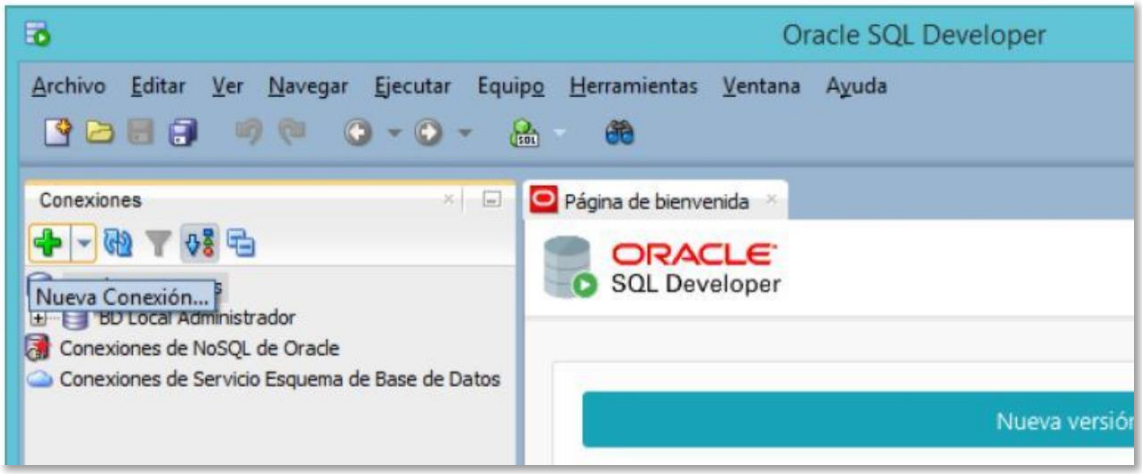
PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Bases de Datos
5ta. Práctica Dirigida
(Segundo Semestre 2020)

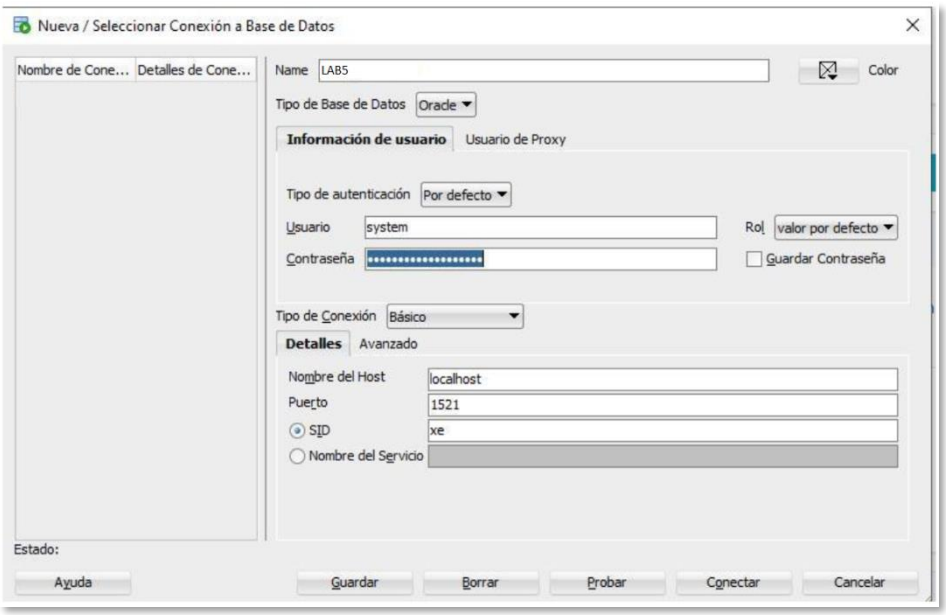
GUÍA – FUNCIONES – PROCEDIMIENTOS CURSORES – TRIGGERS

Conectarse a una base de datos

Primero, ejecute el **Oracle SQL Developer**, cierre la pestaña **Página de bienvenida**, y en el panel de **Conexiones**, haga clic en el ícono **+** para crear una nueva conexión.



Se abrirá la siguiente ventana:

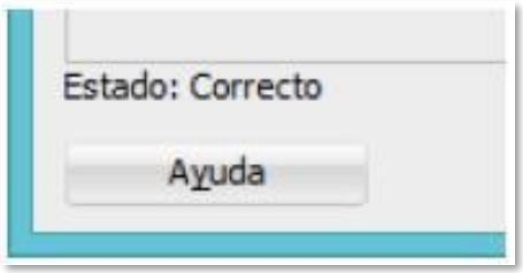


Ingrese los siguientes datos para la conexión:

- Nombre de conexión (Name): **LAB5**
- Usuario: **system**
- Contraseña: Debe escribir la contraseña que ingresó al instalar el Oracle Database Express Edition.

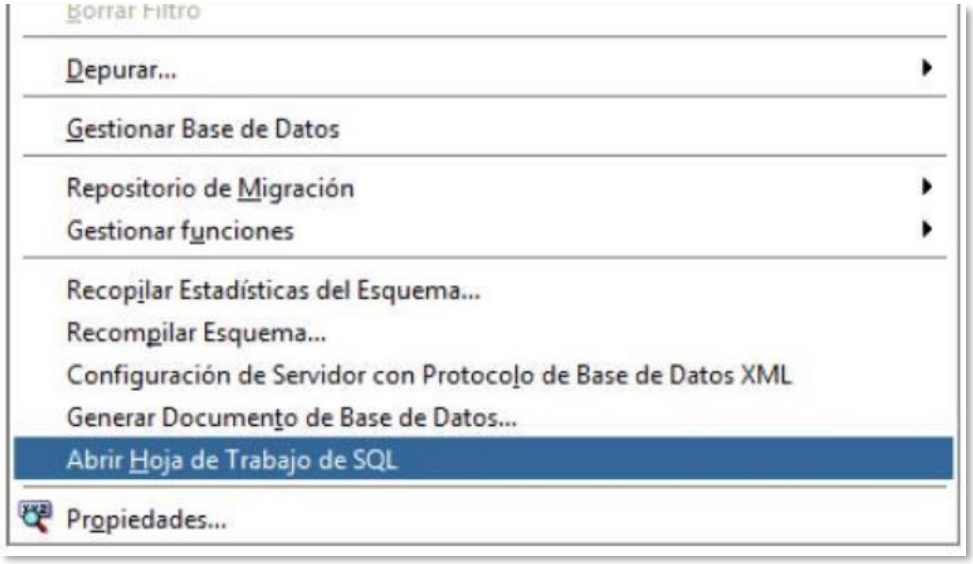
(O en caso de usar la conexión remota proporcionada por la Universidad, ingresar los datos de conexión respectivos).

Haga clic en el botón **Probar**. Si todo está bien configurado, aparecerá el mensaje de **Estado: Correcto** en la parte inferior izquierda.

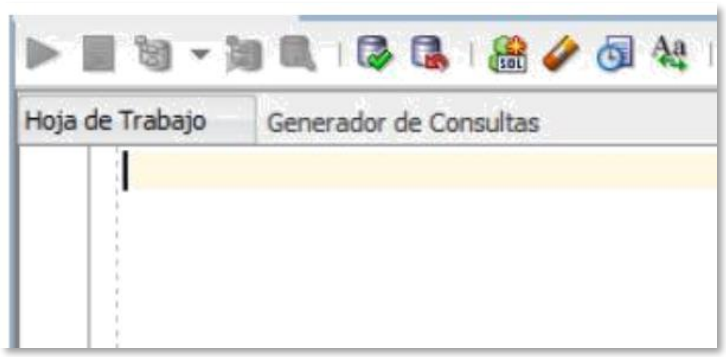


Luego haga clic en el botón **Guardar**, y luego clic en el botón **Conectar**. El nombre de la conexión creada (**LAB5**) aparecerá en el panel de conexiones.

Haga clic derecho sobre el nombre de la conexión **LAB5** y aparecerá el siguiente menú contextual:



Hacemos clic en **Abrir Hoja de Trabajo de SQL**. Nos mostrará una pantalla similar a la siguiente:



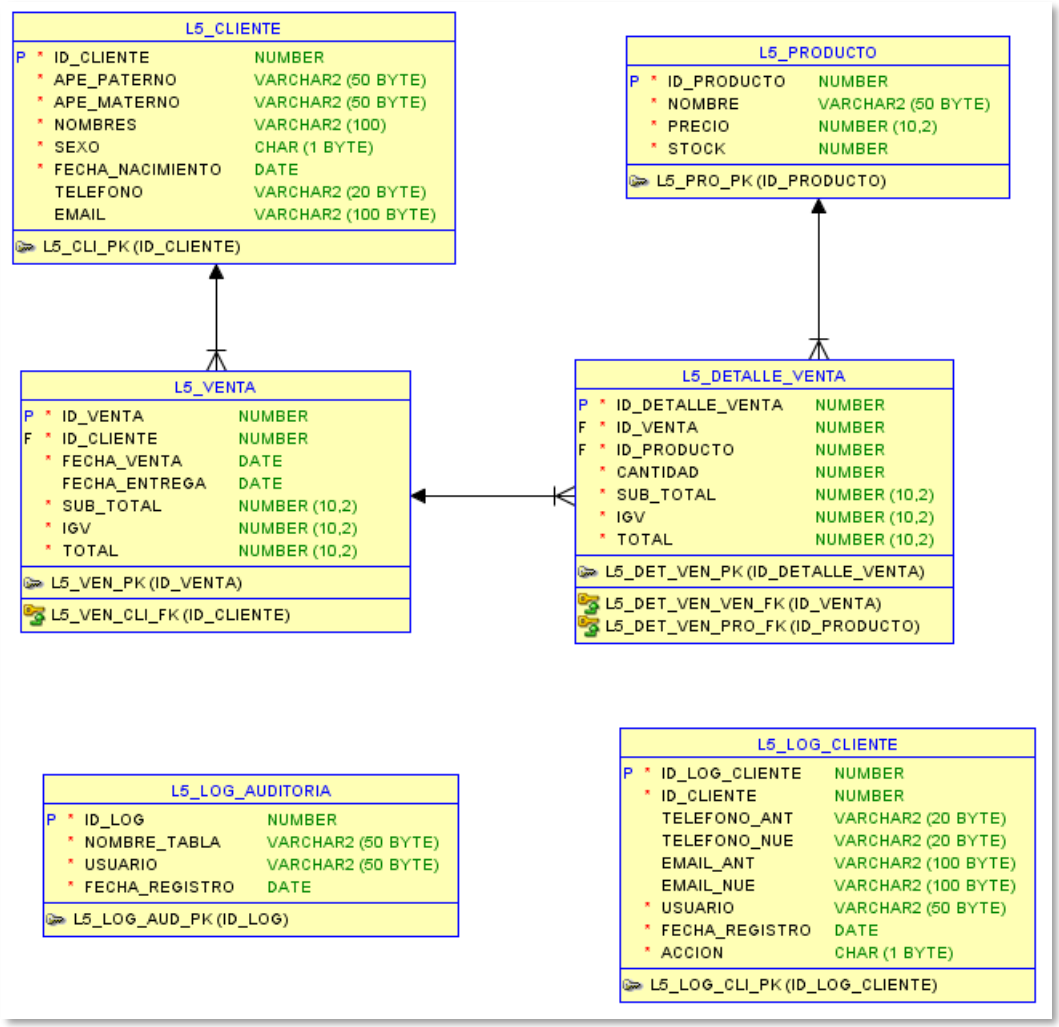
Esta es la **hoja de trabajo**, donde se podrán escribir y ejecutar los scripts SQL.

Parte Dirigida

Para iniciar, ejecute los scripts que se adjuntan en la parte dirigida en el siguiente orden:

- **INF246_2020-2_LAB5_01_dirigida_drops.sql**
- **INF246_2020-2_LAB5_02_dirigida_modelo.sql**
- **INF246_2020-2_LAB5_03_dirigida_inserts.sql**

Para esta parte dirigida de usarán las siguientes tablas:



(También puede revisar el archivo **modelo_dirigida.pdf**)

1.- Funciones

Problema: Se ha identificado que en la base de datos se han calculado los montos de IGV y SUB_TOTAL de manera incorrecta, es decir utilizando el 18% en lugar del 19% como valor del IGV. Para lo cual se pide crear una función que calcule el sub total a partir del total y otra función que calcule el impuesto igv a partir del total.

Solución:

```
CREATE OR REPLACE FUNCTION FN_CALCULAR_SUBTOTAL(P_TOTAL NUMBER) RETURN NUMBER
```

```
IS
BEGIN
  RETURN P_TOTAL / 1.19;
END;
```

```
CREATE OR REPLACE FUNCTION FN_CALCULAR_IGV(P_TOTAL NUMBER) RETURN NUMBER IS
BEGIN
  RETURN P_TOTAL - FN_CALCULAR_SUBTOTAL(P_TOTAL);
END;
```

2.- Procedimientos y Cursores

Problema: Se pide crear un procedimiento que recorra las ventas y los detalle de venta y corrija los montos de igv y subtotales, usando las funciones creadas en el punto anterior.

Solución: Se van a crear dos procedimientos, uno para actualizar el detalle de la venta dado un id_venta y otro para recorrer todas las ventas y actualizar los totales.

```
CREATE OR REPLACE PROCEDURE PR_AJUSTE_DETALLE(P_ID_VENTA NUMBER) IS
BEGIN
  UPDATE L5_DETALLE_VENTA
  SET SUB_TOTAL = FN_CALCULAR_SUBTOTAL(TOTAL),
      IGV = FN_CALCULAR_IGV(TOTAL)
  WHERE ID_VENTA = P_ID_VENTA;
END;
```

```
CREATE OR REPLACE PROCEDURE PR_AJUSTE_VENTAS IS
  CURSOR C_VENTAS IS
    SELECT V.ID_VENTA
    FROM L5_VENTA V;

  V_ID_VENTA NUMBER;
  V_IGV L5_VENTA.IGV%TYPE;
  V_SUB_TOTAL L5_VENTA.SUB_TOTAL%TYPE;
BEGIN
  OPEN C_VENTAS;
  LOOP
    FETCH C_VENTAS INTO V_ID_VENTA;
    EXIT WHEN C_VENTAS%NOTFOUND;

    PR_AJUSTE_DETALLE(V_ID_VENTA);

    SELECT SUM(IGV), SUM(SUB_TOTAL)
    INTO V_IGV, V_SUB_TOTAL
    FROM L5_DETALLE_VENTA WHERE ID_VENTA = V_ID_VENTA;

    UPDATE L5_VENTA SET SUB_TOTAL = V_SUB_TOTAL, IGV = V_IGV WHERE ID_VENTA =
V_ID_VENTA;
  END LOOP;
  CLOSE C_VENTAS;
END;
```

Probamos el procedimiento:

```
SET SERVEROUTPUT ON;
EXEC PR_AJUSTE_VENTAS;
```

Problema: Se pide crear un procedimiento almacenado que imprima un reporte con todas las ventas realizadas, su detalle, así como la ínformación del cliente y productos vendidos en un formato con el siguiente:

```
=========================================================
CÓD. VENTA: 1 CLIENTE: Flores Tong, Juan
FECHA VENTA: 05/09/2020
FECHA ENTREGA: 06/09/2020 ESTADO: Entregado.
    PRODUCTO: Arroz 1kg. Costeño CANTIDAD: 1 SUB_TOTAL: S/ 2.97 TOTAL: S/ 3.5
    PRODUCTO: Azúcar Rubia 1kg. CANTIDAD: 1 SUB_TOTAL: S/ 2.37 TOTAL: S/ 2.8
SUB_TOTAL: S/ 5.34 IGV: S/ .96 TOTAL: S/ 6.3
=========================================================
CÓD. VENTA: 2 CLIENTE: Tapia Delgado, Maria
FECHA VENTA: 28/09/2020
FECHA ENTREGA: 01/10/2020 ESTADO: Entregado.
    PRODUCTO: Galleta Soda Field CANTIDAD: 2 SUB_TOTAL: S/ 2.03 TOTAL: S/ 2.4
    PRODUCTO: Chocolate Sublime CANTIDAD: 3 SUB_TOTAL: S/ 3.81 TOTAL: S/ 4.5
SUB_TOTAL: S/ 5.85 IGV: S/ 1.05 TOTAL: S/ 6.9
.
.
.
=========================================================
CÓD. VENTA: 8 CLIENTE: Quispe Ramirez, Miguel
FECHA VENTA: 12/11/2020
FECHA ENTREGA: --- ESTADO: Por entregar.
    PRODUCTO: Leche Gloria tarro CANTIDAD: 1 SUB_TOTAL: S/ 3.98 TOTAL: S/ 4.7
    PRODUCTO: Harina Blanca Flor 1kg. CANTIDAD: 2 SUB_TOTAL: S/ 11.02 TOTAL:
S/ 13
    PRODUCTO: Spaguetti Don Vittorio CANTIDAD: 1 SUB_TOTAL: S/ 1.95 TOTAL: S/
2.3
SUB_TOTAL: S/ 16.95 IGV: S/ 3.05 TOTAL: S/ 20
```

Consideraciones:

- Las **fechas** deben presentarse en el formato **DD/MM/YYYY**.
- Si aún no ha sido realizada la entrega imprimir la cadena **'---'** y colocar como estado **"Por entregar".**
- Si ya se realizó la entrega imprimir la fecha de entrega y colocar como estado **"Entregado".**

Solución: Para este problema vamos a recorrer todos los elementos de la tabla L5_VENTA, y para cada elemento se recorrerán todos los registros de la tabal L5_DETALLE_VENTA asociados a ese ID_VENTA. Por lo tanto emplearemos 2 cursores, uno sin parámetros y el segundo con parámetros:

```
CREATE OR REPLACE PROCEDURE PR_REPORTE_VENTAS IS
  CURSOR C_VENTAS IS
    SELECT V.ID_VENTA, V.SUB_TOTAL, V.IGV, V.TOTAL, V.FECHA_VENTA,
V.FECHA_ENTREGA,
      C.APE_PATERNO || ' ' || C.APE_MATERNO || ', ' || C.NOMBRES AS CLIENTE
    FROM L5_VENTA V, L5_CLIENTE C
    WHERE V.ID_CLIENTE = C.ID_CLIENTE
    ORDER BY V.FECHA_VENTA;

  CURSOR C_DETALLE_VENTAS(P_ID_VENTA NUMBER) IS
    SELECT P.NOMBRE, D.CANTIDAD, D.SUB_TOTAL, D.TOTAL
    FROM L5_DETALLE_VENTA D, L5_PRODUCTO P
    WHERE D.ID_PRODUCTO = P.ID_PRODUCTO
    AND D.ID_VENTA = P_ID_VENTA
    ORDER BY D.ID_DETALLE_VENTA;
```

```
  -- PODEMOS USAR UN REGISTRO DEL MISMO TIPO DE DATO DE LA FILA DEL CURSOR
  REG_VENTA C_VENTAS%ROWTYPE;
  -- PODEMOS USAR CAMPOS POR SEPARADO PARA GUARDAR LOS RESULTADOS DEL QUERY
  V_NOMBRE_PRODUCTO L5_PRODUCTO.NOMBRE%TYPE;
  V_SUB_TOTAL L5_DETALLE_VENTA.SUB_TOTAL%TYPE;
  V_TOTAL L5_DETALLE_VENTA.TOTAL%TYPE;
  V_CANTIDAD NUMBER;
BEGIN

  OPEN C_VENTAS;
  LOOP
    FETCH C_VENTAS INTO REG_VENTA;
    EXIT WHEN C_VENTAS%NOTFOUND;


DBMS_OUTPUT.PUT_LINE('=========================================================='
);
    DBMS_OUTPUT.PUT_LINE('CÓD. VENTA: ' || REG_VENTA.ID_VENTA || ' CLIENTE: '
|| REG_VENTA.CLIENTE);
    DBMS_OUTPUT.PUT_LINE('FECHA VENTA: ' || TO_CHAR(REG_VENTA.FECHA_VENTA,
'DD/MM/YYYY'));
    IF REG_VENTA.FECHA_ENTREGA IS NULL THEN
      DBMS_OUTPUT.PUT_LINE('FECHA ENTREGA: ' || '---' || ' ESTADO: ' || 'Por
entregar.');
    ELSE
      DBMS_OUTPUT.PUT_LINE('FECHA ENTREGA: ' ||
TO_CHAR(REG_VENTA.FECHA_ENTREGA, 'DD/MM/YYYY') || ' ESTADO: ' ||
'Entregado.');
    END IF;

    OPEN C_DETALLE_VENTAS(REG_VENTA.ID_VENTA);
    LOOP
      FETCH C_DETALLE_VENTAS INTO V_NOMBRE_PRODUCTO, V_CANTIDAD, V_SUB_TOTAL,
V_TOTAL;
      EXIT WHEN C_DETALLE_VENTAS%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('    PRODUCTO: ' || V_NOMBRE_PRODUCTO || '
CANTIDAD: ' || V_CANTIDAD || ' SUB_TOTAL: S/ ' || V_SUB_TOTAL || ' TOTAL: S/ '
|| V_TOTAL);
    END LOOP;
    CLOSE C_DETALLE_VENTAS;

    DBMS_OUTPUT.PUT_LINE('SUB_TOTAL: S/ ' || REG_VENTA.SUB_TOTAL || ' IGV: S/
' || REG_VENTA.IGV || ' TOTAL: S/ ' || REG_VENTA.TOTAL);
  END LOOP;
  CLOSE C_VENTAS;

END;
```

Probamos el procedimiento del reporte:

```
SET SERVEROUTPUT ON;
EXEC PR_REPORTE_VENTAS;
```

Otra forma de solución, usando dos procedimientos separados, uno llama al otro.

Procedimiento para imprimir el detalle de las ventas, recibe como parámetro el ID_VENTA:

```
CREATE OR REPLACE PROCEDURE PR_REPORTE_DETALLE_2(P_ID_VENTA NUMBER) IS
  CURSOR C_DETALLE_VENTAS(P_ID_VENTA NUMBER) IS
    SELECT P.NOMBRE, D.CANTIDAD, D.SUB_TOTAL, D.TOTAL
    FROM L5_DETALLE_VENTA D, L5_PRODUCTO P
```

```
    WHERE D.ID_PRODUCTO = P.ID_PRODUCTO
    AND D.ID_VENTA = P_ID_VENTA
    ORDER BY D.ID_DETALLE_VENTA;

  V_NOMBRE_PRODUCTO L5_PRODUCTO.NOMBRE%TYPE;
  V_SUB_TOTAL L5_DETALLE_VENTA.SUB_TOTAL%TYPE;
  V_TOTAL L5_DETALLE_VENTA.TOTAL%TYPE;
  V_CANTIDAD NUMBER;
BEGIN

  OPEN C_DETALLE_VENTAS(P_ID_VENTA);
    LOOP
      FETCH C_DETALLE_VENTAS INTO V_NOMBRE_PRODUCTO, V_CANTIDAD, V_SUB_TOTAL,
V_TOTAL;
      EXIT WHEN C_DETALLE_VENTAS%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('    PRODUCTO: ' || V_NOMBRE_PRODUCTO || '
CANTIDAD: ' || V_CANTIDAD || ' SUB_TOTAL: S/ ' || V_SUB_TOTAL || ' TOTAL: S/ '
|| V_TOTAL);
    END LOOP;
    CLOSE C_DETALLE_VENTAS;

END;
```

Procedimiento para imprimir los datos de la venta, llama internamente al procedimiento de detalle de ventas pasándole como parámetro el ID_VENTA respectivo:

```
CREATE OR REPLACE PROCEDURE PR_REPORTE_VENTAS_2 IS
  CURSOR C_VENTAS IS
    SELECT V.ID_VENTA, V.SUB_TOTAL, V.IGV, V.TOTAL, V.FECHA_VENTA,
V.FECHA_ENTREGA,
      C.APE_PATERNO || ' ' || C.APE_MATERNO || ', ' || C.NOMBRES AS CLIENTE
    FROM L5_VENTA V, L5_CLIENTE C
    WHERE V.ID_CLIENTE = C.ID_CLIENTE
    ORDER BY V.FECHA_VENTA;

  REG_VENTA C_VENTAS%ROWTYPE;

BEGIN

  OPEN C_VENTAS;
  LOOP
    FETCH C_VENTAS INTO REG_VENTA;
    EXIT WHEN C_VENTAS%NOTFOUND;


DBMS_OUTPUT.PUT_LINE('=====================================================');
    DBMS_OUTPUT.PUT_LINE('CÓD. VENTA: ' || REG_VENTA.ID_VENTA || ' CLIENTE: '
|| REG_VENTA.CLIENTE);
    DBMS_OUTPUT.PUT_LINE('FECHA VENTA: ' || TO_CHAR(REG_VENTA.FECHA_VENTA,
'DD/MM/YYYY'));
    IF REG_VENTA.FECHA_ENTREGA IS NULL THEN
      DBMS_OUTPUT.PUT_LINE('FECHA ENTREGA: ' || '---' || ' ESTADO: ' || 'Por
entregar.');
    ELSE
      DBMS_OUTPUT.PUT_LINE('FECHA ENTREGA: ' ||
TO_CHAR(REG_VENTA.FECHA_ENTREGA, 'DD/MM/YYYY') || ' ESTADO: ' ||
'Entregado.');
    END IF;

    -- AQUI LLAMAMOS AL SEGUNDO PROCEDIMIENTO
    PR_REPORTE_DETALLE_2(REG_VENTA.ID_VENTA);

    DBMS_OUTPUT.PUT_LINE('SUB_TOTAL: S/ ' || REG_VENTA.SUB_TOTAL || ' IGV: S/
```

```
' || REG_VENTA.IGV || ' TOTAL: S/ ' || REG_VENTA.TOTAL);
  END LOOP;
  CLOSE C_VENTAS;

END;
```

Probamos el procedimiento del reporte:

```
SET SERVEROUTPUT ON;
EXEC PR_REPORTE_VENTAS_2;
```

3.- Triggers

Problema: Se pide llevar la trazabilidad de qué usuarios de base de datos realizan inserciones sobre la tabla **L5_CLIENTE**, a manera de auditoría sobre la tabla **L5_LOG_AUDITORIA**.

La operación (**trigger**) debe registrar el **nombre de la tabla**, el **usuario** que realizó el registro y la **fecha de operación**, para estos dos últimos puede usar las funciones **USER** y **SYSDATE** respectivamente.

Solución: El trigger a crear tiene que dispararse cada vez que un usuario **inserte** datos en la tabla **L5_CLIENTE**, por lo tanto será un trigger **"after insert":**

```
CREATE OR REPLACE TRIGGER TR_LOG_CLIENTE
AFTER INSERT ON L5_CLIENTE
FOR EACH ROW
DECLARE
  V_ID_LOG NUMBER;
BEGIN
  SELECT NVL(MAX(ID_LOG), 0) INTO V_ID_LOG FROM L5_LOG_AUDITORIA;

  INSERT INTO L5_LOG_AUDITORIA (ID_LOG, NOMBRE_TABLA, USUARIO, FECHA_REGISTRO)
  VALUES (V_ID_LOG + 1, 'L5_CLIENTE', USER, SYSDATE);
END;
```

Realizamos la prueba de la siguiente manera:

- Insertamos un cliente en **L5_CLIENTE**.
- Comprobamos el registro insertado con un query en **L5_CLIENTE**.
- Comprobamos el funcionamiento correcto del trigger con un query en **L5_LOG_AUDITORIA**.

Ya que el último ID_CLIENTE de la tabla es el 6, usaremos el 7:

```
INSERT INTO L5_CLIENTE (ID_CLIENTE, APE_PATERNO, APE_MATERNO, NOMBRES, SEXO,
FECHA_NACIMIENTO, TELEFONO, EMAIL)
VALUES (7, 'Theo', 'Monzen', 'Otto', 'M', TO_DATE('25/01/1983', 'DD/MM/YYYY'),
'999999999', 'otto.theo@pucp.edu.pe');
```

Verificamos que se insertó en la tabla **L5_CLIENTE**:

```
SELECT * FROM L5_CLIENTE;
```

ID_CLIENTE	APE_PATERNO	APE_MATERNO	NOMBRES	SEXO	FECHA_NACIMIENTO	TELEFONO	EMAIL
1	Flores	Tong	Juan	M	15-JUN-75	972343481	jflores@gmail.com
2	Tapia	Delgado	Maria	F	10-JAN-55	960491786	mtapia@hotmail.com
3	Quispe	Ramirez	Miguel	M	09-MAR-83	962905883	mquispe@gmail.com
4	Arauco	Arana	Carlos	M	11-FEB-00	962609096	carauco@gmail.com
5	Dallas	Miller	Ann	F	15-JUN-99	965918300	adallas@miempresa.com
6	Cueva	Vasquez	Kim	F	03-MAR-96	923950023	kim@hotmail.com
7	Theo	Monzen	Otto	M	25-JAN-83	999999999	otto.theo@pucp.edu.pe

Revisamos la tabla donde el trigger registró el resultado de la acción:

```
SELECT * FROM L5_LOG_AUDITORIA;
```

ID_LOG	NOMBRE_TABLA	USUARIO	FECHA_REGISTRO
1	L5_CLIENTE	INF24620202	28-NOV-20

Y efectivamente se realizó la inserción en la tabla de auditoría. Con esto podemos llevar la trazabilidad de los usuarios del sistema que hicieron inserciones (o cualquier otra acción) sobre cualquier tabla del sistema, para este ejemplo: L5_CLIENTE.

Problema: Se desea tener un mayor control sobre la auditoría de las tabla **L5_CLIENTE** y no necesariamente guardar la información de quién y cuándo se realizaron cambios, sino también se quiere llevar un control de la historia de cambios realizados a los datos de los clientes, en este caso conocer si hubieron correcciones en los campos **EMAIL** y **TELEFONO** (considerar inserción y modificación). Este registro de cambios se llevará en la tabla **L5_LOG_CLIENTE**.

Solución: La tabla **L5_LOG_CLIENTE** guardará entonces los siguientes registros:

- Si es **inserción**: los campos **_ANT** estarán en NULL y los campos **_NUE** tendrán los registros nuevos.
- Si es **modificación**: los campos **_ANT** tendrán los datos anteriores al update y los campos **_NUE** tendrán los registros nuevos.
- El campo **ACCION** tendrá los valores de **'I'** para insert y **'U'** para update.

El trigger a crear tiene que dispararse cada vez que un usuario **inserte** o **actualice** datos en la tabla **L5_CLIENTE**, por lo tanto será un trigger **"after insert or update":**

```
CREATE OR REPLACE TRIGGER TR_TRAZA_CLIENTE
AFTER INSERT OR UPDATE ON L5_CLIENTE
FOR EACH ROW
DECLARE
  V_ID_LOG NUMBER;
BEGIN
  SELECT NVL(MAX(ID_LOG_CLIENTE), 0) INTO V_ID_LOG FROM L5_LOG_CLIENTE;

  IF INSERTING THEN
    INSERT INTO L5_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, TELEFONO_NUE,
EMAIL_NUE, USUARIO, FECHA_REGISTRO, ACCION)
    VALUES (V_ID_LOG + 1, :NEW.ID_CLIENTE, :NEW.TELEFONO, :NEW.EMAIL, USER,
SYSDATE, 'I');
  END IF;

  IF UPDATING THEN
    INSERT INTO L5_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, TELEFONO_ANT,
TELEFONO_NUE, EMAIL_ANT, EMAIL_NUE, USUARIO, FECHA_REGISTRO, ACCION)
```

```
    VALUES (V_ID_LOG + 1, :NEW.ID_CLIENTE, :OLD.TELEFONO, :NEW.TELEFONO,
:OLD.EMAIL, :NEW.EMAIL, USER, SYSDATE, 'U');
  END IF;
END;
```

Realizamos la prueba de la siguiente manera:

- Insertamos un cliente en **L5_CLIENTE**.
- Comprobamos el registro insertado con un query en **L5_CLIENTE**.
- Comprobamos el funcionamiento correcto del trigger con un query en **L5_LOG_AUDITORIA**.
- Actualizamos un cliente en **L5_CLIENTE**.
- Comprobamos el registro actualizado con un query en **L5_CLIENTE**.
- Comprobamos el funcionamiento correcto del trigger con un query en **L5_LOG_AUDITORIA**.

```
INSERT INTO L5_CLIENTE (ID_CLIENTE, APE_PATERNO, APE_MATERNO, NOMBRES, SEXO,
FECHA_NACIMIENTO, TELEFONO, EMAIL)
VALUES (8, 'Mezones', 'Estrada', 'Renzo', 'M', TO_DATE('14/04/1996',
'DD/MM/YYYY'), '88888888', 'renzo@gmail.com');
```

Verificamos que se insertó en la tabla **L5_CLIENTE**:

```
SELECT * FROM L5_CLIENTE;
```

ID_CL...	APE_PATERNO	APE_MATERNO	NOMBRES	SEXO	FECHA_NACIMIENTO	TELEFONO	EMAIL
1	Flores	Tong	Juan	M	15-JUN-75	972343481	jflores@gmail.com
2	Tapia	Delgado	Maria	F	10-JAN-55	960491786	mtapia@hotmail.com
3	Quispe	Ramirez	Miguel	M	09-MAR-83	962905883	mquispe@gmail.com
4	Arauco	Arana	Carlos	M	11-FEB-00	962609096	carauco@gmail.com
5	Dallas	Miller	Ann	F	15-JUN-99	965918300	adallas@miempresa.com
6	Cueva	Vasquez	Kim	F	03-MAR-96	923950023	kim@hotmail.com
7	Theo	Monzen	Otto	M	25-JAN-83	99999999	otto.theo@pucp.edu.pe
8	Mezones	Estrada	Renzo	M	14-APR-96	88888888	renzo@gmail.com

Revisamos la tabla donde el trigger registró el resultado de la acción:

```
SELECT * FROM L5_LOG_CLIENTE;
```

ID_LOG_CLIENTE	ID_CLIENTE	TELEFONO_ANT	TELEFONO_NUE	EMAIL_ANT	EMAIL_NUE	USUARIO	FECHA_REGISTRO	ACCION
1	8	(null)	88888888	(null)	renzo@gmail.com	INF24620202	28-NOV-20	I

```
UPDATE L5_CLIENTE SET TELEFONO = '987987987', EMAIL = 'otheo@pucp.edu.pe'
WHERE ID_CLIENTE = 7;
```

Verificamos que se actualizó en la tabla **L5_CLIENTE**:

```
SELECT * FROM L5_CLIENTE;
```

ID_CLIENTE	APE_PATERNO	APE_MATERNO	NOMBRES	SEXO	FECHA_NACIMIENTO	TELEFONO	EMAIL
1	Flores	Tong	Juan	M	15-JUN-75	972343481	jflores@gmail.com
2	Tapia	Delgado	Maria	F	10-JAN-55	960491786	mtapia@hotmail.com
3	Quispe	Ramirez	Miguel	M	09-MAR-83	962905883	mquispe@gmail.com
4	Arauco	Arana	Carlos	M	11-FEB-00	962609096	carauco@gmail.com
5	Dallas	Miller	Ann	F	15-JUN-99	965918300	adallas@miempresa.com
6	Cueva	Vasquez	Kim	F	03-MAR-96	923950023	kim@hotmail.com
7	Theo	Monzen	Otto	M	25-JAN-83	987987987	otheo@pucp.edu.pe
8	Mezones	Estrada	Renzo	M	14-APR-96	88888888	renzo@gmail.com

Revisamos la tabla donde el trigger registró el resultado de la acción:

```
SELECT * FROM L5_LOG_CLIENTE;
```

ID_LOG_CLIE...	ID_CLIENTE	TELEFONO_ANT	TELEFONO_NUE	EMAIL_ANT	EMAIL_NUE	USUARIO	FECHA_REGISTRO	ACCION
1	8	(null)	88888888	(null)	renzo@gmail.com	INF24620202	28-NOV-20	I
2	7	99999999	987987987	otto.theo@pucp.edu.pe	otheo@pucp.edu.pe	INF24620202	28-NOV-20	U

Con esto podemos registrar la historia o trazabilidad de los cambios de una tabla.

Otra forma: usando parámetros en la sentencia UPDATING (si queremos que el trigger solo se dispare al actualizar un campo en particular):

```
CREATE OR REPLACE TRIGGER TR_TRAZA_CLIENTE
AFTER INSERT OR
UPDATE OF TELEFONO, EMAIL
ON L5_CLIENTE
FOR EACH ROW
DECLARE
  V_ID_LOG NUMBER;
BEGIN

  IF INSERTING THEN
    SELECT NVL(MAX(ID_LOG_CLIENTE), 0) INTO V_ID_LOG FROM L5_LOG_CLIENTE;

    INSERT INTO L5_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, TELEFONO_NUE,
EMAIL_NUE, USUARIO, FECHA_REGISTRO, ACCION)
    VALUES (V_ID_LOG + 1, :NEW.ID_CLIENTE, :NEW.TELEFONO, :NEW.EMAIL, USER,
SYSDATE, 'I');
  END IF;

  IF UPDATING('TELEFONO') THEN
    SELECT NVL(MAX(ID_LOG_CLIENTE), 0) INTO V_ID_LOG FROM L5_LOG_CLIENTE;

    INSERT INTO L5_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, TELEFONO_ANT,
TELEFONO_NUE, USUARIO, FECHA_REGISTRO, ACCION)
    VALUES (V_ID_LOG + 1, :NEW.ID_CLIENTE, :OLD.TELEFONO, :NEW.TELEFONO, USER,
SYSDATE, 'U');
  END IF;

  IF UPDATING('EMAIL') THEN
    SELECT NVL(MAX(ID_LOG_CLIENTE), 0) INTO V_ID_LOG FROM L5_LOG_CLIENTE;

    INSERT INTO L5_LOG_CLIENTE (ID_LOG_CLIENTE, ID_CLIENTE, EMAIL_ANT,
EMAIL_NUE, USUARIO, FECHA_REGISTRO, ACCION)
    VALUES (V_ID_LOG + 1, :NEW.ID_CLIENTE, :OLD.EMAIL, :NEW.EMAIL, USER,
SYSDATE, 'U');
  END IF;
END;
```