Cvičení 7: Klauzule JOIN a SQL views

1) **Vytvořte tabulku ExchangeRate, která bude obsahovat sloupce *ExchangeRate_ID*, *Currency_1, Currency_2* a *ExchangeRate*.** První sloupec je **celočíselným primárním klíčem**, kódy měn mají stejný datový typ jako např. sloupec ***Currency*** v tabulce **Country**, a sloupec se samotnou hodnotou kurzu bude vždy **desetinné číslo se dvěma desetinnými místy.** Následně do tabulky vložte záznamy *('1', 'EUR', 'CZK', '26.10')*, *('2', 'USD', 'CZK', '22.18')* a *('3', 'CZK', 'CZK', '1.00')*.

2) **Napište dotaz, který vrátí následující sloupce:**

   - ***Order_ID*** z tabulky **Order**
   - ***Order_Date*** z tabulky **Order**
   - ***Customer_ID*** z tabulky **Order**
   - ***Customer_ID*** z tabulky **Customer**
   - ***Gender*** z tabulky **Customer**
   - ***City*** z tabulky **Customer**
   - ***Quantity*** z tabulky **Order**
   - ***Product_ID*** z tabulky **Order**

   Hlavní zdrojovou tabulkou je **Order**, ke které lze připojit tabulku **Customer** pomocí sloupce ***Customer_ID***.

3) **Napište obdobný dotaz jako v předchozí úloze, který však bude vracet pouze záznamy, ve kterých existuje vazba mezi tabulkami Order a Customer.**

4) **Napište obdobný dotaz jako v předchozí úloze, kde navíc odfiltrujte záznamy s prázdnou hodnotou ve sloupci *Gender* nebo *City*.** Z dotazu také v rámci klauzule **SELECT** odstraňte sloupce ***Customer.Customer_ID***, který už nyní bude mít vždy stejnou hodnotu jako sloupec ***Order.Customer_ID***.

5) **K danému dotazu připojte tabulku Product pomocí vazby mezi primárním klíčem této tabulky a sloupcem *Order.Product_ID*.** Zobrazeny by měl být jen takové záznamy, kde vazba mezi tabulkami skutečně existuje. V **SELECT** části dotazu poté také přidejte sloupec ***Product.Product_Category*.**

6) **Nyní dotaz upravte tak, abyste pro různé kombinace produktové kategorie a pohlaví zákazníka zjistili celkovou sumu položek na objednávkách spolu s celkovým počtem objednávek.** Data budou tedy seskupena podle kategorie produktů a zároveň pohlaví zákazníka.

7) **Dotaz z předchozího úkolu ještě upravte tak, aby byly odfiltrovány záznamy s produktovou kategorií 'Unknown' a poté také seřazeny podle pohlaví (vzestupně) a sumy položek na objednávkách (sestupně).**

8) **Abychom mohli provádět následující výpočty, změňte datový typ sloupce *Product.Unit_Price* na standardní celé číslo (tedy INT).**

   Následně napište dotaz, který propojí tabulky **Order** a **Product** (stejně jako v předchozích úlohách) a bude zobrazovat následující sloupce:

   - ***Order_ID*** z tabulky **Order**
   - ***Product_ID*** z tabulky **Order**
   - ***Quantity*** z tabulky **Order**
   - ***Unit_Price*** z tabulky **Product**
   - ***Currency*** z tabulky **Product**
   - nový sloupec ***VýšeObjednávky***, který se vypočítá jako **objednané množství** daného produktu vynásobené jeho **jednotkovou cenou**

9) **K dotazu z předchozí úlohy připojte tabulku ExchangeRate pomocí sloupce *Product.Currency* tak, aby bylo následně možné správně převést částky v EUR do CZK.** V **SELECT** části dotazu poté přidejte sloupec ***ExchangeRate*** z připojené tabulky spolu s novým sloupcem ***VýšeObjednávkyCZK***, který bude vracet hodnotu sloupce ***VýšeObjednávky*** převedou do **CZK.**

10) **Dotaz z předchozí úlohy uložte v databázi jako view s názvem V_OrderValue. Tento pohled si poté zkuste spustit pomocí klauzule SELECT.**

11) **Napište dotaz, který vrátí všechny možné kombinace zákazníků a produktů.** Poté ke **každé kombinaci** přiřaďte příslušný **počet záznamů**, který jí odpovídá v tabulce **Order** (tedy, kolikrát byl určitý produkt objednán určitým zákazníkem).

Cvičení 7: Klauzule JOIN a SQL views

1) **Vytvořte tabulku ExchangeRate dle kritérií v zadání.** Následně do tabulky vložte záznamy *('1', 'EUR', 'CZK', '26.10'), ('2', 'USD', 'CZK', '22.18')* a *('3', 'CZK', 'CZK', '1.00')*.

   Danou tabulku vytvoříme pomocí příkazu **CREATE TABLE** a definování vhodných datových typů. Hodnoty do tabulky poté vložíme příkazem **INSERT INTO**. Oba dotazy lze provést naráz, pokud je mezi nimi středník.

```
CREATE TABLE ExchangeRate (ExchangeRate_ID INT PRIMARY KEY
                          ,Currency_1 CHAR(3)
                          ,Currency_2 CHAR(3)
                          ,ExchangeRate DECIMAL(10,2)
                          )
;
INSERT INTO ExchangeRate
VALUES ('1', 'EUR', 'CZK', '26.10')
, ('2', 'USD', 'CZK', '22.18')
, ('3', 'CZK', 'CZK', '1.00')
```

2) **Napište dotaz, který vrátí sloupce *Order_ID, Order_Date, Customer_ID, Quantity*** a ***Product_ID*** z tabulky **Order,** spolu s ***Customer_ID*** a ***Gender*** z tabulky **Customer**. Hlavní zdrojovou tabulkou je **Order**, ke které lze připojit tabulku **Customer**.

   Před názvem každého sloupce musí být specifikováno, ze které tabulky vychází. Za tímto účelem mají také tabulky při „joinování" **alias** (**ord** a **cust**). Jinak by musel být před sloupci napsán celý originální název tabulky.

```
SELECT ord.Order_ID
      ,ord.Order_Date
      ,ord.Customer_ID
      ,cust.Customer_ID
      ,cust.Gender
      ,cust.City
      ,ord.Quantity
      ,ord.Product_ID
  FROM [csv].[Order] ord
  LEFT JOIN [csv].[Customer] cust
  ON ord.Customer_ID = cust.Customer_ID
```

3) **Napište obdobný dotaz jako v předchozí úloze, který však bude vracet pouze záznamy, ve kterých existuje vazba mezi tabulkami Order a Customer.**

Záznamy, pro které neexistuje vazba mezi tabulkami **Order** a **Customer** poznáme tak, že hodnota „připojovacího“ sloupce je u nich **prázdná** (příslušné ***Customer_ID*** z levé tabulky **Order** se tedy nenašlo v pravé tabulce **Customer**). Takové záznamy lze odfiltrovat klauzulí **INNER JOIN** nebo podmínkou ***WHERE cust.Customer_ID IS NOT NULL***.

```
SELECT ord.Order_ID
      ,ord.Order_Date
      ,ord.Customer_ID
      ,cust.Customer_ID
      ,cust.Gender
      ,cust.City
      ,ord.Quantity
      ,ord.Product_ID
  FROM [csv].[Order] ord
  INNER JOIN [csv].[Customer] cust
  ON ord.Customer_ID = cust.Customer_ID
```

Záznamy s prázdnou hodnotou ve sloupci ***Customer_ID*** z tabulky ***Customer*** opravdu zmizely. V tabulce však stále zůstaly záznamy, které mají prázdné hodnoty ve sloupcích ***Gender*** či ***City***. Ty totiž mají tyto sloupce prázdné už v tabulce **Customer**.

	Order_ID	Order_Date	Customer_ID	Customer_ID	Gender	City	Quantity	Product_ID
1	O1000	2020-06-05	C163	C163	M	Bratislava - Podunajské Biskupice	2	P108
2	O1001	2020-06-21	C127	C127	F	Praha - 18	1	P119
3	O1002	2020-10-05	C202	NULL	NULL	NULL	3	P105
4	O1003	2020-07-05	C133	C133	F	Praha - Satalice	1	P103
5	O1004	2021-03-22	C106	C106	M	Praha Čakovice	4	P106
6	O1005	2020-08-08	C121	C121	NULL	Plzeň - Koterov	3	P118
7	O1006	2020-12-29	C194	C194	M	NULL	2	P119
8	O1007	2020-11-14	C142	C142	M	Radíkov	3	P109
9	O1008	2020-08-20	C117	C117	NULL	Brno - Trnitá	3	P100
10	O1009	2020-03-31	C173	C173	F	Žilina-Strážov	2	P117
11	O1010	2020-12-15	C165	C165	F	Bratislava - Raca	1	P120
12	O1011	2020-10-04	C127	C127	F	Praha - 18	3	P105
13	O1012	2020-05-15	C205	NULL	NULL	NULL	3	P101

	Order_ID	Order_Date	Customer_ID	Customer_ID	Gender	City	Quantity	Product_ID
1	O1000	2020-06-05	C163	C163	M	Bratislava - Podunajské Biskupice	2	P108
2	O1001	2020-06-21	C127	C127	F	Praha - 18	1	P119
3	O1003	2020-07-05	C133	C133	F	Praha - Satalice	1	P103
4	O1004	2021-03-22	C106	C106	M	Praha Čakovice	4	P106
5	O1005	2020-08-08	C121	C121	NULL	Plzeň - Koterov	3	P118
6	O1006	2020-12-29	C194	C194	M	NULL	2	P119
7	O1007	2020-11-14	C142	C142	M	Radíkov	3	P109
8	O1008	2020-08-20	C117	C117	NULL	Brno - Trnitá	3	P100
9	O1009	2020-03-31	C173	C173	F	Žilina-Strážov	2	P117
10	O1010	2020-12-15	C165	C165	F	Bratislava - Raca	1	P120
11	O1011	2020-10-04	C127	C127	F	Praha - 18	3	P105
12	O1013	2021-03-16	C184	C184	F	NULL	1	P115
13	O1014	2020-09-28	C104	C104	M	Řimovice	5	P109

4) **Napište obdobný dotaz jako v předchozí úloze, kde navíc odfiltrujte záznamy s prázdnou hodnotou ve sloupci *Gender* nebo *City*.** Z dotazu také v rámci klauzule **SELECT** odstraňte sloupce ***Customer.Customer_ID***, který už nyní bude mít vždy stejnou hodnotu jako sloupec ***Order.Customer_ID***.

```
SELECT ord.Order_ID
      ,ord.Order_Date
      ,ord.Customer_ID
      ,cust.Gender
      ,cust.City
      ,ord.Quantity
      ,ord.Product_ID
  FROM [csv].[Order] ord
  INNER JOIN [csv].[Customer] cust
  ON ord.Customer_ID = cust.Customer_ID
  WHERE cust.Gender IS NOT NULL
  AND cust.City IS NOT NULL
```

5) **K danému dotazu připojte tabulku Product pomocí vazby mezi primárním klíčem této tabulky a sloupcem *Order.Product_ID*.** Zobrazeny by měl být jen takové záznamy, kde vazba mezi tabulkami skutečně existuje. V **SELECT** části dotazu poté také přidejte sloupec ***Product.Product_Category*.**

   Jelikož mají být znovu vráceny jen záznamy s existující vazbou mezi tabulkami **Order** a **Product**, použijeme klauzuli **INNER JOIN**.

```
SELECT ord.Order_ID
      ,ord.Order_Date
      ,ord.Customer_ID
      ,cust.Gender
      ,cust.City
      ,ord.Quantity
      ,ord.Product_ID
      ,prod.Product_Category
  FROM [csv].[Order] ord
  INNER JOIN [csv].[Customer] cust
  ON ord.Customer_ID = cust.Customer_ID
  INNER JOIN [csv].[Product] prod
  ON ord.Product_ID = prod.Product_ID
  WHERE cust.Gender IS NOT NULL
  AND cust.City IS NOT NULL
```

6) **Nyní dotaz upravte tak, abyste pro různé kombinace produktové kategorie a pohlaví zákazníka zjistili celkovou sumu položek na objednávkách spolu s celkovým počtem objednávek.** Data budou tedy seskupena podle kategorie produktů a zároveň pohlaví zákazníka.

Do **SELECT** části dotazu vypíšeme pouze potřebné sloupce, tedy produktovou kategorii, pohlaví zákazníků a výpočty sumy položek i počtu objednávek. V klauzuli **GROUP BY** musí být zahrnuty oba sloupce, na jejichž úrovni seskupování probíhá.

```
SELECT prod.Product_Category
       ,cust.Gender
       ,SUM(ord.Quantity) AS SumaPoložek
       ,COUNT(ord.Quantity) AS PočetObjednávek
FROM [csv].[Order] ord

INNER JOIN [csv].[Customer] cust
ON ord.Customer_ID = cust.Customer_ID
INNER JOIN [csv].[Product] prod
ON ord.Product_ID = prod.Product_ID

WHERE cust.Gender IS NOT NULL
AND cust.City IS NOT NULL

GROUP BY prod.Product_Category, cust.Gender
```

7) **Dotaz z předchozího úkolu ještě upravte tak, aby byly odfiltrovány záznamy s produktovou kategorií 'Unknown' a poté také seřazeny podle pohlaví (vzestupně) a sumy položek na objednávkách (sestupně).**

Jak je vidět níže, do dotazu byla přidána **WHERE** podmínka pro sloupec ***Product.Product_Category*** a klauzule **ORDER BY** s příslušnými pravidly řazení.

```
SELECT prod.Product_Category
       ,cust.Gender
       ,SUM(ord.Quantity) AS SumaPoložek
       ,COUNT(ord.Quantity) AS PočetObjednávek
FROM [csv].[Order] ord

INNER JOIN [csv].[Customer] cust
ON ord.Customer_ID = cust.Customer_ID
INNER JOIN [csv].[Product] prod
ON ord.Product_ID = prod.Product_ID

WHERE cust.Gender IS NOT NULL
AND cust.City IS NOT NULL
AND prod.Product_Category <> 'Unknown'

GROUP BY prod.Product_Category, cust.Gender
ORDER BY cust.Gender, SumaPoložek DESC
```

Z výsledné tabulky poté můžeme vyčíst, jaké produkty nejvíce (či naopak nejméně) nakupují ženy, a jaké zase muži.

	Product_Category	Gender	SumaPoložek	PočetObjednávek
1	Homeware	F	1527	553
2	Toys	F	1461	566
3	Clothes	F	1344	557
4	Furniture	F	1109	290
5	Clothes	M	1004	211
6	Homeware	M	721	223
7	Toys	M	638	205
8	Furniture	M	228	114

8) **Abychom mohli provádět následující výpočty, změňte datový typ sloupce *Product.Unit_Price* na standardní celé číslo (tedy INT).** Následně napište dotaz, který propojí tabulky **Order** a **Product** a bude zobrazovat sloupce ***Order_ID*** z tabulky **Order**, ***Product_ID*** z tabulky **Order**, ***Quantity*** z tabulky **Order**, ***Unit_Price*** z tabulky **Product**, ***Currency*** z tabulky **Product** a nový sloupec ***VýšeObjednávky***, který se vypočítá jako **objednané množství** daného produktu vynásobené jeho **jednotkovou cenou**.

Hodnotu sloupce ***VýšeObjednávky*** získáme vynásobením sloupců ***Order.Quantity*** a ***Product.Unit_Price***. Kdyby datové typy obou zůstaly nastaveny na **SMALLINT**, byl by i výsledek jejich násobení omezen maximální povolenou hodnotou tohoto data typu. To by však mohlo způsobit pád dotazu, jelikož při násobení může vyjít hodnota v řádu desetitisíců, tedy více než je maximum **SMALLINT**.

```
SELECT ord.[Order_ID]
      ,ord.[Product_ID]
      ,ord.[Quantity]
      ,prod.[Unit_Price]
      ,ord.[Quantity] * prod.[Unit_Price] AS VýšeObjednávky
      ,prod.Currency
FROM [csv].[Order] ord
INNER JOIN [csv].[Product] prod
ON ord.Product_ID = prod.Product_ID
```

Jak si lze všimnout ve výsledku dotazu, hodnoty jsou v **různých měnách** v závislosti na měně jednotkové ceny. Kdybychom chtěli zobrazit všechny výše objednávek v CZK, museli bychom do výpočtu zahrnout měnový kurz (viz další úloha).

	Order_ID	Product_ID	Quantity	Unit_Price	VýšeObjednávky	Currency
1	O1000	P108	2	1000	2000	CZK
2	O1001	P119	1	160	160	EUR
3	O1002	P105	3	1700	5100	CZK
4	O1003	P103	1	1800	1800	CZK
5	O1004	P106	4	1200	4800	CZK
6	O1005	P118	3	100	300	EUR

9) **K dotazu z předchozí úlohy připojte tabulku ExchangeRate pomocí sloupce *Product.Currency* tak, aby bylo následně možné správně převést částky v EUR do CZK.** V **SELECT** části dotazu poté přidejte sloupec ***ExchangeRate*** z připojené tabulky spolu s novým sloupcem ***VýšeObjednávkyCZK***, který bude vracet hodnotu sloupce ***VýšeObjednávky*** převedou do **CZK.**

Tabulku **ExchangeRate** je potřeba napojit pomocí vazby mezi sloupci ***Product.Currency*** a ***ExchangeRate.Currency_1***, jelikož právě v tomto sloupci je v tabulce měnových kurzů definována zahraniční měna, jejíž kurz vzhledem k CZK lze poté získat ze sloupce ***ExchangeRate***. Pokud je měna ceny produktu CZK, vezme se z tabulky **ExchangeRate** kurz, který odpovídá vazbě CZK – CZK, tedy hodnota 1,00.

Hodnota sloupce ***VýšeObjednávkyCZK*** se poté vypočítá jednoduše jako ***VýšeObjednávky*** vynásobená **měnovým kurzem**.

```
SELECT ord.[Order_ID]
      ,ord.[Product_ID]
      ,ord.[Quantity]
      ,prod.[Unit_Price]
      ,ord.[Quantity] * prod.[Unit_Price] AS VýšeObjednávky
      ,prod.[Currency]
      ,exrate.[ExchangeRate]
      ,(ord.[Quantity] * prod.[Unit_Price]) * exrate.[ExchangeRate] AS VýšeObjednávkyCZK
FROM [csv].[Order] ord
INNER JOIN [csv].[Product] prod
ON ord.Product_ID = prod.Product_ID
LEFT JOIN [dbo].[ExchangeRate] exrate
ON prod.Currency = exrate.Currency_1
```

Všimněte si, že hodnoty sloupce ***VýšeObjednávkyCZK*** jsou zobrazovány se dvěma desetinnými místy, jelikož převzaly datový typ sloupce ***ExchangeRate***, který byl využit k jejich výpočtu (a tudíž se předpokládá, že výsledek může vyjít také jako desetinné číslo).

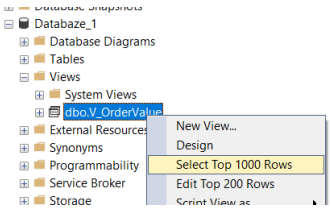
	Order_ID	Product_ID	Quantity	Unit_Price	VýšeObjednávky	Currency	ExchangeRate	VýšeObjednávkyCZK
1	O1000	P108	2	1000	2000	CZK	1.00	2000.00
2	O1001	P119	1	160	160	EUR	26.10	4176.00
3	O1002	P105	3	1700	5100	CZK	1.00	5100.00
4	O1003	P103	1	1800	1800	CZK	1.00	1800.00
5	O1004	P106	4	1200	4800	CZK	1.00	4800.00
6	O1005	P118	3	100	300	EUR	26.10	7830.00

10) Dotaz z předchozí úlohy uložte v databázi jako view s názvem V_OrderValue.

View (pohled) z jakéhokoliv dotazu vytvoříme pomocí klauzule **CREATE VIEW** *název view* **AS** a následně vložením celého dotazu, který do pohledu chcete uložit.

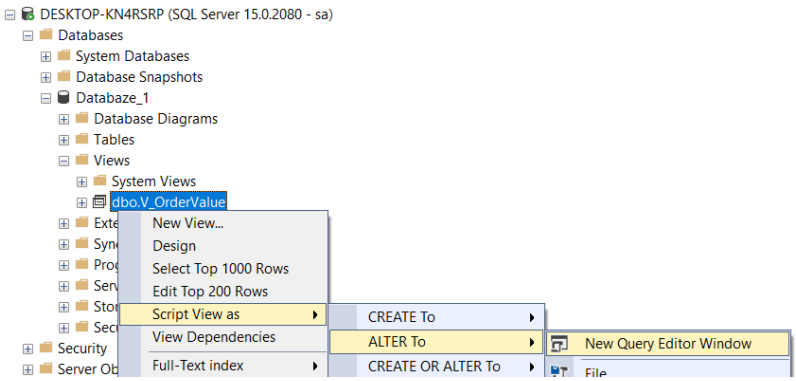
```
CREATE VIEW V_OrderValue AS

SELECT ord.[Order_ID]
      ,ord.[Product_ID]
      ,ord.[Quantity]
      ,prod.[Unit_Price]
      ,ord.[Quantity] * prod.[Unit_Price] AS VýšeObjednávky
      ,prod.[Currency]
      ,exrate.[ExchangeRate]
      ,(ord.[Quantity] * prod.[Unit_Price]) * exrate.[ExchangeRate] AS VýšeObjednávkyCZK
FROM [csv].[Order] ord
INNER JOIN [csv].[Product] prod
ON ord.Product_ID = prod.Product_ID
LEFT JOIN [dbo].[ExchangeRate] exrate
ON prod.Currency = exrate.Currency_1
```

Poté je kdykoliv možné view spustit například pomocí rychlé volby **Select Top 1000 Rows**, díky níž získáme prvních 1000 řádků výsledného result setu dotazu. Stejně tak si můžeme napsat vlastní **SELECT** dotaz, ve kterém se na data z pohledu odkážeme.



	Order_ID	Product_ID	Quantity	Unit_Price	VýšeObjednávky	Currency	ExchangeRate	VýšeObjednávkyCZK
1	O1000	P108	2	1000	2000	CZK	1.00	2000.00
2	O1001	P119	1	160	160	EUR	26.10	4176.00
3	O1002	P105	3	1700	5100	CZK	1.00	5100.00
4	O1003	P103	1	1800	1800	CZK	1.00	1800.00
5	O1004	P106	4	1200	4800	CZK	1.00	4800.00

Případné úpravy v předpisu pohledu lze provádět pomocí možnosti **Script View as – ALTER To – New Query Editor Window**, která otevře okno s daným dotazem, po jehož úpravách stačí příkaz klasicky provést stisknutím tlačítka **Execute**.



11) **Napište dotaz, který vrátí všechny možné kombinace zákazníků a produktů.** Poté ke **každé kombinaci** přiřaďte příslušný **počet záznamů**, který jí odpovídá v tabulce **Order** (tedy, kolikrát byl určitý produkt objednán určitým zákazníkem).

K nalezení všech kombinací zákazníků a produktů použijeme klauzuli **CROSS JOIN**.

```
SELECT Customer_ID, Product_ID
FROM [csv].[Customer] cust
CROSS JOIN [csv].[Product] prod
```

Poté bude potřeba připojit také tabulku **Order**, ze které máme pro všechny kombinace napočítat četnost jejich výskytu. V připojovací **ON** podmínce tedy tuto tabulku napojíme jak k tabulce **Customer**, tak i k tabulce **Product**.

Zároveň záznamy musíme **seskupit**, a to jak podle všech zákazníků z tabulky **Customer**, tak i podle všech produktů z tabulky **Product**, aby se nám žádné kombinace vytvořené klauzulí **CROSS JOIN** neztratily.

```
SELECT cust.[Customer_ID]
    ,prod.Product_ID
    ,COUNT(ord.Order_ID) AS PočetObjednání
FROM [csv].[Customer] cust
CROSS JOIN [csv].[Product] prod

LEFT JOIN [csv].[Order] ord
ON cust.Customer_ID = ord.Customer_ID
AND prod.Product_ID = ord.Product_ID

GROUP BY cust.Customer_ID, prod.Product_ID
ORDER BY cust.Customer_ID
```

	Customer_ID	Product_ID	PočetObjednání
1	C100	P102	1
2	C100	P118	3
3	C100	P115	3
4	C100	P101	3
5	C100	P114	6
6	C100	P117	3
7	C100	P113	2
8	C100	P110	2
9	C100	P105	1
10	C100	P109	4
11	C100	P104	4
12	C100	P107	1
13	C100	P119	1
14	C100	P116	3
15	C100	P120	4
16	C100	P103	6
17	C100	P100	0
18	C100	P108	4
19	C100	P112	2
20	C100	P111	4
21	C100	P106	7
22	C101	P102	5
23	C101	P106	1

Správnost výsledků vrácených dotazem lze ověřit namátkovou kontrolou záznamů v tabulce **Order**.

```
SELECT * FROM [Databaze_1].[csv].[Order]
WHERE Customer_ID = 'C100' AND Product_ID = 'P100'
```

Order_ID	Order_Date	Customer_ID	Product_ID	Quantity	Order_Status	Currency

```
SELECT * FROM [Databaze_1].[csv].[Order]
WHERE Customer_ID = 'C100' AND Product_ID = 'P111'
```

	Order_ID	Order_Date	Customer_ID	Product_ID	Quantity	Order_Status	Currency
1	O2146	2020-04-27	C100	P111	3	1	CZK
2	O3456	2020-11-25	C100	P111	3	1	CZK
3	O3802	2020-02-26	C100	P111	2	1	CZK
4	O6273	2021-02-13	C100	P111	1	3	CZK