

# SQL

- **Structured Query Language**

# Subqueries

- **Introduction to Subqueries**
- **Subquery as a derived table**
- **Subquery as an expression**
- **Subquery to correlate data**
- **IN and NOT IN**
- **EXISTS and NOT EXISTS**

# Subquery as a derived table

- Subquery is a recordset within a query that functions as a table
- It takes the place of a table in the FROM Clause

```
SELECT T.orderid, T.customerid  
FROM ( SELECT orderid, customerid  
      FROM orders ) AS T
```

# Subquery as an expression

- Subquery is evaluated and treated as an expression
- It is executed once for the query

```
SELECT productname, unitprice
      ,( SELECT AVG(unitprice) FROM products) AS average
      ,unitprice-(SELECT AVG(unitprice) FROM products) AS
difference
FROM products
```

# Subquery as an expression

- Subquery can be used to specify condition

```
SELECT productname, unitprice
      ,( SELECT AVG(unitprice) FROM products) AS average
      ,unitprice-(SELECT AVG(unitprice) FROM products) AS
difference
FROM products
WHERE unitprice > ( SELECT AVG(unitprice) FROM products)
```

# Correlated subquery

**1** Outer query passes column values to the inner query

**2** Inner query uses that value to satisfy the inner query

```
SELECT productname, unitprice
      ,( SELECT AVG(unitprice)
        FROM products as p_in
        WHERE p_out.categoryid = p_in.categoryid ) AS
      average
FROM products as p_out
```

**3** Inner query returns a value back to the outer query

**4** The process is repeated for the next row of the outer query



**Back to Step 1**

# Correlated subquery

- Subquery can be used to specify condition

```
SELECT productname, unitprice
      ,( SELECT AVG(unitprice) FROM products as p_wew
        WHERE p_zew.category_id = p_wew.categoryid ) AS
      average
FROM products as p_zewn
WHERE price >
      ( SELECT AVG(unitprice) FROM products as p_wew
        WHERE p_zew.category_id = p_wew.categoryid )
```

# Example

- This example returns a list of products and the largest order ever placed for each product in the order details table

```
SELECT DISTINCT productid, quantity
FROM [order details] AS ord1
WHERE quantity = ( SELECT MAX(quantity)
                   FROM [order details] AS ord2
                   WHERE ord1.productid =
ord2.productid )
```

```
select productid, max(quantity)
from [order details]
group by productid
```



# Example

- This example returns a list of customers who ordered more than 20 pieces of product number 23.

```
SELECT orderid, customerid
FROM orders AS or1
WHERE 20 < (SELECT quantity
            FROM [order details] AS od
            WHERE or1.orderid = od.orderid
            AND  od.productid = 23)
```

# EXISTS

- Use with correlated subqueries
- Determine whether data exists
- Outer query tests for the existence of rows
- Inner query returns TRUE or FALSE
- No data is produced

```
SELECT lastname, employeeid
FROM employees AS e
WHERE EXISTS (SELECT * FROM orders AS o
              WHERE e.employeeid = o.employeeid
              AND o.orderdate = '9/5/97')
```

- This example uses a correlated subquery with an EXISTS operator in the WHERE clause to return a list of employees who took orders on '9/5/97'.

# EXISTS vs JOIN

- subquery

```
SELECT lastname, employeeid
FROM employees AS e
WHERE EXISTS (SELECT * FROM orders AS o
              WHERE e.employeeid = o.employeeid
                 AND o.orderdate = '9/5/97')
```

- join

```
SELECT DISTINCT lastname, e.employeeid
FROM orders AS o
INNER JOIN employees AS e
  ON o.employeeid = e.employeeid
WHERE o.orderdate = '9/5/1997'
```

# Not EXISTS

```
SELECT lastname, employeeid
FROM employees AS e
WHERE not EXISTS (SELECT * FROM orders AS o
                  WHERE e.employeeid = o.employeeid
                  AND o.orderdate = '9/5/97')
```

# IN

- Inner query generates list of elements
- Outer query tests for the existence of rows on a list

```
USE northwind
SELECT lastname, employeeid
FROM employees AS e
WHERE employeeid IN (SELECT employeeid FROM orders AS o
                     WHERE o.orderdate = '9/5/97')
```

- This example uses a subquery with an IN operator in the WHERE clause to return a list of employees who took orders on '9/5/97'.

# Not IN

```
USE northwind
SELECT lastname, employeeid
FROM employees AS e
WHERE employeeid NOT IN (SELECT employeeid FROM orders AS o
                        WHERE o.orderdate = '9/5/97')
```

# JOIN vs EXISTS vs IN

- **JOIN**

```
SELECT DISTINCT lastname, e.employeeid
FROM orders AS o
INNER JOIN employees AS e
  ON o.employeeid = e.employeeid
WHERE o.orderdate = '9/5/1997'
```

- **EXIST**

```
SELECT lastname, employeeid
FROM employees AS e
WHERE EXISTS (SELECT * FROM orders AS o
              WHERE e.employeeid = o.employeeid
              AND   o.orderdate = '9/5/97')
```

- **IN**

```
SELECT lastname, employeeid
FROM employees AS e
WHERE employeeid in (SELECT employeeid FROM orders AS o
                    WHERE o.orderdate = '9/5/97')
```