**Introduction to the system:**

This project is intended for owners of online stores that stores all possible data. Namely: client, invoice, sales, product, product\_category, admins, customer, warehouse, company and others.

**Normalization**

1. **Client(client\_ID, first\_name, email, username, client\_password) - 3NF**

The table is in 3NF because there is no trace of dependencies between attributes.

1. **Invoice(invoice\_ID, customer\_ID, client\_ID, payment\_type, bank\_account\_name, bank\_account\_number, date\_record, totalmount) - 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

1. Sales(sales\_ID, product\_ID, quantity, unit\_price) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

1. Customer(customer\_ID, customer\_name, contact) - **2NF**

The table is in 2NF because there is a partial dependency between customer\_name and contact from customer\_ID.

5) Personal(personal\_ID, country, city, address, email, phone) - **3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

6) Warehouse(warehouse\_number, order\_ID, product\_ID, warehouse\_location) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

1. Order(order\_ID, customer\_ID, product\_ID, supplier\_ID, client\_ID, order\_number, order\_date) **- 3NF**The table is in 3NF because there are no transitive dependencies between attributes.

8) Supplier(supplier\_ID, contact\_first\_name, company\_name, contact\_title) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

9) Recieve\_product(recieve\_ID, client\_ID, product\_ID, supplier\_ID, quantity, unit\_price) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

10) Product(client\_ID, category\_ID, product\_ID, company\_ID, product\_name, until\_price, discount) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

11) Product\_category(category\_ID, product\_name) **- 1NF**

The table is in 1NF because it contains no repeating groups or multi-valued attributes.

12) Household(species\_numberHC, price, quantity) - **2NF**

The table is in 2NF because there is a partial relationship between price and quantity from species\_numberHC.

13) Electronics(species\_numberE, price, quantity) **- 2NF**

The table is in 2NF because there is a partial relationship between price and quantity from species\_numberE.

14) Food(species\_numberF, price, quantity, product\_expiration\_date) **- 2NF**

The table is in 2NF because there is a partial relationship between the price, quantity, and expiration date of the product from kind\_numberF.

15) Admins(admin\_ID, admin\_name, admin\_password) **- 3NF**

The table is in 3NF because there are no transitive dependencies between attributes.

1. Company(company\_ID, product\_name) **- 1NF**

The table is in 1NF because there are no repeating groups or multi-valued attributes.

**Codes:**  
1) Procedure which does group by information

**CREATE OR REPLACE PROCEDURE admin\_role**

**AS**

**cursor employee\_group is**

**select POST, count(\*) as employee\_count**

**from OM\_ADMINS**

**group by POST;**

**BEGIN**

**for post\_group in employee\_group**

**loop**

**DBMS\_OUTPUT.PUT\_LINE(post\_group.POST || '| :' || post\_group.employee\_count);**

**end loop;**

**END;**

**BEGIN**

**admin\_role();**

**END;**  
  
This code is a procedure that retrieves the number of employees associated with each "post" (or job position) from a table called "OM\_ADMINS" and prints the results to the console using the DBMS\_OUTPUT.PUT\_LINE procedure.

2) Function which counts the number of records

**CREATE OR REPLACE FUNCTION count\_records(nameOfTable IN VARCHAR2)**

**RETURN NUMBER IS**

**countOfRecord NUMBER := 0;**

**BEGIN**

**EXECUTE IMMEDIATE 'SELECT COUNT(\*) FROM ' || nameOfTable INTO countOfRecord;**

**RETURN countOfRecord;**

**END;**

**DECLARE**

**numRecords NUMBER;**

**BEGIN**

**numRecords := count\_records('OM\_ADMINS');**

**DBMS\_OUTPUT.PUT\_LINE('Number of records: ' || numRecords);**

**END;**

This is a function that takes the name of a database table as an input parameter and returns the number of records in that table.

3) Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

**CREATE OR REPLACE PROCEDURE update\_wine\_quantity AS**

**BEGIN**

**UPDATE OM\_PRODUCT**

**SET UNTIL\_PRICE = UNTIL\_PRICE - (DISCOUNT + 50)**

**WHERE PRODUCT\_NAME LIKE 'Wine%';**

**DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT || ' rows updated successfully.');**

**END;**

**begin**

**update\_wine\_quantity;**

**end;**

This is a procedure that updates the price of all products whose names begin with 'Wine' by subtracting a fixed amount from their original price. The updated quantity is then displayed on the console using the DBMS\_OUTPUT.PUT\_LINE procedure.

4) Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

**CREATE OR REPLACE TRIGGER check\_password\_length**

**BEFORE INSERT ON OM\_CLIENT**

**FOR EACH ROW**

**DECLARE**

**too\_short exception;**

**BEGIN**

**IF LENGTH(:new.client\_password) < 5 THEN**

**RAISE too\_short;**

**END IF;**

**EXCEPTION**

**WHEN too\_short THEN**

**raise\_application\_error(-20001, 'Password length should be at least 5 characters');**

**END;**

**insert into OM\_CLIENT VALUES (454521, 'Doner', 'donerlong@gmail.com', 'DonerLong', 'sam')**  
  
This is a trigger that validates the length of the password field before inserting a new record into the OM\_CLIENT table. If the password length is less than 5 characters, an exception is raised and the insertion is aborted.

5) Create a trigger before insert on any entity which will show the current number of rows in the table

**CREATE OR REPLACE TRIGGER row\_count\_trigger**

**BEFORE INSERT ON OM\_PRODUCT**

**DECLARE**

**row\_count INT;**

**BEGIN**

**SELECT COUNT(\*) INTO row\_count FROM OM\_PRODUCT;**

**DBMS\_OUTPUT.PUT\_LINE('Current row count: ' || row\_count);**

**END;**

**insert into OM\_PRODUCT VALUES (454521, 585858585, 786548, 123456, 'samsa', 3500, 500)**

This is a trigger that is executed before a new row is inserted into the OM\_PRODUCT table. The trigger retrieves the current row count from the OM\_PRODUCT table and outputs it using the DBMS\_OUTPUT.PUT\_LINE procedure.