Table of Contents

1ο ερώτημα – Διαχείριση Παραγγελιών

i.

```
CREATE OR REPLACE TYPE address_type AS OBJECT
(city VARCHAR2(50),
street VARCHAR2(50),
num NUMBER(3)
);
/

ALTER TABLE Customers ADD (Address address_type);

ALTER TABLE Customers DROP COLUMN Address;

UPDATE Customers
SET Address = address_type(
            CASE WHEN DBMS_RANDOM.VALUE <= 1/7 THEN 'Athens'
                 WHEN DBMS_RANDOM.VALUE <= 2/7 THEN 'Thessaloniki'
                 WHEN DBMS_RANDOM.VALUE <= 3/7 THEN 'Patras'
                 WHEN DBMS_RANDOM.VALUE <= 4/7 THEN 'Heraklion'
                 WHEN DBMS_RANDOM.VALUE <= 5/7 THEN 'Larissa'
                 WHEN DBMS_RANDOM.VALUE <= 6/7 THEN 'Volos'
                 WHEN DBMS_RANDOM.VALUE <= 7/7 THEN 'Ioannina'
            END,
            CASE WHEN DBMS_RANDOM.VALUE <= 1/20 THEN 'Adrianou Street'
                 WHEN DBMS_RANDOM.VALUE <= 2/20 THEN 'Athinas Street'
                 WHEN DBMS_RANDOM.VALUE <= 3/20 THEN 'Ermou Street'
                 WHEN DBMS_RANDOM.VALUE <= 4/20 THEN 'Panepistimiou
Avenue'
                 WHEN DBMS_RANDOM.VALUE <= 5/20 THEN 'Patision Street'
                 WHEN DBMS_RANDOM.VALUE <= 6/20 THEN 'Vasilissis Sofias
Avenue'
                 WHEN DBMS_RANDOM.VALUE <= 7/20 THEN 'Kifisias Avenue'
                 WHEN DBMS_RANDOM.VALUE <= 8/20 THEN 'Syngrou Avenue'
                 WHEN DBMS_RANDOM.VALUE <= 9/20 THEN 'Vouliagmenis
Avenue'
                 WHEN DBMS_RANDOM.VALUE <= 10/20 THEN 'Stadiou Street'
                 WHEN DBMS_RANDOM.VALUE <= 11/20 THEN 'Leoforos
Alexandras'
                 WHEN DBMS_RANDOM.VALUE <= 12/20 THEN 'Akadimias Street'
                 WHEN DBMS_RANDOM.VALUE <= 13/20 THEN 'Solonos Street'
                 WHEN DBMS_RANDOM.VALUE <= 14/20 THEN 'Ploutarchou
Street'
                 WHEN DBMS_RANDOM.VALUE <= 15/20 THEN 'Ermou Street'
                 WHEN DBMS_RANDOM.VALUE <= 16/20 THEN 'Miaouli Street'
                 WHEN DBMS_RANDOM.VALUE <= 17/20 THEN 'Asklipiou Street'
```

```
                WHEN DBMS_RANDOM.VALUE <= 18/20 THEN 'Filellinon
Street'
                WHEN DBMS_RANDOM.VALUE <= 19/20 THEN 'Tritis
Septemvriou Street'
                WHEN DBMS_RANDOM.VALUE <= 20/20 THEN 'Aiolou Street'
            END,
            TRUNC(DBMS_RANDOM.VALUE * 100) + 1
);

SELECT c.customer_id, c.address.num, c.address.street, c.address.city
FROM customers c;
```

OUTPUT

CUSTOMER_ID	ADDRESS.NUM	ADDRESS.STREET	ADDRESS.CITY
8267	97	Vouliagmenis Avenue	Heraklion
5612	58	Patision Street	Patras
12723	31	Patision Street	Heraklion
35782	66	Vasilissis Sofias Avenue	Athens
40226	7	Kifisias Avenue	Patras
2871	18	Panepistimiou Avenue	Heraklion
9982	21	Kifisias Avenue	Thessaloniki
17093	57	Panepistimiou Avenue	Patras
34863	81	Patision Street	Heraklion

ii.

```
CREATE TYPE product_type_list AS TABLE OF VARCHAR2(50);
/

ALTER TABLE Products ADD (ProductTypes product_type_list)
NESTED TABLE ProductTypes STORE AS ProductTypes_tab;

SELECT product_id, categoryname, producttypes FROM products;
```

OUTPUT

PRODUCT_ID	CATEGORYNAME	PRODUCTTYPES
15	Desktop PCs	IT2021088.PRODUCT_TYPE_LIST('computer')
28	Operating Systems	IT2021088.PRODUCT_TYPE_LIST('computer')
113	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
114	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
115	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
116	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
117	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
118	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')
119	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')

iii.

```
CREATE OR REPLACE FUNCTION mapToProductTypes(p_category VARCHAR2) RETURN
product_type_list IS
    l_product_types product_type_list := product_type_list();
BEGIN
    -- Perform the mapping based on CategoryName
    CASE
        WHEN p_category = 'Recordable DVD Discs' THEN
            l_product_types := product_type_list('video', 'storage', 'games');
        WHEN p_category IN ('Camcorders', 'Camera Batteries', 'Camera Media',
'Cameras') THEN
            l_product_types := product_type_list('video');
        WHEN p_category = 'CD-ROM' THEN
            l_product_types := product_type_list('audio', 'storage', 'games');
        WHEN p_category = 'Home Audio' THEN
            l_product_types := product_type_list('audio');
        WHEN p_category = 'Memory' THEN
            l_product_types := product_type_list('storage', 'computer');
        WHEN p_category IN ('Bulk Pack Diskettes', 'Recordable CDs') THEN
            l_product_types := product_type_list('storage');
        WHEN p_category IN ('Game Consoles', 'Y Box Games', 'Y Box
Accessories') THEN
            l_product_types := product_type_list('games');
        WHEN p_category IN ('Modems/Fax', 'Accessories') THEN
            l_product_types := product_type_list('computer', 'other');
        WHEN p_category IN ('Desktop PCs', 'Operating Systems', 'Monitors',
'Portable PCs') THEN
            l_product_types := product_type_list('computer');
        WHEN p_category IN ('Documentation', 'Printer Supplies') THEN
            l_product_types := product_type_list('other');
    END CASE;

    RETURN l_product_types;
END mapToProductTypes;
/

UPDATE Products
SET ProductTypes = mapToProductTypes(CategoryName);

SELECT product_id, categoryname, producttypes FROM products;
```

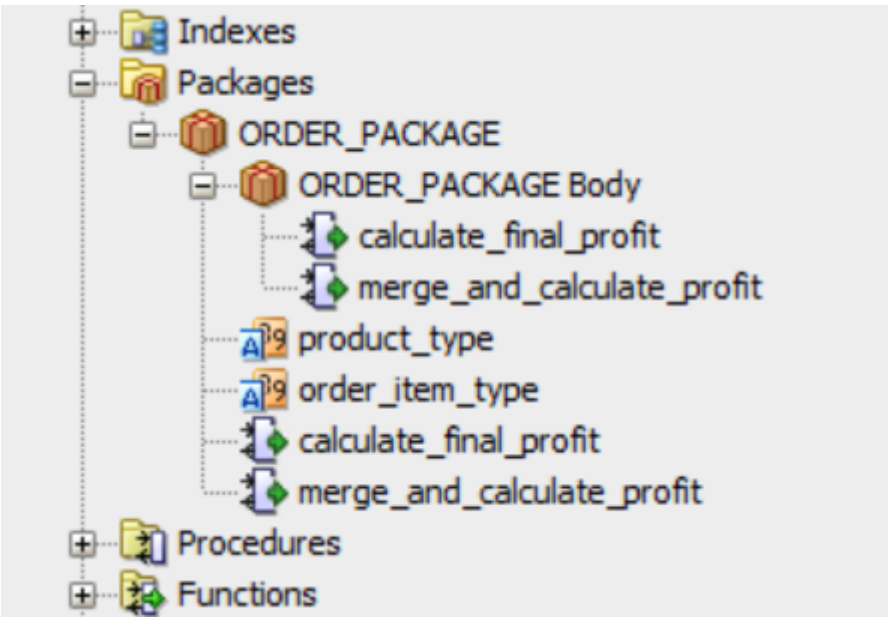
Το **OUTPUT** είναι ίδιο με του ερωτήματος (ii)

PRODUCT_ID	CATEGORYNAME	PRODUCTTYPES
15	Desktop PCs	IT2021088.PRODUCT_TYPE_LIST('computer')
28	Operating Systems	IT2021088.PRODUCT_TYPE_LIST('computer')
113	Recordable CDs	IT2021088.PRODUCT_TYPE_LIST('storage')

iv.

```
CREATE OR REPLACE PACKAGE order_package AS
  -- Ορισμός του τύπου product
  TYPE product_type IS RECORD (
    product_id NUMBER,
    productname VARCHAR(46),
    categoryname VARCHAR(50),
    producttypes product_type_list,
    listprice NUMBER
  );

  -- Ορισμός του τύπου order_item
  TYPE order_item_type IS RECORD (
    days_to_process NUMBER,
    price NUMBER(10,2),
    cost NUMBER,
    channel VARCHAR2(20),
    product product_type
  );

  -- Ορισμός του τύπου order_items_list ως πίνακας από order_item
  TYPE order_items_list IS TABLE OF order_item_type;

  -- Ορισμός του τύπου order_info
  TYPE order_info_type IS RECORD (
    customer_id NUMBER,
    order_items order_items_list
  );

  FUNCTION calculate_final_profit(order_info order_info_type) RETURN
NUMBER; -- (v)
  FUNCTION merge_and_calculate_profit(
    existing_order order_info_type,
    new_order order_info_type
  ) RETURN NUMBER; -- (vi)
END order_package;
/
```

Η δομή του πακέτου είναι:



v & vi.

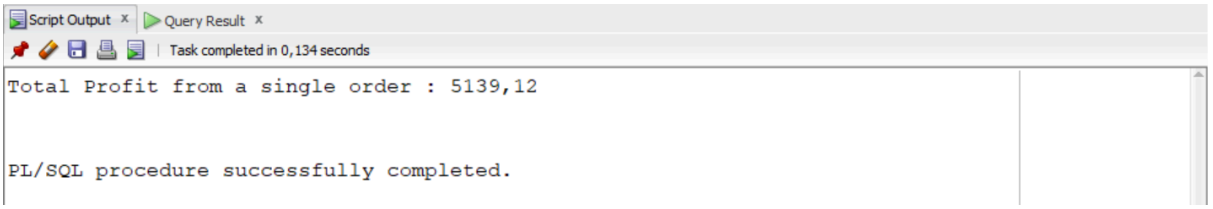
```
CREATE OR REPLACE PACKAGE BODY order_package AS
    -- Implement the function to calculate final profit
    FUNCTION calculate_final_profit(order_info order_info_type) RETURN
NUMBER IS
        total_profit NUMBER := 0;
    BEGIN
        -- Iterate through order items and calculate profit for each
product
        FOR i IN 1..order_info.order_items.COUNT LOOP
            total_profit := total_profit +
(order_info.order_items(i).price - order_info.order_items(i).cost);
        END LOOP;

        RETURN total_profit;
    END calculate_final_profit;


    FUNCTION merge_and_calculate_profit(
        existing_order order_info_type,
        new_order order_info_type
        ) RETURN NUMBER IS
        merged_order order_info_type;
        total_profit NUMBER := 0;
    BEGIN
    -- Merge orders
        merged_order := existing_order;
```

```
        -- Add new order items to the merged order
        FOR i IN 1..new_order.order_items.COUNT LOOP
          merged_order.order_items.EXTEND;
          merged_order.order_items(merged_order.order_items.COUNT) :=
new_order.order_items(i);
        END LOOP;

        -- Calculate total profit for the merged order
        -- epanaxrhsimopoioume thn etoimh function apo to prohgoumeno
erotima
        total_profit := calculate_final_profit(merged_order);


        -- Return the total profit
        RETURN total_profit;
    END merge_and_calculate_profit;
END order_package;
/
```

v example usage

```
DECLARE
  -- Declare a variable to store the calculated profit
  v_total_profit NUMBER;
  v_order_info order_package.order_info_type;

BEGIN
    v_order_info := createAndPopulateOrderInfo(3);
    ---- !!! ALLAZOUME TON ARITHMO (ORDER_ID) STHN PARENTHESH TOU
createAndPopulateOrderInfo GIA NA DOKIMASOUME ME ALLA DEDOMENA !!! ----
    v_total_profit :=
order_package.calculate_final_profit(v_order_info);

    -- Display the result for each row
    DBMS_OUTPUT.PUT_LINE('Total Profit from a single order : ' ||
v_total_profit);
END;
/
```

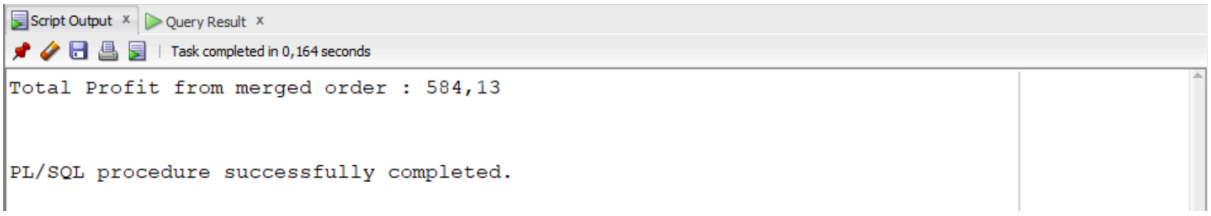
OUTPUT



vi example usage

```
DECLARE
  -- Declare a variable to store the calculated profit
  v_total_profit NUMBER;

  v_existing_order order_package.order_info_type;
  v_new_order order_package.order_info_type;

BEGIN
  v_existing_order := createAndPopulateOrderInfo(1);
  v_new_order := createAndPopulateOrderInfo(2);
  -- Open the cursor

    -- Calculate the total profit using the function for each row
    v_total_profit :=
order_package.merge_and_calculate_profit(v_existing_order, v_new_order);

    -- Display the result for each row
    DBMS_OUTPUT.PUT_LINE('Total Profit from merged order : ' ||
v_total_profit);
END;
/
```

OUTPUT



Για ευκολία δημιουργήσαμε μία συνάρτηση που δέχεται ένα order id και επιστρέφει τον αντίστοιχο τύπο order info:

```
-- extra function that's used to populate order_info
CREATE OR REPLACE FUNCTION createAndPopulateOrderInfo(desired_order_id
NUMBER) RETURN order_package.order_info_type
IS

  v_order_info order_package.order_info_type;

  -- Declare variables to store values from the SELECT statement
  v_customer_id NUMBER;
  v_days_to_process NUMBER;
```

```
  v_price NUMBER;
  v_cost NUMBER;
  v_channel VARCHAR2(50);
  v_product_id NUMBER;
  v_productname VARCHAR2(50);
  v_categoryname VARCHAR2(50);
  v_list_price NUMBER;
  v_producttypes product_type_list; -- Adjust the data type as needed

  -- Declare a cursor to fetch data
  CURSOR order_cursor IS
    SELECT o.customer_id, o.days_to_process, o.price, o.cost, o.channel,
p.product_id, p.productname, p.categoryname, p.list_price,
p.producttypes
    FROM Orders o JOIN Products p ON o.product_id = p.product_id
    WHERE o.order_id = desired_order_id;

BEGIN
    v_order_info.order_items := order_package.order_items_list();
  -- Open the cursor
  OPEN order_cursor;

  -- Fetch and process each row
  LOOP

    FETCH order_cursor INTO v_customer_id, v_days_to_process, v_price,
v_cost, v_channel, v_product_id, v_productname, v_categoryname,
v_list_price, v_producttypes;

    EXIT WHEN order_cursor%NOTFOUND;

    -- Manually populate the example order_items_list with fetched
values for each row
    v_order_info.customer_id := v_customer_id;
    v_order_info.order_items.EXTEND;

v_order_info.order_items(v_order_info.order_items.LAST).days_to_process
:= v_days_to_process;
    v_order_info.order_items(v_order_info.order_items.LAST).price :=
v_price;
    v_order_info.order_items(v_order_info.order_items.LAST).cost :=
v_cost;
    v_order_info.order_items(v_order_info.order_items.LAST).channel :=
v_channel;

v_order_info.order_items(v_order_info.order_items.LAST).product.product_
```

```
id := v_product_id;

v_order_info.order_items(v_order_info.order_items.LAST).product.productn
ame := v_productname;

v_order_info.order_items(v_order_info.order_items.LAST).product.category
name := v_categoryname;

v_order_info.order_items(v_order_info.order_items.LAST).product.listpric
e := v_list_price;

v_order_info.order_items(v_order_info.order_items.LAST).product.producttt
ypes := v_producttypes;

  END LOOP;

  -- Close the cursor
  CLOSE order_cursor;
  RETURN v_order_info;
END createAndPopulateOrderInfo;
/
```

vii.

```
CREATE OR REPLACE TYPE result_record IS OBJECT (
    customer_id NUMBER,
    address VARCHAR2(100),
    product_count NUMBER,
    total_profit NUMBER
);
/

CREATE TABLE results OF result_record;

DROP TABLE results;

CREATE OR REPLACE PROCEDURE process_address(p_address address_type,
p_start_num NUMBER, p_end_num NUMBER) AS
    TYPE customer_id_list IS TABLE OF NUMBER;
    TYPE full_address_list IS TABLE OF VARCHAR2(100);
    TYPE product_count_list IS TABLE OF NUMBER;
    TYPE total_profit_list IS TABLE OF NUMBER;

    l_customer_ids    customer_id_list := customer_id_list();
    l_full_addresses   full_address_list := full_address_list();
```

```
    l_product_counts   product_count_list := product_count_list();
    l_total_profits    total_profit_list := total_profit_list();
BEGIN
    DELETE FROM results;

    SELECT
        c.customer_id,
        p_address.city || ', ' || p_address.street || ' ' ||
c.address.num AS full_address,
        COUNT(o.order_id) AS product_count,
        SUM(o.price - o.cost) AS total_profit
    BULK COLLECT INTO l_customer_ids, l_full_addresses,
l_product_counts, l_total_profits
    FROM Customers c
    LEFT JOIN Orders o ON c.customer_id = o.customer_id
    WHERE c.address.city = p_address.city
      AND c.address.street = p_address.street
      AND c.address.num BETWEEN p_start_num AND p_end_num
    GROUP BY c.customer_id, c.address.num;

    -- Bulk insert into the results table
    FOR i IN 1..l_customer_ids.COUNT
    LOOP
        INSERT INTO results
    SELECT result_record(
        l_customer_ids(i),
        l_full_addresses(i),
        l_product_counts(i),
        l_total_profits(i)
    ) FROM DUAL;
    END LOOP;
END;
/

EXEC process_address(address_type('Patras', 'Vasilissis Sofias Avenue',
NULL), 34, 56);

SELECT * FROM results WHERE product_count <> 0;
```

OUTPUT

CUSTOMER_ID	ADDRESS	PRODUCT_COUNT	TOTAL_PROFIT
4732	Patras, Vasilissis Sofias Avenue 41	65	384,88
588	Patras, Vasilissis Sofias Avenue 42	154	2872,32
4715	Patras, Vasilissis Sofias Avenue 53	223	2775,31
5330	Patras, Vasilissis Sofias Avenue 48	168	6052,6
48811	Patras, Vasilissis Sofias Avenue 35	80	1351,21

Επαλήθευση

```
--------- Epalitheush ----------------

SELECT c.customer_id, c.Address.city AS City, c.Address.street AS
Street,
       c.Address.num AS Num, SUM(o.price - o.cost) AS Total_Profit
FROM Orders o JOIN Customers c ON o.customer_id = c.customer_id
WHERE o.customer_id = 4732 -- πρώτη γραμμή
GROUP BY c.customer_id, c.Address.city, c.Address.street, c.Address.num;
```

OUTPUT

Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0,087 seconds

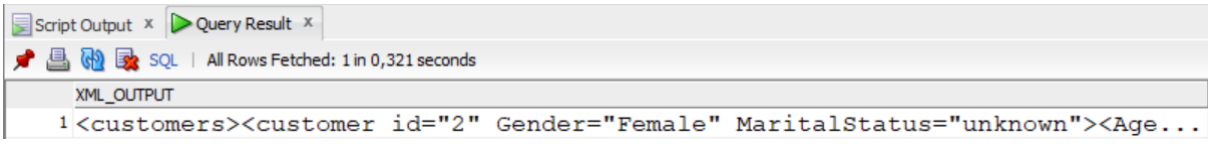
	CUSTOMER_ID	CITY	STREET	NUM	TOTAL_PROFIT
1	4732	Patras	Vasilissis Sofias Avenue	41	384,88

2º ερώτημα – Εξαγωγή σε XML

```
SELECT
  XMLElement("customers",
    XMLAgg(
      XMLElement("customer",
        XMLATTRIBUTES(
          c.customer_id AS "id",
          c.gender AS "Gender",
          c.marital_status AS "MaritalStatus"
        ),
        XMLForest(
          c.age_group AS "AgeGroup",
          c.income_level AS "IncomeLevel",
          XMLAgg(
            XMLElement("Product",
              XMLATTRIBUTES(p.product_id AS "id"),
              XMLForest(
                p.productname AS "ProductName",
                p.categoryname AS "ProductCategory"
              ),
              XMLELEMENT("ProductTypes",
                  (SELECT
                      XMLAgg(
                        XMLElement("ProductType", column_value)
                      )
                    FROM TABLE(ProductTypes) t
```

```
                  )
              )
            )
          ) AS "Products",
          XMLForest(
            c.address.street AS "Street",
            c.address.num AS "Number",
            c.address.city AS "City"
          ) AS "Address"
        )
      )
    )
  ).getClobVal() AS xml_output
FROM Customers c
JOIN (
WITH RankedOrders AS (
  SELECT o.order_id, p.product_id, c.customer_id,
    ROW_NUMBER() OVER (PARTITION BY p.product_id, c.customer_id ORDER BY
o.order_id) AS rnk
  FROM
    Customers c
    JOIN Orders o ON o.customer_id = c.customer_id
    JOIN Products p ON o.product_id = p.product_id
    CROSS JOIN TABLE(ProductTypes) t
  WHERE o.days_to_process <= 20 AND c.customer_id <= 30
)
SELECT order_id, product_id, customer_id
FROM RankedOrders
WHERE rnk = 1
) o ON o.customer_id = c.customer_id
JOIN Products p ON o.product_id = p.product_id
GROUP BY c.customer_id,
         c.gender, c.marital_status, c.age_group, c.income_level,
         c.address.street, c.address.num, c.address.city;
```

OUTPUT



ΠΡΟΣΟΧΗ!!!

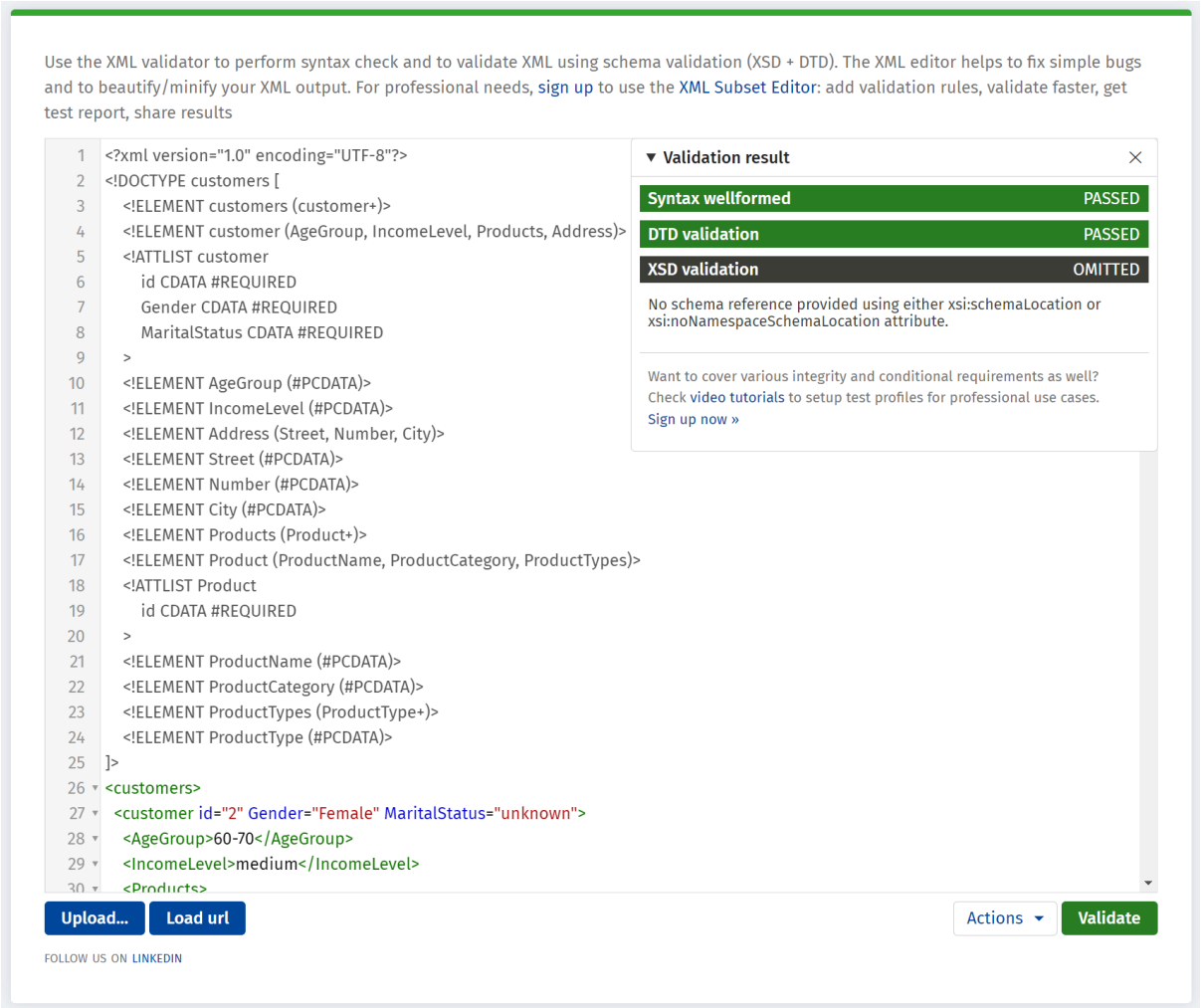
Το αποτέλεσμα περικόπτεται σε προηγούμενες εκδόσεις της sql developer. Το bug όμως διορθώθηκε στην πιο πρόσφατη έκδοση SQL Developer 23.1.1 και τώρα μπορούμε να κάνουμε αντιγραφή το column χωρίς να χάνεται πληροφορία.

Το XML που παράγεται για δύο τυχαίους customers με days_to_process ≤ 20 είναι:

```xml
<customers>
	<customer id="308" Gender="Female" MaritalStatus="married">
		<AgeGroup>40-50</AgeGroup>
		<IncomeLevel>low</IncomeLevel>
		<Products>
			<Product id="145">
				<ProductName>Finding Fido</ProductName>
				<ProductCategory>Y Box Games</ProductCategory>
				<ProductTypes>
					<ProductType>games</ProductType>
				</ProductTypes>
			</Product>
		</Products>
		<Address>
			<Street>Panepistimiou Avenue</Street>
			<Number>83</Number>
			<City>Athens</City>
		</Address>
	</customer>
	<customer id="929" Gender="Male" MaritalStatus="married">
		<AgeGroup>above 70</AgeGroup>
		<IncomeLevel>medium</IncomeLevel>
		<Products>
			<Product id="138">
				<ProductName>256MB Memory Card</ProductName>
				<ProductCategory>Camera Media</ProductCategory>
				<ProductTypes>
					<ProductType>video</ProductType>
				</ProductTypes>
			</Product>
		</Products>
		<Address>
			<Street>Kifisias Avenue</Street>
			<Number>34</Number>
			<City>Heraklion</City>
		</Address>
	</customer>
```

και όπως φαίνεται συμμορφώνεται απολύτως με το DTD της εκφώνησης. Το πραγματικό XML αρχείο φυσικά (για τους πρώτους 30 customers, καθώς ολόκληρο το αρχείο ήταν υπερβολικά μεγάλο για να χρησιμοποιηθεί σε online validators), έχει customers με περισσότερα από ένα <Product> στη λίστα των <Products>, καθώς και product categories με περισσότερα από ένα <ProductType> στη λίστα των <ProductTypes>.

Εδώ επαληθευουμε το XML για όλους τους customers με customer_id ≤ 30, χρησιμοποιώντας το DTD της εκφώνησης και έναν online validator (**ΠΡΟΣΟΧΗ** ότι βγάλαμε το MaritalStatus απ' την λίστα των elements και το χρησιμοποιούμε μόνο ως attribute):



xml validator: https://www.truugo.com/xml_validator/

3ο ερώτημα – Ερωτήσεις XPath

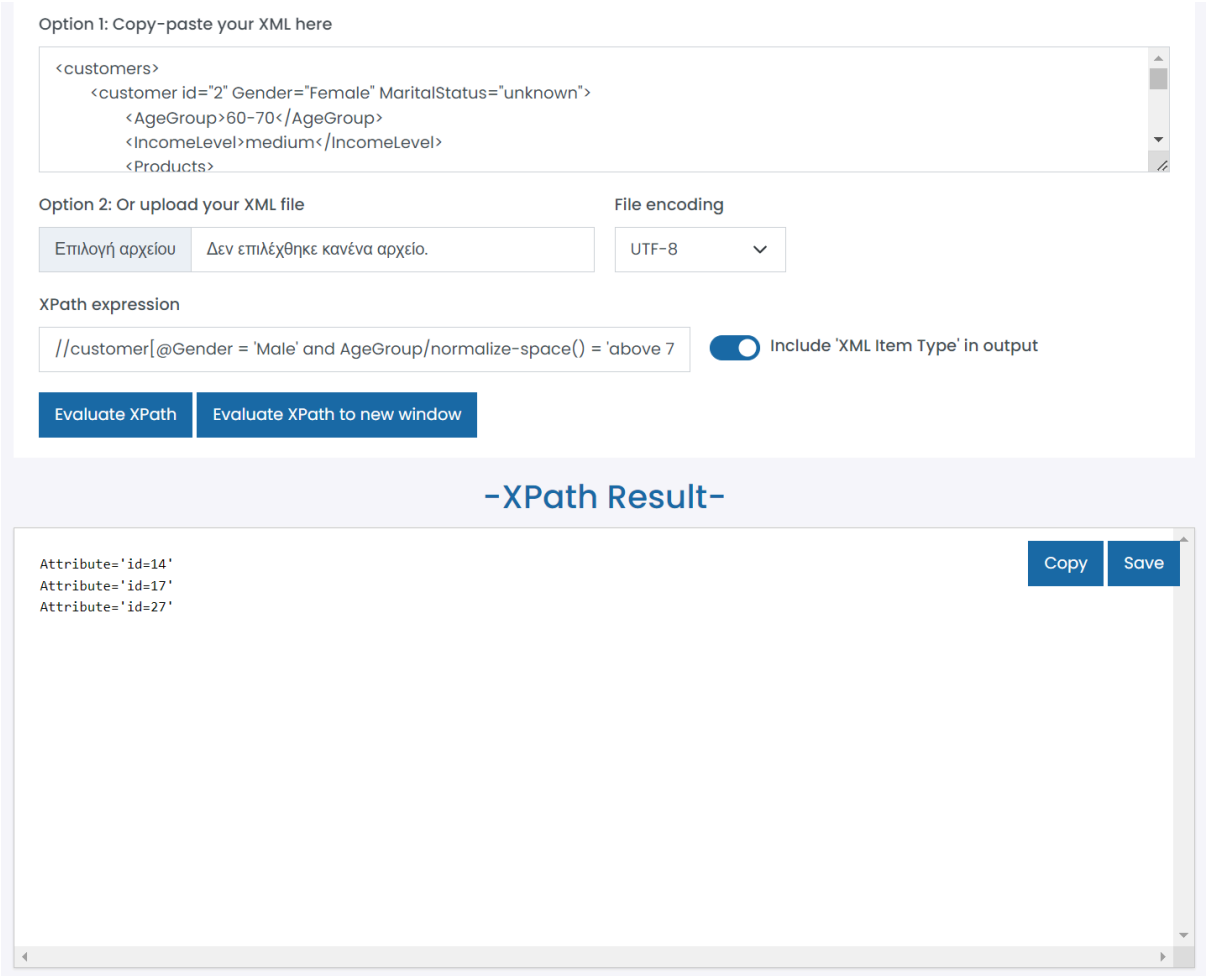
Για την επαλήθευση των ερωτήσεων τροποποιήσαμε ελαφρώς τον πίνακα Customers ως εξής:

```
-- για επαλήθευση
UPDATE Customers
SET gender = 'Male', age_group = 'above 70'
WHERE customer_id IN (
    SELECT c.customer_id
    FROM Customers c
    JOIN Orders o ON o.customer_id = c.customer_id
```

```sql
    JOIN Products p ON o.product_id = p.product_id
    WHERE o.days_to_process <= 20 AND c.customer_id <= 30
    AND p.categoryname = 'Monitors'
);

UPDATE Customers c
SET c.address.city = 'Volos', c.address.street = 'Ermou Street'
WHERE customer_id IN (
    SELECT c.customer_id
    FROM Customers c
    JOIN Orders o ON o.customer_id = c.customer_id
    WHERE o.days_to_process <= 20 AND c.customer_id <= 30
    AND c.income_level = 'high'
);
```

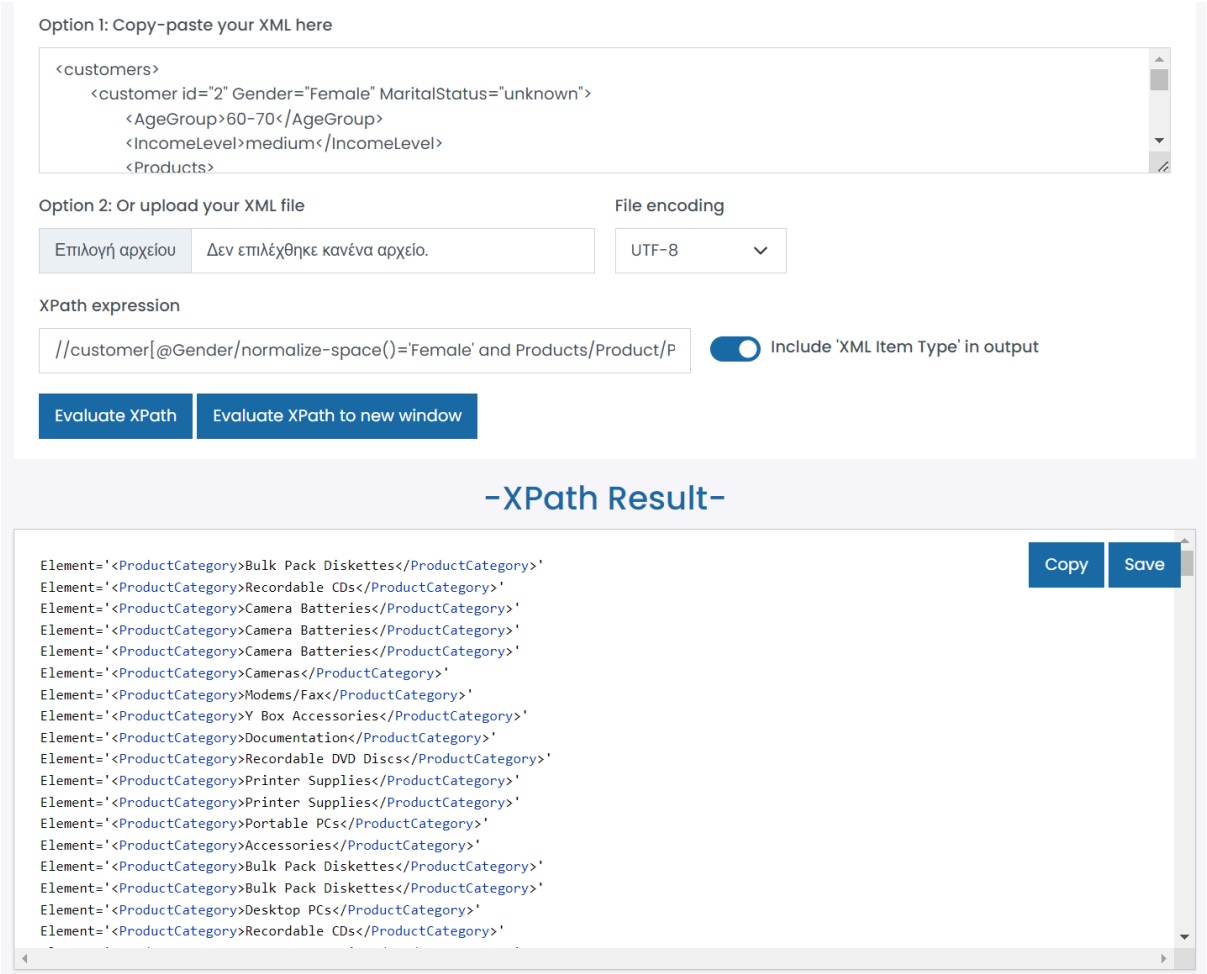
1.

```xpath
//customer[@Gender = 'Male' and AgeGroup/normalize-space() = 'above 70'
and Products/Product/ProductCategory/normalize-space() = 'Monitors']/@id
```



2.

```
//customer[@Gender/normalize-space()='Female' and
Products/Product/ProductTypes/ProductType/normalize-space()='games']/Pro
ducts/Product/ProductCategory
```



3.

```
//customer[IncomeLevel/normalize-space()='high' and
Address/City/normalize-space()='Volos' and
Address/Street/normalize-space()='Ermou
Street']/Products/Product/ProductTypes
```

Option 1: Copy-paste your XML here

```
<customers>
    <customer id="2" Gender="Female" MaritalStatus="unknown">
        <AgeGroup>60-70</AgeGroup>
        <IncomeLevel>medium</IncomeLevel>
        <Products>
```

Option 2: Or upload your XML file

Επιλογή αρχείου | Δεν επιλέχθηκε κανένα αρχείο.

File encoding: UTF-8

XPath expression

```
//customer[IncomeLevel/normalize-space()='high' and Address/City/norm
```

Include 'XML Item Type' in output

Evaluate XPath | Evaluate XPath to new window

-XPath Result-

Copy | Save

```
Element='<ProductTypes>
                <ProductType>computer</ProductType>
                <ProductType>other</ProductType>
            </ProductTypes>'
Element='<ProductTypes>
                <ProductType>computer</ProductType>
            </ProductTypes>'
Element='<ProductTypes>
                <ProductType>video</ProductType>
            </ProductTypes>'
Element='<ProductTypes>
                <ProductType>games</ProductType>
            </ProductTypes>'
Element='<ProductTypes>
                <ProductType>video</ProductType>
            </ProductTypes>'
Element='<ProductTypes>
                <ProductType>video</ProductType>
```

xpath tester: https://www.freeformatter.com/xpath-tester.html#before-output