**Jewelry store**

**Task 1**

After a long search, we decided to choose a jewelry store dataset.

There were unnecessary parts in this dataset, so we eliminated them during the import process.

Now this dataset matches requirements of this project

The link:

<https://data.world/jfreex/products-catalog-from-newchiccom/workspace/file?filename=jewelry.csv>

**Task 2**

We created 2 derived columns as required by using this sql commands and procedures. Also we made functions which simplifies the usage for the clients.

1. Updating price to be able to work with it

UPDATE JEWELRYS

SET PURCHASE\_PRICE = REPLACE(PURCHASE\_PRICE, '.', '');

2)Creating 2 derived columns

ALTER TABLE JEWELRYS

ADD price VARCHAR2(38);

ALTER TABLE JEWELRYS

ADD color\_options VARCHAR2(26);

3)Procedures which provides the seting and updating this fields

CREATE OR REPLACE PROCEDURE set\_price IS

CURSOR get\_data IS

SELECT PURCHASE\_PRICE,DISCOUNT,ID FROM JEWELRYS;

v\_PURCHASE\_PRICE JEWELRYS.PURCHASE\_PRICE%TYPE;

v\_DISCOUNT JEWELRYS.DISCOUNT%TYPE;

v\_ID JEWELRYS.ID%TYPE;

BEGIN

OPEN get\_data();

LOOP

FETCH get\_data INTO v\_PURCHASE\_PRICE,v\_DISCOUNT,v\_ID;

EXIT WHEN get\_data%NOTFOUND;

UPDATE JEWELRYS

SET price = v\_PURCHASE\_PRICE - v\_DISCOUNT

WHERE id = v\_ID;

END LOOP;

CLOSE get\_data;

END set\_price;

UPDATE JEWELRYS

SET price = 777

WHERE id = 1692610;

CREATE OR REPLACE PROCEDURE set\_co IS

CURSOR get\_col IS

SELECT color1,color2,ID FROM JEWELRYS;

v\_color1 JEWELRYS.color1%TYPE;

v\_color2 JEWELRYS.color2%TYPE;

v\_ID JEWELRYS.ID%TYPE;

v\_c\_op JEWELRYS.color\_options%TYPE;

BEGIN

OPEN get\_col();

LOOP

FETCH get\_col INTO v\_color1,v\_color2,v\_ID;

EXIT WHEN get\_col%NOTFOUND;

IF (v\_color1 IS NULL) OR (v\_color2 IS NULL) THEN

v\_c\_op := 'NO';

ELSE

v\_c\_op := 'YES';

END IF;

UPDATE JEWELRYS

SET color\_options = v\_c\_op

WHERE id = v\_ID;

END LOOP;

CLOSE get\_col;

END set\_co;

4)This provided function counts how many jewels are currently in stock by a parameter(name)

create or replace function jewelrys\_count

(p\_name IN jewelrys.name%TYPE)

Return NUMBER IS

v\_in\_stock jewelrys.in\_stock%TYPE:=0;

begin

select count(in\_stock) into v\_in\_stock from jewelrys

where name = p\_name;

return v\_in\_stock;

Exception

When NO\_DATA\_FOUND THEN

Return null;

end jewelrys\_count;

Invoking

select id ,color1, jewelrys\_count(name)

from jewelrys where name = 'Collier pompon papillon';

5) This procedure increases the price of jewelry by 10%

CREATE OR REPLACE PROCEDURE raise\_purchase\_price

(p\_id IN jewelrys.id%TYPE,

p\_percent IN NUMBER)

IS

BEGIN

UPDATE jewelrys

SET purchase\_price = purchase\_price \* (1 + p\_percent/100)

WHERE id = p\_id;

END raise\_purchase\_price;

BEGIN

raise\_purchase\_price(1513785, 10);

END;

6) These created triggers were made with purpose of showing the changes in database , and by whom they were done , showing exact time and username (which also turns out to be important)

create table log\_jew(

WHO VARCHAR2(20),

WHEN\_1 DATE

);

CREATE OR REPLACE TRIGGER log\_emp\_changes

AFTER UPDATE ON jewelrys BEGIN

INSERT INTO log\_jew (who, when\_1)

VALUES (USER, SYSDATE);

END;

select \* from log\_jew;

7) The second trigger detects the updates which was done by a user , showing in the output “You just updated the row”

CREATE OR REPLACE TRIGGER trigg

BEFORE UPDATE ON jewelrys

FOR EACH ROW

ENABLE

DECLARE

v\_user VARCHAR2(20);

BEGIN

select user into v\_user from dual;

DBMS\_OUTPUT.PUT\_LINE('You just updated row.'|| v\_user);

END;