Vistas

```sql
CREATE VIEW duenio_auto AS
SELECT cliente.id_cliente, cliente.nombre_apellido, auto.marca, auto.modelo
FROM cliente, auto
WHERE cliente.id_cliente = auto.id_cliente;
```

- Una vista que muestre el nombre del cliente junto con su auto (marca, modelo)

```sql
CREATE VIEW trabajo_empleado AS
SELECT auto.id_auto, auto.marca, auto.modelo, trabajo.num_empleado
FROM auto, trabajo
WHERE auto.id_auto = trabajo.id_auto;
```

- Muestra el auto (marca, modelo) junto con el empleado que realizó el trabajo

```sql
CREATE VIEW producto_util AS
SELECT producto.id_producto, producto.marca, producto.litros, trabajo.num_empleado
FROM producto, trabajo
WHERE producto.id_producto = trabajo.id_auto;
```

- Muestra la marca y litros del producto utilizado, junto con el empleado que realizó el trabajo

```sql
CREATE VIEW fue_check AS
SELECT trabajo.id_trabajo, trabajo.num_empleado, chequeo.revisado
FROM trabajo, chequeo
WHERE trabajo.id_trabajo = chequeo.id_trabajo;
```

- Muestra el empleado que realizó el trabajo, y si el vehículo ya fue revisado

```sql
CREATE VIEW completo AS
SELECT trabajo.id_trabajo, trabajo.num_empleado, producto.marca, producto.litros, chequeo.revisado
FROM trabajo, producto, chequeo
WHERE trabajo.id_trabajo = chequeo.id_trabajo = producto.id_trabajo;
```

- Muestra el producto (marca, litros) junto al empleado que lo realizó y si fue chequeado

Funciones

```
CREATE FUNCTION `auto_cliente`(par_id_cliente INT)
RETURNS char(255)
READS SQL DATA
BEGIN
        declare resultado char (255);
    set resultado = (select modelo from auto where id_cliente = par_id_cliente);
RETURN resultado;
```

- Devuelve el vehículo (modelo) del número de cliente que se ingrese

```
CREATE FUNCTION `revision`(id_auto_p INT)
RETURNS char(255)
READS SQL DATA
BEGIN
        declare var1 int;
    declare var2 int;
    set var1 = (select id_trabajo from trabajo where id_auto_p = id_auto);
    set var2 = (select revisado from chequeo where var1 = id_trabajo);
    if (var2) > 0 then
                return ('El auto fue chequeado');
        else
                return ('El auto no fue chequeado');
        end if;
```

- Ingresando el id del auto, devuelve un mensaje en función de si el vehículo ya fue revisado o no

Stored Procedures

```
CREATE PROCEDURE `order_by` (in var char(50), in var2 int)
BEGIN
        if var2 = 0 then
                set @orden = concat ('order by ', var, ' desc');
    else
                set @orden = concat ('order by ', var, ' asc');
    end if;
                set @final = concat ( 'select * from cliente ', @orden);
    prepare runSQL from @final;
    execute runSQL;
    deallocate prepare runSQL;
```

- Ingresando un campo de la tabla ‘cliente’ + el numero ‘1’ ordena en base a ese campo de manera ascendente. En caso de ingresar ‘0’ ordena de manera descendente

```
CREATE PROCEDURE `sp_insert`(in nom char(200), in tel int, in p_dni int)
BEGIN
        insert into cliente ( nombre_apellido, telefono, dni ) values (nom, tel, p_dni);
```

- Inserta los valores ‘nombre’, ‘telefono’ y ‘dni’ en la tabla ‘cliente’

Triggers

```
CREATE TRIGGER AFT_INS_auto
AFTER INSERT ON auto
FOR EACH ROW
INSERT INTO insert_auto
VALUES (NEW.id_auto, NEW.marca, NEW.modelo, NEW.kms, NEW.id_cliente);
```

- Copia todos los datos insertados en la tabla ‘auto’ en una segunda tabla ‘insert_auto’

```
CREATE TRIGGER BEF_DEL_CLIENTE_LOGS
BEFORE DELETE ON cliente
FOR EACH ROW
INSERT INTO logs
VALUES (CURDATE(), CURTIME(), USER());
```

- Almacena en la tabla ‘logs’ la fecha, hora y usuario que elimina un campo de la tabla ‘cliente’