alq.id_cliente, alq.id_auto,
alq.fecha_inicio,
alq.fecha_fin, alq.id_reserva
FROM AlquilerPLSQL alq
JOIN ReservaPLSQL res ON
alq.id_reserva = res.id_reserva
JOIN SucursalPLSQL suc ON
res.id_sucursal = suc.id_sucursal
WHERE suc.id_sucursal IN (1,
2, 3);

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

La consulta muestra la información de los alquileres de septiembre asociados a id cliente 1,2,3. Se corrige el código configurando las fechas:
SELECT * FROM AlquilerPLSQL WHERE TO_DATE (fecha_inicio, 'DD-MM-YY') BETWEEN TO_DATE('01-09-23','DD-MM-YY') AND TO_DATE ('30-09-23','DD-MM-YY') AND id_cliente IN (1, 2, 3);

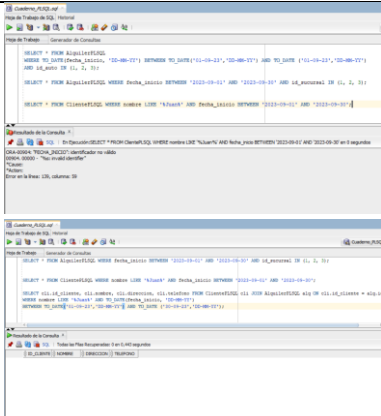
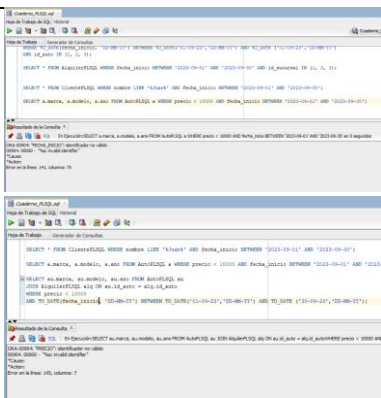
SELECT * FROM
AlquilerPLSQL WHERE
fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30' AND
id_auto IN (1, 2, 3);

La consulta muestra toda la información de los alquileres de septiembre relacionados con id auto 1,2,3. Se corrige el código configurando las fechas:
SELECT * FROM AlquilerPLSQL WHERE TO_DATE(fecha_inicio, 'DD-MM-YY') BETWEEN TO_DATE('01-09-23','DD-MM-YY') AND TO_DATE ('01-09-23','DD-MM-YY') AND id_auto IN (1, 2, 3);

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

La consulta muestra toda la información de los alquileres de septiembre relacionados con id sucursal 1,2,3. Se corrige el código configurando las fechas y haciendo un join entre la tabla alquiler, reserva y sucursal:
SELECT alq.id_alquiler, alq.id_cliente, alq.id_auto, alq.fecha_inicio, alq.fecha_fin, alq.id_reserva FROM AlquilerPLSQL alq JOIN ReservaPLSQL res ON alq.id_reserva = res.id_reserva JOIN SucursalPLSQL suc ON

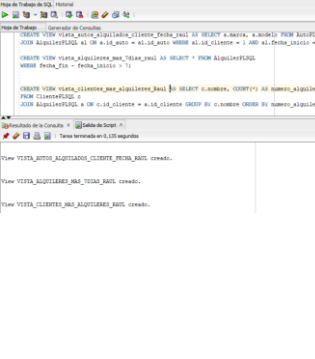
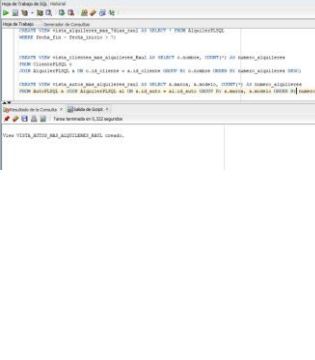
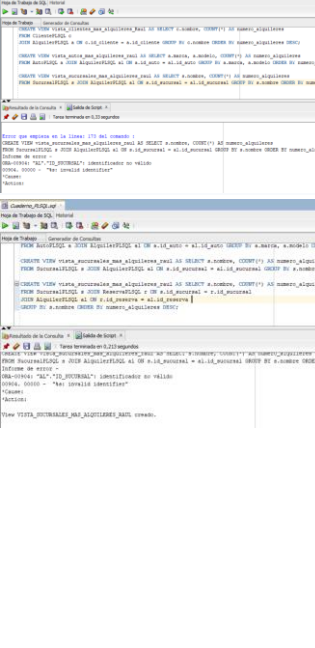
		<pre> res.id_sucursal = suc.id_sucursal WHERE TO_DATE(alq.fecha_inicio, 'DD-MM-YY') BETWEEN TO_DATE('01-09-23','DD-MM- YY') AND TO_DATE ('01-09- 23','DD-MM-YY') AND suc.id_sucursal IN (1, 2, 3); </pre>
<pre> SELECT c.nombre, COUNT(*) AS numero_alquileres FROM ClientePLSQL c JOIN AlquilerPLSQL a ON c.id_cliente = a.id_cliente GROUP BY c.nombre ORDER BY numero_alquileres DESC LIMIT 1; </pre>	Ejercicio repetido = respuesta ejercicio 44	Ejercicio repetido = respuesta ejercicio 44
<pre> 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; </pre>	Ejercicio repetido = respuesta ejercicio 45	Ejercicio repetido = respuesta ejercicio 45
<pre> 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; </pre>	Ejercicio repetido = respuesta ejercicio 46	Ejercicio repetido = respuesta ejercicio 46
<pre> 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; </pre>	Ejercicio repetido = respuesta ejercicio 47	Ejercicio repetido = respuesta ejercicio 47
<pre> SELECT EXTRACT(DAYOFWEEK FROM fecha_inicio) AS dia_semana, COUNT(*) AS </pre>	Ejercicio repetido = respuesta ejercicio 48	Ejercicio repetido = respuesta ejercicio 48

numero_alquileres FROM AlquilerPLSQL GROUP BY EXTRACT(DAYOFWEEK FROM fecha_inicio) ORDER BY numero_alquileres DESC LIMIT 1;		
SELECT * FROM AlquilerPLSQL ORDER BY precio DESC LIMIT 1;	Ejercicio repetido = respuesta ejercicio 49	Ejercicio repetido = respuesta ejercicio 49
SELECT * FROM AlquilerPLSQL ORDER BY precio ASC LIMIT 1;	Ejercicio repetido = respuesta ejercicio 50	Ejercicio repetido = respuesta ejercicio 50
SELECT * FROM ClientePLSQL WHERE nombre LIKE '%Juan%' AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';		<p>La consulta muestra los nombres de clientes que contengan la palabra juan y hayan alquilado durante el mes de septiembre. Se corrige el código ajustando las fechas y realizando join entre la tabla alquiler y cliente.</p> <pre> SELECT cli.id_cliente, cli.nombre, cli.direccion, cli.telefono FROM ClientePLSQL cli JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.id_cliente WHERE nombre LIKE '%Juan%' AND TO_DATE(fecha_inicio, 'DD-MM-YY') BETWEEN TO_DATE('01-09-23','DD-MM-YY') AND TO_DATE ('30-09-23','DD-MM-YY');</pre>
SELECT a.marca, a.modelo, a.ano FROM AutoPLSQL a WHERE precio < 10000 AND fecha_inicio BETWEEN '2023-09-01' AND '2023-09-30';		<p>La consulta muestra los datos de autos cuyo precio sea menor a 10000 y el alquiler se haya realizado en septiembre. A este código se le puede corregir el formato de fecha y realizar el join entre las tablas autos y alquiler, sin embargo, saldrá error porque no existe la consulta precio en las tablas:</p>

		SELECT au.marca, au.modelo, au.ano FROM AutoPLSQL au JOIN AlquilerPLSQL alq ON au.id_auto = alq.id_auto WHERE precio < 10000 AND TO_DATE(fecha_inicio, 'DD-MM-YY') BETWEEN TO_DATE('01-09-23','DD-MM- YY') AND TO_DATE ('30-09- 23','DD-MM-YY');
--	--	--

EJERCICIOS TERCER CICLO (81-90):

CREATE VIEW vista_clientes_alquilados_sucurs al 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';		La vista muestra los clientes y autos alquilados teniendo en cuenta el nombre de la sucursal. Se corrige el código haciendo el join correspondiente para relacionar alquiler y sucursal: CREATE VIEW vista_clientes_alquilados_sucurs al_raul AS SELECT cli.nombre, au.marca, au.modelo FROM ClientePLSQL cli JOIN AlquilerPLSQL alq ON cli.id_cliente = alq.id_cliente JOIN AutoPLSQL au ON alq.id_auto = au.id_auto JOIN ReservaPLSQL res ON alq.id_reserva = res.id_reserva JOIN SucursalPLSQL suc ON res.id_sucursal = suc.id_sucursal WHERE suc.nombre = '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';		La vista muestra los datos de los autos del cliente id 01 que haya realizado el alquiler el 27 de septiembre de 2023. Si se ejecuta la respuesta sale vacía porque no se cumplen las condiciones requeridas.

<p>CREATE VIEW vista_alquileres_mas_7dias AS SELECT * FROM AlquilerPLSQL WHERE fecha_fin - fecha_inicio > 7;</p>		<p>La vista muestra los alquileres que hayan durado más de 7 días.</p>
<p>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;</p>		<p>La vista muestra el nombre de los clientes junto con la cantidad de alquileres realizados.</p>
<p>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;</p>		<p>La vista muestra los autos y la cantidad de alquileres reportados por cada uno.</p>
<p>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;</p>		<p>La vista muestra las sucursales con mayor numero de alquileres, se corrige el código realizando un join entre las tablas sucursal, reserva y alquiler.</p> <p>CREATE VIEW vista_sucursales_mas_alquileres_raul AS SELECT s.nombre, COUNT(*) AS numero_alquileres FROM SucursalPLSQL s JOIN ReservaPLSQL r ON s.id_sucursal = r.id_sucursal JOIN AlquilerPLSQL al ON r.id_reserva = al.id_reserva GROUP BY s.nombre ORDER BY numero_alquileres DESC;</p>

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;

La vista muestra el mes que
tenga más cantidad de
alquileres.

CREATE VIEW
vista_dias_semana_mas_alqu
ieres 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;

La vista muestra el numero
acumulado de alquileres por
día de la semana, se corrige el
código cambiando el uso de
DAYOFWEEK

CREATE VIEW
vista_dias_semana_mas_alquile
res_raul
AS SELECT
TO_CHAR(fecha_inicio,'D') AS
dia_semana, COUNT(*) AS
numero_alquileres
FROM AlquilerPLSQL GROUP BY
TO_CHAR(fecha_inicio,'D')
ORDER BY numero_alquileres
DESC;

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

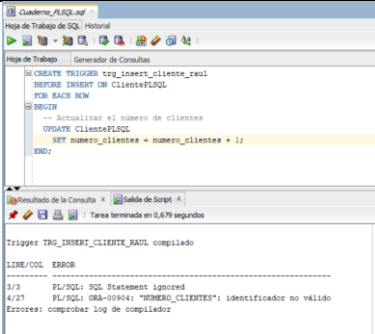
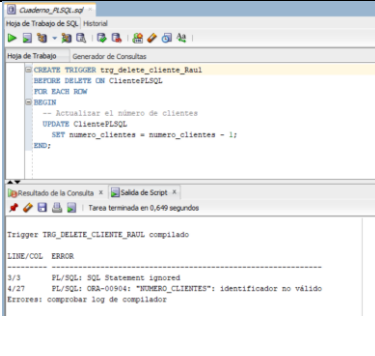
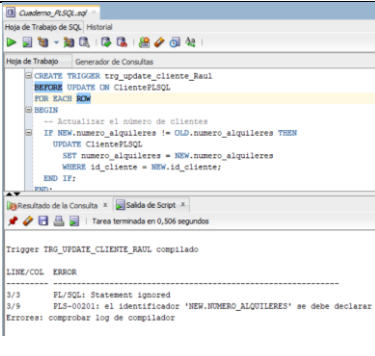
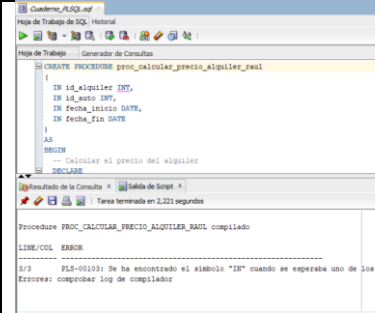
La vista muestra los autos
ordenados por precio más alto,
pero genera error porque
ninguna de las tablas tiene la
columna precio.

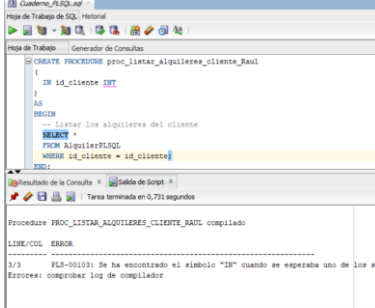
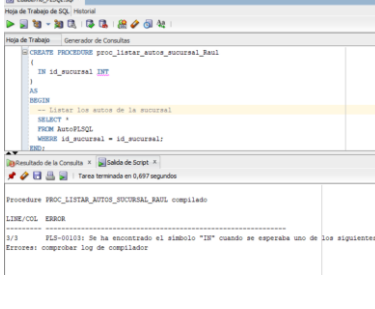
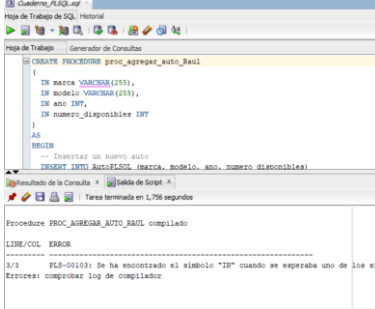
CREATE VIEW
vista_alquileres_mas_baratos
AS SELECT * FROM
AlquilerPLSQL ORDER BY
precio ASC;

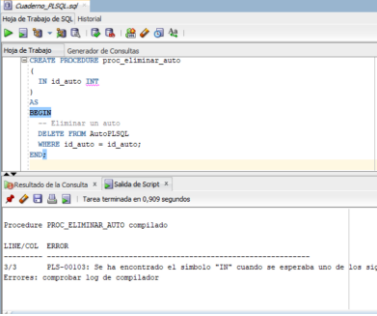
La vista muestra los autos
ordenados por precio más
bajo, pero genera error porque
ninguna de las tablas tiene la
columna precio.

EJERCICIOS TERCER CICLO (91-100):

<pre>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;</pre>		<p>El trigger actualiza el número de autos disponibles al insertar un nuevo registro en la tabla auto, pero presenta un error porque esta tratando de actualizar el campo "numero disponibles" el cual no existe en la tabla.</p>
<pre>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;</pre>		<p>El trigger actualiza el numero de autos disponibles antes de eliminar un registro en la tabla auto, pero presenta un error porque está tratando de actualizar el campo "numero disponibles" el cual no existe en la tabla.</p>
<pre>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;</pre>		<p>El trigger actualiza el numero de autos disponibles cambiando el dato antiguo por el nuevo en caso de encontrar diferencias. Presenta error tratando de actualizar el campo numero disponibles el cual no existe en la tabla auto.</p>

<pre>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;</pre>		<p>El trigger actualiza el numero de clientes sumando una unidad antes de insertar un nuevo cliente en la tabla cliente. Presenta error tratando de actualizar el campo numero_clientes que no existe en la tabla clientes.</p>
<pre>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;</pre>		<p>El trigger actualiza el numero de clientes restando una unidad antes de eliminar un cliente de la tabla clientes. Presenta error tratando de actualizar el campo numero_clientes que no existe en la tabla clientes.</p>
<pre>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;</pre>		<p>El trigger actualiza el numero de alquileres de un cliente antes de actualizar el registro de la tabla cliente. Presenta error porque no existe una columna llamada numero de alquileres que se pueda actualizar.</p>
<pre>CREATE PROCEDURE proc_calcular_precio_alquiler (IN id_alquiler INT, IN id_auto INT, IN fecha_inicio DATE, IN fecha_fin DATE) AS BEGIN</pre>		<p>El procedimiento almacenado calcula el precio de un alquiler de auto considerando fecha inicio, fecha fin y un valor base de auto. Presenta error debido a que la sintaxis usada no funciona</p>

<pre>-- 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;</pre>		<p>en Oracle. Además, menciona la columna "precio" en relación al procedimiento almacenado, pero dicha columna no existe.</p>
<pre>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;</pre>		<p>El procedimiento almacenado busca la información de los alquileres asociados a un cliente utilizando el "id_cliente" como dato de entrada, pero presenta un error debido a que la sintaxis utilizada no funciona en Oracle.</p>
<pre>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;</pre>		<p>El procedimiento almacenado lista los autos asociados a una sucursal utilizando id sucursal como dato de entrada. Pero presenta un error porque busca el campo id sucursal en la tabla auto y este campo no se encuentra en la tabla.</p>
<pre>CREATE PROCEDURE proc_agregar_auto (IN marca VARCHAR(255), IN modelo VARCHAR(255), IN ano INT, IN numero_disponibles INT) AS BEGIN</pre>		<p>El procedimiento almacenado ingresa un registro en la tabla Auto con los campos marca, modelo y año. Pero presenta un error debido a que la sintaxis no es compatible con Oracle.</p>

<pre>-- Insertar un nuevo auto INSERT INTO AutoPLSQL (marca, modelo, ano, numero_disponibles) VALUES (marca, modelo, ano, numero_disponibles); END;</pre>		
<pre>CREATE PROCEDURE proc_eliminar_auto (IN id_auto INT) AS BEGIN -- Eliminar un auto DELETE FROM AutoPLSQL WHERE id_auto = id_auto; END;</pre>	 <p>The screenshot shows a code editor with the following SQL script:</p> <pre>CREATE PROCEDURE proc_eliminar_auto (IN id_auto INT) AS BEGIN -- Eliminar un auto DELETE FROM AutoPLSQL WHERE id_auto = id_auto; END;</pre> <p>Below the script, the compilation output is displayed:</p> <pre>Procedure PROC_ELIMINAR_AUTO compilado LINE/COL ERROR ----- 3/3 PL/SQL: Se ha encontrado el simbolo "IN" cuando se esperaba uno de los sig Errores: consultar log de compilador</pre>	<p>El procedimiento almacenado elimina un auto de la tabla auto utilizando el id como parámetro de entrada. Pero presenta un error debido a que la sintaxis no es compatible con Oracle.</p>