SQL

ALICJA SAWICKA

APRIL 2022

Agenda

FIRST MEETING:

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

SECOND MEETING:

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null
11. Between

THIRD MEETING:

12. Numeric functions: min, max, avg, count, sum
13. Len, concat, left, right
14. Upper, lower
15. Reverse, Replace, Substring

FOURTH MEETING:

13. Join
14. Group by, having

FIFTH MEETING:

15. Case when
16. Subqueries

SIXTH MEETING:

17. Datatype
18. Keys
19. Create, alter, drop
20. Insert, update, delete

SQL - overview

SELECT

SELECT * FROM *table*

Use DRBR

SELECT * FROM SQL.STORY
SELECT TOP 20 * FROM SQL.STORY

SELECT TOP 20 AssigneeName FROM SQL.STORY
SELECT AssigneeName FROM SQL.STORY

SELECT *kolumna1 + kolumna2* from *tabela*

SELECT *kolumna1 + ',' + kolumna2 + ',* from *tabela*

SELECT *kolumna1 + ' ' + kolumna2 + ' '* from *tabela*

SELECT 'zadanie: ' + *kolumna1 + ', ' + kolumna2 + '.'* from *tabela*

SELECT *kolumna1* from *tabela*

SELECT *kolumna1* AS wynik from *tabela*

Aliases

```
SELECT IssueType, IssueKey, IssueStatus, CreatedAt FROM sql.story

SELECT IssueType AS TYP, IssueKey AS [KEY], IssueStatus AS Status, CreatedAt AS [CREATE DATE] FROM sql.story
```

100 %

Results | Messages

	IssueType	IssueKey	IssueStatus	CreatedAt
1	Epic	DIDMB-1106	Abandoned	2020-07-07 07:19:05.0000000 -05:00
2	Epic	DIDMB-1398	Ready for Development	2020-10-29 07:39:47.0000000 -05:00
3	Epic	DIDMB-1464	Blocked/Waiting/On Hold	2020-12-02 06:15:43.0000000 -06:00
4	Epic	DIDMB-1551	Abandoned	2021-01-07 05:10:40.0000000 -06:00
5	Epic	DIDMB-1675	Ready for Development	2021-01-26 03:23:50.0000000 -06:00
6	Epic	DIDMB-1841	Done	2021-02-24 05:21:24.0000000 -06:00
7	Epic	DIDMB-1945	Done	2021-03-24 04:28:43.0000000 -05:00
8	Epic	DIDMB-2027	In Progress	2021-04-23 04:25:01.0000000 -05:00
9	Epic	DIDMB-2033	Ready for Development	2021-04-24 12:15:42.0000000 -05:00
10	Epic	DIDMB-2035	Ready for Development	2021-04-27 07:37:11.0000000 -05:00
11	Epic	DIDMB-2036	Done	2021-04-27 07:50:48.0000000 -05:00
12	Epic	DIDMB-2037	Done	2021-04-27 09:21:16.0000000 -05:00

	TYP	KEY	Status	CREATE DATE
1	Epic	DIDMB-1106	Abandoned	2020-07-07 07:19:05.0000000 -05:00
2	Epic	DIDMB-1398	Ready for Development	2020-10-29 07:39:47.0000000 -05:00
3	Epic	DIDMB-1464	Blocked/Waiting/On Hold	2020-12-02 06:15:43.0000000 -06:00
4	Epic	DIDMB-1551	Abandoned	2021-01-07 05:10:40.0000000 -06:00
5	Epic	DIDMB-1675	Ready for Development	2021-01-26 03:23:50.0000000 -06:00
6	Epic	DIDMB-1841	Done	2021-02-24 05:21:24.0000000 -06:00
7	Epic	DIDMB-1945	Done	2021-03-24 04:28:43.0000000 -05:00
8	Epic	DIDMB-2027	In Progress	2021-04-23 04:25:01.0000000 -05:00
9	Epic	DIDMB-2033	Ready for Development	2021-04-24 12:15:42.0000000 -05:00
10	Epic	DIDMB-2035	Ready for Development	2021-04-27 07:37:11.0000000 -05:00
11	Epic	DIDMB-2036	Done	2021-04-27 07:50:48.0000000 -05:00
12	Epic	DIDMB-2037	Done	2021-04-27 09:21:16.0000000 -05:00

WHERE

Operatory dla WHERE

=	Równy
>	Większy
<	Mniejszy
>=	Większy lub równy
<=	Mniejszy lub równy
<>	Nie równy
BETWEEN	Pomiędzy
LIKE	Wartości „jak”, używamy np. z %

```
SELECT * FROM sql.story
WHERE AssigneeUPIN = 'A9CDLZZ'

SELECT * FROM sql.story
WHERE AssigneeUPIN LIKE '%A9CDLZZ%'
```

Napisz zapytanie, które wyświetli:

1. wszystkie przypisane do ABVNYZZ taski (AssigneeUPIN);
2. wszystkie założone przez USASIMADB001 taski (ReporterUPIN);
3. 10 zadań w statusie Done;
4. kolumny: ReporterUPIN oraz IssueStatus z 30 wynikami;
5. kolumnę Status ze statusami innymi niż In Progress;
6. w kolumnie „Numer i data” IssueKey i CreatedAt;
7. 30 tasków założonych po 2022-01-01;
8. 30 tasków zamkniętych w 2021 roku;

AND, OR, NOT, IN

AND: WSZYSTKIE warunki są prawdą

OR: KTÓRYKOLWIEK warunek jest prawdą

NOT: warunek/warunki NIE są/jest prawdą

IN: warunek zwraca wiele wartości, skrót dla wielu stanów OR [możliwe używanie z NOT]

SELECT * FROM *sql.story*
WHERE *AssigneeUPIN*= 'a9cdlzz' AND *IssueStatus* = 'Done'

SELECT * FROM sql.story
WHERE *AssigneeUPIN* = 'a9cdlzz' OR *IssueStatus* = 'Done'

```
SELECT * FROM sql.story
WHERE ReporterName = 'a9cdlzz’
AND NOT IssueStatus = 'To Do'
```

```
JOIN td.STEP ON RN_RUN_ID = ST_RUN_ID
LEFT JOIN td.RELEASE_CYCLES ON RN_ASSIGN_RCYC = RCYC_ID
LEFT JOIN td.RELEASES ON RCYC_PARENT_ID = REL_ID
WHERE RN_USER_TEMPLATE_01 = 'Not Approved'
AND RN_STATUS = 'Passed'
AND REL_NAME = 'Monthly - 2019-09-Sep'
    AND (
        (
        (ST_DESCRIPTION LIKE '%SHOT%' OR ST_DESCRIPTION LIKE '%SCREENSHOT%' OR ST_DESCRIPTION = '%ATTACH[^MENT]%')
        OR
        (ST_EXPECTED LIKE '%SHOT%' OR ST_EXPECTED LIKE '%SCREENSHOT%' OR ST_EXPECTED = '%ATTACH[^MENT]%')
        )
AND (ST_ACTUAL NOT LIKE '%SKIP%' AND ST_ACTUAL NOT LIKE '%NOT APPLICABLE%' AND ST_ACTUAL NOT LIKE '%NA%' AND ST_ACTUAL NOT LIKE '%N/A%')
AND ST_ATTACHMENT IS NULL
AND RN_COMMENTS IS NULL
OR (ST_EXPECTED = ST_ACTUAL OR ST_ACTUAL LIKE '%XX%' OR ST_ACTUAL LIKE '%.PNG%' OR ST_ACTUAL LIKE '%.GIF%' OR ST_ACTUAL LIKE '%.JPG%' OR ST_ACTUAL LIKE '%.JPEG%'))
order by Reason, RUN_ID
```

Napisz zapytanie, które wyświetli:
1. wszystkie przypisane do ABVNYZZ taski (AssigneeUPIN);
2. wszystkie założone przez USASIMADB001 taski (ReporterUPIN);
3. 10 zadań w statusie Done;
4. kolumny: ReporterUPIN oraz IssueStatus z 30 wynikami;
5. kolumnę Status ze statusami innymi niż In Progress;
6. w kolumnie „Numer i data” IssueKey i CreatedAt;
7. 30 tasków założonych po 2022-01-01;
8. 30 tasków zamkniętych w 2021 roku;

9. taski przypisane do A9CDLZZ i zamknięte w roku 2022;
10. taski o numerze większym niż DIDMB-2000, które są Epicami w statusie Done;
11. 25 tasków z „testing” w Summary, będące Sub-taskami;
12. 15 tasków z tego roku, które są Story w kolumnach: Numer Taska, Data, Typ;
13. taski, które nie są w statusie To do, In Progress albo Done;
14. taski, które są Sub-taskami, nie mające „testing” w Summary oraz posiadające Juana lub Johna w polu AssigneeName lub ReporterName;

SQL part 2

Agenda

~~FIRST MEETING:~~

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

SECOND MEETING:

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null

THIRD MEETING:

11. Numeric functions: min, max, avg, count, sum
12. Len, concat, left, right
13. Upper, lower
14. Reverse, Replace, Substring

FOURTH MEETING:

15. Join
16. Group by, having

FIFTH MEETING:

17. Case when
18. Subqueries

SIXTH MEETING:

19. Datatype
20. Keys
21. Create, alter, drop
22. Insert, update, delete

REMINDER

WILDCARDS

WILDCARDS

%	Reprezentuje dowolny znak, ciąg znaków lub zero znaków
[ ]	Reprezentuje dowolny znak z nawiasu
^	Reprezentuje dowolny znak, którego nie ma w nawiasie
_	Reprezentuje pojedynczy znak
-	Reprezentuje zakres znaków

Podaj zapis wartości, która:

1. rozpoczyna się znakiem „a”
2. kończy się znakiem „a”
3. ma znak „a” na dowolnej pozycji
4. ma znak „a” na drugim miejscu
5. rozpoczyna się od „a” i jest co najmniej trzyznakowa
6. rozpoczyna się znakiem „a”, kończy „t”

Jaki wynik dadzą poniższe zapisy?:

1. h_t
2. h[oa]t
3. h[^oa]t
4. h[a-b]t
5. %attach%
6. attach[ment]

TOP

DISTINCT

ORDER BY

DESC
ASC

```
SELECT TOP 10 * from sql.story
ORDER BY IssueStatus

SELECT TOP 10 * from sql.story
ORDER BY IssueStatus, IssueType

SELECT TOP 10 * from sql.story
ORDER BY IssueStatus ASC, IssueType DESC
```

NULL

IS NULL

IS NOT NULL

BETWEEN

Napisz zapytanie, które wyświetli:

1. Wszystkie Story wykonane między 12-12-2020 a 14-02-2022;
2. 10 najstarszych Epiców (najwcześniejsza CreatedAt);
3. Sub-taski z 2021r. ułożone alfabetycznie po AssigneeName;
4. Story z „Enhancements” w IssueSummary;
5. Taski, które w IssueKey mają numer „12”;
6. Taski w statusie „Abandoned” stworzone dla GSC PL (IssueSummary);
7. Taski stworzone dla GSC PL, które nie mają AssigneeName;
8. Story z tą samą osobą w polu ReporterName i AssigneeName;
9. Wszystkie taski, dla których AssigneeUPIN ma ZZZ na końcu;

SQL part 3

Agenda

~~FIRST MEETING:~~

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

~~SECOND MEETING:~~

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null

THIRD MEETING:

11. Numeric functions: min, max, avg, count, sum
12. Len, concat, left, right
13. Upper, lower
14. Reverse, Replace, Substring

FOURTH MEETING:

15. Join
16. Group by, having

FIFTH MEETING:

17. Case when
18. Subqueries

SIXTH MEETING:

19. Datatype
20. Keys
21. Create, alter, drop
22. Insert, update, delete

REMINDER

NUMERIC FUNCTIONS

NUMERIC FUNCTIONS

MIN	Zwraca minimalną wartości z zestawu wartości
MAX	Zwraca maksymalną wartość z zestawu wartości
AVG	Zwraca średnią wartość wyrażeń
COUNT	Zwraca liczbę rekordów dla danego zapytania
SUM	Zwraca sumę wartości

```
SELECT MIN(StoryPoints) FROM sql.details
SELECT MIN(ReporterName) FROM sql.story

SELECT MAX(StoryPoints) FROM sql.details
SELECT MAX(ReporterName) FROM sql.story

SELECT AVG(StoryPoints) FROM sql.details

SELECT SUM(StoryPoints) FROM sql.details

SELECT COUNT(StoryPoints) FROM sql.details
SELECT COUNT(ReporterName) FROM sql.story
```

LEN, CONCAT, LEFT, RIGHT

LEN	Zwraca długość łańcucha
CONCAT	Dodaje dwa lub więcej (254) ciągów
LEFT	Zwraca ciąg znaków (zaczynając od lewej)
RIGHT	Zwraca ciąg znaków (zaczynając od prawej)

```
SELECT LEN('SQL')
SELECT LEN(' SQL')
SELECT LEN('SQL ')

SELECT CONCAT('SQL')
SELECT CONCAT('SQL', 'PYTHON')
SELECT CONCAT('SQL' , '  ' ,'PYTHON')
```

```
SELECT RIGHT (AssigneeName, 3) FROM sql.story

SELECT  LEFT (ReporterName, 3) FROM sql.story
```

UPPER, LOWER

LOWER	Zwraca ciąg znaków pisany małymi literami
UPPER	Zwraca ciąg znaków pisany dużymi literami

```
SELECT LOWER (ReporterUPIN) FROM sql.story

SELECT UPPER (ReporterUPIN) FROM sql.story
```

REVERSE, REPLACE, SUBSTRING

REVERSE	Odwraca ciąg znaków i zwraca wynik
REPLACE	Zastępuje ciąg znaków i podmienia go na nowy
SUBSTRING	Wyodrębnia znaki z ciągu znaków

```
SELECT REVERSE (ReporterName) FROM sql.story

SELECT REPLACE (AssigneeName, 'Jan Nowak', 'TEST')

FROM sql.story

SELECT SUBSTRING (IssueKey, 3 , 6) FROM sql.story
```

Napisz zapytanie, które wyświetli:

1. Najdłuższy czas spędzony nad zadaniem (SpentHrs, sql.details);
2. Assignee, którzy mają najkrótszy i najdłuższy staż w firmie (AssigneeUPIN, sql.story);
3. 10 zadań o najdłuższym IssueSummary ułożonych malejąco (IssueSummary, sql.details);
4. Zamieni „9” w AssigneeUPIN na „BO TAK” i „2” na „bo nie” (AssigneeUPIN, sql.epic);
5. W kolumnie „Numer i data” IssueKey i CreatedAt (sql.story);
6. Średnią ilość Story Pointów (StoryPoints, sql.details);
7. Ilość czasu poświęconą na wszystkie zadania dla IssueKey > DIDMB-2400 (SpentHrs, sql.details);
8. 2 ostatnie litery ReporterName, których nazwisko kończy się samogłoską (ReporterName, sql.epic);

SQL part 4

Agenda

~~FIRST MEETING:~~

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

~~SECOND MEETING:~~

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null

~~THIRD MEETING:~~

11. Numeric functions: min, max, avg, count, sum
12. Len, concat, left, right
13. Upper, lower
14. Reverse, Replace, Substring

FOURTH MEETING:

15. Join
16. Group by, having

FIFTH MEETING:

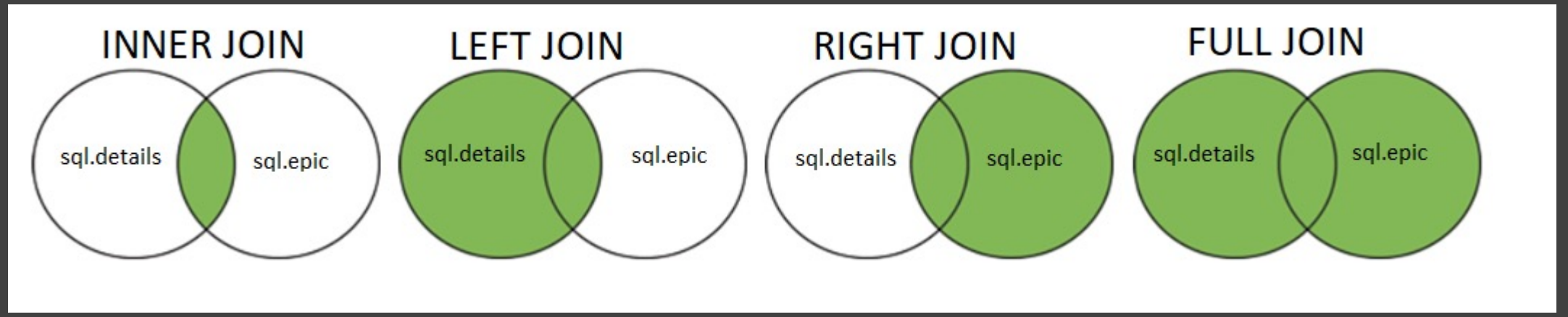
17. Case when
18. Subqueries

SIXTH MEETING:

19. Datatype
20. Keys
21. Create, alter, drop
22. Insert, update, delete

REMINDER

JOIN



SELECT A.IssueKey, B.Epic, A.EpicLink, B.AssigneeName, B.ReporterName FROM sql.details A

INNER JOIN sql.epic B

ON A.EpicLink = B.Epic

SELECT A.IssueKey, B.Epic, A.EpicLink, B.AssigneeName, B.ReporterName FROM sql.details A

LEFT OUTER JOIN sql.epic B

ON A.EpicLink = B.Epic

SELECT A.IssueKey, B.Epic, A.EpicLink, B.AssigneeName, B.ReporterName FROM sql.details A

RIGHT OUTER JOIN sql.epic B

ON A.EpicLink = B.Epic

SELECT A.IssueKey, B.Epic, A.EpicLink, B.AssigneeName, B.ReporterName FROM sql.details A

FULL OUTER JOIN sql.epic B

ON A.EpicLink = B.Epic

dbo.CAR

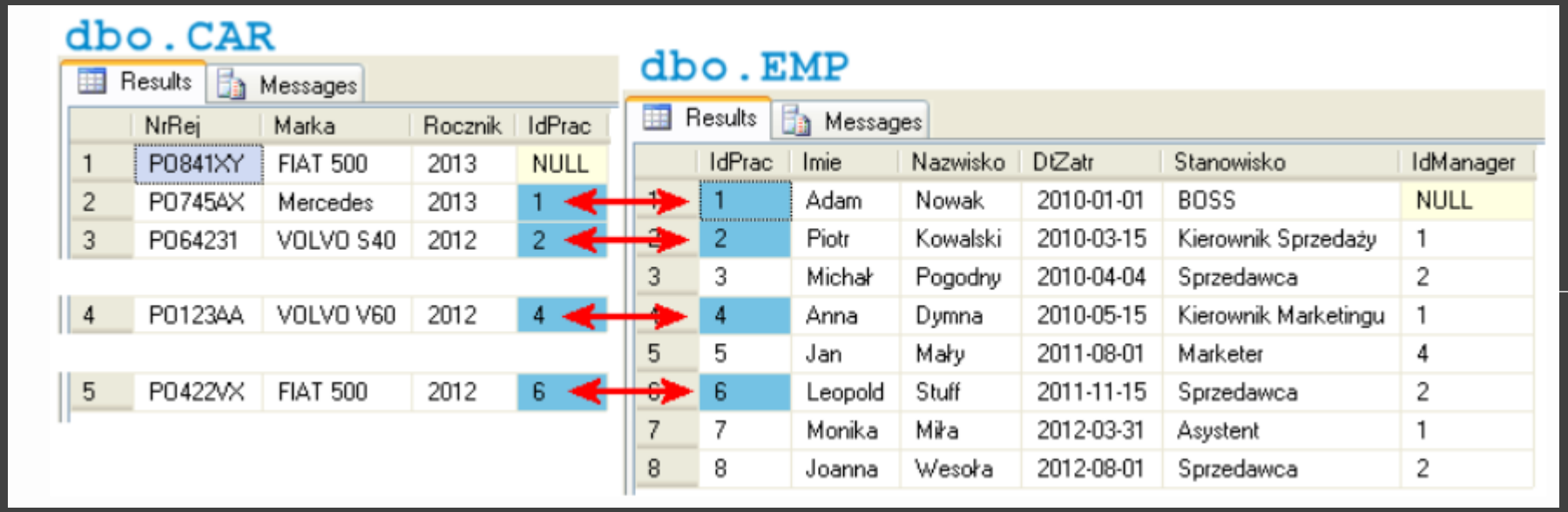
Results | Messages

	NrRej	Marka	Rocznik	IdPrac
1	P0123AA	VOLVO V60	2012	4
2	P0422VX	FIAT 500	2012	6
3	P064231	VOLVO S40	2012	2
4	P0745AX	Mercedes	2013	1
5	P0841XY	FIAT 500	2013	NULL

dbo.EMP

Results | Messages

	IdPrac	Imie	Nazwisko	DtZatr	Stanowisko	IdManager
1	1	Adam	Nowak	2010-01-01	BOSS	NULL
2	2	Piotr	Kowalski	2010-03-15	Kierownik Sprzedaży	1
3	3	Michał	Pogodny	2010-04-04	Sprzedawca	2
4	4	Anna	Dymna	2010-05-15	Kierownik Marketingu	1
5	5	Jan	Mały	2011-08-01	Marketer	4
6	6	Leopold	Stuff	2011-11-15	Sprzedawca	2
7	7	Monika	Miła	2012-03-31	Asystent	1
8	8	Joanna	Wesoła	2012-08-01	Sprzedawca	2



Results Messages

	IdPrac	Imie	Nazwisko	DtZatr	Stanowisko	IdManager	NrRej	Marka	Rocznik	IdPrac
1	1	Adam	Nowak	2010-01-01	BOSS	NULL	PO745AX	Mercedes	2013	1
2	2	Piotr	Kowalski	2010-03-15	Kierownik Sprzedaży	1	PO64231	VOLVO S40	2012	2
3	4	Anna	Dymna	2010-05-15	Kierownik Marketingu	1	PO123AA	VOLVO V60	2012	4
4	6	Leopold	Stuff	2011-11-15	Sprzedawca	2	PO422VX	FIAT 500	2012	6

```
SELECT * FROM dbo.CAR as E
JOIN dbo.EMP as C
ON e.idPrac = c.idPrac
```

dbo.CAR

Results | Messages

	NrRej	Marka	Rocznik	IdPrac
1	P0123AA	VOLVO V60	2012	4
2	P0422VX	FIAT 500	2012	6
3	P064231	VOLVO S40	2012	2
4	P0745AX	Mercedes	2013	1
5	P0841XY	FIAT 500	2013	NULL

dbo.EMP

Results | Messages

	IdPrac	Imie	Nazwisko	DtZatr	Stanowisko	IdManager
1	1	Adam	Nowak	2010-01-01	BOSS	NULL
2	2	Piotr	Kowalski	2010-03-15	Kierownik Sprzedaży	1
3	3	Michał	Pogodny	2010-04-04	Sprzedawca	2
4	4	Anna	Dymna	2010-05-15	Kierownik Marketingu	1
5	5	Jan	Mały	2011-08-01	Marketer	4
6	6	Leopold	Stuff	2011-11-15	Sprzedawca	2
7	7	Monika	Miła	2012-03-31	Asystent	1
8	8	Joanna	Wesoła	2012-08-01	Sprzedawca	2

```
SELECT e.Imie, e.Nazwisko, E.Stanowisko, C.Marka
FROM dbo.EMP as E
LEFT JOIN dbo.CAR as C
ON e.idPrac = c.idPrac
```

Results | Messages

	Imie	Nazwisko	Stanowisko	Marka
1	Adam	Nowak	BOSS	Mercedes
2	Piotr	Kowalski	Kierownik Sprzedaży	VOLVO S40
3	Michał	Pogodny	Sprzedawca	NULL
4	Anna	Dymna	Kierownik Marketingu	VOLVO V60
5	Jan	Mały	Marketer	NULL
6	Leopold	Stuff	Sprzedawca	FIAT 500
7	Monika	Miła	Asystent	NULL
8	Joanna	Wesoła	Sprzedawca	NULL

dbo.CAR

Results | Messages

	NrRej	Marka	Rocznik	IdPrac
1	PO123AA	VOLVO V60	2012	4
2	PO422VX	FIAT 500	2012	6
3	PO64231	VOLVO S40	2012	2
4	PO745AX	Mercedes	2013	1
5	PO841XY	FIAT 500	2013	NULL

dbo.EMP

Results | Messages

	IdPrac	Imie	Nazwisko	DtZatr	Stanowisko	IdManager
1	1	Adam	Nowak	2010-01-01	BOSS	NULL
2	2	Piotr	Kowalski	2010-03-15	Kierownik Sprzedaży	1
3	3	Michał	Pogodny	2010-04-04	Sprzedawca	2
4	4	Anna	Dymna	2010-05-15	Kierownik Marketingu	1
5	5	Jan	Mały	2011-08-01	Marketer	4
6	6	Leopold	Stuff	2011-11-15	Sprzedawca	2
7	7	Monika	Miła	2012-03-31	Asystent	1
8	8	Joanna	Wesoła	2012-08-01	Sprzedawca	2

Results | Messages

	Marka	NrRej	Rocznik	Pracownik
1	VOLVO V60	PO123AA	2012	Anna Dymna
2	FIAT 500	PO422VX	2012	Leopold Stuff
3	VOLVO S40	PO64231	2012	Piotr Kowalski
4	Mercedes	PO745AX	2013	Adam Nowak
5	FIAT 500	PO841XY	2013	NULL

```
SELECT c.Marka, c.NrRej, c.Rocznik, e.Imie + ' ' + e.Nazwisko as Pracownik FROM dbo.EMP e
RIGHT JOIN dbo.CAR as C
ON e.idPrac = c.idPrac
```

dbo . CAR

Results | Messages

	NrRej	Marka	Rocznik	IdPrac
1	P0123AA	VOLVO V60	2012	4
2	P0422VX	FIAT 500	2012	6
3	P064231	VOLVO S40	2012	2
4	P0745AX	Mercedes	2013	1
5	P0841XY	FIAT 500	2013	NULL

dbo . EMP

Results | Messages

	IdPrac	Imie	Nazwisko	DtZatr	Stanowisko	IdManager
1	1	Adam	Nowak	2010-01-01	BOSS	NULL
2	2	Piotr	Kowalski	2010-03-15	Kierownik Sprzedaży	1
3	3	Michał	Pogodny	2010-04-04	Sprzedawca	2
4	4	Anna	Dymna	2010-05-15	Kierownik Marketingu	1
5	5	Jan	Mały	2011-08-01	Marketer	4
6	6	Leopold	Stuff	2011-11-15	Sprzedawca	2
7	7	Monika	Miła	2012-03-31	Asystent	1
8	8	Joanna	Wesoła	2012-08-01	Sprzedawca	2

Results | Messages

	Imie	Nazwisko	Stanowisko	Marka
1	Adam	Nowak	BOSS	Mercedes
2	Piotr	Kowalski	Kierownik Sprzedaży	VOLVO S40
3	Michał	Pogodny	Sprzedawca	NULL
4	Anna	Dymna	Kierownik Marketingu	VOLVO V60
5	Jan	Mały	Marketer	NULL
6	Leopold	Stuff	Sprzedawca	FIAT 500
7	Monika	Miła	Asystent	NULL
8	Joanna	Wesoła	Sprzedawca	NULL
9	NULL	NULL	NULL	FIAT 500

```
SELECT e.Imie.Nazwisko, e. Stanowisko, c.Marka FROM dbo.EMP e
FULL JOIN dbo.CAR as C
ON e.idPrac = c.idPrac
```

GROUP BY

```
SELECT IssueStatus, IssueKey FROM sql.story
GROUP BY IssueKey
```

```
SELECT IssueStatus, IssueType FROM sql.story
WHERE AssigneeName = 'Jan Nowak'
GROUP BY IssueStatus, IssueType
```

```
SELECT MAX(Issuekey), IssueType FROM sql.story
```

HAVING

```
SELECT IssueStatus, IssueType FROM sql.story
WHERE AssigneeName = 'Jan Nowak'
GROUP BY IssueStatus, IssueType
HAVING IssueType = 'Story'
```

Kolejność wykonywania zapytań:

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

Napisz zapytanie, które wyświetli:

1. Joina dla wszystkich wyników w tabelach:
   a) left i right: sql.story i sql.details
   (wymagane kolumny: IssueKey x2, ReporterName, AssigneeName, IssueStatus, IssueSummary x2, StoryPoints);
   b) left i right: sql.details i sql.epic
   (wymagane kolumny: IssueKey, EpicLink, Epic, StoryPoints, ReporterName, AssigneeName, EpicStatus);
   c) inner, left i right: sql.story i sql.epic
   (wymagane kolumny: IssueKey x2, ReporterName x2, AssigneeName x2, IssueStatus, IssueSummary x2, StoryPoints, EpicLink, Epic, EpisStatus);

2. IssueKey, IssueStatus i StoryPoints dla wszystkich danych (sql.story, sql.details);
3. Ilość czasu poświęconą na wszystkie zadania w roku 2022 (sql.story, sql.details);
4. IssueKey, Epic, EpicStatus dla zadań, które nie mają StoryPointsów (sql.details, sql.epic);
5. Epic, IssueKey, EpicStatus, IssueStatus, ReporterName, AssigneeName dla Epiców w statusie „In Progress” i brakiem ReporterName dla Issue (sql.story, sql.details, sql.epic);
6. Issue stworzone w 2021 bez AssigneeName – dla tych wyników wskaż StoryPointsy (wymagane kolumny: IssueKey, CreatedAt, StoryPoints) (sql.story, sql.details);

SQL part 5

Agenda

~~FIRST MEETING:~~

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

~~SECOND MEETING:~~

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null

~~THIRD MEETING:~~

11. Numeric functions: min, max, avg, count, sum
12. Len, concat, left, right
13. Upper, lower
14. Reverse, Replace, Substring

~~FOURTH MEETING:~~

15. Join
16. Group by, having

FIFTH MEETING:

17. Case when
18. Subqueries

SIXTH MEETING:

19. Datatype
20. Keys
21. Create, alter, drop
22. Insert, update, delete

CASE WHEN

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END
```

```
SELECT  IssueStatus,
        CASE
        WHEN IssueStatus = 'Done' THEN 'OK'
        WHEN IssueStatus = 'In Progress' THEN 'Waiting'
        ELSE 'Ask Product Owner'
        END AS Statusy
from sql.story
```

```
SELECT IssueKey, IssueStatus, AssigneeName, ReporterName
FROM sql.story
ORDER BY IssueStatus,
CASE WHEN IssueStatus = 'Done' THEN AssigneeName END ASC,
CASE WHEN IssueStatus = 'In Progress' THEN ReporterName END
```

SUBQUERIES

ZAPYTANIE NIESKORELOWANE

tzw. podzapytanie proste, niezależne

1. Wykonujemy najpierw podzapytanie
2. A następnie zwrócone przez nie wartości podstawiamy do zapytania zewnętrznego, które zostaje wykonane

```
SELECT AssigneeName FROM sql.story
WHERE IssueKey =
                    (SELECT IssueKey FROM sql.details
                    WHERE EpicLink = 'DIDMB-2331')
```

```
SELECT ReporterName FROM sql.story
WHERE IssueKey =
        (SELECT IssueKey FROM sql.details
        WHERE EpicLink LIKE 'DIDMB-%12%')
```

```
SELECT ReporterName FROM sql.story
WHERE IssueKey IN
        (SELECT IssueKey FROM sql.details
        WHERE EpicLink LIKE 'DIDMB-%12%')
```

ZAPYTANIE SKORELOWANE

powiązane z zapytaniem nadrzędnym

1. Zapytanie główne odczytuje wszystkie rekordy z tabeli NAD
2. Dla pierwszego odczytanego rekordu wykonywane jest zapytanie podrzędne
3. Jeżeli wartość odczytana w zapytaniu nadrzędnym jest większa niż wartość obliczona w zapytaniu podrzędnym, dany rekord wchodzi do wyniku rozwiązania, a jeżeli nie to zostaje odrzucony

Wyświetl te IssueKey, których StoryPointsy są większe równe niż średnia StoryPointsów dla danego SpentHrs

```
SELECT IssueKey, StoryPoints, SpentHrs

FROM sql.details NAD

WHERE StoryPoints >=
            (
            SELECT AVG(StoryPoints)
            FROM sql.details POD
            WHERE NAD.SpentHrs = POD.SpentHrs
            )
ORDER BY IssueKey
```

Krok pierwszy:
Zapytanie główne odczytuje wszystkie rekordy z tabeli sql.details

Krok drugi:
Dla pierwszego odczytanego rekordu wykonywane jest zapytanie podrzędne (które znajduje AVG(Story_Points)):
(co możliwe jest dzięki wykonaniu korelacji w klauzuli WHERE, łączącego wyniki zapytań nadrzędnego i podrzędnego)

Krok trzeci:
Jeżeli wartość StoryPoints odczytana w zapytaniu nadrzędnym jest większa niż wartość obliczona w zapytaniu podrzędnym, dany rekord wchodzi do wyniku rozwiązania, a jeżeli nie to zostaje odrzucony

Operatory EXISTS/ NOT EXISTS

zwracają informacje, czy podzbiór zapytania jest pusty czy nie (czy spełnia warunki podzapytania czy nie)

Wyświetla informacje o Issue, na które poświęcono więcej niż 12

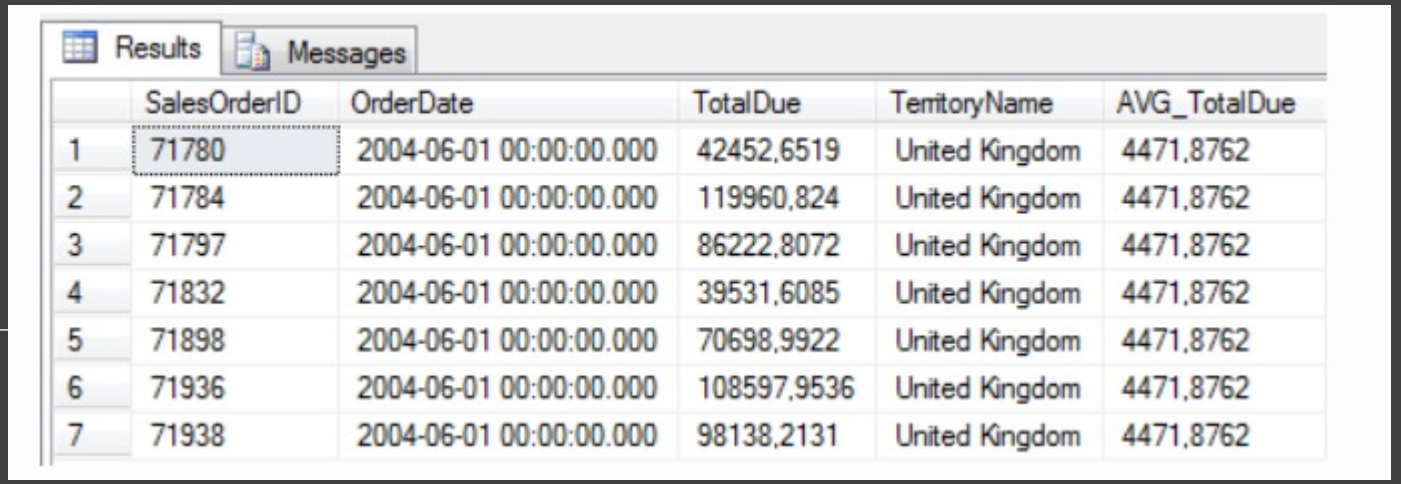
```
SELECT *
FROM sql.details NAD
WHERE EXISTS
        (
        SELECT 1
        FROM sql.details POD
        WHERE NAD.SpentHrs = POD.SpentHrs
        AND SpentHrs > 12
        )
```

Gdzie i jak występują podzapytania?

MOGĄ występować WSZĘDZIE wewnątrz zapytania, ale gdzie POWINNY zależy od tego ile wartości zwracają:

a) Wewnątrz listy pobieranych wartości (po SELECT) – zbiór jednoelementowy

b) Wewnątrz klauzuli FROM – jakikolwiek zbiór



	SalesOrderID	OrderDate	TotalDue	TerritoryName	AVG_TotalDue
1	71780	2004-06-01 00:00:00.000	42452,6519	United Kingdom	4471,8762
2	71784	2004-06-01 00:00:00.000	119960,824	United Kingdom	4471,8762
3	71797	2004-06-01 00:00:00.000	86222,8072	United Kingdom	4471,8762
4	71832	2004-06-01 00:00:00.000	39531,6085	United Kingdom	4471,8762
5	71898	2004-06-01 00:00:00.000	70698,9922	United Kingdom	4471,8762
6	71936	2004-06-01 00:00:00.000	108597,9536	United Kingdom	4471,8762
7	71938	2004-06-01 00:00:00.000	98138,2131	United Kingdom	4471,8762

```sql
SELECT SalesOrderID, OrderDate, TotalDue, st.Name AS TerritoryName,
       (
            -- podzapytanie w SELECT – średnia dla wszystkich zleceń
           SELECT AVG(TotalDue)
           FROM [Sales].[SalesOrderHeader]
       ) AS AVG_TotalDue
FROM [Sales].[SalesOrderHeader] soh
     inner join [Sales].[SalesTerritory] st ON soh.TerritoryID = st.TerritoryID
WHERE st.CountryRegionCode = 'GB' and OrderDate between '2004-06-01' and '2004-06-30'
      and  TotalDue >=
         (
           -- podzapytanie w filtracji w WHERE
           SELECT AVG(TotalDue) AS  AVG_TotalDue
           FROM [Sales].[SalesOrderHeader]
         )
```

c) Tworzenie warunków połączeń w ON, wyrażeń w WHERE oraz filtracji grup w HAVING

zależy od wybranego operatora:

operatory >, <, =, <> : zwracają zbiory jednoelementowe

operatory IN, NOT IN, ALL, ANY: zwracają zbiory wieloelementowe

Operatory ALL/ANY

operator ALL: zwraca rezultat jeśli wszystkie wartości w podzapytaniu spełniają warunek

operator ANY: zwraca rezultat jeśli którakolwiek wartość w podzapytaniu spełnia warunek

Napisz zapytanie, które wyświetli:

1. Wiedząc, że zespół testerski to: A9CDLZZ oraz ABBR9ZZ, policz ile zadań jest przypisane do testera1 (A9CDLZZ) i testera2 (ABBR9ZZ) (sql.story);

2. Wyświetl kolumny: Epic, EpicStatus, ReporterName, AssigneeName dla Epiców stworzonych od 2021 i ułóż je tak, by spełniały warunki:
   - jeśli EpicStatus jest „In Progress” to malejąco po ReporterName,
   - jeśli AssigneeName jest puste to malejąco po EpicStatusie (sql.epic);

3. Wymień EpicLinki, które podpięte są do Issue w statusie „Abandoned” lub „Blocked/Waiting/On Hold” i updatowane były w 2021 (sql.story, sql.details);

4. Wyświetl Epic, które nie posiadają żadnych StoryPointsów (sql.details, sql.epic);

5. Wyświetl wszystkie dane o Issue, na które poświęcono co najmniej 3h więcej niż minimalny czas poświęcony na Issue dla tych samych StoryPointsów (sql.details);

SQL part 6

Agenda

~~FIRST MEETING:~~

1. SQL – overview
2. Select
3. Aliases
4. Where
5. And, or, not, in

~~SECOND MEETING:~~

6. Top
7. Wildcards
8. Distinct
9. Order by
10. Null

~~THIRD MEETING:~~

11. Numeric functions: min, max, avg, count, sum
12. Len, concat, left, right
13. Upper, lower
14. Reverse, Replace, Substring

~~FOURTH MEETING:~~

15. Join
16. Group by, having

~~FIFTH MEETING:~~

17. Case when
18. Subqueries

SIXTH MEETING:

19. Datatype
20. Create, alter, drop
21. Insert, update, delete
22. Keys

DATATYPE

Kategoria	Typ danych	Opis
Liczbowe	DECIMAL (NUMBER)	Liczby zmiennoprzecinkowe
	INTEGER	Liczby całkowite

Nazwa	Zakres	Waga
bigint	-2^63 (-9,223,372,036,854,775,808) - 2^63-1 (9,223,372,036,854,775,807)	8b
int	-2^31 (-2,147,483,648) - 2^31-1 (2,147,483,647)	4b
smallint	-2^15 (-32,768) - 2^15-1 (32,767)	2b
tinyint	0 - 255	1b

Kategoria	Typ danych	Opis
Daty i czasu	DATE	data
	TIME	czas
	DATETIME2	data i czas

Nazwa	Zakres	Waga
date	01.01.0001 – 31.12.9999	3b
time	Dokładność do 100 nanosekund (10^-9)s	3 - 5b
datetime2	01.01.0001 – 31.12.9999 z dokładnością do 100 nanosekund	6 - 8b
timestamp	Unikalna wartość, która jest aktualizowana przy każdorazowej modyfikacji lub stworzeniu rekordu	

Kategoria	Typ danych	Opis
Znakowe	CHAR	ciąg znaków o stałej długości
	VARCHAR	ciąg znaków o zmiennej długości
	NCHAR, NVARCHAR	ciąg znaków o stałej lub zmiennej długości zakodowanych w UNICODE (polskie znaki)

Nazwa	Zakres	Waga
char (n)	8 000 znaków	zależna od długości n
varchar (n)		2b + n
varchar (max)	1 073 741 824 znaków	2GB
nchar (n)	4 000 znaków	n x 2
nvarchar (n)		(2b + n) x 2
nvarchar (max)	536 870 912 znaków	2GB

CREATE SCHEMA sql

```
use DRBR
Create schema sql
Create table sql.story
(IssueKey varchar(30) PRIMARY KEY
, IssueSummary varchar(max)
, ProjectKey varchar(30)
, IssueType varchar(30)
, IssueStatus varchar(50)
, ReporterName varchar(max)
, ReporterUPIN varchar(20)
, AssigneeName varchar(max)
, AssigneeUPIN varchar(20)
, CreatedAt datetimeoffset(7)
, UpdatedAt datetimeoffset(7)
, ClosedAt datetimeoffset(7)
, RunningTimestamp datetime2(7))
```

CREATE TABLE
ALTER TABLE
DROP TABLE
TRUNCATE TABLE

CREATE TABLE sql.nowe
(IssueName varchar(max))

CREATE TABLE nazwa_tabeli
(nazwa_kolumny typ_danych)

ALTER TABLE sql.nowe
ADD IssueNumber varchar(25)

ALTER TABLE nazwa_tabeli
ADD kolumna typ_danych

ALTER TABLE sql.nowe
DROP IssueNumber

ALTER TABLE nazwa_tabeli
DROP kolumna

ALTER TABLE sql.nowe
ALTER COLUMN IssueNumber tinyint

ALTER TABLE nazwa_tabeli
ALTER COLUMN kolumna typ_danych

DROP sql.nowe

DROP nazwa_tabeli

TRUNCATE sql.nowe

TRUNCATE nazwa_tabeli

INSERT INTO
UPDATE
DELETE

INSERT INTO nazwa_tabeli (kolumna1, kolumna2, kolumna3, ...)
VALUES (*wartość1, wartość2, wartość3, ...);*

INSERT INTO sql.nowe

VALUES ('ITQM-1', 'drugie');

INSERT INTO nazwa_tabeli (kolumna1, kolumna2, ...)
VALUES (wartość dla kolumny1, wartość dla kolumny2, ...)

INSERT INTO sql.story (IssueKey, AssigneeName, ReporterName)

VALUES ('ITQM-4', 'Jan Nowak', 'Tomasz Kowalski');

INSERT INTO nazwa_tabeli (kolumna1, kolumna3, kolumna7 ...)
VALUES (wartość dla kolumny1, wartość dla kolumny2, ...)

```
UPDATE sql.nowe
SET IssueNumber =  'ITQM-1000'
```

```
UPDATE nazwa_tabeli
SET kolumna = wartość
```

```
UPDATE sql.nowe
SET IssueName =  'INNA'
WHERE IssueName = 'drugie'
```

```
UPDATE nazwa_tabeli
SET kolumna = wartość
WHERE kolumna = wartość
```

DELETE FROM sql.nowe

DELETE FROM nazwa_tabeli

DELETE FROM sql.nowe
WHERE IssueNumber = 'ITQM-1000'

DELETE FROM nazwa_tabeli
WHERE kolumna = wartość

KEYS

PRIMARY KEY

- klucz podstawowy

- unikalna wartość każdego rekordu w tabeli (not null)

- tabela ma jeden klucz podstawowy (który może składać się z jednej lub kilku kolumn)

```
use DRBR
Create table sql.story
(IssueKey varchar(30) PRIMARY KEY
, IssueSummary varchar(max)
, ProjectKey varchar(30)
, IssueType varchar(30)
, IssueStatus varchar(50)
, ReporterName varchar(max)
, ReporterUPIN varchar(20)
, AssigneeName varchar(max)
, AssigneeUPIN varchar(20)
, CreatedAt datetimeoffset(7)
, UpdatedAt datetimeoffset(7)
, ClosedAt datetimeoffset(7)
, RunningTimestamp datetime2(7))
```

```
use DRBR
Create table sql.story
(IssueKey varchar(30)
, IssueSummary varchar(max)
, ProjectKey varchar(30)
, IssueType varchar(30)
, IssueStatus varchar(50)
, ReporterName varchar(max)
, ReporterUPIN varchar(20)
, AssigneeName varchar(max)
, AssigneeUPIN varchar(20)
, CreatedAt datetimeoffset(7)
, UpdatedAt datetimeoffset(7)
, ClosedAt datetimeoffset(7)
, RunningTimestamp datetime2(7)
, CONSTRAINT PK_STORY PRIMARY KEY (IssueKey, CreatedAt)
)
```

FOREIGN KEY

- klucz obcy
- to pole (lub zbiór pól) w jednej tabeli, które odwołuję się do klucza podstawowego w innej tabeli
- używany do łączenia dwóch tabel

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

KONIEC