ΠΕΡΙΕΧΟΜΕΝΑ:

1. Υποερωτήμα (Subquery)
    1. Χρήση σε SELECT
    2. Χρήση σε FROM
    3. Χρήση σε WHERE
        1. Υποερώτημα που επιστρέφει μία τιμή
        2. Υποερώτημα που επιστρέφει μία στήλη
    4. Χρήση σε HAVING

Νίκος Θ.

Σμαραγδένιος Χορηγός Μαθήματος

Πάνος Γ.

Ασημένιος Χορηγός Μαθήματος

Ένα υποερώτημα (subquery):

- είναι ένα SQL ερώτημα το οποίο χρησιμοποιείται ως μέρος ενός μεγαλύτερου SQL ερωτήματος
- Ένα υποερώτημα μπαίνει μέσα σε παρενθέσεις και μπορεί να χρησιμοποιηθεί σε διαφορετικά clauses:

Υποερώτημα	SELECT	FROM	WHERE	HAVING
Επιστρέφει μία τιμή	√	√	√	√
Επιστρέφει μία στήλη	X	√	√	√
Επιστρέφει έναν πίνακα	X	√	X	X

Υποερώτημε σε SELECT clause

- Συντακτικό:

```
SELECT col1, col2,…,( subquery ) AS colk, …, colN
```

- Για κάθε γραμμή του τελικού αποτελέσματος του ερωτήματος:
  - Υπολογίζεται το subquery (το οποίο πρέπει να επιστρέφει ακριβώς μία τιμή)
  - και ενσωματώνεται στο τελικό αποτέλεσμα ως μία ακόμη στήλη (η οποία έχει υποχρεωτικά συνώνυμο)
- Επίσης το SELECT μπορεί να έχει οσαδήποτε υποερωτήματα.

Παράδειγμα 1: DB: classic, select.sql

```
-- example 1.1
SELECT c.customerName,
       (
        SELECT MAX(p.amount)
        FROM payments p
        WHERE p.customerNumber = c.customerNumber
       ) AS max_amount
FROM customers c
ORDER BY 2 DESC;
```

- Το subquery το λέμε και **"εσωτερικό" (inner) ερώτημα**
- Ενώ αυτό που το περικλείει το λέμε και **"εξωτερικό" (outer) ερώτημα.**
- και ισχύει ότι το εσωτερικό ερώτημα "βλέπει" τους πίνακες του εξωτερικού ερωτήματος (ενώ το αντίστροφο δεν ισχύει)
- Το παραπάνω ερώτημα καλείται και **"σχετιζόμενο" υποερώτημα (correlated subquery)** διότι σχετίζει τους πίνακες του εσωτερικού και του εξωτερικού ερωτήματος.

Παρατήρηση:

- Θα αποφεύγουμε να γράψουμε ερωτήματα με subquery στο SELECT, διότι μπορεί να γραφεί κομψότερα με JOIN.

Άσκηση 1:

Εξάγετε το ίδιο αποτέλεσμα με αυτό του παραδείγματος 1, με JOIN.

Υποερώτημα σε FROM clause

- Συντακτικό:

```
FROM t1, t2,…,( subquery ) tk, …, tN
```

- Κάθε υποερώτημα στο FROM clause υπολογίζει έναν πίνακα.
- Υπολογίζονται πρώτα οι πίνακες-υποερωτήματα και στη συνέχεια μπορούμε να τους συνδυάσουμε με άλλους πίνακες στη συνήθη δομή ερωτημάτων
- Το FROM μπορεί να έχει οσαδήποτε υποερωτήματα.
- Πρέπει κάθε υποερώτημα να συνοδεύεται από ένα συνώνυμο του πίνακα που κατασκευάστηκε.

Παρατηρήσεις:

- Με χρήση πινάκων που προκύπτουν από υποερωτήματα, μπορούμε να κατασκευάσουμε προσωρινούς πίνακες που είναι πιο εύληπτοι και αναγνώσιμοι και έπειτα να τους συνδυάσουμε με άλλους πίνακες.
  - Δες στο παρ.2.1 ότι ο πίνακας του υποερωτήματος (αναφέρεται και ως inline view - ένθετη όψη) διασπά εννοιολογικά καλύτερα το συνολικό ερώτημα
  - Ωστόσο το ίδιο ερώτημα θα μπορούσε να γίνει και μόνο με τη χρήση JOIN (βλ. script from.sql)
- Ωστόσο στο παρ.2.2. βλέπουμε μία χρήση που γίνεται μόνο με χρήση υποερωτημάτων: Η χρήση μιας συνάρτησης σώρευσης επί του αποτελέσματος μιας συνάρτησης σώρευσης μπορεί να γίνει μόνο με χρήση υποερωτήμος.

```
-- example 2.1: DB: Sakila, Script: from.sql
SELECT cust_address_details.full_name,
       cust_address_details.country,
       SUM(amount) AS total_amount
FROM payment p
JOIN (
      SELECT c.customer_id,
            CONCAT(c.first_name, ' ', c.last_name) AS full_name,
              a.address, a.district, a.postal_code, ct.city, cn.country
      FROM customer c
            JOIN address a ON c.address_id = a.address_id
            JOIN city ct ON a.city_id = ct.city_id
            JOIN country cn ON ct.country_id = cn.country_id
       ) cust_address_details
       ON p.customer_id = cust_address_details.customer_id
GROUP BY p.customer_id
```

```
-- example 2.2: DB: Sakila, Script: from.sql
SELECT AVG(max_amount) AS result
FROM (
       SELECT CONCAT(c.first_name, ' ', c.last_name) AS full_name,
              MAX(p.amount) AS max_amount
       FROM customer c
             JOIN payment p ON c.customer_id = p.customer_id
       GROUP BY p.customer_id
       ) max_payments
```

Υποερώτημα σε WHERE clause

- Μπορούμε να χρησιμοποιήσουμε υποερώτημα που επιστρέφει μία τιμή, σε κάθε τύπο συνθήκης που χρησιμοποιούμε τιμή. Δηλαδή μπορούμε να αντικαταστήσουμε το "value" με υποερώτημα στις συνθήκες:

```
WHERE operand OP operand
operand = column | value
WHERE column BETWEEN low_value AND high_value
```

```sql
-- example 3.1: DB: world, Script: where1.sql
SELECT name AS country, population
FROM country
WHERE population > (
        SELECT AVG(population)
        FROM country
      )
ORDER BY population DESC;

-- example 3.2: DB: classic
SELECT productName, buyPrice
FROM products
WHERE buyPrice = (SELECT MAX(buyPrice) FROM products)
      OR buyPrice = (SELECT MIN(buyPrice) FROM products);
```

Άσκηση 2: **DB: classic - Πίνακας products**

Επιστρέψτε το όνομα και την τιμή αγοράς, των προϊόντων με τιμή από x έως y, όπου x η μέση τιμή και y η μέγιστη τιμή προϊόντος

Άσκηση 3: **DB: classic - Πίνακας customers, orders**

Εντοπίστε τους πελάτες που έχουν παραγγείλει (ανεξάρτητα με το αν έχει διεκπεραιωθεί - ολοκληρωθεί η αγορά) τουλάχιστον ένα ακριβό προϊόν (ως ακριβό, θεωρούμε ένα προϊόν με τιμή από x έως y, όπου x η μέση τιμή και y ή μέγιστη τιμή προϊόντος). Επιστρέψτε μόνο τις ονομασίες των πελατών (customerName), ταξινομημένες αλφαβητικά.

ΜΑΘΗΜΑ 1.7: ΥΠΟΕΡΩΤΗΜΑΤΑ 1.3. Χρήση σε WHERE (2. Υποερ. που επιστρέφει μία στήλη 1/2)

SQL psounis

Υποερώτημα σε WHERE clause με την IN

- Μπορούμε να χρησιμοποιήσουμε υποερώτημα που επιστρέφει μία στήλη (λίστα τιμών), σε συνθήκη IN:

```
WHERE column [NOT] IN (subquery)
```

```
-- example 4.1: DB: classic, Script: from2.sql
SELECT CONCAT(firstName, ' ', lastName) AS supervisedSF
FROM employees
WHERE reportsTo IN
        (
         SELECT employeeNumber
         FROM offices o JOIN employees e
              ON o.officeCode = e.officeCode
         WHERE o.city = 'San Francisco'
        )
```

Παρατήρηση:

- Γενικά, ένα ερώτημα με την IN σε υποερώτημα μπορεί να επαναδιατυπωθεί με χρήση INNER JOIN.

Άσκηση 4:

Επαναδιατυπώστε το παράδειγμα 4.1 με χρήση INNER JOIN.

Υποερώτημα σε WHERE clause με την EXISTS

- Μπορούμε να χρησιμοποιήσουμε υποερώτημα που επιστρέφει μία οτιδήποτε, σε συνθήκη με την EXISTS:

```
WHERE [NOT] EXISTS (subquery)
```

- Η EXISTS αληθεύει αν το υποερώτημα επιστρέψει έστω μία τιμή

```
-- example 4.2:
SELECT customerNumber
FROM customers c
WHERE NOT EXISTS
              (SELECT *
              FROM orders o
              WHERE o.customerNumber = c.customerNumber)
```

Παρατηρήσεις:

- Ο έλεγχος γίνεται για κάθε γραμμή που προκύπτει από το FROM
- Η επιλογή πεδίων στο υποερώτημα, δεν έχει σημασία, γι' αυτό συνήθως βάζουμε το *)

Άσκηση 5:

Επαναδιατυπώστε το παράδειγμα 4.2 με χρήση OUTER JOIN.

ΜΑΘΗΜΑ 1.7: ΥΠΟΕΡΩΤΗΜΑΤΑ 1.3. Χρήση σε WHERE (2. Υποερ. που επιστρέφει μία στήλη 2/2)

SQL psounis

Υποερώτημα σε WHERE clause με ALL|ANY|SOME

- Μπορούμε να χρησιμοποιήσουμε υποερώτημα που επιστρέφει μία στήλη (λίστα τιμών), απαιτώντας ένα πεδίο να σχετίζεται με όλες ή τουλάχιστον μία από τος τιμές της λίστας:

```
WHERE column op ALL|ANY|SOME (subquery)
```

  - op: Σχεσιακός τελεστής

Παρατηρήσεις:

- Το ALL μεταφράζεται σε "όλα" έτσι π.χ. το "WHERE column > ALL (subquery) σημαίνει ότι το πεδίο column πρέπει να είναι > από όλες τις τιμές που επέστρεψε το ερώτημα, άρα > από τη μέγιστη τιμή που περιέχει η λίστα τιμών.
- Το SOME (και ισοδύναμα το ANY) μεταφράζεται σε "κάποιο", έτσι π.χ. το "WHERE column > SOME (subquery) σημαίνει ότι το πεδίο column πρέπει να είναι > από κάποια τιμή που επέστρεψε το ερώτημα, άρα > από την ελάχιστη τιμή που περιέχει η λίστα.

	ALL	ANY|SOME
>	> μέγιστο(subquery)	> ελάχιστο(subquery)
>=	>= μέγιστο(subquery)	>= ελάχιστο(subquery)
=	= v1 AND =v2 AND…	IN(v1, v2, …)
<=	<= ελάχιστο(subquery)	<= μέγιστο(subquery)
<	< ελάχιστο(subquery)	< μέγιστο(subquery)
<>	NOT IN (v1, v2, …)	<> 1 OR <> 2

```
-- example 5.1: DB: world, script: where3.sql
SELECT name AS country, population
FROM country
WHERE population > ALL (
        SELECT population
        FROM country
        WHERE Continent IN ('Europe', 'North America')
      );
```

Άσκηση 6:

Α/ Χρησιμοποιώντας υποχρεωτικά την ALL, βρείτε όλες τις χώρες που έχουν έτος ανεξαρτησίας μικρότερο από οποιαδήποτε χώρα που βρίσκεται στην Ευρώπη.

Β/ Επαναλάβετε το ερώτημα χωρίς να χρησιμοποιήσετε την ANY.

Σημείωση:

- Καθότι μπορούμε να τα αντικαταστήσουμε με άλλα συντακτικά στοιχεία, σπανίως θα χρησιμοποιούμε τα SOME/ANY/ALL στην πράξη.

Υποερώτημα σε HAVING

- Όλες οι περιπτώσεις που μελετήσαμε για το WHERE ισχύουν και για το HAVING:

```
HAVING condition
```

```
-- example 6.1: DB: world, Script: from2.sql
SELECT Continent, AVG(LifeExpectancy)
FROM country
GROUP BY Continent
HAVING AVG(LifeExpectancy) >
        (
         SELECT AVG(LifeExpectancy)
         FROM country
        )
ORDER BY 2 DESC;
```

Άσκηση 7:

Εμφανίστε τις ηπείρους και το πλήθος των πόλεών τους, όπου το πλήθος των πόλεων πρέπει να είναι μικρότερο από το πλήθος των πόλεων που βρίσκονται στην Κίνα και την Ινδία

Επισήμανση:

- Δεν επιτρέπεται να έχουμε ORDER BY (δεν έχει νόημα) σε ένα subquery.

DISS: Subqueries vs Joins

- Πλεονεκτήματα Join:
  - Είναι πιο μαζεμένο σαν σκέψη (παράγουμε έναν τελικό πίνακα επί του οποίου π.χ. κάνουμε επιλογή)
  - Η SELECT μπορεί να διαλέξει πεδία από όλους τους πίνακες.
- Πλεονεκτήματα Subqueries:
  - Μπορούμε να διασπάσουμε τη σκέψη μας σε βήματα (π.χ. θα πάρω αυτό το αποτέλεσμα από αυτό το subquery, μετά το άλλο αποτέλεσμα από το άλλο subquery, και έπειτα θα τα εισάγω στο μεγάλο query που θα τα περιέχει)
  - Η διάσπαση σε βήματα, φαίνεται προτιμότερη όταν το ερώτημα είναι πραγματικά μεγάλο (π.χ. να έχει δεδομένα από 10 πίνακες)
  - Ότι μπορούμε να διοχετεύσουμε αποτελέσματα σωρευτικών συναρτήσεων, σε ερωτήματα.
- Ωστόσο:
  - Αν εξαιρέσουμε το τελευταίο (σωρευτικά αποτελέσματα) όλα τα υπόλοιπα γίνονται και με JOIN. Ακόμη και το FROM μπορεί να γίνει εναλλακτικά με views (βλ. επόμενο μάθημα)
- Η συζήτηση δεν σταματά, αν συνεκτιμήσουμε την ταχύτητα των εναλλακτικών τρόπων (μάθημα 1.8), google στον ελεύθερο χρόνο: “Join vs Subquery MySQL”
- Μάλλον δείχνει να επικρατεί το JOIN για τη MySQL οπότε αυτό θα προτιμάμε γενικά στην κατασκευή ερωτημάτων, εκτός και αν εμπίπτει στις 2 κατηγορίες: Πραγματικά τεράστιο query ή που απαιτεί εσωτερικά, σωρευτικό αποτέλεσμα.