**Přednáška**: Datové struktury

**Cvičení Function:** Kurz SQL

**Mail:** jiri.meska@silicium.cz

**Cíl cvičení**:

* Práce s analytickými funkcemi

## Struktura databáze

Dnes budeme pracovat s upraveným modelem, kde objednávku rozdělíme na objednávku a položky objednávky. Dále budeme pracovat s katalogem zboží, který je popsán v xml, využijeme později. Pracovat budete s pohledem.



Semantics of the database model:

|  |  |
| --- | --- |
| Adresa  Město  Ulice | Address  City  Street |
| Region  Kraj | Region  State |
|  |  |
| Obchodník  jmenoObchodnika  provize  vekObchodnika  idManager  idRegion  idAdresa | Salesman  name of salesman  provision  age  foreign key to boss  sale district  address |
| Zakaznik  jmenoZakaznika  bonita  vekZakaznika | Customer  name of customer  evaluation  age |
| ZboziKatalog  popisK  popisxml  cenajson | Goods, commodity  description  xml document  json document |
| ZboziKategorie  popisKategorie | Goods, category  description |
| Objednavka  datum | Order  date of order |
| ObjednavkaPolozka  mnozstviKus  poznamka | OrderItem  number  comment |

## Analytické funkce přehled

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Průvodce tématikou analytických funkcí v postgresSQL:  <http://www.postgresqltutorial.com/postgresql-window-function> | |  |  |  |

Functions such as AVG(), MIN(), MAX(), SUM(), and COUNT() can be also used as window functions.

| **Name** | **Description** |
| --- | --- |
| [CUME\_DIST](http://www.postgresqltutorial.com/postgresql-cume_dist-function/) | Return the relative rank of the current row. |
| [DENSE\_RANK](http://www.postgresqltutorial.com/postgresql-dense_rank-function/) | Rank the current row within its partition without gaps. |
| [FIRST\_VALUE](http://www.postgresqltutorial.com/postgresql-first_value-function/) | Return a value evaluated against the first row within its partition. |
| [LAG](http://www.postgresqltutorial.com/postgresql-lag-function/) | Return a value evaluated at the row that is at a specified physical offset row before the current row within the partition. |
| [LAST\_VALUE](http://www.postgresqltutorial.com/postgresql-last_value-function/) | Return a value evaluated against the last row within its partition. |
| [LEAD](http://www.postgresqltutorial.com/postgresql-lead-function/) | Return a value evaluated at the row that is offset rows after the current row within the partition. |
| [NTILE](http://www.postgresqltutorial.com/postgresql-ntile-function/) | Divide rows in a partition as equally as possible and assign each row an integer starting from 1 to the argument value. |
| [NTH\_VALUE](http://www.postgresqltutorial.com/postgresql-nth_value-function/) | Return a value evaluated against the nth row in an ordered partition. |
| [PERCENT\_RANK](http://www.postgresqltutorial.com/postgresql-percent_rank-function/) | Return the relative rank of the current row (rank-1) / (total rows – 1) |
| [RANK](http://www.postgresqltutorial.com/postgresql-rank-function/) | Rank the current row within its partition with gaps. |
| [ROW\_NUMBER](http://www.postgresqltutorial.com/postgresql-row_number/) | Number the current row within its partition starting from 1. |

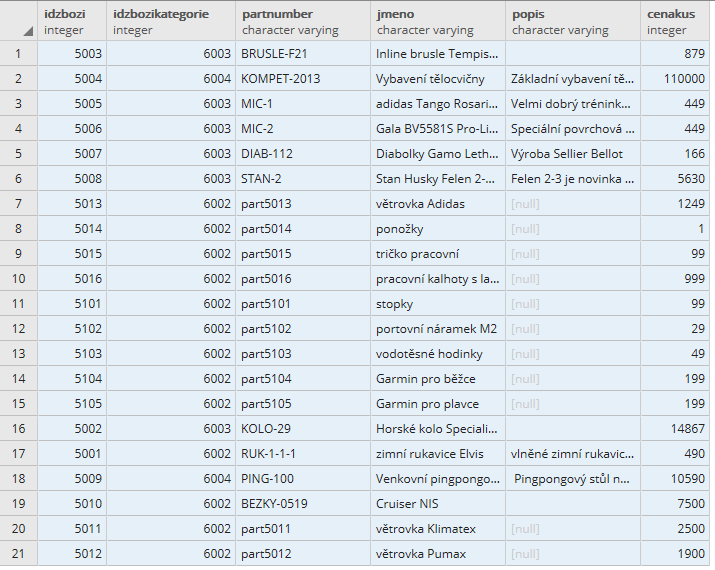
## Vytvoření nové databáze a import dat

Vytvoření databáze pro dnešní lekci.

|  |  |
| --- | --- |
| Úkol | Vytvoření databáze „AnalytickeFunkce“ viz postup v úvodním „Uvodu do cvičení“ |
|  | Podobně jako v minulém cvičení spustit script „dbscript.sql“, který tentokrát vytvoří pomocí DDL cvičnou databázi a naplní ji cvičnými daty pro účely tohoto cvičení. |

## Pohled Zbozi

Pro toto cvičení předpokládáme pohled Zbozi, který vytvořte podle návodu dále. Zatím nezkoumejte, jak pracujeme s XML, to se naučíme později.



|  |  |
| --- | --- |
| SQL Query | create view Zbozi as  select  idZboziKatalog as idZbozi,  cast(cast((xpath('/zbozi/@idZboziKategorie', popisxml))[1] as varchar) as integer) as idZboziKategorie,  cast((xpath('/zbozi/part\_number/text()', popisxml))[1] as varchar) as partNumber,  cast((xpath('/zbozi/jmeno/text()', popisxml))[1] as varchar) as jmeno,  cast((xpath('/zbozi/popis/text()', popisxml))[1] as varchar) as popis,  cast(cast((xpath('/zbozi/cena/text()', popisxml))[1] as varchar) as integer) as cenaKus  from zboziKatalog; |
| Result |  |

|  |  |
| --- | --- |
| SQL Query | select \* from Zbozi; |
| Result |  |

## Analytické funkce

Analytické funkce umožňují ve výstupu kombinovat vlastnosti řádků a skupiny řádků. Již umíme pracovat se skupinami řádků pomocí GROUP BY. Lze ale vytvářet další rozdělení řádků do skupin pomocí klausule OVER (PARTITION by …).

Partition lze deklarovat jako WIŇDOW a odkazovat na ně. Následující zápisy jsou ekvivalentní:

|  |  |
| --- | --- |
| Select  partNumber,  jmeno,  popisKategorie,  count(cenaKus) OVER (PARTITION by idZboziKategorie) as pocet  From zbozi join zboziKategorie using(idZboziKategorie)  order by idZboziKategorie, cenaKus; | Select  partNumber,  jmeno,  popisKategorie,  count(cenaKus) OVER w as pocet  From zbozi join zboziKategorie using(idZboziKategorie)  WINDOW w AS (PARTITION by idZboziKategorie)  order by idZboziKategorie, cenaKus; |

1. Zobrazte pohled zboží a v každém řádku zobrazte průměrnou cenu v rámci kategorie zboží.

Zobrazte ve výsledku:

Jmeno

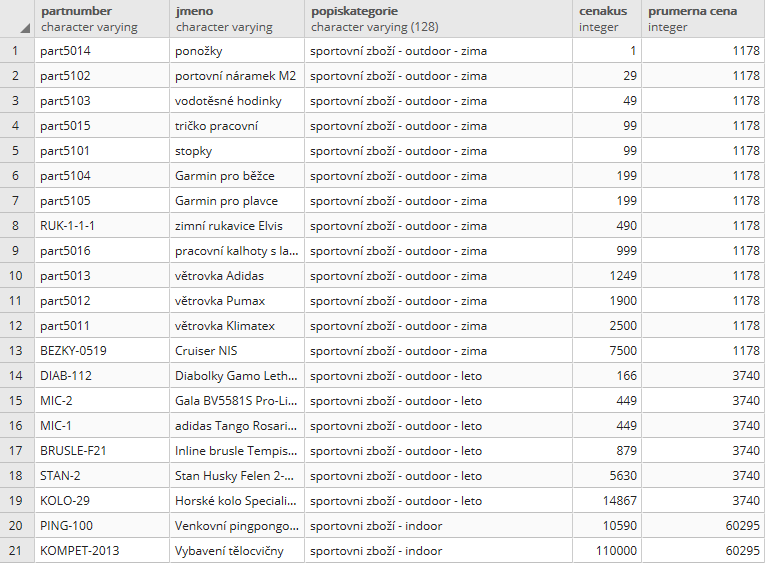
PartNumber

Popis kategorie

Cenu

Průměrnou cenu v rámci kategorie

Setřiďte podle idZboziKategorie, cenaKus



|  |  |
| --- | --- |
| SQL Query |  |
| Result |  |

Porovnejte výsledký následujících dvou dotazů, které se liší pouze v třídění PARTTION:

|  |
| --- |
| Select  partNumber,  jmeno,  popisKategorie,  cenaKus,  cast ((AVG(cenaKus) OVER w) as integer) as "prumerna cena",  count(cenaKus) OVER w as pocet,  min(cenaKus) OVER w as min,  max(cenaKus) OVER w as max  From zbozi join zboziKategorie using(idZboziKategorie)  WINDOW w AS (PARTITION by idZboziKategorie)  order by idZboziKategorie, cenaKus; |
|  |

|  |
| --- |
| Select  partNumber,  jmeno,  popisKategorie,  cenaKus,  cast ((AVG(cenaKus) OVER w) as integer) as "prumerna cena",  count(cenaKus) OVER w as pocet,  min(cenaKus) OVER w as min,  max(cenaKus) OVER w as max  From zbozi join zboziKategorie using(idZboziKategorie)  WINDOW w AS (PARTITION by idZboziKategorie order by cenaKus)  order by idZboziKategorie, cenaKus; |
|  |

1. Setřiďte členění do kategorie z posledního příkladu podle ceny zboží. Modifikujte přidáním dalšího sloupce, který zobrazí pořadí zboží v kategorii, tj. podle ceny.

Zobrazte ve výsledku:

Jmeno

PartNumber

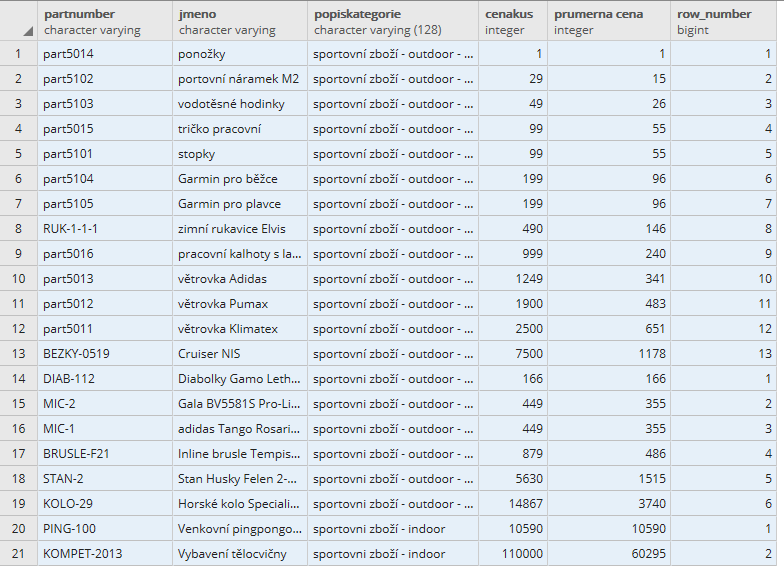
Popis kategorie

Cenu

Průměrnou cenu v rámci kategorie

Číslo řádku v rámci kategorie – podle ceny (row\_number())

Setřiďte podle idZboziKategorie, cenaKus



|  |  |
| --- | --- |
| SQL Query |  |
| Result |  |

1. Modifikace posledního příkladu, místo row\_number() použijte rank().

Zobrazte ve výsledku:

Jmeno

PartNumber

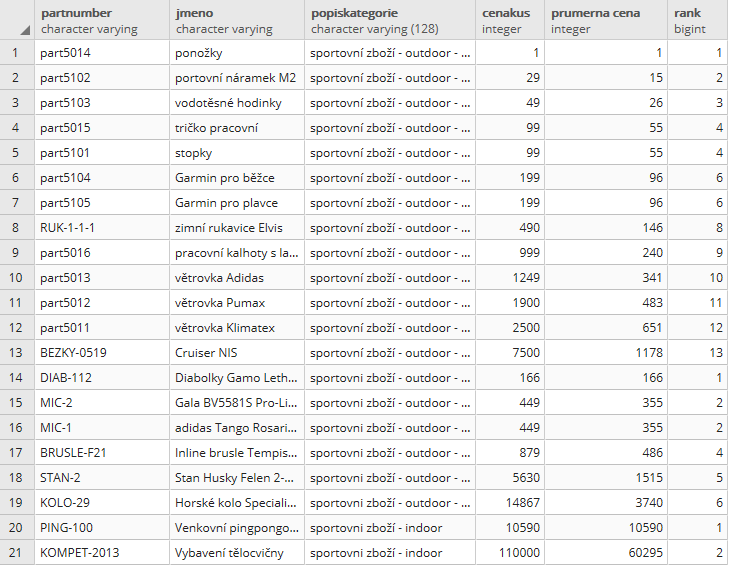
Popis kategorie

Cenu

Průměrnou cenu v rámci kategorie

Pořadí v rámci kategorie – podle ceny (rank())

Setřiďte podle idZboziKategorie, cenaKus



Jaky je rozdíl mezi posledními dvěma výstupy?

Srovnejte výsledky b),c).

|  |  |
| --- | --- |
| SQL Query |  |
| Result |  |

1. Vraťme se k seznamu zboží tříděného podle ceny pro potřeby seskupení podle kategorie. Zobrazíme cenu zboží a cenu zboží nejblíže nižší a cenu zboží nejblíže vyšší (odpovídá třídění) k platnému řádku

Zobrazte ve výsledku:

Popis

Popis kategorie

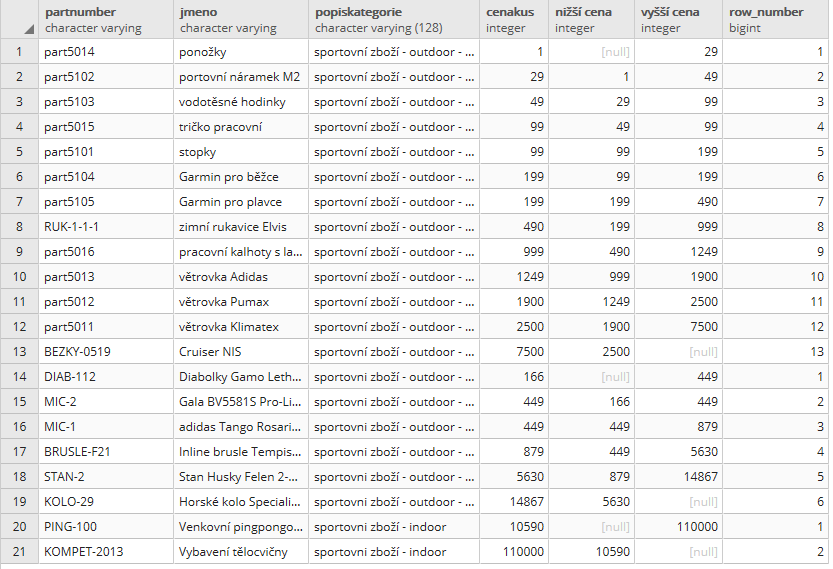
Cenu

Cena nejblíže nižší v rámci kategorie (v rámci kategorie třídit dle ceny)

Cena nejblíže vyšší v rámci kategorie (v rámci kategorie třídit dle ceny)

Číslo řádku v rámci kategorie – podle ceny (row\_number())

Setřiďte podle idZboziKategorie, cenaKus



|  |  |
| --- | --- |
| SQL Query |  |
| Result |  |