

Sistema de Gestión de Empresas Gasolineras

Santiago Velasco García – Mayo, 2025

Documentación de funcionalidad (consultas, vistas, trigger, procedimiento almacenado)

Dos consultas con diferentes tipos de joins

```
SQL> -- CONSULTAS CON DISTINTOS TIPOS DE JOINS
SQL>
SQL> -- INNER JOIN
SQL> -- Mostrar el nombre del cliente (pilaCte, apPatCte) y su montoTotalGastado, solo si han
SQL> -- realizado al menos una venta.
SQL>
SQL> SELECT c.pilaCte, c.apPatCte, c.montoTotalGastado
2 FROM CLIENTE c
3 INNER JOIN VENTA v ON c.idCte = v.idCte
4 GROUP BY c.idCte, c.pilaCte, c.apPatCte, c.montoTotalGastado;
```

PILACTE	APPATCTE	MONTOTOTALGASTADO
Luis	Herrera	1996.5
Brandon	Fernandez	515.9
Jose	Martinez	844.2
Raul	Hernandez	1034.4
Paola	Garcia	117.25
Amy	Franco	646.5
Martin	Muller	1107.6
Fernanda	Zamora	1420
Ximena	Francisco	351.75

9 filas seleccionadas.

```
SQL> -- LEFT JOIN
SQL> -- Mostrar todas las bombas (idBomba, estado) e incluir los detalles de fallas si las hay.
SQL> -- Las bombas sin fallas también deben aparecer.
SQL> SELECT b.idBomba, b.estado, f.fechaFalla, f.descFalla, f.perdidas, f.fechaReparacion
2 FROM BOMBA_SURTIDORA b
3 LEFT JOIN FALLA f ON b.idBomba = f.idBomba
4 ORDER BY b.idBomba;
```

IDBOMBA	ESTADO	FECHAFAL	DESCFALLA	PERDIDAS	FECHAREP
123440	Activa	21/04/24	mala calibracion	20500	26/04/24
123441	Activa	21/04/24	No funciona	0	23/04/24
123442	Activa	26/06/23	mala calibracion	34095	27/06/23
123442	Activa	21/04/24	Se le safo una pieza	0	22/04/24
123443	Activa	21/04/24	No envia gasolina	0	22/04/24
123443	Activa	04/10/23	Se le safo una pieza	0	05/10/23
123450	En_Falla	12/01/24	Se le safo una pieza	0	13/01/24
123450	En_Falla	26/05/25	Da mal el precio	45000.41	27/05/25
123451	Activa	22/05/25	mala calibracion	54300.32	26/05/25
123452	Activa				
123453	Activa				
IDBOMBA	ESTADO	FECHAFAL	DESCFALLA	PERDIDAS	FECHAREP
123454	Activa				
123455	Activa				
123456	En_Mantenimiento	26/05/25	No marca nada	0	30/05/25
123457	Activa				
123458	En_Falla				
123459	Activa				

17 filas seleccionadas.

Consultas con contrastes de agregación

```
SQL> -- CONSULTAS CON FUNCIONES DE AGREGACIÓN
SQL> -- 1. Sucursal con mayor y menor cantidad de combustible
SQL> WITH total_combustible_por_sucursal AS (
2     SELECT
3         s.idSucursalSub,
4         s.noSucursal,
5         s.direccion,
6         SUM(c.litrosDisponibles) AS total_combustible
7     FROM
8         SUCURSAL s
9     JOIN
10        CISTERNA c ON s.idSucursalSub = c.idSucursalSub
11    GROUP BY
12        s.idSucursalSub, s.noSucursal, s.direccion
13 )
14 SELECT *
15 FROM total_combustible_por_sucursal
16 WHERE total_combustible = (
17     SELECT MAX(total_combustible) FROM total_combustible_por_sucursal
18 )
19 OR total_combustible = (
20     SELECT MIN(total_combustible) FROM total_combustible_por_sucursal
21 );
```

IDSUCURSALSUB	NOSUCURSAL	DIRECCION	TOTAL_COMBUSTIBLE
5	1004	Blvd. Adolfo López Mateos 2173, Los Alpes, Álvaro Obregón, 01010 Ciudad de México, CDMX	2600
9	1008	Eje 10 Sur, Av. Pedro Henríquez Ureña, Pedregal de San Francisco, 04369 Ciudad de México	65500

```
SQL> -- 2. Promedio de litros vendidos por sucursal
SQL> SELECT
2     s.idSucursalSub,
3     s.noSucursal,
4     ROUND(AVG(v.litros), 2) AS promedio_litros_vendidos
5 FROM
6     VENTA v
7 JOIN
8     EMPLEADO e ON v.idEmp = e.idEmp
9 JOIN
10    EMP_SUC es ON e.idEmp = es.idEmp
11 JOIN
12    SUCURSAL s ON es.idSucursalSub = s.idSucursalSub
13 GROUP BY
14     s.idSucursalSub, s.noSucursal, s.direccion;
```

IDSUCURSALSUB	NOSUCURSAL	PROMEDIO_LITROS_VENDIDOS
1	1000	18.67
2	1001	15
3	1002	12
5	1004	18
9	1008	22.5

Dos vistas útiles para las tablas

```
SQL> -- Mostrar cada empleado con el número total de ventas realizadas, litros vendidos y
SQL> -- monto total generado.
SQL> CREATE OR REPLACE VIEW view_resumen_ventas_empleado AS
2  SELECT
3      e.idEmp,
4      e.pilaEmp || ' ' || e.apPatEmp || NVL(' ' || e.apMatEmp, '') AS nombreEmpleado,
5      COUNT(v.idVenta) AS totalVentas,
6      SUM(v.litros) AS litrosVendidos,
7      SUM(v.monto) AS montoGenerado
8  FROM
9      EMPLEADO e
10 JOIN
11     VENTA v ON e.idEmp = v.idEmp
12 GROUP BY
13     e.idEmp, e.pilaEmp, e.apPatEmp, e.apMatEmp;
```

Vista creada.

```
SQL>
SQL> SELECT * FROM VIEW_RESUMEN_VENTAS_EMPLEADO;
```

IDEMP	NOMBREEMPLEADO	TOTALVENTAS	LITROSVENDIDOS	MONTOGENERADO
1231	Lucho Hernandez	1	21	492.45
1232	Jose Mendoza	1	15	351.75
1235	Javier Normal Normal	1	17	398.65
1237	Ximena Benites Cerrano	1	18	465.48
1240	Johan Aguirre Aguilar	1	12	281.4
1233	Naydelin Buendia Martinez	1	20	517.2
1234	Marcos Lomeli	1	25	710
1236	Leinad Rodriguez Barrera	1	30	852
1238	Lucia Mendez Hernandez	1	18	421.2
1239	Isaias Becerril	1	15	388

10 filas seleccionadas.

```
SQL> -- Vista útil para monitorear el estado de las bombas, su última falla y su
SQL> -- empleado asignado.
SQL> CREATE OR REPLACE VIEW view_monitoreo_bombas AS
2  SELECT
3      b.idBomba,
4      b.estado,
5      e.idEmp,
6      e.pilaEmp || ' ' || e.apPatEmp || NVL(' ' || e.apMatEmp, '') AS nombreEmpleado,
7      f.fechaFalla,
8      f.descFalla,
9      f.fechaReparacion
10 FROM
11     BOMBA_SURTIDORA b
12 LEFT JOIN EMP_ASSIGN_BOM ab ON b.idBomba = ab.idBomba
13 LEFT JOIN EMPLEADO e ON ab.idEmp = e.idEmp
14 LEFT JOIN (
15     SELECT f1.*
16     FROM FALLA f1
17     WHERE (f1.idBomba, f1.fechaFalla) IN (
18         SELECT idBomba, MAX(fechaFalla)
19         FROM FALLA
20         GROUP BY idBomba
21     )
22 ) f ON b.idBomba = f.idBomba;
```

Vista creada.

```
SQL> SELECT * FROM VIEW_MONITOREO_BOMBAS;
```

IDBOMBA	ESTADO	IDEMP	NOMBREEMPLEADO	FECHAFAL
DESCFALLA				FECHAREP
123440	Activa			21/04/24
mala calibracion				26/04/24
123441	Activa			21/04/24
No funciona				23/04/24
123442	Activa			21/04/24
Se le safo una pieza				22/04/24

IDBOMBA	ESTADO	IDEMP	NOMBREEMPLEADO	FECHAFAL
DESCFALLA				FECHAREP
123443	Activa			21/04/24
No envia gasolina				22/04/24
123457	Activa	1237	Ximena Benites Cerrano	26/05/25
No marca nada				30/05/25
123450	En_Falla	1240	Johan Aguirre Aguilar	26/05/25
Da mal el precio				27/05/25

IDBOMBA	ESTADO	IDEMP	NOMBREEMPLEADO	FECHAFAL
DESCFALLA				FECHAREP
123451	Activa	1231	Lucho Hernandez	22/05/25
mala calibracion				26/05/25
123458	En_Falla	1238	Lucia Mendez Hernandez	
123454	Activa	1234	Marcos Lomeli	

IDBOMBA	ESTADO	IDEMP	NOMBREEMPLEADO	FECHAFAL
DESCFALLA				FECHAREP
123455	Activa	1235	Javier Normal Normal	
123452	Activa	1232	Jose Mendoza	
123453	Activa	1233	Naydelin Buendia Martinez	

IDBOMBA	ESTADO	IDEMP	NOMBREEMPLEADO	FECHAFAL
DESCFALLA				FECHAREP
123456	En_Mantenimiento	1236	Leinad Rodriguez Barrera	
123459	Activa	1239	Isaias Becerril	

14 filas seleccionadas.

Consultas de álgebra relacional

1. Obtener los nombres de los clientes que han comprado más de 30 litros

$\pi_{\text{pilaCte}, \text{apPatCte}} (\sigma_{\text{litrosComprados} > 30} (\text{CLIENTE}))$

2. Clientes que no son VIP

$\pi_{\text{idCte}} (\text{CLIENTE}) - \pi_{\text{idCte}} (\text{CLIENTE_VIP})$

3. Total de pérdidas por bomba ordenado de mayor a menor

$\gamma_{\text{idBomba}; \text{SUM(perdidas)} \rightarrow \text{total_perdidas}} (\text{FALLA})$
 $\rightarrow \tau_{\downarrow \text{total_perdidas}} (\text{resultado})$

4. Mostrar el nombre del cliente, el empleado y el tipo de combustible de cada venta

$\text{VENTA} \bowtie \text{CLIENTE} \bowtie \text{EMPLEADO}$

$\pi_{\text{pilaCte}, \text{apPatCte}, \text{pilaEmp}, \text{apPatEmp}, \text{tipoComb}} (\text{resultado})$

5. Bombas activas que no aparecen en la tabla FALLA

$R1 \leftarrow \pi_{\text{idBomba}} (\sigma_{\text{estado}='Activa'} (\text{BOMBA_SURTIDORA}))$

$R2 \leftarrow \pi_{\text{idBomba}} (\text{FALLA})$

$R3 \leftarrow R1 - R2$

Consultas de álgebra relacional en SQL

```
SQL> -- CONSULTAS CON ÁLGEBRA RELACIONAL
SQL> -- 1. Obtener los nombres de los clientes que han comprado más de 30 litros
SQL> SELECT pilaCte, apPatCte
  2   FROM CLIENTE
  3  WHERE litrosComprados > 30;
```

PILACTE	APPATCTE
Luis	Herrera
Jose	Martinez
Martin	Muller
Raul	Hernandez
Roberto	Jimenez
Fernanda	Zamora

6 filas seleccionadas.

```
SQL>
SQL> -- 2. Clientes que no son VIP
SQL> SELECT idCte
  2   FROM CLIENTE
  3  MINUS
  4  SELECT idCte
  5   FROM CLIENTE_VIP;
```

IDCTE
11112228
11112229

```
SQL> -- 3. Total de pérdidas por bomba ordenado de mayor a menor
SQL> SELECT idBomba, SUM(perdidas) AS total_perdidas
  2   FROM FALLA
  3  GROUP BY idBomba
  4  ORDER BY total_perdidas DESC;
```

IDBOMBA	TOTAL_PERDIDAS
123451	54300.32
123450	45000.41
123442	34095
123440	20500
123457	0
123441	0
123443	0

7 filas seleccionadas.

```
SQL>
SQL> -- 4. Mostrar el nombre del cliente, el empleado y el tipo de combustible de cada venta
SQL> SELECT
  2   c.pilaCte,
  3   c.apPatCte,
  4   e.pilaEmp,
  5   e.apPatEmp,
  6   v.tipoComb
  7  FROM
  8   VENTA v
  9  JOIN CLIENTE c ON v.idCte = c.idCte
 10  JOIN EMPLEADO e ON v.idEmp = e.idEmp;
```

PILACTE	APPATCTE	PILAEMP	APPATEMP	TIPOCOM
Luis	Herrera	Lucho	Hernandez	Magna
Brandon	Fernandez	Jose	Mendoza	Magna
Amy	Franco	Naydelin	Buendia	Premium
Martin	Muller	Marcos	Lomeli	Diesel
Jose	Martinez	Javier	Normal	Magna
Fernanda	Zamora	Leinad	Rodriguez	Diesel
Raul	Hernandez	Ximena	Benites	Premium
Ximena	Francisco	Lucia	Mendez	Magna
Brandon	Fernandez	Isaias	Becerril	Premium
Paola	Garcia	Johan	Aguirre	Magna

10 filas seleccionadas.

```

SQL>
SQL> -- 5. Bombas activas que no aparecen en la tabla FALLA (bombas que no tienen ni han tenido fallas)
SQL> SELECT idBomba
      2 FROM BOMBA_SURTIDORA
      3 WHERE estado = 'Activa'
      4 MINUS
      5 SELECT idBomba
      6 FROM FALLA;

```

IDBOMBA

123452
123453
123454
123455
123459

Trigger

El siguiente trigger nos sirve para detectar automáticamente cuando una cisterna tiene poco combustible, es decir, cada vez que se insertar o modifican datos de la tabla CISTERNA, el disparador revisa la columna "litroDispo", si llega a haber menos de 6000 litros, automáticamente marcará que está en un nivel bajo, por lo que tomará el valor de 1 en esa columna, caso contrario pondrá 0 si hay más de 6000 litros de combustible.

```
SQL> CREATE OR REPLACE TRIGGER trg_nivel_bajo
  2 BEFORE INSERT OR UPDATE ON CISTERNA
  3 FOR EACH ROW
  4 BEGIN
  5     IF :NEW.litroDispo <= 6000 THEN
  6         :NEW.nivelBajo := 1;
  7     ELSE
  8         :NEW.nivelBajo := 0;
  9     END IF;
 10 END;
 11 /
```

Disparador creado.

```
SQL>
SQL> -- PROBANDO LOS DOS CASOS DEL TRIGGER
SQL>
SQL> SELECT * FROM CISTERNA;
```

IDE	NOSUCURSAL	TIPOCOM	CAPACIDADMAX	LITRODISPO	NIVELBAJO	NOCISTERNA
001	1000	Magna	65000	60000	0	1
001	1000	Premium	50000	46000	0	2
001	1000	Diesel	45000	10000	0	3
002	1001	Magna	65000	5100	1	1
003	1002	Premium	50000	3400	1	2
004	1003	Premium	50000	6200	0	2
005	1004	Diesel	45000	2600	1	3
009	1008	Magna	65000	29000	0	1
009	1008	Premium	50000	15000	0	2
009	1008	Diesel	45000	21500	0	3
007	1006	Magna	65000	61500	0	1

11 filas seleccionadas.


```
SQL> UPDATE CISTERNA
  2 SET litroDispo = 5000
  3 WHERE idEmpGas = '001' AND noSucursal = 1000 AND tipoComb = 'Magna' AND noCisterna = 1;
```

1 fila actualizada.

```
SQL>
SQL> UPDATE CISTERNA
  2 SET litroDispo = 65000
  3 WHERE idEmpGas = '003' AND noSucursal = 1002 AND tipoComb = 'Premium' AND noCisterna = 2;
```

1 fila actualizada.

```
SQL>
SQL> SELECT * FROM CISTERNA;
```

IDE	NOSUCURSAL	TIPOCOM	CAPACIDADMAX	LITRODISPO	NIVELBAJO	NOCISTERNA
001	1000	Magna	65000	5000	1	1
001	1000	Premium	50000	46000	0	2
001	1000	Diesel	45000	10000	0	3
002	1001	Magna	65000	5100	1	1
003	1002	Premium	50000	65000	0	2
004	1003	Premium	50000	6200	0	2
005	1004	Diesel	45000	2600	1	3
009	1008	Magna	65000	29000	0	1
009	1008	Premium	50000	15000	0	2
009	1008	Diesel	45000	21500	0	3
007	1006	Magna	65000	61500	0	1

11 filas seleccionadas.

- **Procedimiento almacenado**

El siguiente proceso almacenado registrar_venta nos permitirá insertar una venta de combustible asegurando la integridad de los datos y el cumplimiento de las reglas de negocio, tales como:

- ❖ Verificar que la bomba esté activa (C.S.6).
- ❖ Validar que el empleado esté asignado a la bomba (integridad referencial con EMP_ASSIGN_BOM).
- ❖ Confirmar que exista suficiente combustible en la cisterna correspondiente (C.S.7).
- ❖ Actualizar los datos del cliente y su condición de cliente VIP (C.S.4).
- ❖ Descontar los litros vendidos de la cisterna correspondiente y actualizar su estado.

```

1  -- PROCEDIMIENTO ALMACENADO
2  CREATE OR REPLACE PROCEDURE registrar_venta (
3      p_idVenta      IN VENTA.idVenta%TYPE,
4      p_idCte        IN VENTA.idCte%TYPE,
5      p_idEmp        IN VENTA.idEmp%TYPE,
6      p_idBomba      IN VENTA.idBomba%TYPE,
7      p_tipoComb     IN VENTA.tipoComb%TYPE,
8      p_fecha        IN VENTA.fecha%TYPE,
9      p_litros       IN VENTA.litros%TYPE,
10     p_monto        IN VENTA.monto%TYPE
11 ) AS
12     v_estadoBomba   BOMBA_SURTIDORA.estado%TYPE;
13     v_es_vip        NUMBER := 0;
14     v_noCompras      CLIENTE_VIP.noCompras%TYPE;
15     v_litrosAcumulados CLIENTE_VIP.litrosAcumulados%TYPE;
16     v_litrosRegalados CLIENTE_VIP.litrosRegalados%TYPE;
17     v_idCisternaSub  CISTERNA.idCisternaSub%TYPE;
18     v_litrosDisponibles CISTERNA.litrosDisponibles%TYPE;
19 BEGIN
20     -- 1. Validar que la bomba esté activa
21     SELECT estado INTO v_estadoBomba
22     FROM BOMBA_SURTIDORA
23     WHERE idBomba = p_idBomba;
24
25     IF v_estadoBomba <> 'Activa' THEN
26         RAISE_APPLICATION_ERROR(-20001, 'La bomba no está activa.');
```

```

27     END IF;

28
29     -- 2. Obtener la cisterna asociada a la bomba y tipo de combustible
30     SELECT cb.idCisternaSub
31     INTO v_idCisternaSub
32     FROM CIS_BOMB cb
33     JOIN CISTERNA cis ON cb.idCisternaSub = cis.idCisternaSub
34     WHERE cb.idBomba = p_idBomba
35           AND cis.tipoComb = p_tipoComb;
36
37     -- 3. Validar que haya suficiente combustible disponible
38     SELECT litrosDisponibles
39     INTO v_litrosDisponibles
40     FROM CISTERNA
41     WHERE idCisternaSub = v_idCisternaSub;
42
43     IF v_litrosDisponibles < p_litros THEN
44         RAISE_APPLICATION_ERROR(-20002, 'No hay suficientes litros en la cisterna.');
```

```

45     END IF;

46
47     -- 4. Registrar la venta
48     INSERT INTO VENTA (
49         idVenta, idCte, idEmp, idBomba, tipoComb, fecha, litros, monto
50     ) VALUES (
51         p_idVenta, p_idCte, p_idEmp, p_idBomba, p_tipoComb, p_fecha, p_litros, p_monto
52     );
53
54     -- 5. Actualizar CLIENTE (total acumulado histórico)
55     UPDATE CLIENTE
56     SET litrosComprados = litrosComprados + p_litros,
57         montoTotalGastado = NVL(montoTotalGastado, 0) + p_monto
58     WHERE idCte = p_idCte;
```

```

60      -- 6. Si es VIP, actualizar su historial de compras y litros acumulados
61      SELECT COUNT(*) INTO v_es_vip
62      FROM CLIENTE_VIP
63      WHERE idCte = p_idCte;
64
65      IF v_es_vip = 1 THEN
66          SELECT noCompras, litrosAcumulados, litrosRegalados
67          INTO v_noCompras, v_litrosAcumulados, v_litrosRegalados
68          FROM CLIENTE_VIP
69          WHERE idCte = p_idCte;
70
71          v_noCompras := v_noCompras + 1;
72          v_litrosAcumulados := v_litrosAcumulados + p_litros;
73
74          IF MOD(v_noCompras, 5) = 0 THEN
75              -- Cada 5 compras se acumulan 3 litros regalados
76              v_litrosRegalados := v_litrosRegalados + 3;
77          END IF;
78
79          UPDATE CLIENTE_VIP
80          SET noCompras = v_noCompras,
81              litrosAcumulados = v_litrosAcumulados,
82              litrosRegalados = v_litrosRegalados
83          WHERE idCte = p_idCte;
84      END IF;
85
86      -- 7. Descontar litros de la cisterna
87      UPDATE CISTERNA
88      SET litrosDisponibles = litrosDisponibles - p_litros,
89      nivelBajo = CASE
90          WHEN (litrosDisponibles - p_litros) <= 6000 THEN 1
91          ELSE 0
92      END
93      WHERE idCisternaSub = v_idCisternaSub;
94
95  EXCEPTION
96      WHEN NO_DATA_FOUND THEN
97          RAISE_APPLICATION_ERROR(-20003, 'Datos inválidos: bomba o cisterna no encontrada.');
```

```

98      WHEN OTHERS THEN
99          RAISE_APPLICATION_ERROR(-20004, 'Error inesperado: ' || SQLERRM);
100  END;
101  /

```

Se prueba el procedimiento con algunas consultas:

```
SQL> SELECT idBomba, estado, idSucursalSub, tipoComb, litrosDisponibles FROM
2 BOMBA_SURTIDORA
3 JOIN CIS_BOMB USING (idBomba)
4 JOIN CISTERNA USING (idCisternaSub)
5 WHERE idBomba=123451;
```

IDBOMBA	ESTADO	IDSUCURSALSUB	TIPOCOM	LITROSDisponibles
123451	Activa	1	Magna	4970
123451	Activa	1	Premium	46000
123451	Activa	1	Diesel	10000

```
SQL>
SQL> SELECT * FROM CLIENTE WHERE idCte = 000011112220;
```

IDCTE	PILACTE	APPATCTE	APMATCTE	TELEFONO	LITROSComprados	MONTO	TIPOCOM
11112220	Luis	Herrera		5511223300	80	1996.5	Premium

```
SQL>
SQL> SELECT * FROM VENTA;
```

IDVENTA	IDCTE	IDEMP	IDBOMBA	TIPOCOM	FECHA	LITROS	MONTO
10000001	11112220	1231	123451	Magna	16/05/25	21	492.45
10000002	11112221	1232	123452	Magna	17/05/25	15	351.75
10000003	11112224	1235	123455	Magna	20/05/25	17	398.65
10000004	11112226	1237	123457	Premium	22/05/25	18	465.48
10000005	11112229	1240	123450	Magna	25/05/25	12	281.4
10000006	11112223	1233	123453	Premium	23/05/25	20	517.2
10000007	11112225	1234	123454	Diesel	24/05/25	25	710
10000008	11112228	1236	123456	Diesel	24/05/25	30	852
10000009	11112222	1238	123458	Magna	23/05/25	18	421.2
10000010	11112221	1239	123459	Premium	23/05/25	15	388

10 filas seleccionadas.

Se ejecuta:

```
SQL> BEGIN
2   registrar_venta(
3     p_idVenta   => 10000011,
4     p_idCte     => 000011112220,
5     p_idEmp     => 001231,
6     p_idBomba   => 123451,
7     p_tipoComb  => 'Magna',
8     p_fecha     => SYSDATE,
9     p_litros    => 30,
10    p_monto     => 703.50
11  );
12 END;
13 /
```

Procedimiento PL/SQL terminado correctamente.

Se vuelve a consultar:

```
SQL> SELECT idBomba, estado, idSucursalSub, tipoComb, litrosDisponibles FROM
2 BOMBA_SURTIDORA
3 JOIN CIS_BOMB USING (idBomba)
4 JOIN CISTERNA USING (idCisternaSub)
5 WHERE idBomba=123451;

IDBOMBA ESTADO          IDSUCURSALSUB TIPOCOM  LITROSDisponibles
-----
123451 Activa            1 Magna       4940
123451 Activa            1 Premium     46000
123451 Activa            1 Diesel      10000

SQL>
SQL> SELECT * FROM CLIENTE WHERE idCte = 000011112220;

IDCTE PILACTE          APPATCTE          APMATCTE
-----
CORREO
11112220 Luis          Herrera
LuisH@correo.com
5511223300          110          2700 Premium

SQL>
SQL> SELECT * FROM VENTA;

IDVENTA IDCTE IDEMP IDBOMBA TIPOCOM FECHA LITROS MONTO
-----
10000011 11112220 1231 123451 Magna 26/05/25 30 703.5
10000001 11112220 1231 123451 Magna 16/05/25 21 492.45
10000002 11112221 1232 123452 Magna 17/05/25 15 351.75
10000003 11112224 1235 123455 Magna 20/05/25 17 398.65
10000004 11112226 1237 123457 Premium 22/05/25 18 465.48
10000005 11112229 1240 123450 Magna 25/05/25 12 281.4
10000006 11112223 1233 123453 Premium 23/05/25 20 517.2
10000007 11112225 1234 123454 Diesel 24/05/25 25 710
10000008 11112228 1236 123456 Diesel 24/05/25 30 852
10000009 11112222 1238 123458 Magna 23/05/25 18 421.2
10000010 11112221 1239 123459 Premium 23/05/25 15 388

11 filas seleccionadas.
```

Se descontaron los litros correspondientes de la cisterna asociada a la bomba, mismos que se sumaron al campo litrosComprados del cliente correspondiente y se registró exitosamente la venta.

Intentemos ahora hacer esto en una bomba inactiva:

```
SQL> SELECT * FROM BOMBA_SURTIDORA WHERE idBomba = 123456;

IDBOMBA ESTADO
-----
123456 En_Mantenimiento

SQL> BEGIN
2 registrar_venta(
3 p_idVenta => 10000012,
4 p_idCte => 000011112220,
5 p_idEmp => 001236,
6 p_idBomba => 123456,
7 p_tipoComb => 'Magna',
8 p_fecha => SYSDATE,
9 p_litros => 30,
10 p_monto => 703.50
11 );
12 END;
13 /
BEGIN
*
ERROR en l nea 1:
ORA-20004: Error inesperado: ORA-20001: La bomba no est  activa.
ORA-06512: en "CURSOBDD.REGISTRAR_VENTA", l nea 99
ORA-06512: en l nea 2
```

Ahora probaremos la dinámica VIP:

```
69 -- Probar dinámica VIP
70 SELECT * FROM CLIENTE_VIP WHERE idCte = 000011112221;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 1 in 0.082 seconds

	IDCTE	NOCOMPRAS	LITROSCOMPRADOS	LITROSREGALADOS
1	11112221	3	70	0

Al cliente le faltan dos compras para poder reclamar sus 3 litros de regalo así que hacemos esas transacciones:

```
72 BEGIN
73     registrar_venta(
74         p_idVenta => 10000012,
75         p_idCte   => 000011112221,
76         p_idEmp   => 001232,
77         p_idBomba => 123452,
78         p_tipoComb => 'Magna',
79         p_fecha   => SYSDATE,
80         p_litros  => 30,
81         p_monto   => 703.5
82     );
83 END;
84
```

Después de otra compra:

All rows fetched: 1 in 0.041 seconds

	IDCTE	NOCOMPRAS	LITROSACUMULADOS	LITROSREGALADOS
1	11112221	4	100	0

```
84
85 BEGIN
86     registrar_venta(
87         p_idVenta => 10000013,
88         p_idCte   => 000011112221,
89         p_idEmp   => 001232,
90         p_idBomba => 123452,
91         p_tipoComb => 'Magna',
92         p_fecha   => SYSDATE,
93         p_litros  => 30,
94         p_monto   => 703.5
95     );
96 END;
```

Volvemos a consultar y vemos que al cliente se le han regalado sus 3 litros:

All rows fetched: 1 in 0.063 seconds

	IDCTE	NOCOMPRAS	LITROSACUMULADOS	LITROSREGALADOS
1	11112221	5	130	3