Curso
2020/21

ALBAEATS - BBDD DE UNA APP DE COMIDA RÁPIDA

ORACLE: DBDD_34 / QTAKXEVX
CARLOS BUENDÍA JIMÉNEZ – CARLOS.BUENDIA1@ALU.UCLM.ES
ALBERTO NOVILLO CHINCHILLA – ALBERTO.NOVILLO@ALU.UCLM.ES
ALFONSO MARTÍNEZ PIÑANGO – ALFONSO.MARTINEZ8@ALU.UCLM.ES

Índice

Imágenes

1. Funcionamiento del sistema

Se desea modelar una BD para guardar los datos necesarios relativos a una aplicación de envío comida a domicilio. Para ello:

- Se requiere almacenar todas las personas que han registrado en la aplicación (usuarios). De los usuarios se requiere almacenar: IdUsuario, Nombre, Apellidos, Teléfono, CorreoE, Ciudad, País. Dentro de usuario, se heredan dos ramas: cliente y repartidor.
    - Los clientes podrán realizar pedidos a los restaurantes seleccionando las ofertas que posean y seleccionando un tipo de pago. Para los clientes se requiere almacenar: Dirección, Código postal.
    - Los repartidores se encargarán de llevar el pedido a los clientes. Para los repartidores se requiere almacenar: DNI, Número seguridad social, Fecha de alta y la Fecha de baja.

- Para saber cómo va a pagar el cliente, se ha creado la clase método de pago que contiene un IdPago y una fecha del pago. De esta clase, heredan Contra reembolso y Tarjeta Crédito. Para Contra reembolso se almacena: un boleano si da propina y observaciones. Para Tarjeta de Crédito se almacena: Número, Fecha de Caducidad, CVV, Propietario.

- Por otra parte, los restaurantes se encargan de suministrar los productos necesarios para realizar el pedido que ha solicitado el cliente. Los datos necesarios para el restaurante son: Id_Restaurante, Nombre, Dirección, Codigo_Postal, Teléfono, TipoRestaurante, Hora de apertura, Hora de cierre, Calificación. Y para los productos que maneja se almacenan los siguientes datos: ID_Producto, Nombre, Descripción, Precio_Unit, Stock, TipoProducto.

  Además, algunos de los productos a la venta estarán en oferta y en caso de eliminar el producto, se eliminará también la oferta. La clase oferta almacenará: Código oferta, Descuento, MaximoDescuento, Finalización.

- El cliente selecciona los productos que desee, guardándose en la clase L.Pedido. Esta clase, contendrá el ID_LPedido y la cantidad. Cuando el cliente ha terminado de realizar su pedido, la información pasa de la L.Pedido al Pedido que almacena: Id_Pedido, Precio, Distancia, Fecha, Pagado, Urgencia, Estado. En caso de cancelarse el pedido, los datos de la clase pedido se eliminarían.

- Los repartidores tendrán asociado un vehículo. La clase vehículo almacenará: Matricula, Modelo, Marca, Disponibilidad, Peso. Además, tiene dos clases que heredan de esta: Gasolina y Eléctrico. Gasolina almacenará: TipoLicencia, Emisiones y Eléctrico almacenará: Autonomía y CargaMaxima.

- Por otra parte, si el vehículo se estropea, será llevado ante un mecánico para ser reparado. La clase mecánico almacenará: DNI, PeridoContratación, Nombre, Apellidos, NombreEmpresa.

2. Requisitos

2.1 Requisitos funcionales

- RF-1: Realizar pedido
- RF-2: Recibir ofertas
- RF-3: Seleccionar método de pago
- RF-4: Reparar vehículo
- RF-5: Ver el estado del pedido actual

2.2 Restricciones del sistema

- Un repartidor solo puede tener un vehículo.
- Un pedido está asociado a un solo repartidor.
- Si una línea de pedido se borra, se borrará también el pedido.
- Un restaurante está asociado a un producto (Ej. menú).
- Si se elimina el producto, se eliminarán las ofertas relacionadas a ese producto.

3. Diagrama entidad relación extendida y UML

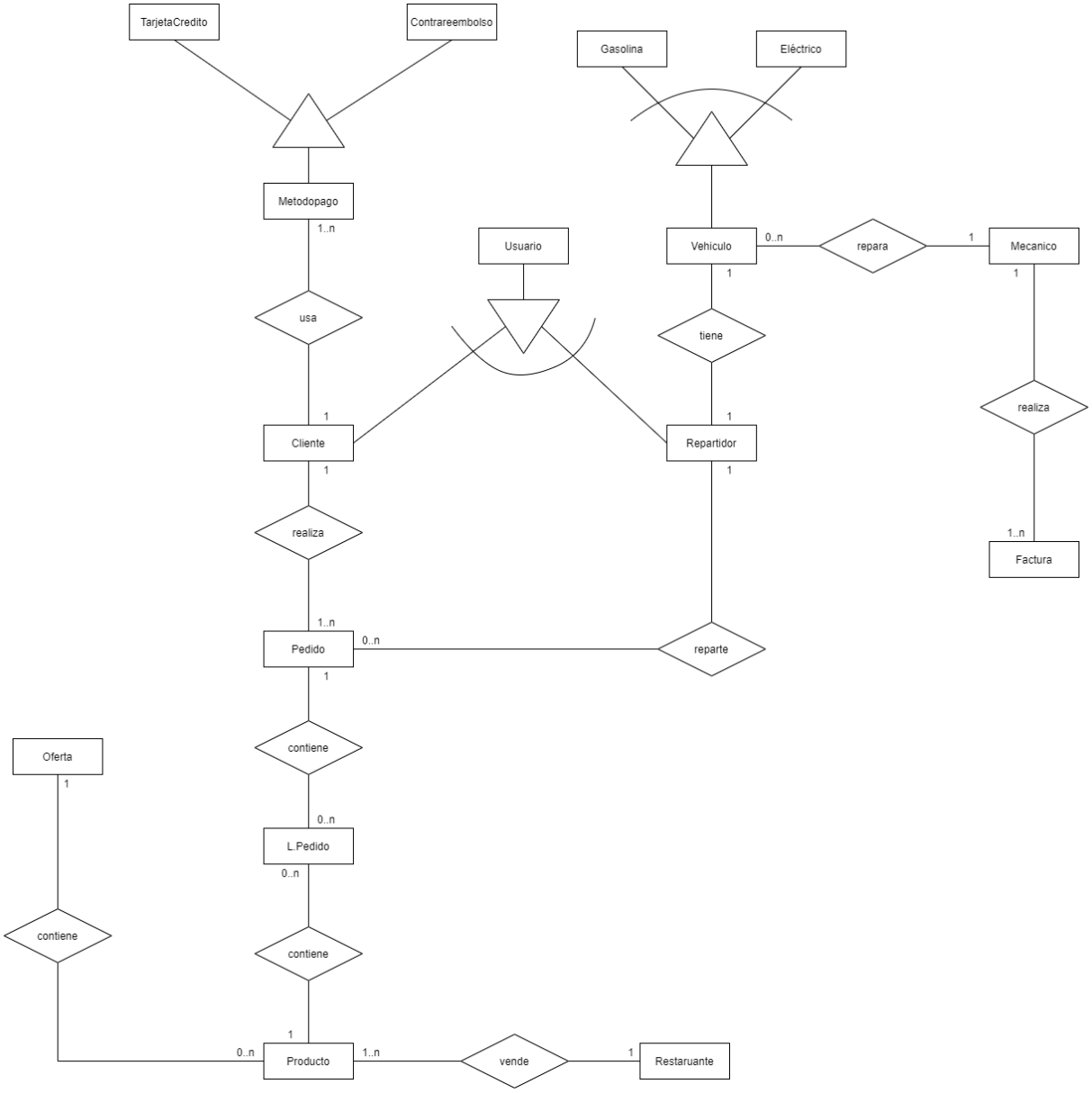


Ilustración 1. Diagrama Entidad-Relación

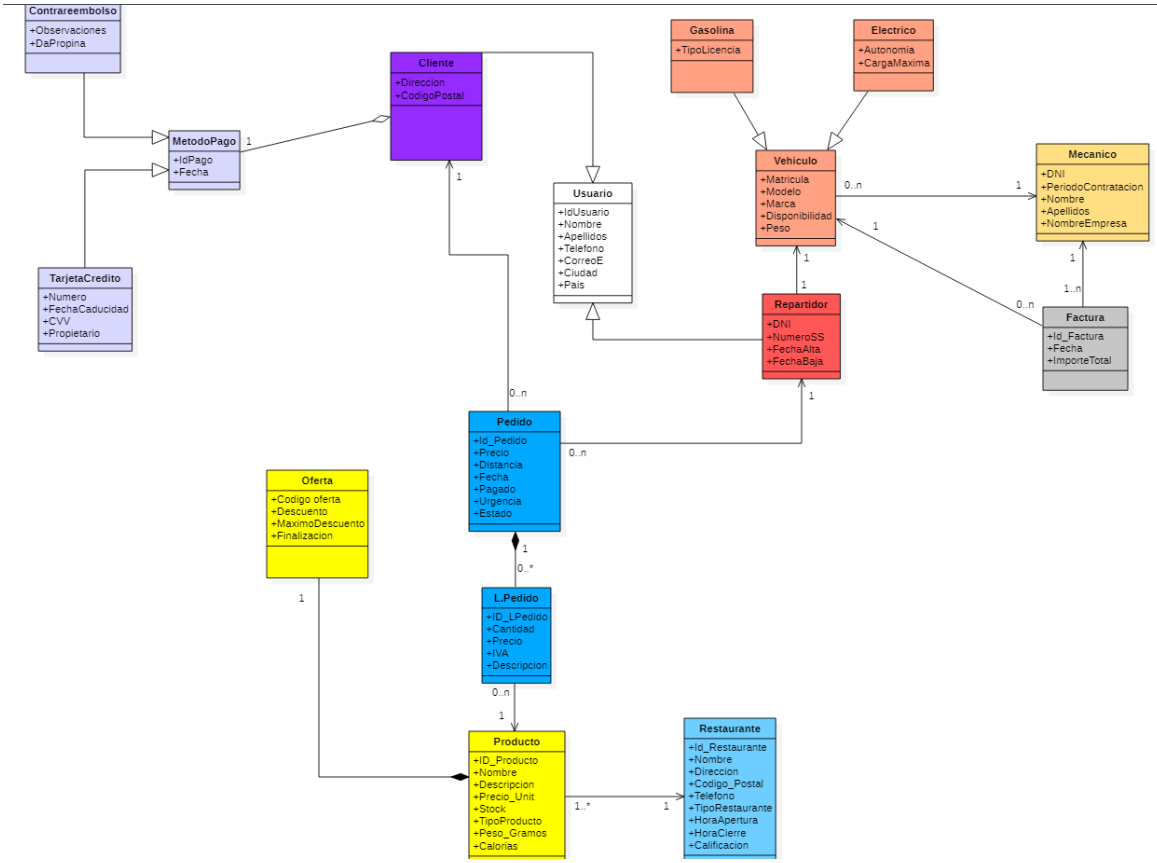


Ilustración 2. Diagrama UML

Diagrama UML que representa las clases creadas en la base de datos. Cada color corresponde con una tabla en la base de datos.

4. Script de type

```
DROP TYPE CLIENTE_OBJ FORCE;/
DROP TYPE CONTRAREEMBOLSO_OBJ FORCE;/
DROP TYPE FACTURA_OBJ FORCE;/
DROP TYPE LPEDIDO_OBJ FORCE;/
DROP TYPE MECANICO_OBJ FORCE;/
DROP TYPE METODOPAGO_OBJ FORCE;/
DROP TYPE PEDIDO_OBJ FORCE;/
DROP TYPE PRODUCTO_OBJ FORCE;/
DROP TYPE REPARTIDOR_OBJ FORCE;/
DROP TYPE RESTAURANTE_OBJ FORCE;/
DROP TYPE TARJETA_OBJ FORCE;/
DROP TYPE USUARIO_OBJ FORCE;/
```

```
DROP TYPE VEHELECTRICO_OBJ FORCE;/
DROP TYPE VEHGASOLINA_OBJ FORCE;/
DROP TYPE VEHICULO_OBJ FORCE;/
DROP TYPE LFACTURA_OBJ FORCE;/
DROP TYPE LPEDIDO_NTABTYP FORCE;/
DROP TYPE OFERTA_OBJ FORCE;/
DROP TYPE OFERTA_NTABTYP FORCE;/

CREATE OR REPLACE TYPE RESTAURANTE_OBJ AS OBJECT(
id_restaurante NUMBER(10,0),
nombre VARCHAR2(20),
direccion VARCHAR2(200),
ciudad VARCHAR(30),
codigo_postal NUMBER(5),
telefono NUMBER(9),
tipo_restaurante VARCHAR(30),
hora_apertura TIMESTAMP (1),
hora_cierre TIMESTAMP(1),
calificacion NUMBER(1),
ORDER MEMBER FUNCTION CompareRestaurantes(r RESTAURANTE_OBJ) RETURN INTEGER
);
/


CREATE OR REPLACE TYPE BODY RESTAURANTE_OBJ AS
        ORDER MEMBER FUNCTION CompareRestaurantes(r RESTAURANTE_OBJ) RETURN INTEGER IS
        BEGIN
                IF id_restaurante = r.id_restaurante THEN
                        IF nombre< r.nombre THEN RETURN 1;
                        ELSIF nombre > r.nombre THEN RETURN -1;
                        ELSE RETURN 0;
                        END IF;
                ELSE
                        IF id_restaurante < r.id_restaurante THEN RETURN 1;
                        ELSIF id_restaurante > r.id_restaurante THEN RETURN -1;
                        ELSE RETURN 0;
                        END IF;
                END IF;
        END;
END;
/

CREATE OR REPLACE TYPE OFERTA_OBJ AS OBJECT(
codigo_oferta NUMBER(6),
descuento NUMBER(2),
maximo_descuento NUMBER(2),
finalizacion DATE
);
/

CREATE TYPE OFERTA_NTABTYP AS TABLE OF OFERTA_OBJ;
/

CREATE OR REPLACE TYPE PRODUCTO_OBJ AS OBJECT(
id_producto NUMBER(10),
nombre VARCHAR(20),
descripcion VARCHAR2(200),
precio_unit NUMBER(6,2),
```

```
stock NUMBER(6),
tipo_producto VARCHAR2(40),
peso_gramos NUMBER(3),
calorias NUMBER(3),
restaurante REF RESTAURANTE_OBJ,
oferta OFERTA_NTABTYP
);
/


CREATE OR REPLACE TYPE MECANICO_OBJ AS OBJECT(
   dni VARCHAR2(9),
   nombre VARCHAR2(20),
   apellidos VARCHAR2(30),
   periodo DATE,
   empresa VARCHAR2(30)
);
/
CREATE OR REPLACE TYPE LPEDIDO_OBJ AS OBJECT(
id_lpedido NUMBER(6),
cantidad NUMBER(3),
precio NUMBER(3),
iva NUMBER(2),
descripcion VARCHAR(200),
producto REF PRODUCTO_OBJ
);
/
CREATE TYPE LPEDIDO_NTABTYP AS TABLE OF LPEDIDO_OBJ;
/


CREATE OR REPLACE TYPE USUARIO_OBJ AS OBJECT(
id_usuario NUMBER(6),
nombre VARCHAR2(30),
apellidos VARCHAR2(50),
telefono NUMBER(9),
correoE VARCHAR2(60),
ciudad VARCHAR2(20),
pais VARCHAR2(20)
)NOT FINAL;
/




CREATE OR REPLACE TYPE METODOPAGO_OBJ AS OBJECT(
   idpago NUMBER(10),
   fecha DATE
)NOT FINAL;
/


CREATE OR REPLACE TYPE TARJETA_OBJ UNDER METODOPAGO_OBJ (
   numero NUMBER(16),
   fecha_caducidad NUMBER(4),
   cvv NUMBER(3),
   propiertario VARCHAR2(50)
);
```

```
/

CREATE OR REPLACE TYPE CONTRAREEMBOLSO_OBJ UNDER METODOPAGO_OBJ (
   observaciones VARCHAR2(300),
   daPropina NUMBER(1)
);
/
CREATE OR REPLACE TYPE CLIENTE_OBJ UNDER USUARIO_OBJ(
direccion VARCHAR2(60),
codigo_postal NUMBER(5),
metodoPago REF METODOPAGO_OBJ
);
/

CREATE OR REPLACE TYPE VEHICULO_OBJ AS OBJECT(
   matricula VARCHAR2(7),
   modelo VARCHAR2(20),
   marca VARCHAR2(30),
   disponibilidad NUMBER(2,0),
   peso NUMBER(7,2),
   mecanico REF MECANICO_OBJ,
        ORDER MEMBER FUNCTION CompareMatricula(v VEHICULO_OBJ) RETURN INTEGER

)NOT FINAL;
/

CREATE OR REPLACE TYPE BODY VEHICULO_OBJ AS
        ORDER MEMBER FUNCTION CompareMatricula(v VEHICULO_OBJ) RETURN INTEGER IS
        BEGIN
                IF matricula < v.matricula THEN
                        RETURN -1;
                ELSIF matricula > v.matricula THEN
                        RETURN 1;
                ELSE
                        RETURN 0;
                END IF;
        END;
END;
/


CREATE OR REPLACE TYPE VEHGASOLINA_OBJ UNDER VEHICULO_OBJ (
   tipolicencia VARCHAR(2),
   emisiones NUMBER(5,2)
);
/

CREATE OR REPLACE TYPE VEHELECTRICO_OBJ UNDER VEHICULO_OBJ (
   autonomia NUMBER(3,0),
   emisiones NUMBER(5,2)
);
/
CREATE OR REPLACE TYPE REPARTIDOR_OBJ UNDER USUARIO_OBJ (
   dni VARCHAR2(9),
   numeross NUMBER(12,0),
   fechaalta DATE,
   fechabaja DATE,
   vehiculo REF VEHICULO_OBJ
```

```
);
/


CREATE OR REPLACE TYPE PEDIDO_OBJ AS OBJECT(
id_pedido NUMBER(9),
precio NUMBER(5,2),
distancia NUMBER(6,2),
fecha DATE,
pagado NUMBER(1),
urgencia NUMBER(1),
estado VARCHAR2(20),
lpedido LPEDIDO_NTABTYP,
repartidor REF REPARTIDOR_OBJ,
pedido REF CLIENTE_OBJ
);
/

CREATE OR REPLACE TYPE FACTURA_OBJ AS OBJECT(
   id_factura NUMBER(10,0),
   descripcion VARCHAR2(50),
   importe NUMBER(8,2),
   mecanico REF MECANICO_OBJ,
   vehiculo REF VEHICULO_OBJ
);
/
```

5. Script de las tablas

```
DROP TABLE RESTAURANTE_TAB FORCE;/
DROP TABLE MECANICO_TAB FORCE;/
DROP TABLE METODOPAGO_TAB FORCE;/
DROP TABLE PRODUCTO_TAB FORCE;/
DROP TABLE OFERTA_TAB FORCE;/
DROP TABLE CLIENTE_TAB FORCE;/
DROP TABLE VEHICULO_TAB FORCE;/
DROP TABLE REPARTIDOR_TAB FORCE;/
DROP TABLE PEDIDO_TAB FORCE;/
DROP TABLE LINEASPEDIDO_TAB FORCE;/
DROP TABLE FACTURA_TAB FORCE;/
DROP TABLE LRECIBO_TAB FORCE;/

CREATE TABLE RESTAURANTE_TAB OF RESTAURANTE_OBJ (
   id_restaurante PRIMARY KEY,
   nombre NOT NULL,
   direccion NOT NULL,
   ciudad NOT NULL,
   telefono NOT NULL
);/

CREATE TABLE MECANICO_TAB OF MECANICO_OBJ (
   dni PRIMARY KEY,
   nombre NOT NULL,
   empresa NOT NULL
);/
```

```
CREATE TABLE METODOPAGO_TAB OF METODOPAGO_OBJ(
   idpago PRIMARY KEY,
   fecha NOT NULL
);/

CREATE TABLE PRODUCTO_TAB OF PRODUCTO_OBJ (
   id_producto PRIMARY KEY,
   nombre NOT NULL,
   CHECK(precio_unit >0),
   CHECK(stock >0),
   restaurante SCOPE IS RESTAURANTE_TAB)
   NESTED TABLE oferta STORE AS OFERTA_TAB;

ALTER TABLE OFERTA_TAB ADD (PRIMARY KEY (codigo_oferta ),    CHECK (descuento>0),
   CHECK (maximo_descuento>0));

CREATE TABLE CLIENTE_TAB OF CLIENTE_OBJ (
   id_usuario  PRIMARY KEY,
   nombre NOT NULL,
   telefono NOT NULL,
   correoE NOT NULL,
   ciudad NOT NULL,
   direccion NOT NULL,
   codigo_postal NOT NULL,
   SCOPE FOR (metodoPago) IS METODOPAGO_TAB

);/



CREATE TABLE VEHICULO_TAB OF VEHICULO_OBJ(
   matricula PRIMARY KEY,
   SCOPE FOR (mecanico) IS MECANICO_TAB
);/


CREATE TABLE REPARTIDOR_TAB OF REPARTIDOR_OBJ(
   id_usuario  PRIMARY KEY,
   nombre NOT NULL,
   telefono NOT NULL,
   correoE NOT NULL,
   ciudad NOT NULL,
   dni  UNIQUE,
   numeross UNIQUE,
   fechaalta NOT NULL,
   SCOPE FOR (vehiculo) IS VEHICULO_TAB
);/



CREATE TABLE PEDIDO_TAB OF PEDIDO_OBJ(
   id_pedido PRIMARY KEY,
```

```
    precio NOT NULL,
    fecha NOT NULL,
    urgencia NOT NULL,
    CHECK (urgencia > 0),
    CHECK (urgencia <=4),
    SCOPE FOR (repartidor) IS REPARTIDOR_TAB)
    NESTED TABLE lpedido STORE AS LINEASPEDIDO_TAB;/

ALTER TABLE LINEASPEDIDO_TAB ADD (SCOPE FOR (producto) IS  PRODUCTO_TAB, PRIMARY KEY (id_lpedido),
CHECK(cantidad > 0));/



CREATE TABLE FACTURA_TAB OF FACTURA_OBJ(
    id_factura PRIMARY KEY,
    importe NOT NULL,
    CHECK(importe >= 0),
    SCOPE FOR (mecanico) IS MECANICO_TAB,
    SCOPE FOR (vehiculo) IS VEHICULO_TAB
    );/
```

6. Scripts de los Insert

```
INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(1,'Pizzeria Antonio','Calle Falsa 123','Madrid', 00037, 666444777, 'Pizzeria', TO_DATE ('20:00:00',
'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 7)); /

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(2,'Dominos Pizza','Avenida España','Albacete', 15637, 644775693, 'Pizzeria', TO_DATE ('20:00:00',
'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 4));/


INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(3,'Big Bang Burger','Calle Nueva','Cuenca', 13695, 699885214, 'Comida rápida', TO_DATE ('20:00:00',
'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 8));
/

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(4,'KFC','Calle Imaginalia','Albacete', 15695, 655325214, 'Comida rápida', TO_DATE ('12:00:00', 'HH24:MI:SS'),
TO_DATE ('17:30:00', 'HH24:MI:SS'), 9));
/

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(5,'Taco Bell','Calle Princesa','Albacete', 15644, 688521499, 'Comida rápida', TO_DATE ('14:00:00',
'HH24:MI:SS'), TO_DATE ('17:30:00', 'HH24:MI:SS'), 9));
/

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(6,'WOK','Calle Antonio Machado ','Madrid', 00044, 652149988, 'Comida asiatica', TO_DATE ('20:00:00',
'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 7));
/
```

```
INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(7,'Honk Kong','Avenida de los Reyes Catolicos','Cuenca', 13674, 688149952, 'Comida asiatica', TO_DATE
('20:00:00', 'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 5));
/

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(8,'Restaurante barrio','Calle Fermin Caballero','Cuenca', 13616, 677164237, 'Restaurante', TO_DATE
('13:00:00', 'HH24:MI:SS'), TO_DATE ('16:30:00', 'HH24:MI:SS'), 3));
/

INSERT INTO RESTAURANTE_TAB VALUES (RESTAURANTE_OBJ
(9,'Restaurante Poli','Paseo de los Estudiantes ','Albacete', 15617, 646275978, 'Restaurante',TO_DATE
('20:00:00', 'HH24:MI:SS'), TO_DATE ('23:30:00', 'HH24:MI:SS'), 9));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(1, 'Pizza BBQ', 'Pizza con queso, jamon y salsa barbacoa', 12.00, 40, 'Pizza', 310, 210, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '1' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(2, 'Pizza Carbonara', 'Pizza con queso, jamon, tomate, peperoni y aceitunas', 11.00, 30, 'Pizza', 340, 220,
(SELECT REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '1' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(3, 'Pizza Jamon y queso', 'Pizza con queso, jamon y tomate',8.00, 30, 'Pizza', 250, 190, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '2' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(4, 'Pizza Cuatro quesos', 'Pizza con varios quesos y tomate', 8.00, 50, 'Pizza', 200, 210, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '2' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(5, 'H queso', 'Hamburgesa con queso, lechuga y tomate', 2.50, 90, 'Carne',300, 190, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '3' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(6, 'H Big Bang', '3 hamburgesas con queso, lechuga, tomate y huevo', 3.99, 40, 'Carne', 500, 230, (SELECT
REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '3' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(7, 'Alitas de pollo', 'Racion de 12 alitas de pollo', 2.99, 60, 'Carne', 400, 240, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '4' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(8, 'Muslos de pollo', 'Racion de 12 muslos de pollo', 3.99, 45, 'Carne', 450, 240, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '4' ), OFERTA_NTABTYP()));
```

```
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(9, 'Taco', 'Taco con carne picada, verduras y salsa a elegir', 2.99, 70, 'Carne y verduras', 250, 220, (SELECT
REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '5' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(10, 'Taco picante', 'Taco con carne picada, verduras y guacamole', 4.99, 63, 'Carne y verduras', 350, 240,
(SELECT REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '5' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(11, 'Pollo al limón', 'Pollo con salsa al limón', 5.90, 125, 'Carne', 350, 215, (SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '6' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(12, 'Rollitos primavera', 'Ración de 2 piezas de hojaldre relleno de verduras. Incluye salsa agridulce o soja',
4.90, 250, 'Verdura',225, 205, (SELECT REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '6' ),
OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(13, 'Sushi', 'Racion de 6 trozos de pescado fresco sin cocinar con arroz', 4.57, 97, 'Pescado', 142, 182, (SELECT
REF(r) FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '7' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(14, 'Curry', 'Salsa especiada con arroz', 3.57, 230, 'Arroz', 160, 189, (SELECT REF(r) FROM RESTAURANTE_TAB r
WHERE r.ID_RESTAURANTE = '7' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(15, 'Filete con patatas', 'Filete de ternera con una racion de patatas', 7.50, 35, 'Carne', 300, 215,(SELECT REF(r)
FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '8' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(16, 'Cordero asado', 'Cordero asado con una racion de patatas', 8.57, 27, 'Carne', 260, 209,(SELECT REF(r)
FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '8' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(17, 'Paella', 'Arroz con trozos de pollo, pimiento, gisantes y gambas', 4.20, 40, 'Arroz', 220, 200, (SELECT REF(r)
FROM RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '9' ), OFERTA_NTABTYP()));
/

INSERT INTO PRODUCTO_TAB VALUES (PRODUCTO_OBJ
(18, 'Pechugas con salsa', 'Pechugas con salsa de setas', 4.57, 45, 'Carne', 260, 209,(SELECT REF(r) FROM
RESTAURANTE_TAB r WHERE r.ID_RESTAURANTE = '9' ), OFERTA_NTABTYP()));
/
```

```
INSERT INTO TABLE (SELECT oferta FROM PRODUCTO_TAB  WHERE id_producto  =1 ) VALUES(OFERTA_OBJ
(000001, 15, 30, TO_DATE ('2021/11/12', 'yyyy/mm/dd,' )));
/

INSERT INTO TABLE (SELECT oferta FROM PRODUCTO_TAB  WHERE id_producto  =3 ) VALUES(OFERTA_OBJ
(000031, 10, 24, TO_DATE ('2021/11/12', 'yyyy/mm/dd,' )));
/

INSERT INTO TABLE (SELECT oferta FROM PRODUCTO_TAB  WHERE id_producto  =7 ) VALUES(OFERTA_OBJ
(000200, 10, 10, TO_DATE ('2021-08-27', 'yyyy/mm/dd,' )));
/

INSERT INTO TABLE (SELECT oferta FROM PRODUCTO_TAB  WHERE id_producto  =8 ) VALUES(OFERTA_OBJ
(000456, 20, 40, TO_DATE ('2021/06/03', 'yyyy/mm/dd,' )));
/

INSERT INTO TABLE (SELECT oferta FROM PRODUCTO_TAB  WHERE id_producto  =14 ) VALUES(OFERTA_OBJ
(456781, 5, 25,  TO_DATE ('2021/11/1', 'yyyy/mm/dd,' )));
/

INSERT INTO MECANICO_TAB VALUES (MECANICO_OBJ
('45678321L', 'Fernando', 'Perez Fernandez', TO_DATE('31/01/2025', 'dd/mm/yyyy'), 'GenRush'));
/

INSERT INTO MECANICO_TAB VALUES (MECANICO_OBJ
('56217971E', 'Manolo', 'Gomez Romero', TO_DATE('30/06/2026', 'dd/mm/yyyy'), 'Motores manolo'));
/

INSERT INTO MECANICO_TAB VALUES (MECANICO_OBJ
('41247895J', 'Diego', 'Rodrigez Lopez', TO_DATE('30/06/2026', 'dd/mm/yyyy'), 'Reparaciones rodrigez'));
/

INSERT INTO METODOPAGO_TAB VALUES (CONTRAREEMBOLSO_OBJ
(1, TO_DATE('21/07/2021 21:00:00', 'dd/mm/yyyy hh24:mi:ss'), '', 0 ));
/

INSERT INTO METODOPAGO_TAB VALUES (CONTRAREEMBOLSO_OBJ
(2, TO_DATE('21/07/2021 20:00:00', 'dd/mm/yyyy hh24:mi:ss'), 'Habitacion 273', 1));
/

INSERT INTO METODOPAGO_TAB VALUES (TARJETA_OBJ
(3, TO_DATE('21/07/2021 20:30:00', 'dd/mm/yyyy hh24:mi:ss'),  1111222233334444, 0725, 111, 'Antonio'));
/

INSERT INTO METODOPAGO_TAB VALUES (TARJETA_OBJ
(4, TO_DATE('12/03/2021 20:30:00', 'dd/mm/yyyy hh24:mi:ss'), 5362133333334444, 0227, 934, 'Don Pepe'));
/

INSERT INTO METODOPAGO_TAB VALUES (TARJETA_OBJ
(5, TO_DATE('14/03/2021 15:00:00', 'dd/mm/yyyy hh24:mi:ss'), 5325452663534345, 0227, 934, 'Don
Valentiniano'));
/
```

```
INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
(1, 'Antonio', 'Perez Gomez', 612345679, 'antoniopg@gmail.com', 'Albacete', 'España', 'Calle 111', 00202,
(SELECT REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = 1 )));
/

INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
(2, 'Lucia', 'Fajardo Gonzalez', 645782391, 'luciafg@gmail.com', 'Albacete', 'España', 'Calle 222', 00202, (SELECT
REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = 2 )));
/

INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
(3, 'Mario', 'Gomez Martinez', 698754324, 'mariogm@gmail.com', 'Cuenca', 'España', 'Calle 333', 13655,
(SELECT REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = 3 )));
/

INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
(4, 'Alicia', 'Romero Tortola', 677248798, 'aliciart@gmail.com', 'Cuenca', 'España', 'Calle 444', 13684, (SELECT
REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = 4 )));
/

INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
(5, 'Rebeca', 'Navarro Gomez', 654778968, 'rebecang@gmail.com', 'Madrid', 'España', 'Calle 555', 00045,
(SELECT REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = 5 )));
/



INSERT INTO VEHICULO_TAB VALUES ( VEHGASOLINA_OBJ
('1111abc', '245RP4', 'Suzuki', 1 , 210.96, (SELECT REF(m) FROM MECANICO_TAB m WHERE m.dni =
'45678321L' ),'B', 250.12));
/

INSERT INTO VEHICULO_TAB VALUES ( VEHGASOLINA_OBJ
('2222def', '946LT7', 'Renault', 1 , 200.12, (SELECT REF(m) FROM MECANICO_TAB m WHERE m.dni =
'56217971E' ),'C', 305.45));
/

INSERT INTO VEHICULO_TAB VALUES( VEHELECTRICO_OBJ
('3333ghi', '327TFT7', 'Toyota', 1 , 225.47, (SELECT REF(m) FROM MECANICO_TAB m WHERE m.dni =
'45678321L' ),300, 100.02));
/

INSERT INTO VEHICULO_TAB VALUES( VEHELECTRICO_OBJ
('4444jkl', '429BC45', 'BMW', 1 , 213.23, (SELECT REF(m) FROM MECANICO_TAB m WHERE m.dni = '41247895J'
),200, 075.43));
/

INSERT INTO REPARTIDOR_TAB VALUES( REPARTIDOR_OBJ
(6, 'Juan', 'Gonzalez Romero', 673215984, 'juangr@gmail.com', 'Albacete', 'España',
'44444447F',444444444444, TO_DATE('21/01/2021', 'dd/mm/yyyy'), TO_DATE('21/01/2022', 'dd/mm/yyyy'),
(SELECT REF(V) FROM VEHICULO_TAB v WHERE v.matricula = '3333ghi')));
/
```

```
INSERT INTO REPARTIDOR_TAB VALUES( REPARTIDOR_OBJ
(7, 'Marta', 'Sevilla Martinez', 628497533, 'martasm@gmail.com', 'Cuenca', 'España',
'22222224T',222222222222, TO_DATE('16/03/2021', 'dd/mm/yyyy'), TO_DATE('16/03/2022', 'dd/mm/yyyy'),
(SELECT REF(V) FROM VEHICULO_TAB v WHERE v.matricula = '4444jkl')));
/

INSERT INTO REPARTIDOR_TAB VALUES( REPARTIDOR_OBJ
(8, 'Pedro', 'Plaza Fernandez', 629477821, 'pedropf@gmail.com', 'Madrid', 'España',
'33333339N',333333333333, TO_DATE('09/01/2021', 'dd/mm/yyyy'), TO_DATE('09/06/2022', 'dd/mm/yyyy'),
(SELECT REF(V) FROM VEHICULO_TAB v WHERE v.matricula = '1111abc')));
/

INSERT INTO REPARTIDOR_TAB VALUES( REPARTIDOR_OBJ
(9, 'Antonio', 'Martinez Fernandez', 623748202, 'antoniooof@gmail.com', 'Albacete', 'España',
'2232333j',232332323, TO_DATE('09/01/2021', 'dd/mm/yyyy'), TO_DATE('09/06/2022', 'dd/mm/yyyy'),
(SELECT REF(V) FROM VEHICULO_TAB v WHERE v.matricula = '4444jkl')));
/

INSERT INTO PEDIDO_TAB VALUES( PEDIDO_OBJ
(1, 12.30,12, TO_DATE('21/07/2021 22:00:00', 'dd/mm/yyyy hh24:mi:ss') ,0,2, 'en
camino',LPEDIDO_NTABTYP(), (SELECT REF(r) FROM REPARTIDOR_TAB r WHERE r.ID_USUARIO = 6),(SELECT
REF(c) FROM CLIENTE_TAB c WHERE c.ID_USUARIO = 1) ));
/

INSERT INTO PEDIDO_TAB VALUES(PEDIDO_OBJ
(2, 21.30,12, TO_DATE('21/07/2021 23:00:00', 'dd/mm/yyyy hh24:mi:ss') ,1,1, 'en
preparacion',LPEDIDO_NTABTYP(), (SELECT REF(r) FROM REPARTIDOR_TAB r WHERE r.ID_USUARIO =
6),(SELECT REF(c) FROM CLIENTE_TAB c WHERE c.ID_USUARIO = 2) ));
/

INSERT INTO PEDIDO_TAB VALUES(PEDIDO_OBJ
(3, 37.16,3.4, TO_DATE('31/07/2021 12:00:00', 'dd/mm/yyyy hh24:mi:ss') ,1,1,
'completado',LPEDIDO_NTABTYP(), (SELECT REF(r) FROM REPARTIDOR_TAB r WHERE r.ID_USUARIO =
8),(SELECT REF(c) FROM CLIENTE_TAB c WHERE c.ID_USUARIO = 4) ));
/

INSERT INTO PEDIDO_TAB VALUES(PEDIDO_OBJ
(4, 13.2,5.3, TO_DATE('12/03/2021 12:00:00', 'dd/mm/yyyy hh24:mi:ss') ,1,3,
'completado',LPEDIDO_NTABTYP(), (SELECT REF(r) FROM REPARTIDOR_TAB r WHERE r.ID_USUARIO =
7),(SELECT REF(c) FROM CLIENTE_TAB c WHERE c.ID_USUARIO = 3) ));
/

INSERT INTO PEDIDO_TAB VALUES(PEDIDO_OBJ
(5, 11.30,2.7, TO_DATE('21/07/2021 23:00:00', 'dd/mm/yyyy hh24:mi:ss') ,0,1,
'completado',LPEDIDO_NTABTYP(), (SELECT REF(r) FROM REPARTIDOR_TAB r WHERE r.ID_USUARIO =
6),(SELECT REF(c) FROM CLIENTE_TAB c WHERE c.ID_USUARIO = 2) ));
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO ='1') VALUES(
1, 1, 13.45, 12, 'x1', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 1)
);
```

```
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO ='1') VALUES(
2, 3, 4.10, 12, 'x4', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 5)
);
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO ='2') VALUES(
3, 2, 8.45, 12, 'X2', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 4)
);
/
INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO =3) VALUES(
4, 4, 2.35, 13, 'X4', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 4)
);
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO =4) VALUES(
5, 1, 5.5, 13, 'X1', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 1)
);
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO =4) VALUES(
6, 1, 8.3, 12, 'X1', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 9)
);
/

INSERT INTO TABLE (SELECT p.LPEDIDO
FROM PEDIDO_TAB p WHERE p.ID_PEDIDO =4) VALUES(
7, 1, 2.3, 12, 'X1', (SELECT REF(pro)FROM PRODUCTO_TAB pro
WHERE pro.ID_PRODUCTO = 11)
);
/

INSERT INTO FACTURA_TAB VALUES ( FACTURA_OBJ
(000001, 'Cambio ruedas', 450.52,(SELECT REF(m) FROM MECANICO_TAB m WHERE m.DNI = '56217971E'),
(SELECT REF(v) FROM VEHICULO_TAB v WHERE v.matricula = '4444jkl')));
/

INSERT INTO FACTURA_TAB VALUES ( FACTURA_OBJ
(000002, 'Rep. motor', 1200.49,(SELECT REF(m) FROM MECANICO_TAB m WHERE m.DNI = '56217971E'),
(SELECT REF(v) FROM VEHICULO_TAB v WHERE v.matricula = '3333ghi')));
/
```

```
INSERT INTO FACTURA_TAB VALUES ( FACTURA_OBJ
(000003, 'Rep. dirección', 450.52,(SELECT REF(m) FROM MECANICO_TAB m WHERE m.DNI = '41247895J'),
(SELECT REF(v) FROM VEHICULO_TAB v WHERE v.matricula = '2222def')));
/
```

7. Consultas y vistas.

Consultas Alberto Novillo Chinchilla

1. OFERTAS_HOY
   Vista que selecciona los productos con una oferta cuya fecha de caducidad es hoy (Para obtener resultados se ha puesto un día en específico). También se muestra el nombre del restaurante del que procede el producto.

   ```
   CREATE OR REPLACE VIEW OFERTAS_HOY AS
   SELECT oft.codigo_oferta AS CODIGO, pro.nombre AS PRODUCTO, oft.descuento,
   oft.finalizacion AS FECHA, VALUE(pro).restaurante.nombre AS RESTAURANTE
   FROM producto_tab pro, TABLE(pro.oferta) oft
   WHERE oft.finalizacion = '12/11/21';
   ```

2. VEHICULOS_SIN_USO
   Vista que slecciona aquellos vehículos que aun no han realizado ningún reparto. Muestra la matricula, marca y modelo y el nombre del repartidor asociado al vehículo.

   ```
   CREATE VIEW VEHICULOS_SIN_USO AS
   SELECT value(r).vehiculo.matricula as Matricula, value(r).vehiculo.marca as Marca,
   value(r).vehiculo.modelo as Modelo, r.nombre
   FROM repartidor_tab r WHERE id_usuario NOT IN (
   SELECT value(p).repartidor.id_usuario FROM PEDIDO_TAB p    GROUP BY
   value(p).repartidor.id_usuario
   );
   ```

3. REPARTIDORES_CON_MAS_KM
   Vista que muestra el repartidor con más kilómetros acumulado. Muestra el nombre, numero de la SS y el total de kilómetros acumulados. Solo se calcularan los kilómetros de aquellos pedidos cuyo estado sea ‘en camino’ o ‘completado’.

   ```
   CREATE OR REPLACE VIEW REPARTIDORES_CON_MAS_KM AS
   SELECT  value(ped).repartidor.nombre AS NOMBRE_REPARTIDOR,
   value(ped).repartidor.numeross AS NUMERO_SS, ROUND(sum(ped.distancia), 3) as
   KM_TOTALES
   FROM PEDIDO_TAB ped
   WHERE ped.estado='en camino' or ped.estado='completado no pagado' or ped.estado =
   'completado' GROUP BY value(ped).repartidor
   ORDER BY KM_TOTALES DESC
   FETCH FIRST 1 ROWS ONLY;
   ```

4. LINEAS_PEDIDO_TOTALES
   Vista que calcula el numero de líneas que tiene un pedido. Muestra la información del pedido, del cliente (nombre) y del repartidor (nombre), así como las líneas totales del producto.

   ```
   CREATE OR REPLACE VIEW LINEAS_PEDIDO_TOTALES AS
   SELECT ped.id_pedido, ped.fecha, ped.pagado, ped.precio, value(ped).pedido.nombre AS
   CLIENTE, value(ped).repartidor.nombre AS REPARTIDOR,  COUNT(*) AS LINEAS
   FROM PEDIDO_TAB ped,
   TABLE (SELECT lpedido FROM PEDIDO_TAB WHERE id_pedido = ped.id_pedido)lp
   GROUP BY ped.id_pedido, ped.fecha, ped.pagado, ped.precio, value(ped).pedido.nombre,
   value(ped).repartidor.nombre
   ```

5. CLIENTES_CON_TARJETA
   Almacena en una vista los clientes que atualmente tienen como tarjeta su pago establecido. Muestra la información más importante del cliente y los datos de su tarjeta.

   ```
   CREATE OR REPLACE VIEW CLIENTE_CON_TARJETA AS
   SELECT nombre, apellidos, telefono, correoE, direccion,
   TREAT(DEREF(metodopago)AS TARJETA_OBJ).numero as numeroTarjeta,
   TREAT(DEREF(metodopago)AS TARJETA_OBJ).cvv as CVV,
   TREAT(DEREF(metodopago)AS TARJETA_OBJ).fecha_caducidad as FechaCaducidad,
   TREAT(DEREF(metodopago)AS TARJETA_OBJ).propiertario as Propietario
   FROM CLIENTE_TAB clie
   WHERE value(clie).metodopago.idpago  IN (
      SELECT TREAT(VALUE(mp) AS tarjeta_obj).idpago FROM metodopago_tab mp
      WHERE TREAT(VALUE(mp) AS tarjeta_obj).numero IS NOT NULL
   )
   ```

   NOTA: La función compila y ejecuta perfectamente, obteniendo todos los datos satisfactoriamente, pero a la hora de mostrar los datos de la vista, ORACLE da un error 01445:

   ORA-01445: Cannot Select ROWID from a Join View without a Key-Preserved Table

   Esta vista es necesaria para un *trigger*, por lo que el error ha sido omitido.

6. Eliminar pedidos ya completados.
   Esta consulta PL/SQL elimina los pedidos de la Base de Datos cuyo estado sea completado. Para evitar perder información de pedidos que aún quedan por pagar, comprueba si el pedido está pagado o no. En caso de que este pagado lo elimina y si no lo está, cambia su estado a ‘completado no pagado’.

```
DECLARE
CURSOR mycursor IS
SELECT id_pedido, estado, pagado
FROM PEDIDO_TAB
FOR UPDATE NOWAIT;
BEGIN
  FOR ped IN mycursor LOOP
    IF (ped.estado = 'completado' or ped.estado = 'completado no pagado' ) THEN
      IF (ped.pagado = 0) THEN
        UPDATE PEDIDO_TAB SET estado = 'completado no pagado' WHERE CURRENT OF mycursor;
      ELSE
        DELETE FROM PEDIDO_TAB WHERE CURRENT OF mycursor;
      END IF;
    END IF;
  END LOOP; -- se ejecuta close implícitamente
END;
```

Consultas Carlos Buendía Jiménez

1. En esta consulta, se puestran la id del pedido, su precio, si acutalimente se encuentra pagado o no, y el estado de todos aquellos pedidos, los cuales cuesten mas de 10€

```
CREATE VIEW PedMayor10 AS
  SELECT id_pedido, precio, pagado, estado
  FROM PEDIDO_TAB
  WHERE Precio > 10;
```

2. En esta vista, se muestran todos los restaurantes que actualmente esten en la Madrid, y de aquellos, mostraremos los cuales tienen pizza como producto.

```
CREATE OR REPLACE VIEW RestMyP AS
  (SELECT p.restaurante.nombre, p.restaurante.direccion
  FROM PRODUCTO_TAB p
  WHERE  p.tipo_producto='Pizza'  )
  INTERSECT
  (SELECT nombre, direccion
  FROM RESTAURANTE_TAB
  WHERE ciudad='Madrid');
```

3. Mostraremos, todos aquellos productos con su nombre, descripccion y stock, los cuales tengan actualemente mas de 5 productos en stock.

```
CREATE VIEW StockMAY5 AS
  SELECT nombre, descripcion, stock
  FROM Producto_TAB
  WHERE Stock<5;
```

4. En esta vista, vamos a postrar todo el dinero que un usuario especifico, en este caso Lucia, se ha gastado. Ademas de esto, mostraremos su id, nombre, apellidos y el dinero que se ha gastado.

```
CREATE OR REPLACE VIEW DINERO_LUCIA AS
(SELECT p.pedido.id_usuario, p.pedido.nombre, p.pedido.apellidos, p.precio
FROM PEDIDO_TAB p
WHERE p.PAGADO = 1
GROUP BY p.pedido.id_usuario, p.pedido.nombre, p.pedido.apellidos, p.precio
```

```
HAVING COUNT(p.PRECIO) > 0)
INTERSECT
(SELECT id_usuario, nombre, apellidos, p.precio
FROM CLIENTE_TAB , pedido_tab p
WHERE nombre = 'Lucia' );
```

5. Creamos una vista, para mostrar las pizzas, de las cuales, sus ofertas siguen en vigor, es decir. A dia de hoy, no se ha acabado la oferta todavia, y se puede optar a ellas

```
CREATE OR REPLACE VIEW OFERTAS_SIN_FINALIZAR AS
SELECT p.nombre, p.id_producto, o.finalizacion
FROM producto_tab p,
TABLE
(SELECT oferta
FROM PRODUCTO_TAB
WHERE producto_tab.id_producto = p.id_producto
)o
WHERE p.tipo_producto = 'Pizza' AND o.finalizacion > sysdate
ORDER BY(o.FINALIZACION);
```

6. Borramos todos los productos, cuyo stock ya no este disponible, es decir, ya no hay mas existencias del producto y por lo tanto no se puede consumir.

```
DELETE
FROM PRODUCTO_TAB p
WHERE p.stock <0 OR p.stock =0;
```

Consultas Alfonso Martínez Piñango

1. MECANICOS_VIGENTES

Muestra los mecánicos que actualmente estén en periodo de contratación y haya realizado una reparación. Para ello se comprobará con la fecha actual y se mirará cuantas facturas se han generado con su dni.

```
create view MECANICOS_VIGENTES as

select * from

        (select * from

                (select f.mecanico.dni as dni, f.mecanico.nombre as nombre, f.mecanico.empresa as empresa from factura_tab f)

                intersect

                (select dni, nombre, empresa from mecanico_tab

                where periodo > sysdate))

        natural join

        (select  f.mecanico.dni as dni, f.mecanico.nombre as nombre, f.mecanico.empresa as empresa, count(f.mecanico.dni) as facturas_totales

        from factura_tab f
```

```
group by f.mecanico.dni, f.mecanico.nombre, f.mecanico.empresa
having count(f.mecanico.dni) > 0)
```

2. AUTONOMIA_F

- Mostrar todos los vehículos eléctricos cuya autonomia sea superior a 190km y la persona que lo repara es 'Fernando'. Para ello, especificaremos la búsqueda a los eléctricos y comprobaremos que existen las matrículas.

```
create view AUTONOMIA_F as
select treat(value(v) as vehelectrico_obj).matricula as matricula , treat(value(v) as vehelectrico_obj).autonomia as autonomia,
treat(value(v) as vehelectrico_obj).emisiones as emisiones ,v.modelo, v.marca, v.disponibilidad
from vehiculo_tab v
where treat(value(v) as vehelectrico_obj).matricula <> 'null'
and treat(value(v) as vehelectrico_obj).autonomia > 190
and v.mecanico.nombre = (select nombre from mecanico_tab
        where nombre = 'Fernando')
```

3. FACTURAS_MECANICO

- Mostrará las facturas del vehículo con matrícula x y reparados por el mecánico y. Para este caso, se ha usado el dni 56217971E y la matrícula 4444jkl.

```
create view FACTURAS_MECANICO as
select f.id_factura, f.descripcion, f.importe, f.mecanico.dni, f.vehiculo.matricula
from FACTURA_TAB f
where f.mecanico.dni in (select dni from MECANICO_TAB where dni = '56217971E' )
    and
    f.vehiculo.matricula in (select matricula from VEHICULO_TAB where matricula = '4444jkl')
```

4. VEHICULOS_USO

- Mostrará los vehículos que están en uso, es decir, los vehículos que están llevando un pedido. Para ello, usaremos principalmente la tabla pedido, ya que, contendrá el estado del pedido y así se podrá comprobar si un vehículo está ocupado.

```
create view VEHICULOS_USO as
select * from
        (select * from
                (select r.id_usuario from repartidor_tab r)
                intersect
                (select p.repartidor.id_usuario as idup from pedido_tab p
                where p.estado = 'en camino'))
        natural join
        (select re.id_usuario, re.vehiculo.matricula as matricula from repartidor_tab re);
```

5. PRODUCTOS_CA

- Mostrará los restaurantes que tengan carne como producto y se localicen en Cuenca o Albacete. Para ello, usaremos intersect que nos delimitará los resultados.

```
create view PRODUCTOS_CA as
(select p.restaurante.nombre, p.restaurante.ciudad from producto_tab p
where p.tipo_producto = 'Carne' )
intersect
(select nombre, ciudad from restaurante_tab
where ciudad = 'Cuenca' or ciudad = 'Albacete')
```

6. PRODUCTOSA

- Obtendrá el total de todos los productos de los restaurantes pertenecientes a Albacete.

```
create view PRODUCTOSA as
select count(p.id_producto), p.restaurante.ciudad
from producto_tab p
where p.restaurante.ciudad = 'Albacete'
group by p.restaurante.ciudad
Having count(p.id_producto) > 0;
```

8. Disparadores

Disparadores Alberto Novillo Chinchilla

1. ASSIGNACION_REPARTIDOR.

   EVENTO: Insertar o actualizar un pedido en PEDIDOS_TAB
   ACCIÓN:

   - Si se inserta pedido:

     1. Comprueba que el repartidor asignado al pedido esta dado de alta (fecha de baja < hoy). Si no lo está, lanzará una excepción.
     2. Comprueba que el vehículo del repartidor está disponible en este momento. Si no lo está, lanzará una excepción.
     3. Comprueba si el vehículo del repartidor es eléctrico. Si es de tipo eléctrico, comprueba si la autonomía del vehículo es menor que la distancia.
     Si es menor, lanzará una excepción.

   - Si se actualiza la columna de estado de PEDIDOS_TAB:
     1. Si el pedido cambia a 'en camino', ajusta la disponibilidad del vehículo a 0.
     2. Si por el contrario cambia a otro estado (cancelado, en preparación, ...) pone la disponibilidad del vehículo a 1.

```
CREATE OR REPLACE TRIGGER ASSIGNACION_REPARTIDOR
FOR INSERT OR UPDATE  ON PEDIDO_TAB
COMPOUND TRIGGER

  -- Coleccion que almacena los ID de los pedidos que son insertados
  TYPE TID_PEDIDO IS TABLE OF PEDIDO_TAB.id_pedido%TYPE INDEX BY BINARY_INTEGER;
  VTPEDIDO_ID TID_PEDIDO;

  -- Coleccion que almacena las distancias de los pedidos que son insertados
  TYPE TDISTANCIAS IS TABLE OF PEDIDO_TAB.distancia%TYPE INDEX BY BINARY_INTEGER;
  VTDISTANCIAS TDISTANCIAS;

  -- Coleccion que almacena los estados de los pedidos que son insertados
  TYPE estados IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
  VESTADOS estados;

  -- Indice para la insercicion de los filas iinsertadas en las colecciones
  IND BINARY_INTEGER := 0;

  -- Coleccion que almacena las matriculas de todos los vehiculos de tipo electrico
  TYPE MATRICULAS_ELECTRICOS IS TABLE OF VEHICULO_TAB.matricula%TYPE;
  VMATRICULAS_ELECTRICOS MATRICULAS_ELECTRICOS;

  -- Coleccion que almacena la autonomia de todos los vehiculos de tipo electrico
  TYPE AUTONOMIAS_ELECTRICOS IS TABLE OF NUMBER(3,0);
  VAUTONOMIAS_ELECTRICOS AUTONOMIAS_ELECTRICOS;
```

```
-- Variables necesarias para la ejecucion del trigger
repartidor_id  REPARTIDOR_TAB.id_usuario%TYPE;
fechabaja_repartidor REPARTIDOR_TAB.fechabaja%TYPE;
vehiculo_matricula  VEHICULO_TAB.matricula%TYPE;
currentdisp_vehiculo VEHICULO_TAB.disponibilidad%TYPE;
NUEVO_ESTADO varchar2(30);
esElectrico NUMBER(1);
indiceElectrico NUMBER(20);


BEFORE STATEMENT IS
BEGIN

   --OBTENEMOS LAS MATRICULAS DE TODOS LOS ELECTRICOS Y LAS ALMACENAMOS EN LA COLLECCION
   SELECT TREAT(VALUE(v) AS vehelectrico_obj).matricula, TREAT(VALUE(v) AS vehelectrico_obj).autonomia
   BULK COLLECT INTO VMATRICULAS_ELECTRICOS, VAUTONOMIAS_ELECTRICOS
   FROM vehiculo_tab v
   WHERE TREAT(VALUE(v) AS vehelectrico_obj).matricula <> 'null' ;

END BEFORE STATEMENT;

BEFORE EACH ROW IS
BEGIN

   -- ALMACENAMOS EL ID, ESTADO Y DISTANCIA DE CADA NUEVO PEDIDO EN SUS COLECCIONES
   IND := IND +1;
   VTPEDIDO_ID(IND) := :new.id_pedido;
   VESTADOS(IND):= :new.estado;
   VTDISTANCIAS(IND):= :new.distancia;

END BEFORE EACH ROW;


AFTER STATEMENT IS
BEGIN
   -- Recorremos todos los datos almacenados
   FOR i IN 1..IND LOOP
      esElectrico := 0;
      indiceElectrico := 0;
      NUEVO_ESTADO:= VESTADOS(i);

      -- Obtenemos el id y la fecha de baja del repartidor de esta iteracion
      SELECT VALUE(p).repartidor.id_usuario, VALUE(p).repartidor.fechabaja
      INTO repartidor_id, fechabaja_repartidor
      FROM PEDIDO_TAB p
      WHERE ID_PEDIDO =  VTPEDIDO_ID(i);

      -- Obtenemos el la matricula y la disponibilidad del vehiculo de esta iteracion
      SELECT VALUE(r).vehiculo.matricula, VALUE(r).vehiculo.disponibilidad
      INTO vehiculo_matricula, currentdisp_vehiculo
      FROM REPARTIDOR_TAB r
      WHERE r.id_usuario = repartidor_id;
```

```
        -- Recorremos todas las matriculas almacenadas de los vehiculos que son electricos
        FOR j IN 1..VMATRICULAS_ELECTRICOS.COUNT LOOP
          -- Si una de ellas coincide con la matricula actual, el vehiculo es electrico
          IF vehiculo_matricula =  VMATRICULAS_ELECTRICOS(j) THEN
            esElectrico := 1;
            indiceElectrico := j;
          END IF;
        END LOOP;

        --En caso de que estemos insertando
        IF INSERTING then
          --Comprobamos si el vehiculo esta disponible y si el repartidor esta en nomina
          IF currentdisp_vehiculo = 0 THEN
            RAISE_APPLICATION_ERROR (-20001, '¡No se puede asignar un pedido a un repartidor que ya está
repartiendo!');
          ELSIF fechabaja_repartidor < SYSDATE THEN
            RAISE_APPLICATION_ERROR (-20001, '¡No se puede asignar un pedido a un repartidor que está
fuera de contrato!');
          END IF;

          --Si el vehiculo es electrico, comprobamos si su autonomia es menor que la distancia del pedido
          IF esElectrico = 1 THEN
            IF VAUTONOMIAS_ELECTRICOS(indiceElectrico) < VTDISTANCIAS(i) THEN
            RAISE_APPLICATION_ERROR (-20001, 'No se puede asignar este pedido a este repartirdor. Su
autonomia (' || VAUTONOMIAS_ELECTRICOS(indiceElectrico) || ') es menor que la distancia (' ||
VTDISTANCIAS(i) || ').');
          END IF;
        END IF;

        --En caso de actualizar, cambiamos la diponibilidad del vehiculo
        ELSIF UPDATING ('estado') then
          IF (NUEVO_ESTADO) = 'en camino' then
            UPDATE VEHICULO_TAB SET DISPONIBILIDAD = 0 where matricula = vehiculo_matricula;
          ELSE
            UPDATE VEHICULO_TAB SET DISPONIBILIDAD = 1 where matricula = vehiculo_matricula;
          END IF;
        END IF;
      END LOOP;

   END AFTER STATEMENT;
END ASSIGNACION_REPARTIDOR;
```

2. INSERT_ON_CLIENTE_TARJETA

EVENTO: Insertar una nueva fila en la vista CLIENTE_CON_TARJETA
ACCIÓN: Si se inserta un nuevo cliente con tarjeta en la vista, el disparador insertará un tipo CLIENTE_OBJ en la tabla CLIENTE_TAB y un tipo TARJETA_OBJ en la tabla METODOPAGO_TAB con los datos que han sido introducidos en la vista.

```
create or replace TRIGGER insert_on_cliente_tarjeta
INSTEAD OF INSERT
ON CLIENTE_CON_TARJETA
DECLARE

BEGIN

   INSERT INTO METODOPAGO_TAB VALUES (TARJETA_OBJ
   (:new.idpago, SYSDATE,  :new.numerotarjeta, :new.fecha_caducidad, :new.cvv, :new.propietario));

   INSERT INTO CLIENTE_TAB VALUES (CLIENTE_OBJ
   ( :new.id_usuario,:new.nombre, :new.apellidos, :new.telefono, :new.CORREOE, :new.ciudad, 'N/A',
:new.direccion,
   :new.codigo_postal, (SELECT REF(m) FROM METODOPAGO_TAB m WHERE m.idpago = :new.idpago )));

END;
```

Disparadores Carlos Buendía Jiménez

Actualizaremos el precio de los pedidos, cada vez que insertemos o actualicemos la base de datos.

Esta actualización la haremos en base a la autonomía del vehículo eléctrico y la distancia recorrida.

EVENTO: Actualización o inserción de un nuevo elemento en la tabla PEDIDO_TAB

ACCION:

1.- Obtenemos todos los vehículos eléctricos.

2.- Obtenemos todos los repartidores que tienen vehículos eléctricos

3.- Obtenemos todos los pedidos, cuyos repartidores tienen vehículos eléctricos.

4.- Hacemos un update al precio, en base a la autonomía y la distancia.

```
create or replace TRIGGER ACTUALIZACION_PRECIO
FOR INSERT OR UPDATE  ON  PEDIDO_TAB
COMPOUND TRIGGER

    --GUARDO MATRICULA de VEHICULO_TAB
     TYPE MATRICULAS IS TABLE OF VEHICULO_TAB.matricula%TYPE;
          V_MATRICULAS MATRICULAS;
```

```
    --GUARDO DNI REPARTIDOR
    TYPE DNI_REP IS TABLE OF REPARTIDOR_TAB.DNI%TYPE;
        V_DNI_REP DNI_REP;
   --GUARDO LA ID DE PEDIDO
    TYPE IDPEDIDO IS TABLE OF pedido_tab.id_pedido%TYPE;
        V_IDPEDIDO IDPEDIDO;

    TYPE repartidores_con_v_e IS TABLE OF repartidor_tab.vehiculo%type;
        v_repartidores_con_v_e repartidores_con_v_e;



    --GUARDO LA AUTONOMIA DE LOS VEHICULOS ELECTRICOS
    TYPE AUTONOMIA_E IS TABLE OF NUMBER(3);
     V_AUTONOMIA_E AUTONOMIA_E;

   --VARIABLES
     v_precio pedido_tab.precio%TYPE;
     v_distancia pedido_tab.distancia%TYPE;
     v_count binary_integer := 0;
   --Executed before DML statement
   BEFORE STATEMENT IS
   BEGIN
        --OBTENGO MATRICULAS Y AUTONOMIA DE LOS VEHICULOS ELECTRICOS
        SELECT TREAT(VALUE(v) AS vehelectrico_obj).matricula, TREAT(VALUE(v) AS
vehelectrico_obj).autonomia
        BULK COLLECT INTO  V_MATRICULAS, V_AUTONOMIA_E
        FROM vehiculo_tab v
        WHERE TREAT(VALUE(v) AS vehelectrico_obj).matricula is not null ; --POSIBLE ERROR EN EL IS
NOT NULL
        --OBTENGO LOS DNIS DE LOS REPARTIDORES
        SELECT DNI
        BULK COLLECT INTO V_DNI_REP
```

```
        FROM REPARTIDOR_TAB
        WHERE DNI IS NOT NULL;
    END BEFORE STATEMENT;
    --Executed before each row change- :NEW, :OLD are available
    BEFORE EACH ROW IS
    BEGIN
      v_count := v_count +1;
    END BEFORE EACH ROW;
    --Executed after DML statement
    AFTER STATEMENT IS
    BEGIN
      FOR v_i in 1..V_MATRICULAS.count loop
        SELECT r.vehiculo.matricula into V_MATRICULAS(v_i) FROM REPARTIDOR_TAB r  WHERE
r.vehiculo.matricula = V_MATRICULAS(v_i); --OBTENGO LOS REPARTIDORES CON VEHICULOS
ELECTRICOS;
        FOR v_j in 1..V_DNI_REP.count loop
          SELECT p.id_pedido, p.precio, p.distancia into V_IDPEDIDO(v_j), v_precio, v_distancia
FROM PEDIDO_TAB p WHERE p.repartidor.DNI = V_DNI_REP(v_j);
          UPDATE PEDIDO_TAB set PRECIO = (v_precio * (1+ (v_distancia)/4)/
V_AUTONOMIA_E(v_i));
        end loop;
      end loop;
    END AFTER STATEMENT;
END ACTUALIZACION_PRECIO;
```

Disparadores Alfonso Martínez Piñango

```
/*
Cambiar la disponibilidad del vehículo cuando se lleve a un mecánico y generar una factura de la reparación.
Se usara un proceso para generar la factura y crear las relaciones necesarias.

EVENTO: Insertar una factura y actualizar los datos del vehículo.

PASOS:
        1. Insertar una matrícula y comprobar que su disponibilidad está a 1.
```

2. Cambiar la disponibilidad a 0 de ese vehículo.
3. Generar una factura con datos insertados que añadamos.
3.1. En caso de insertar mal los datos saltará una excepción o datos inexistentes.

OPCIONAL:
a) Se usara un procedimiento para actualizar la disponibilidad a 1 simulando que se ha reparado el vehículo.

*/

```
CREATE OR REPLACE TRIGGER reparacion
FOR INSERT OR UPDATE ON VEHICULO_TAB
COMPOUND TRIGGER

  -- Coleccion que almacena las matriculas de todos los vehiculos de tipo electrico
  TYPE MATRICULAS_ELECTRICOS IS TABLE OF VEHICULO_TAB.matricula%TYPE;
  VMATRICULAS_ELECTRICOS MATRICULAS_ELECTRICOS;

  -- Coleccion que almacena la disponibilidad de todos los vehiculos de tipo electrico
  TYPE DISPONIBILIDAD_ELECTRICOS IS TABLE OF VEHICULO_TAB.disponibilidad%TYPE;
  VDISP_ELECTRICOS DISPONIBILIDAD_ELECTRICOS;

  -- Coleccion que almacena los dni de los mecanicos de todos los vehiculos de tipo electrico
  TYPE MECANICODNI_ELECTRICOS IS TABLE OF VEHICULO_TAB.mecanico%TYPE;
  VMDNI_ELECTRICOS MECANICODNI_ELECTRICOS;

  -- Coleccion que almacena las matriculas de todos los vehiculos de tipo gasolina
  TYPE MATRICULAS_GASOLINA IS TABLE OF VEHICULO_TAB.matricula%TYPE;
  VMATRICULAS_GASOLINA MATRICULAS_GASOLINA;

  -- Coleccion que almacena las matriculas de todos los vehiculos de tipo gasolina
  TYPE DISPONIBILIDAD_GASOLINA IS TABLE OF VEHICULO_TAB.disponibilidad%TYPE;
  VDISP_GASOLINA DISPONIBILIDAD_GASOLINA;

  -- Coleccion que almacena los dni de los mecanicos de todos los vehiculos de tipo electrico
  TYPE MECANICODNI_GASOLINA IS TABLE OF VEHICULO_TAB.mecanico%TYPE;
  VMDNI_GASOLINA MECANICODNI_GASOLINA;

  -- Coleccion que almacena los dni de todos los mecanicos
  TYPE DNI_MECANICO IS TABLE OF MECANICO_TAB.dni%TYPE;
  DNIM DNI_MECANICO;

  -- Coleccion que almacena los id de todas las facturas
  TYPE ID_FACTURA IS TABLE OF FACTURA_TAB.id_factura%TYPE;
  IDF ID_FACTURA;


  --Otras variables
  vehiculo_matricula  VEHICULO_TAB.matricula%TYPE;
  esElectrico NUMBER(1);
  mat VARCHAR2(7);
  contfactura NUMBER(10);
  auxloop NUMBER(10);
  nid_factura NUMBER(10,0);
```

```
   ndescripcion VARCHAR2(20);
   nimporte NUMBER(8,2);
   guardadni varchar2(9);

   BEFORE STATEMENT IS
   BEGIN
      --Obtenemos las matriculas de todos los electricos y las almacenamos en la coleccion
      SELECT TREAT(VALUE(v) AS vehelectrico_obj).matricula, TREAT(VALUE(v) AS
vehelectrico_obj).disponibilidad, TREAT(VALUE(v) AS vehelectrico_obj).mecanico
      BULK COLLECT INTO VMATRICULAS_ELECTRICOS, VDISP_ELECTRICOS, VMDNI_ELECTRICOS
      FROM vehiculo_tab v
      WHERE TREAT(VALUE(v) AS vehelectrico_obj).matricula <> 'null' ;

      --Obtenemos las matriculas de todos los gasolina y las almacenamos en la coleccion
      SELECT TREAT(VALUE(v) AS vehgasolina_obj).matricula, TREAT(VALUE(v) AS
vehgasolina_obj).disponibilidad, TREAT(VALUE(v) AS vehgasolina_obj).mecanico
      BULK COLLECT INTO VMATRICULAS_GASOLINA, VDISP_GASOLINA, VMDNI_GASOLINA
      FROM vehiculo_tab v
      WHERE TREAT(VALUE(v) AS vehelectrico_obj).matricula <> 'null' ;

      --Obtenemos los dni de todos los mecanicos y los almacenaos en la coleccion
      SELECT VALUE(m).dni
      BULK COLLECT INTO DNIM
      FROM MECANICO_TAB m
      WHERE VALUE(m).dni IS NOT NULL;

      --Obtenemos los dni de todos los mecanicos y los almacenaos en la coleccion
      SELECT VALUE(f).id_factura
      BULK COLLECT INTO IDF
      FROM FACTURA_TAB f
      WHERE VALUE(f).id_factura IS NOT NULL;

   END BEFORE STATEMENT;

   AFTER STATEMENT IS
   BEGIN

   contfactura := 1;

   --Añadimos la matrícula del vehiculo a reparar y si es electrico o no (1 --> SI, 0 -->NO)
   mat := '2222def';
   esElectrico := 0;


   --Comprobamos si es electrico o de gasolina
   IF esElectrico = 1 THEN
      -- Recorremos todas las matriculas almacenadas de los vehiculos que son electricos
      FOR i IN 1..VMATRICULAS_ELECTRICOS.COUNT LOOP
         -- Comprobacion de que esta la matricula y el vehiculo esta disponible
         IF mat =  VMATRICULAS_ELECTRICOS(i) THEN
            IF VDISP_ELECTRICOS(i) = 1 THEN
               UPDATE VEHICULO_TAB SET DISPONIBILIDAD = 0 where mat = VMATRICULAS_ELECTRICOS(i);
            END IF;
```

```
      END IF;
   END LOOP;


ELSE
-- Recorremos todas las matriculas almacenadas de los vehiculos que son gasolina
   FOR i IN 1..VMATRICULAS_GASOLINA.COUNT LOOP
      -- Comprobacion de que esta la matricula y el vehiculo esta disponible
      IF mat =  VMATRICULAS_GASOLINA(i) THEN
         IF VDISP_GASOLINA(i) = 1 THEN
           UPDATE VEHICULO_TAB SET DISPONIBILIDAD = 0 where mat = VMATRICULAS_GASOLINA(i);
         END IF;
       END IF;
   END LOOP;
 END IF;

--Agregamos los datos que queremos generar en la factura
nid_factura := 000001;
ndescripcion := 'Rep. frenos';
nimporte :=  600;

--Realiza un blucle para evitar duplicados de id
FOR j IN 1..IDF.COUNT LOOP
   IF nid_factura = IDF(j) THEN
      nid_factura := nid_factura + 000001;
   END IF;
END LOOP;

--Llamada al procedimiento para añadir los datos
IF esElectrico = 1 THEN
   FOR k IN 1..VMATRICULAS_ELECTRICOS.COUNT LOOP
      IF mat =  VMATRICULAS_ELECTRICOS(k) THEN
         FOR ki IN 1..DNIM.COUNT LOOP
           FOR kj IN 1..VMDNI_ELECTRICOS.COUNT LOOP
              IF DNIM(k) = VMDNI_ELECTRICOS(kj) THEN
              CREAR_FACTURA(nid_factura, ndescripcion, nimporte, mat, DNIM(ki));
              END IF;
            END LOOP;
         END LOOP;
      END IF;
   END LOOP;
ELSE
    FOR k IN 1..VMATRICULAS_GASOLINA.COUNT LOOP
      IF mat =  VMATRICULAS_GASOLINA(k) THEN
         FOR ki IN 1..DNIM.COUNT LOOP
           FOR kj IN 1..VMDNI_GASOLINA.COUNT LOOP
              IF DNIM(ki) = VMDNI_GASOLINA(kj) THEN
              CREAR_FACTURA(nid_factura, ndescripcion, nimporte, mat, DNIM(ki));
              END IF;
            END LOOP;
         END LOOP;
      END IF;
   END LOOP;
```

```
    END IF;
    END AFTER STATEMENT;
END reparacion;
```

9. Procedimientos y Funciones

Procedimientos y Funciones de Alberto Novillo Chinchilla

1. FUNCTION CALCULAR_PRECIO_PEDIDO

   Función que, dado un numero de pedido, calcula y devuelve el precio total del pedido, sumando el precio multiplicado por la cantidad y aplicando el porcentaje IVA.

   Recibe un numero de pedido y devuelve el total.

```
CREATE or REPLACE FUNCTION CALCULAR_PRECIO_PEDIDO (varid NUMBER) RETURN NUMBER IS
BEGIN

DECLARE
cantidadLinea  NUMBER (8,2);
total NUMBER (10,2);
CURSOR mycursor IS
   SELECT lp.precio, lp.cantidad, lp.iva
   FROM PEDIDO_TAB ped,
   TABLE (SELECT lpedido FROM PEDIDO_TAB WHERE id_pedido = ped.id_pedido)lp
   WHERE ped.id_pedido = varid;
BEGIN
   IF (varid is null) or (varid < 1) then
      RAISE_APPLICATION_ERROR (-20001, 'Error: ID del pedido es nulo o menor que 1');
   END IF;
   total:= 0;
   for i in mycursor loop
      cantidadLinea := i.precio * i.cantidad;
      cantidadLinea := cantidadLinea + (cantidadLinea*(i.iva/100));
      total := total + cantidadLinea;
      cantidadLinea := 0;
   end loop;
   RETURN total;

END;
END CALCULAR_PRECIO_PEDIDO;
```

   Caso de prueba:

   Si ejecutamos esta función, no podremos ver nada, ya que el planteamiento de esta función es auxiliar a otro procedimiento. Aun así, podemos ejecutar esta función, invocándola y pasandole un id:

```
DECLARE
 VARID NUMBER;
 v_Return NUMBER;
BEGIN
 VARID := 3;
```

```
 v_Return := CALCULAR_PRECIO_PEDIDO(
  VARID => VARID
 );
 :v_Return := v_Return;
 DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
```

v_Return = 11,28

2. PROCEDURE MOSTRAR_HISTORIAL

Procedimiento que muestra el historial de pedidos de una persona. Dado el correo electrónico de una persona, muestra todos los pedidos y las líneas de cada pedido, indicando la descripción del producto, la cantidad, el iva, la fecha y el importe total. Se apoya en la función CALCULAR_PRECIO_PEDIDO para calcular el precio.

Además, si cuando se calcula el precio total del pedido se detecta que no concuerda con la base de datos, se actualizará con el valor calculado.

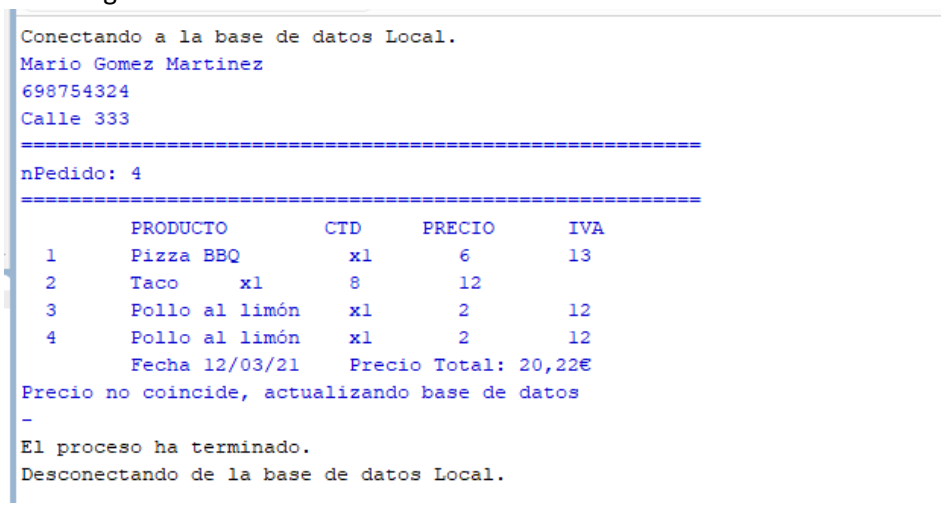
```
create or replace PROCEDURE MOSTRAR_HISTORIAL (correo VARCHAR2) IS
   --muestra el historial de pedidos de un cliente
   TYPE PED_TAB IS TABLE OF REF PEDIDO_OBJ;
   TPED PED_TAB;

   cnombre CLIENTE_tab.nombre%TYPE;
   capellidos CLIENTE_tab.apellidos%TYPE;
   ctelefono  CLIENTE_tab.telefono%TYPE;
   cdireccion CLIENTE_tab.direccion%TYPE;

   pid PEDIDO_TAB.id_pedido%TYPE;
   pprecio PEDIDO_TAB.precio%TYPE;
   pfecha PEDIDO_TAB.fecha%TYPE;

   productoNombre PRODUCTO_TAB.nombre%TYPE;
   precioCalculado NUMBER(8,2);

   indiceLinea number(3);
   BEGIN

   SELECT c.nombre, c.apellidos, c.telefono, c.direccion into  cnombre, capellidos, ctelefono,
cdireccion
   FROM CLIENTE_TAB c WHERE c.correoe = correo;

   DBMS_OUTPUT.PUT_LINE(cnombre || ' ' || capellidos);
   DBMS_OUTPUT.PUT_LINE(ctelefono);
   DBMS_OUTPUT.PUT_LINE(cdireccion);


   SELECT REF(p) BULK COLLECT INTO TPED
   FROM PEDIDO_TAB p
```

```
   WHERE PEDIDO = /*deberia de ser cliente, pero se llama pedido*/ (SELECT REF(c) FROM
CLIENTE_TAB c
                  WHERE correoe = correo);

   FOR I IN 1..TPED.COUNT LOOP
      indicelinea:=0;
      SELECT DEREF(TPED(I)).id_pedido, DEREF(TPED(I)).precio, DEREF(TPED(I)).fecha  INTO
pid,pprecio, pfecha FROM DUAL;
      DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('nPedido: ' || pid );
      DBMS_OUTPUT.PUT_LINE('--------------------------------------------------');
      DBMS_OUTPUT.PUT_LINE('    PRODUCTO       CTD    PRECIO     IVA');

      FOR linea in (
         SELECT lp.*
         FROM PEDIDO_TAB ped, TABLE (SELECT lpedido FROM PEDIDO_TAB WHERE id_pedido =
ped.id_pedido)lp
         WHERE ped.id_pedido = DEREF(TPED(I)).id_pedido)
      LOOP

         --Buscamos el nombre del producto
         SELECT pr.nombre into productoNombre
         FROM PRODUCTO_TAB pr
         WHERE id_producto =  DEREF(linea.producto).id_producto;

         indicelinea := indicelinea + 1;
         DBMS_OUTPUT.PUT_LINE('  ' || indicelinea || chr(9)||productoNombre || CHR(9)||'x'||
linea.cantidad || chr(9)|| linea.precio ||CHR(9)||linea.iva);

      END LOOP;
      precioCalculado := CALCULAR_PRECIO_PEDIDO(pid);

      DBMS_OUTPUT.PUT_LINE(CHR(9)|| 'Fecha '|| pfecha ||CHR(9)||'Precio Total: ' ||
precioCalculado ||'€');
      IF ( pprecio != precioCalculado) then
         DBMS_OUTPUT.PUT_LINE('Precio no coincide, actualizando base de datos');
         UPDATE PEDIDO_TAB x SET precio = precioCalculado WHERE x.id_pedido = pid;
      END IF;


   END LOOP;
   DBMS_OUTPUT.PUT_LINE('-');

END;
```

Caso de Prueba:
Para comprobar el funcionamiento de este procedimiento, debemos de introducir un correo electrónico válido:

```
DECLARE
  CORREO VARCHAR2(200);
BEGIN
  CORREO := 'mariogm@gmail.com';

  MOSTRAR_HISTORIAL(
    CORREO => CORREO
  );
--rollback;
END;
```

Una imagen del resultado se muestra:



```
Conectando a la base de datos Local.
Mario Gomez Martinez
698754324
Calle 333
=========================================================
nPedido: 4
=========================================================
          PRODUCTO         CTD      PRECIO       IVA
  1       Pizza BBQ          x1        6         13
  2       Taco      x1       8         12
  3       Pollo al limón     x1        2         12
  4       Pollo al limón     x1        2         12
          Fecha 12/03/21     Precio Total: 20,22€
Precio no coincide, actualizando base de datos
-
El proceso ha terminado.
Desconectando de la base de datos Local.
```

Como podemos ver, se muestra el historial de pedidos de este cliente (en este caso solo tiene un pedido). Además, el precio calculado no coincide con el que hay actualmente en la base de datos, por lo que se realiza un *update* para introducir el nuevo precio.

3. PROCEDURE DESPEDIR_REPARTIDOR

Este procedimiento se encarga de despedir a un trabajador. Para ello, se introduce el dni del repartidor al que se desea despedir. Una vez localizado en la base de datos, establece la fecha de baja del repartidor a día de hoy (CURRENT_DATE) y se elimina el vehículo propio del repartidor.

```
CREATE OR REPLACE PROCEDURE DESPEDIR_REPARTIDOR (dniFired VARCHAR2) IS
BEGIN

DECLARE

    repart REF REPARTIDOR_OBJ;
    idREP  REPARTIDOR_TAB.id_usuario%TYPE;
    dniREP REPARTIDOR_TAB.dni%TYPE;

    vehi REF VEHICULO_OBJ;
    matriculaVEH   VEHICULO_TAB.matricula%TYPE;
BEGIN
   --Buscamos y almacenamos la ref del repartidor
```

```
    SELECT REF(c) INTO repart FROM REPARTIDOR_TAB c WHERE dniFired = c.dni;

    --almacenamos la ref del vehiculo del repartidor
    SELECT DEREF(repart).vehiculo INTO vehi FROM dual;

    --buscamos el vehiculo del repartidor y guardamos su matricula.
    SELECT DEREF(vehi).matricula into matriculaVEH from dual;

    --Actualizamos la fecha de baja del repartidor a hoy y eliminamos el vehiculo de la lista.

    DBMS_OUTPUT.PUT_LINE('Actualizando fecha de baja del repartidor');
    UPDATE REPARTIDOR_TAB SET fechabaja = '01/01/2024' WHERE  id_usuario= idREP;
    DBMS_OUTPUT.PUT_LINE('Eliminando vehiculo del repartidor de la DB');
    DELETE FROM VEHICULO_TAB WHERE matricula = matriculaVEH;


END;


END DESPEDIR_REPARTIDOR;
```

Caso de prueba:
Para ejecutar este procedimiento, debemos de introducir el DNI del repartidor al que queremos despedir:

```
DECLARE
  DNIFIRED VARCHAR2(200);
BEGIN
  DNIFIRED := '44444447F';

  DESPEDIR_REPARTIDOR(
    DNIFIRED => DNIFIRED
  );
END;
```

Una vez ejecutado, el sistema nos informará de que el repartidor ha sido despedido, actualizando la fecha de baja a hoy y eliminando su vehículo.

Procedimientos y Funciones de Carlos Buendía Jiménez

Procedimientos y Funciones de Alfonso Martínez Piñango

1. CREAR_FACTURA

Procedimiento que se encargará de crear la factura con los datos que se le pasen, haciendo que las relaciones entre vehículo y mecánico tengan sentido.

En el caso de introducirse mal los datos, o no tengan conexión entre ellos saltará un error.

```
CREATE OR REPLACE PROCEDURE CREAR_FACTURA(nid_factura number, ndescripcion varchar2, nimporte
number,
m_dni varchar2, v_matricula varchar2) IS
FACTURA_N REF FACTURA_OBJ;
BEGIN

   SELECT REF(M) INTO MECANICO FROM MECANICO_TAB M
   WHERE DNI = m_dni;

   INSERT INTO FACTURA_OBJ VALUES (nid_factura, ndescripcion, nimporte,
   (SELECT ref(m) FROM MECANICO_TAB m WHERE dni = m_dni ),
   (SELECT ref(v) FROM VEHICULO_TAB v WHERE matricula = v_matricula));

EXCEPTION
WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('ERROR,EL MECANICO O EL VEHICULO NO EXISTEN');

END CREAR_FACTURA;
```