Груписање података

Честа је ситуација да резултати неког упита треба да прикажу сумиране податке.
У SQL сереврy постоји низ уграђених функција за агрегацију, тј сумирање података из више редова.

Агрегатне функције

Најчешће се користе следеће агрегатне функције:

- SUM има аргумент који може бити: константа, назив, колоне или било која комбинација аритметичког стринга и бит оператора; она враћа збир свих вредности или само DISTINCT вредности; може да се користи само са нумеричким колонама док се NULL ознаке игноришу
- MIN, MAX има аргумент који може бити: константа, назив, колоне или било која комбинација аритметичког стринга и бит оператора; враћа минималну (максималну) врендост и може се користити са колонама типа decimal, char, varshar, datetime; враћа најмањи (највећи) број, најранији (најкаснији) датум или први (последњи) стринг
- AVG враћа просечну вредност за све не-NULL нумеричке вреднности у колони
- COUNT, COUNT_BIG раде исто осим што прва враћа резултат типа int а друга тип bigint; обе могу да користе као аргумент * и тада се укључују и NULL вредности; ако је аргумент име колоне, обе враћају број редова у којима су врендости различите од NULL

Пример: Написати упит који показује рад са агрегатним функцијама

```
SELECT  MIN(Kolicina) AS min_kol,
        AVG(KomadCena) AS pros_cena,
        MAX(Porez) AS max_porez
FROM dbo.ProizvodMagacin;
```

Пример: Када се користе агрегатне функције у SELECT листи, све наведене колоне у SELECT листи морају да се користе као улази за агрегатне функције или да се референциране у GROUP BY клаузули; иначе јавља се грешка:

```
SELECT  NarudzbenicaID,
        SUM(Kolicina * KomadCena) AS suma
FROM dbo.ProizvodMagacin;
```

Овакав упит враћа грешку, јер се у SELECT листи налази колона NarudzbenicaID, која није аргумент за агрегатну функцију, нити је део GROUP BY клаузуле.

Пример: један од начина да се избегне грешка јесте уклањање колоне NarudybenicaID из SELECT листе:

```
SELECT SUM(Kolicina * KomadCena) AS suma
FROM dbo.ProizvodMagacin;
```

Упит враћа укупну продајну вредност свих наруџбеница која је добијена као сума производа количине и цене.

Пример: Као део позива агрегатне функције, може се користити и DISTINCT и тада пре израчунавања сумарних врендости уклања дупле вредности из улазне колоне. У примеру се приказују муштерије, који су направили наруџбенице, груписани по години.

```
SELECT YEAR(NarudzbenicaDatum) AS godina,
       COUNT(MusterijaID) AS svi_kupci,
       COUNT(DISTINCT MusterijaID) AS jedinstveni_kupci
FROM dbo.Narudzbenice
GROUP BY YEAR(NarudzbenicaDatum)
ORDER BY godina;
```

После извршења упита, за сваку приказану годину разликује се укупан број купаца и број јединствених купаца:

godina	svi_kupci	jedinstveni_kupci
2013	19450	625
2014	21199	640
2015	23292	657
2016	9601	663

Пример: Употреба агрегатних функција са карактер типовима података

```
SELECT MIN(MusterijaIme) AS prva_musterija,
       MAX(MusterijaIme) AS poslednja_misterija
FROM dbo.Musterije;
```

Резултат упита приказује муштерију чије је име прво као и муштерију чије је име последње по абецеди:

prvi_kupac	poslednji_kupac
Avram	Zoran

Пример: Приказати употребу агрегатних функција са датумским типовима податак

```
SELECT MIN(NarudzbenicaDatum) AS prva_narudzbenica,
       MAX(NarudzbenicaDatum) AS poslednja_narudzbenica
FROM dbo.Narudzbenice;
```

Резултат извршавања упита приказује датум креирања најстарије, прве наруџбенице, као и датум последње креиране наруџбенице.

Пример: употреба COUNT функције. COUNT(*) пребројава све редове у табели Gradovi, док је резултат рада функције COUNT(BrojStanovnika) број редова који имају уписан број становника у колону BrojStanovnika,тј број редова чија врендост није NULL.

```
SELECT COUNT(*) AS ukpan_broj_redova,
       COUNT(BrojStanovnika) AS broj_redova_sa_stanovnicima
FROM dbo.Gradovi;
```

Пример: За илустрацију понашања NULL ознаке код коришћења агрегатних функција, прво ће бити креирана привремена табела, а затим ће у табелу бити унето неколико редова.

```
CREATE TABLE #t1
(
	kolona1 int IDENTITY NOT NULL PRIMARY KEY,
	kolona2 int NULL
);
```

После тога су унете неке вредности у колону 2.
Са упитом SELECT * FROM #t1; се добије:

kolona1	kolona2
1	NULL
2	10
3	20
4	30
5	40
6	50

Над подацима је извршен упит:

```
SELECT
	SUM(kolona2)			AS suma_vrednosti_u_koloni2,
	COUNT(*)			AS broj_svih_redova,
	COUNT(kolona2)			AS broj_redova_sa_popunjenom_kolonom2,
	AVG(kolona2)			AS prosecna_vrednost_1,
	(SUM(kolona2) / COUNT(*))	AS prosecna_vrednost_2,
	AVG(COALESCE(kolona2, 0))	AS prosecna_vrednost_3
FROM #t1;
```

Резултат упита:

suma_kol2	br_svih_red	br_red_kol2	prosek_1	prosek_2	prosek_3
150	6	5	30	25	25

Види се да је сума свих вредности у колони 2, које нису NULL, 150.
Од 6 редова, 5 нису NULL.
Просек 1 даје средњу вредност редова који нису NULL.
Просек 2 даје средњу вредност свих редова без обзира да ли имају NULL.
Да би се сумирали сви редови без обзира да ли имају NULL, NULL ознаке треба заменити са другим вредностима, које ће користити агрегатна функција.
За промену NULL вредности пре агрегације, може се користити COALESCE функција.
У примеру COALESCE функција замењује NULL са 0 и тако се добија тачна вредност.

GROUP BY и HAVING

Агрегатне функције често се користе за анализу података организованих у групе.
GROUP BY клаузула користи се са SELECT наредбом да креира групе редова према јединственој комбинацији вредности специфицираних колона или израза.

```
SELECT select_lista
FROM tabela_ili_pogled
WHERE uslov_za_filtriranje
GROUP_BY kolone_ili_izrazi
HAVING uslov_za_filtriranje_grupa;
```

GROUP BY клаузула се извршава пре SELECT и замењује резултате FROM и WHERE клаузула сопственим резултатима у облику група.
Коначан резултат упита враћа само један ред по групи.
Зато, све операције које се обављају после GRUOP BY се извршавају над групама, а не над редовима.
Понекад је неопходно после GROUP BY филтрирати резултате са HAVING клаузулом.

Пример: упит за сваког продавца (колона ProdavacID) креира групу и за сваку групу рачуна број креираних наруџбеница

```
SELECT ProdavacID AS prodavac,
       COUNT(*) AS broj
FROM dbo.Narudzbenice
GROUP BY ProdavacID
ORDER BY broj DESC;
```

На врху резултујућег скупа су, због коришћења ORDER BY клаузуле са DESC опцијом, продавци са највише креираних наруџбеница.
Укупан број резултујућих редова одговара броју продаваца у табели

prodavac	broj
16	7532
2	7474
13	7400
20	7387

Пример: Ако је потребно да се види за колико је купаца продавац 16 креирао наруџбеница по годинама, може се креирати следећи упит

```
SELECT MusterijaID AS musterija,
          YEAR(NarudzbenicaDatum) AS godina,
          COUNT(*) AS broj
FROM dbo.Narudzbenice
WHERE ProdavacID = 16
GROUP BY MusterijaID, YEAR(NarudzbenicaDatum)
ORDER BY musterija, godina;
```

Филтрирање редова у табели dbo.Narudzbenice је омогућено преко клаузуле WHERE.
Пошто се GROUP BY извршава логички после WHERE, групе се креирају само од редова табеле у којима је задовољен услов да је `ProdavacID = 16` .
Групе ће бити креиране за сваког муштерију и за сваку годину, у којој је мушетрија обавио куповину.
Резултат приказује да је муштерија 1 обавио 6 куповина 2013. године, 3 куповине 2014.године и једну куповину 2015.године...

musterija	godina	broj
1	2013	6
1	2014	3
1	2015	1
2	2013	7
150	2016	5

Пример: Упит враћа списак купаца са којима је радио продавац 16, а који су имали више од 10 куповина годишње. Филтрирање група се обавља помоћу HAVING клаузуле и услова COUNT(*) > 10

```
SELECT MusterijaID AS musterija,
           YEAR(NarudzbenicaDatum) AS godina,
           COUNT(*) AS broj
FROM dbo.Narudzbenice
WHERE ProdavacID = 16
GROUP BY MusterijaID, YEAR(NarudzbenicaDatum)
HAVING COUNT(*) > 10
ORDER BY musterija, godina;
```

musterija	godina	broj
55	2015	12
107	2013	11

<u>Задаци за самосталан рад</u>

1. Написати SELECT упит који ће вратити број наруџбеница (NarudybenicaID), број ставки по свакој наруџбеници као и укупан износ за сваку наруџбеницу. Резултат сортирати у опадајућем редоследу у односу на износ наруџбеница.

2. Филтрирати резултате претходног упита тако да се прикажу наруџбенице које имају само једну ставку.

3. Написати упит који ће вратити називе градова у којима се налази више од једног купца.

4. Написати упит који ће показати све купце и укупан износ куповине коју су обавили. На почетку резултујећ скупа стваити оне купце који су обавили куповине са највећим укупним износом.

5. НАписати упит који ће приказати укупне износе продаје по годинама

6. Написати упит који ће вратити 3 најбоља продавца. Најбољи продавац има највећи износ продаје. Резултујући скуп података треба да садржи: шифру продавца, његово пуно име и укупан износ продаје.

7. Упит враћа 5 најбољих продаваца у 2016.години.

8. Написати упит који ће приказати износе куповине по категоријама купаца. Резултујући скуп података би требало да садржи шифру категорије купаца и назив категорије купаца. Такође, резултат извршавања упита би требало да за сваку категорију купца садржи: број ставки, укупан износ куповине, вредност просечне куповине.