# Databases

## 9. Subqueries

HO
GENT

# Basisvorm

- Subqueries
  - Basisvorm

```
SELECT
FROM
WHERE condition
```



Bevat in de linker- en/of rechterkant van de instructie nog een SELECT

    - Buitenste query = eerste SELECT = de belangrijkste vraag
    - Binneste query = SELECT in WHERE/HAVING = de subquery
      - Wordt altijd eerst uitgevoerd
      - Staat altijd tussen (en)
      - Subqueries kunnen op hun beurt genest worden
  - Een subquery kan **een waarde of een lijst van waarden** retourneren

# Subquery die één waarde retourneert

- Resultaat van de subquery = kan overal gebruikt worden waar een expressive zou gebruikt worden
  - Met alle relationele operators: =, >, <, <=,>=,<>
  - Voorbeeld:
    - Wat is de UnitPrice van het duurste product?

```
SELECT MAX(UnitPrice) As MaxPrice FROM Products
```

| | MaxPrice |
|---|---|
| 1 | 263,50 |

    - Wat is het duurste product?

```
SELECT ProductID, ProductName, UnitPrice As MaxPrice
FROM Products
WHERE UnitPrice = (SELECT MAX(UnitPrice) FROM Products)
```

| | ProductID | ProductName | MaxPrice |
|---|---|---|---|
| 1 | 38 | Côte de Blaye | 263,50 |

    - Wat is het verschil met het volgende?

```
SELECT ProductID, ProductName, UnitPrice As MaxPrice
FROM Products
ORDER BY UnitPrice DESC
LIMIT 1
```

# Subquery die één waarde retourneert

- Andere voorbeelden
  - Geef alle producten die meer kosten dan de gemiddelde prijs

```
SELECT ProductID, ProductName, UnitPrice As MaxPrice
FROM Products
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products)
```

  - Wie is de jongste werknemer uit Amerika?

```
SELECT LastName, FirstName
FROM Employees
WHERE Country = 'USA'
AND BirthDate = (SELECT MAX(BirthDate) FROM Employees WHERE Country = 'USA')
```



HO GENT

# Subquery die één kolom retourneert

- De resulterende kolom kan gebruikt worden als een lijst
  - Operators IN, NOT IN, ANY, ALL
  - IN operator (=ANY operator)
    - Voorbeeld: Geef alle employees die orders verwerkt hebben

```
SELECT e.EmployeeID, CONCAT(e.FirstName, ' ', e.LastName) As FullName
FROM Employees e
WHERE e.EmployeeID IN (SELECT DISTINCT EmployeeID FROM Orders)
```

    - Dit kan ook bekomen worden met een JOIN

```
SELECT DISTINCT e.EmployeeID, CONCAT(e.FirstName, ' ', e.LastName) As FullName
FROM Employees e JOIN Orders o ON e.EmployeeID = o.EmployeeID
```

# Subquery die één kolom retourneert

- De resulterende kolom kan gebruikt worden als een lijst
  - Operators IN, NOT IN, ANY, ALL
  - NOT IN operator
    - Voorbeeld: Geef alle klanten die nog geen orders geplaatst hebben tot nu toe

```
SELECT *
FROM Customers
WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM Orders)
```

    - Dit kan ook bekomen worden met een JOIN

```
SELECT *
FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.CustomerID is NULL
```



HO GENT

# Subquery die één kolom retourneert

- Andere voorbeelden
  - Geef  de companyname van de transporteurs die nog niets hebben vertransporteerd

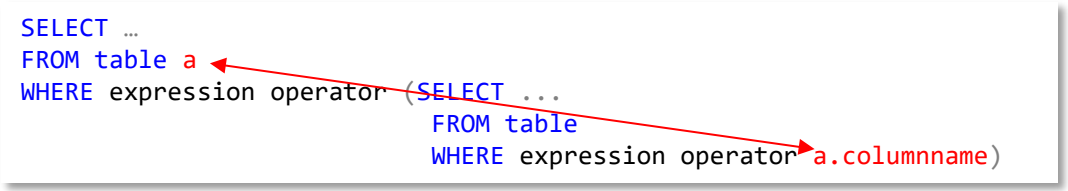
```
SELECT DISTINCT CompanyName
FROM Shippers
WHERE ShipperID NOT IN (Select ShipperID FROM Orders)
```



HO GENT

# Gecorreleerde subqueries

- Principe:
  In de subquery worden velden gebruikt uit de buitenste query




- Opmerking: in de binnenste query kan men velden uit de tabellen van de buitenste query gebruiken, maar NIET andersom.
- Gebruik om prestatieredenen waar mogelijk joins of eenvoudige subquery's.



HO GENT

# Gecorreleerde subqueries

- Gecorreleerde subquery => de binnenste query is afhankelijk van informatie uit de buitenste query.
  - Subquery bevat voorwaarde die verwijst naar hoofdquery.
  - Subquery wordt uitgevoerd voor elke rij in hoofdquery.
- Volgorde van uitvoering is van boven naar beneden → $O(n^2)$!
  - Niet van beneden naar boven zoals bij eenvoudige subquery's → O(n).

# Gecorreleerde subqueries

1. Volgende rij in de buitenste query

2. De buitenste query geeft kolomwaarden voor die rij door aan de binnenste query

3. De binnenste query gebruikt deze waarden bij het bepalen van het resultaat van de binnenste query

4. De binnenste query retourneert een waarde naar de buitenste query, die beslist of de rij in de buitenste query behouden blijft

Terug naar stap 1.



HO GENT

# Gecorreleerde subqueries

- Voorbeeld: Geef de producten die meer kosten dan gemiddeld

```
SELECT ProductId, ProductName, UnitPrice
FROM Products
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products)
```

- Voorbeeld: Geef de producten die meer kosten dan de gemiddelde prijs van de producten uit dezelfde categorie

Volledig Verschillend!

```
SELECT ProductId, ProductName, UnitPrice, CategoryId
FROM Products p
WHERE UnitPrice > (SELECT AVG(UnitPrice) FROM Products WHERE categoryid =
p.categoryId)
```



HO GENT

# Subqueries en de EXISTS operator

- De operator EXISTS test of er een result set bestaat
  - Voorbeeld: Geef alle customers die al een order hebben geplaatst

```
SELECT *
FROM Customers As c
WHERE EXISTS
(SELECT * FROM Orders WHERE CustomerID = c.customerID)
```

- Er bestaat ook iets als NOT EXISTS
  - Voorbeeld: Geef alle customers die al nog géén order hebben geplaatst

```
SELECT *
FROM Customers As c
WHERE NOT EXISTS
(SELECT * FROM Orders WHERE CustomerID = c.customerID)
```



HO GENT

# 3 verschillende manier om hetzelfde resultaat te bekomen

- Voorbeeld: Geef alle customers die al nog géén order hebben geplaatst
  - OUTER JOIN

```
SELECT *
FROM Customers c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.CustomerID is NULL
```

  - Eenvoudige subquery

```
SELECT *
FROM Customers
WHERE CustomerID NOT IN (SELECT DISTINCT CustomerID FROM Orders)
```

  - Gecorreleerde subquery

```
SELECT *
FROM Customers As c
WHERE NOT EXISTS
(SELECT * FROM Orders WHERE CustomerID = c.customerID)
```

Welke werkwijze zal het traagst zijn?

HO GENT

# Subqueries in SELECT en FROM

- De vorige voorbeelden lieten zien hoe subquery's kunnen worden gebruikt in de WHERE.
- Aangezien het resultaat van een query een tabel is, kan deze ook worden gebruikt in de FROM-clausule.
- We kunnen in plaats daarvan ook CTE's gebruiken (zie RDB&DWH).



HO GENT

# Subqueries in de SELECT

- Subqueries kunnen gebruikt worden in de SELECT (eenvoudige of gecorreleerde SELECT)
  - Voorbeeld: Geef voor elk product hoeveel duurder het is dan producten uit dezelfde category.
  - Enkel ter illustratie, we zullen CTEs gebruiken!

```
SELECT ProductId, ProductName, UnitPrice, UnitPrice –
(SELECT AVG(UnitPrice)
 FROM Products
 WHERE categoryId = p.categoryId) As PriceDifference
FROM Products p
```



HO GENT

# Oefeningen

- Geef een lijst van de gebruikers uit hetzelfde land als Maison Dewey
- Geef alle orders waarvoor het leveradres verschillend is van het adres van de klant
- Geef per jobtitel de persoon die laatst aangeworven werd
- Geef voor elk order detail met korting hoeveel de hoeveelheid verschilt van de gemiddelde hoeveelheid van alle order details van hetzelfde product



HO GENT

# Oplossing

- Geef een lijst van de gebruikers uit hetzelfde land als Maison Dewey

```
--CompanyName  country
--Maison Dewey Belgium
--Suprêmes délices Belgium

SELECT CompanyName, country
FROM customers
WHERE country = (SELECT country from customers WHERE
companyName = 'Maison Dewey')
```



HO GENT

# Oplossing

- Geef alle orders waarvoor het leveradres verschillend is van het adres van de klant

```
-- 48 records

SELECT *
FROM Orders o
WHERE ShipAddress <> (SELECT Address FROM Customers c WHERE
o.customerid = c.customerID)
```



HO GENT

# Oplossing

- Geef per jobtitel de persoon die laatst aangeworven werd

```
--title FullName  HireDate
--Vice President, Sales   Andrew Fuller  2012-08-14
--Sales Representative    Anne Dodsworth 2014-11-15
--Sales Manager   Steven Buchanan   2013-10-17
--Inside Sales Coordinator   Laura Callahan 2014-03-05

SELECT title, firstname + ' ' + lastname As FullName, HireDate
FROM employees e
WHERE HireDate = (SELECT MAX(HireDate) FROM employees WHERE
title = e.title)
```



HO GENT

# Oplossing

- Geef voor elk order detail met korting hoeveel de hoeveelheid verschilt van de gemiddelde hoeveelheid van alle order details van hetzelfde product

```
-- 48 records

--OrderID  ProductID  Quantity   quantityDifferenceToAvg
--10250    51         35         12.2821
```

```
SELECT OrderID, ProductID, Quantity, Quantity-(SELECT
AVG(Quantity) FROM order_details
WHERE ProductID = od.ProductID) As quantityDifferenceToAvg
FROM order_details od
WHERE Discount <> 0
```



HO GENT

HO
GENT