Zadanie Z1

---

a) Działanie operatora

WHERE (RowNumber > 51) AND (RowNumber < 100)

Operator ten, na pierwszy rzut oka stanowi rozwiązanie dla problemu "*stronicowania*" wyników, natomiast tak nie jest.

Klauzula *WHERE* określa te wiersze, które mają zostać zaznaczone, a funkcje pozycjonujące działają na **gotowych** już wynikach. Tak więc występuję błąd, ponieważ klauzula *WHERE* jest uwzględniana jeszcze **przed** wywołaniem funkcji Row_Number. By wykorzystać tą linię w poprawny sposób, musimy skorzystać z **wyrażeń tablicowych** (CTE) lub wewnętrznych i zewnętrznych zapytań. Wtedy ten operator da nam wyniki, których **RowNumber** znajduje się w przedziale (51, 100);

b) "Stronicowanie"

I. (Tabela temporary)

```
-- Korzystamy z bazy AdventureWorks
USE AdventureWorks
GO

-- Jeśli obiekt temp już istnieje, to DROP
IF OBJECT_ID('temp') IS NOT NULL
   DROP TABLE temp;
GO

-- Tworzymy tabele temporary
CREATE TABLE temp (
    RowNumber int,
    ContactID int,
    FirstName nvarchar(50),
    LastName nvarchar(50)
);

-- Korzystamy z INSERT from SELECT (INSERT INTO ... SELECT) by wypelnic
tabele temporary
INSERT INTO temp SELECT Row_Number() OVER (ORDER BY LastName) AS
RowNumber, ContactID, FirstName, LastName
FROM Person.Contact ORDER BY RowNumber

-- Z tabeli temporary wyciagamy stronicowane wyniki
SELECT * from temp WHERE (RowNumber > 51) AND (RowNumber < 100);
```

II. (Wyrażenie CTE)

```
-- Korzystamy z bazy AdventureWorks
USE AdventureWorks
GO

-- Korzystamy z CTE
WITH CTE_Person AS
(
      SELECT Row_Number() OVER (ORDER BY LastName)
      AS RowNumber, ContactID, FirstName, LastName
      FROM Person.Contact
)

SELECT * from CTE_Person WHERE (RowNumber > 51) AND (RowNumber < 100);
```

Zadanie Z2

Raport podający ilość dostawców

```
-- Korzystamy z bazy AdventureWorks
USE AdventureWorks
GO

-- Wyciagamy dane z odpowiednich tabel
SELECT psp.Name AS State, pa.City AS City, COUNT(psp.Name) AS "Number of
vendors" FROM Purchasing.Vendor pv
JOIN Purchasing.VendorAddress pva ON pv.VendorID = pva.VendorID
JOIN Person.Address pa ON pva.AddressID = pa.AddressID
JOIN Person.StateProvince psp ON pa.StateProvinceID = psp.StateProvinceID

-- Korzystamy z funkcji grupujących
GROUP BY GROUPING SETS(
    (psp.Name),
    (psp.Name, pa.City)
)

ORDER BY State;
```

Zadanie Z3

a) CASE (zapytanie dla przykładu z Z2)

```
-- Korzystamy z bazy AdventureWorks
USE AdventureWorks
GO

-- Wyciagamy dane z odpowiednich tabel
-- Korzystamy z CASE do ogolnego okreslenia liczby dostawcow i opisania
NULL'i
SELECT
psp.Name AS State,
CASE
    WHEN pa.City IS NULL THEN 'Not initialized'
    ELSE pa.City
END AS City,
CASE
    WHEN COUNT(psp.Name) > 2 THEN 'More than average'
    WHEN COUNT(psp.Name) = 2 THEN 'Average'
    WHEN COUNT(psp.Name) < 2 THEN 'Less than average'
END AS "Number of vendors"
FROM Purchasing.Vendor pv
JOIN Purchasing.VendorAddress pva ON pv.VendorID = pva.VendorID
JOIN Person.Address pa ON pva.AddressID = pa.AddressID
JOIN Person.StateProvince psp ON pa.StateProvinceID =
psp.StateProvinceID

-- Korzystamy z funkcji grupujących
GROUP BY GROUPING SETS(
    (psp.Name),
    (psp.Name, pa.City)
)

ORDER BY State;
```

---

b) Utworzenie tabeli z pomiarami i wykorzystanie PIVOT dla pokazania agregatów

---

```
-- Korzystamy z bazy tempdb
USE tempdb
GO

-- Jeśli obiekt measurements już istnieje, to DROP
IF OBJECT_ID('measurements') IS NOT NULL
    DROP TABLE measurements;
GO

-- Tworzymy tabele na pomiary
CREATE TABLE measurements(
    hour int NOT NULL,
    minute int NOT NULL,
    co2 float NOT NULL,
    vehicle_count int NOT NULL
);

-- Dodajemy dane do tabeli
INSERT INTO measurements VALUES(7, 20, 0.30, 45);
INSERT INTO measurements VALUES(7, 30, 0.24, 34);
INSERT INTO measurements VALUES(7, 40, 0.26, 35);
INSERT INTO measurements VALUES(8, 10, 0.19, 30);
INSERT INTO measurements VALUES(8, 30, 0.13, 18);
INSERT INTO measurements VALUES(10, 35, 0.08, 12);

-- Pokazujemy PIVOT'em agregaty dla mierzonej wartosci ilosci pojazdow
GO
SELECT * FROM measurements
PIVOT (SUM(vehicle_count) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;

SELECT * FROM measurements
PIVOT (MAX(vehicle_count) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;

SELECT * FROM measurements
PIVOT (MIN(vehicle_count) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;

-- Pokazujemy PIVOT'em agregaty dla mierzonej wartosci CO2
SELECT * FROM measurements
PIVOT (SUM(co2) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;

SELECT * FROM measurements
PIVOT (MAX(co2) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;

SELECT * FROM measurements
PIVOT (MIN(co2) FOR hour IN ([7], [8], [9], [10])) AS a
ORDER BY minute;
```
