2w223422222134333212345

Car Retail System Database

Youssef Moustafa 20104848

Anas Magdy 20104847

Youssef Hatem 20108429

Khaled Alaa 20108424

Mahmoud Ahmed 20104039

1.1 Problem Description

El Laithy Auto Groups, a car retail company, requested a database for their system that keep track of the company’s interactions and be able to extract useful information from such database. The available entities that can interact with the company are vendors, clients, and insurance companies. Moreover, Such Entities are defined by several factors (will be mentioned in **section** **1.2**). The company itself has several warehouses, each of which has a number of cars (products). In addition, there’re several departments in the company where several employees work in.

The life cycle of the company is as follows:

From the Supplier side, the warehouse and the purchase department work in cooperation to check the stock of the cars (products) whether there’s a certain type of car that reached the Re-Order point or not-a limit through which it’s required to stock the product again-and, if the product reached it, they would later send a SO (supply order) to the vendor supplying said product. Furthermore, the purchase department must approve and record the SO sent to the vendor.

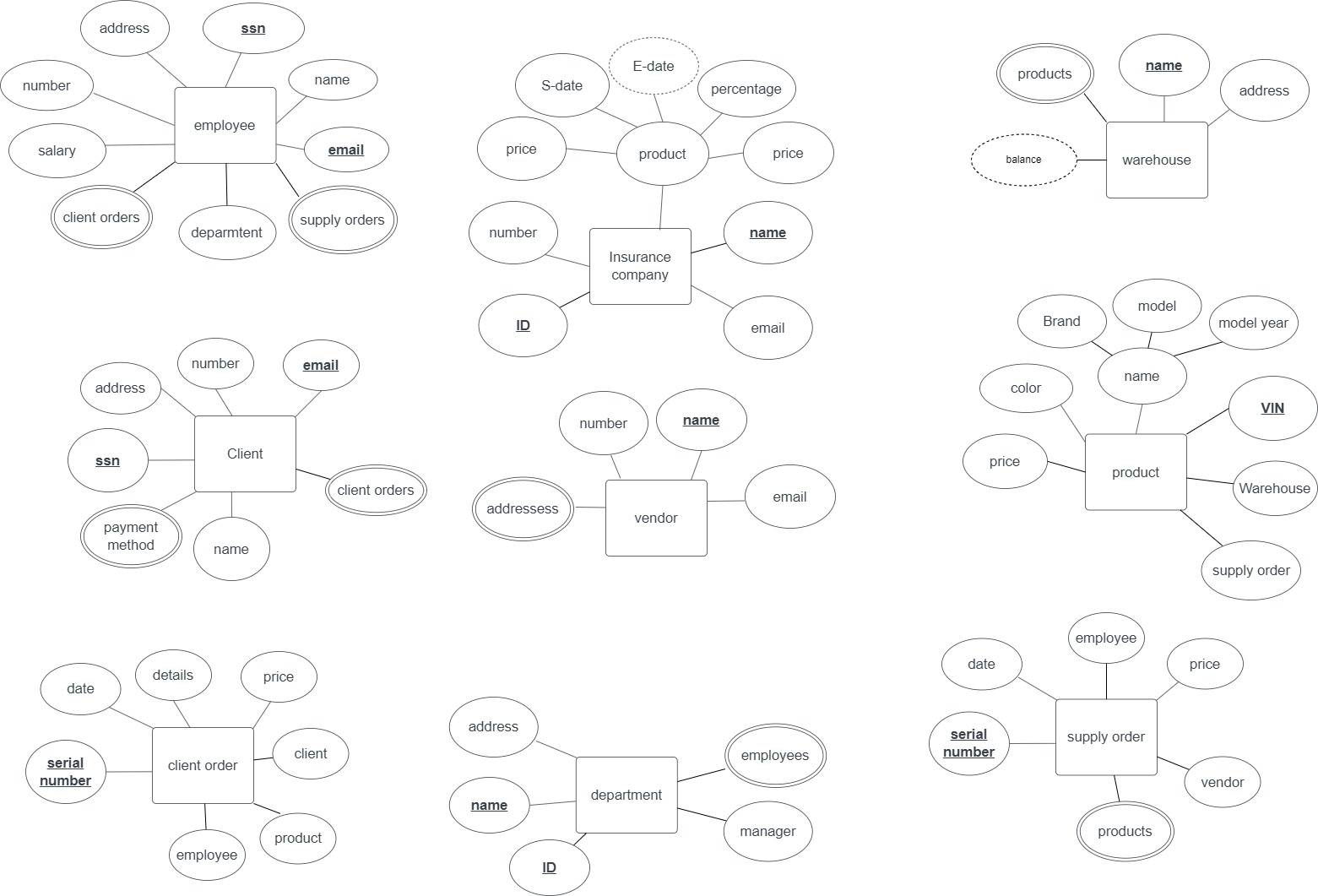
From the Client side, when the client requests a car, he has the option to choose whether they can add an insurance to the car or not. For the former, it’s required to contact an insurance company for the completion of the transaction. All the transactions are recorded by the employees working in sales department by creating CO (Client Order) for each transaction. Surely, there are details for the car and, if required, the insurance provided by the insurance company.

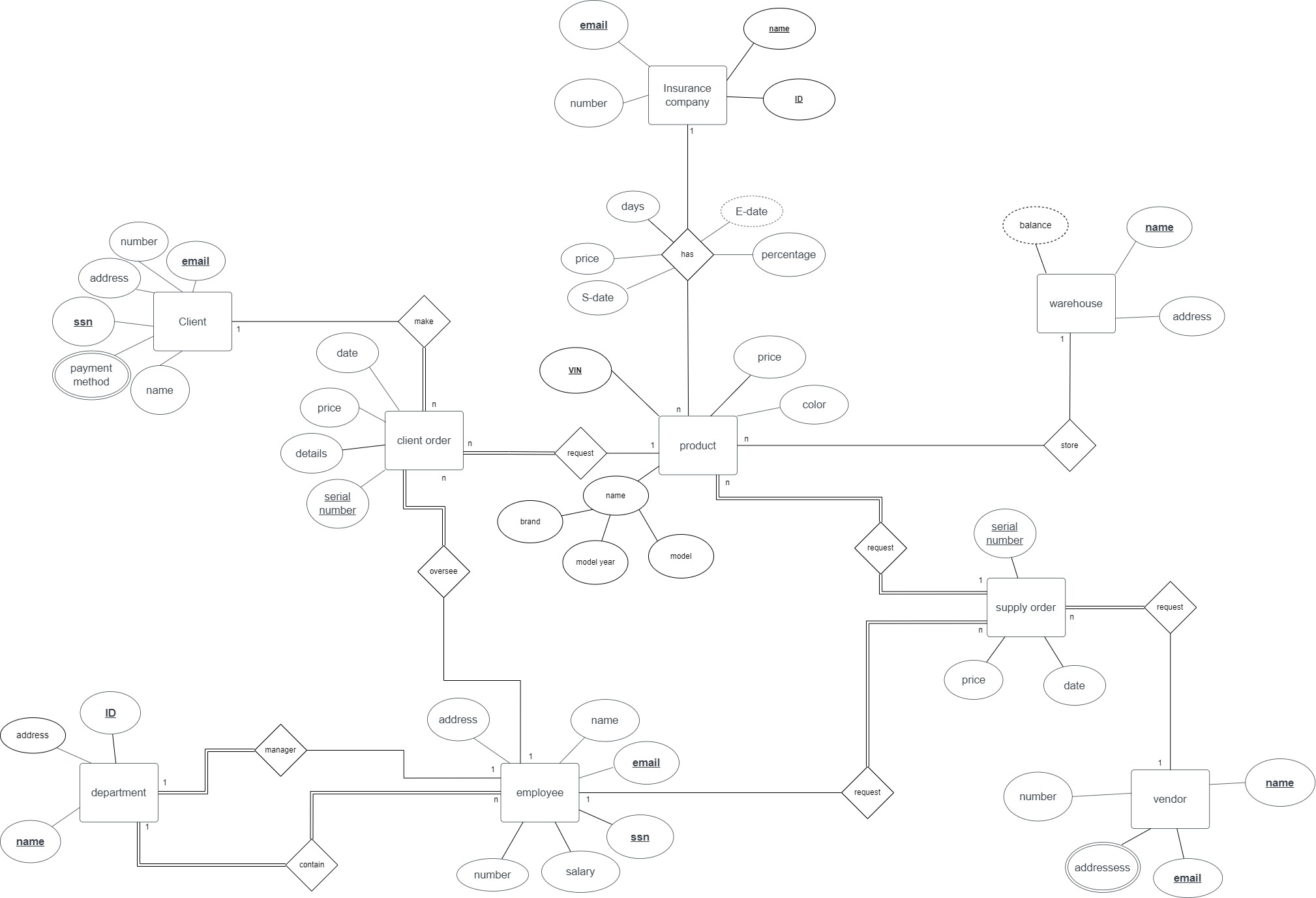
Any further details will be provided in **section 1.2**.

1.2 Problem Analysis

It’s required to design database system that keep track of a car retail system the company keeps track on:

* Vendor that has a unique name, number, several addresses and email.
* Client that has SSN that define the client, the address, Number, email, Name and payment methods that have several choices.
* Warehouse that has a unique name, address and have a product with several types. It also calculates balance of the remaining cars.
* Product that has unique VIN, Brand and model year of the product, and every product has a price and color, and every quantity of the Product are saved in the warehouse with a re-order point.
* Insurance Company that gives the product some insurance. The DB will record start date, the end date, the percentage of the insurance, and price of insurance of a certain product. Every insurance company has name, number and email. A product may or may not have an insurance.
* Supply Order: Every Supply Order has a unique serial number and it contains the request of the product quantity that the warehouse specifies from the supplier, and it contains date, employee making such order, and cost.
* Client Order: Every Client Order has a unique serial number and it presents the request of the client through the product, and it contains and Date for every Order, and price.
* Employee that has a unique SSN and Name, Number and Address, Salary and email.
* Department that has a unique name, address and the departments observe the Warehouse & the orders, and every department contain number of employees.
* Every Department has a manager that manages all the operations in the department
* The Employee Oversee the Client Orders and confirm or deny it. Only one employee can oversee the order, but many orders can be overseen by an employee. All Client orders must be overseen by employees.
* A supplier can supply several products, and a product can be supplied by a vendor.
* There’s a purchase department that observes the warehouse, and a financial department that observes the client/supply orders.
* When a product’s quantity reaches the re-order point, the purchase department receives a refill message.
* Every Product must have a supplier and might exist in a warehouse.

2.1 Rough ER Schema

2.2 Final ER Schema

2.3 Mapping

**Product**: FK FK FK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **VIN** | model  year | model | brand | color | price | SO serial number | WHname | IC\_ID |  |

PK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| price | percentage | Days | S-date |  |  |  |  |  |  |

**Warehouse**:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **name** | address |  |  |  |  |  |  |  |  |

Pk

**Insurance company**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | name | email | number |  |  |  |  |  |

PK

**vendor**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **name** | email | number |  |  |  |  |  |

Pk

**supply order**: FK FK

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **serial number** | date | V\_Name | E\_ssn |  |  |  |  |

Pk

**department**: FK

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **name** | address | M\_SSN |  |  |  |  |

Pk

**client order**: FK FK FK

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **serial number** | price | details | date | ESSN | VIN | CSSN |  |

Pk

**employee**: FK

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ssn** | name | salary | number | address | email | Dname |  |  |  |

Pk

**Client**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ssn** | name | number | address | email |  |  |  |

Pk

**Addresses**:

**FK**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **V\_NAME** | **Addresses** |  |  |  |  |  |  |

Pk

**Payment method**:

**FK**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SSN** | **Payment method** |  |  |  |  |  |  |

Pk

3.1 DDL/DML SQL Sample Statements

\* Will be done using ORACLE SQL dictionary

\* The SQL statements aren’t complete neither are they in order. Please reference to the SQL File associated with the folder uploaded to view all SQL statements.

🡪 Product:

create table product (

VIN Varchar(17) Primary Key,

brand Varchar(20),

model varchar2(20),

modelyear decimal(4),

color Varchar(10),

p\_price decimal,

W\_name Varchar(20) References warehouse,

IC\_ID Varchar(6) References Insurance\_company,

I\_price decimal,

I\_percentage number,

I\_Days number,

I\_Sdate date,

so\_serialnumber varchar2(6) references supplyorder not null

);

insert into product (VIN,brand,model,modelyear,color,p\_price,so\_serialnumber,W\_name,IC\_ID,I\_price,I\_percentage,I\_Days,I\_Sdate) values ('JH4KA3170LC006787','hyundai','tucson',2006,'red',455000,'SOX21','b01','IC1',16000,100,1095,'20-oct-2014');

🡪Warehouse:

create table warehouse(

w\_name Varchar2(20) Primary Key,

w\_address varchar2(50)

);

insert into warehouse values('b01','Nasr City');

insert into warehouse values('b02','El Obour');

🡪Supply Order:

create table supplyorder (

so\_serialnumber varchar2(6) primary key,

so\_price number,

so\_date date,

v\_name VARCHAR2(20) REFERENCES vendor not null,

e\_ssn number(12) REFERENCES employee not null

);

insert into supplyorder(so\_price,so\_date,v\_name,e\_ssn) VALUES (370000,'10-oct-2010','Hyundai',12509893456);

insert into supplyorder(so\_price,so\_date,v\_name,e\_ssn) VALUES (325000,'6-oct-2011','Toyota',567377893456);

🡪Vendor:

create table vendor(

V\_name Varchar(20) Primary key,

V\_email varchar(20) unique,

V\_number number

);

create table Addresses(

V\_name varchar(20) References Vendor,

Addresses varchar(20),

constraint Addressespk primary key (V\_name,Addresses)

);

insert into vendor(V\_name,V\_email,V\_number) values ('Chevrolet','a111@gmail.com',01029495310);

insert into vendor(V\_name,V\_email,V\_number) values ('Kia','a112@gmailcom',01029495311);

🡪Insurance Company:

create table Insurance\_Company(

IC\_ID varchar2(6) primary key,

IC\_name varchar2(50) unique,

IC\_email VARCHAR2(50) unique,

IC\_number number

);

insert into Insurance\_Company(IC\_name,IC\_email,IC\_number) values ('Misr Insurance','a212@gmail.com',01029495320);

insert into Insurance\_Company(IC\_name,IC\_email,IC\_number) values ('Misr Life Insurance Company','a213@gmail.com',01029495321);

🡪Client:

create table Client(

C\_SSN NUMBER(12) Primary Key,

C\_name VARCHAR2(20),

C\_number NUMBER,

C\_address VARCHAR2(20),

C\_email VARCHAR2(40) unique

);

create table Payment\_method(

C\_SSN NUMBER(12) References Client,

Payment\_method VARCHAR2(20),

constraint Payment\_methodpk primary key(C\_SSN,Payment\_method)

);

insert into Client(C\_SSN,C\_name,C\_number,C\_adress,C\_email) values (896225676768 ,'Lynn',01029495330,'Nasr city','a313@gmail.com');

insert into Client(C\_SSN,C\_name,C\_number,C\_adress,C\_email) values (948375838636 ,'Taylor',01029495331,'Masr Al Jadidah','a321@gmail.com');

🡪Department with Employee:

create table department(

d\_id number primary key,

d\_name varchar(15) unique not null,

d\_address varchar(30)

);

insert into department (d\_name,d\_address) values ('hr','nasr city');

insert into department (d\_name,d\_address) values ('accounting','sheraton');

create table employee (

e\_ssn number(12) primary key,

e\_name varchar(17) ,

e\_address varchar(36),

e\_email varchar(23) unique,

salary number(7),

e\_number number(11),

D\_id number References department not null

);

insert into employee values (012377893456,'ali','nasr city','youssefsdr3@yahoo.com',10000,01001109871,'10');

insert into employee values (012509893456,'youssef','sheraton','fghy1q32@yahoo.com',10000,01010334567,'20');

alter table department add (

e\_ssn number references employee

);

update department set e\_ssn=012377893456 where d\_id=10;

update department set e\_ssn=012509893456 where d\_id=20;

ALTER TABLE department

ADD CONSTRAINT essnn CHECK(e\_ssn IS NOT NULL);

🡪Client Order:

create table clientorder (

co\_serialnumber varchar2(6) primary key,

co\_details varchar(10), /\*rent or full\*/

co\_date date,

co\_price number,

e\_ssn number(12) References employee ,

c\_ssn number(12) References client ,

vin varchar2(17) References product

);

insert into clientorder (C\_ssn,VIN,co\_details,co\_date,e\_ssn) values (978275838636,'1NPTXUEXX8N766170','Full',To\_date('05/08/2010','dd/mm/yyyy'),'567377893456');

insert into clientorder (C\_ssn,VIN,co\_details,co\_date,e\_ssn) values (948375838636,'1GNDT13W3W2249640','Full',To\_date('17/03/2012','dd/mm/yyyy'),'567377893456');

3.2 Sample SQL Triggers

🡪There’re triggers that correct the input of insert statements, for example:

create or replace trigger cname\_correction before INSERT ON Client

for each row

begin

:new.c\_name := initcap(:new.c\_name);

end;

create or replace trigger empname\_correction before INSERT ON employee

for each row

begin

:new.e\_name := upper(:new.e\_name);

:new.e\_address := initcap(:new.e\_address);

end;

🡪 There’re triggers that automatically generate IDs for certain tables with fixed sequences, for example:

create sequence did start with 10 INCREMENT BY 10;

create or replace trigger deptID\_generator before INSERT ON department

for each row

begin

:new.d\_id := did.nextval;

:new.d\_name := upper(:new.d\_name);

:new.d\_address := initcap(:new.d\_address);

end;

create sequence icid start with 1;

create or replace trigger Ic\_generator BEFORE insert on Insurance\_Company

for each row

begin

:new.IC\_ID := concat('IC',icid.nextval);

end;

🡪 There’s a trigger that automatically calculate the total price of the client orders:

create or replace trigger calc\_co\_price

BEFORE insert

on clientorder

for each row

declare

carprice number;

insuranceprice number;

begin

select p\_price

into carprice

from product p

where :new.vin = p.vin;

select I\_price

into insuranceprice

from product p

where :new.vin = p.vin;

if insuranceprice > 0 then

:new.co\_price := (carprice+insuranceprice);

else

:new.co\_price := (carprice);

end if;

end;

🡪 There’s a query that validates the client orders by checking that the cars already sold can’t be sold again:

create or replace trigger co\_valdation

before insert

on clientorder

for each row

declare

recordcount number;

usr\_exp exception;

begin

select count(\*)

into recordcount

from clientorder co

where :new.vin = co.vin and co.co\_details = 'Full';

if recordcount > 0 then

raise usr\_exp;

end if;

end;

3.3 SQL Queries with functions

1. A Query to get the most sold brands

select p.brand,count(co.co\_serialnumber) as number\_sold

from clientorder co inner join product p

on co.vin=p.vin

where co.co\_details='Full'

group by p.brand

order by count(co.co\_serialnumber) desc;

1. A Query to get the most sold colors

select p.color,count(co.co\_serialnumber) as number\_sold

from clientorder co inner join product p

on co.vin=p.vin

where co.co\_details='Full'

group by p.color

order by count(co.co\_serialnumber) desc;

1. A Query to select the employees that need promotion. Promotion process is done by selecting employees with more than average orders per employee and have a salary that is less than the average

create or replace function AvgOrderPerEmp

return number

is

v\_avgOrders number;

begin

select avg(count(co\_serialnumber))

into v\_avgorders

from clientorder

group by e\_ssn;

return v\_avgOrders;

end;

create or replace function AvgSalary

return number

is

V\_avgsalary number;

begin

select avg(salary)

into v\_avgsalary

from employee e

where not exists (

select \*

from department d

where e.e\_ssn = d.e\_ssn

);

return v\_avgsalary;

end;

select e.e\_ssn,e.e\_name,e.e\_email,e.salary,count(co.co\_serialnumber) as orders\_made

from employee e inner join clientorder co

on e.e\_ssn = co.e\_ssn

where salary < AvgSalary

group by e.e\_ssn,e.e\_name,e.e\_email,e.salary

having count(co.co\_serialnumber) > AvgOrderPerEmp;

1. A Query to return how many clients that have ordered cars from each city

select lower(c\_address) as City,count(co\_serialnumber) as Clients

from client c inner join clientorder co

on c.c\_ssn = co.c\_ssn

group by lower(c\_address);

1. A Query that gets the clients who purchased the cars but the license has expired

create or replace function checkExpiry(P\_Idate date,P\_days number)

return varchar2

is

V\_calcdate date;

begin

v\_calcdate := p\_idate + p\_days;

if v\_calcdate > sysdate then

return 'false';

else

return 'true';

end if;

end;

select c.c\_name,c.c\_email,p.brand||' '||p.model||' '||p.modelyear as car, p.i\_sdate, p.i\_sdate + p.i\_days as Expiry

from client c

inner join clientorder co

on c.c\_ssn = co.c\_ssn

inner join product p

on co.vin = p.vin

where checkExpiry(p.i\_sdate,p.i\_days) = 'true' and p.i\_days>0;