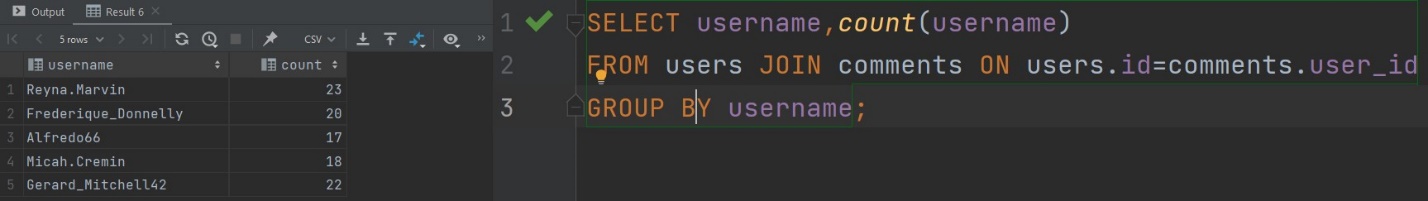
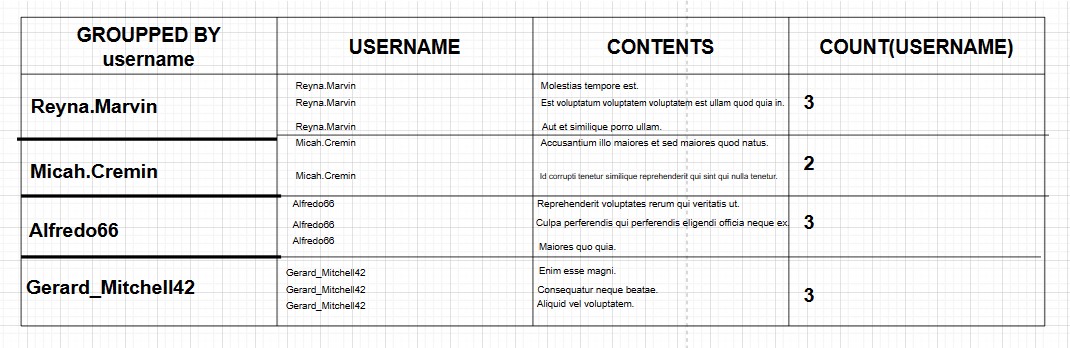
Əvvəlki dərsliklərdə olan mövzularda biz query yazdığımız an **resultset** də həmən yazmış olduğumuz query-ə əsasən bir və ya daha çox sətirdə data qayıdırdı. Bu dərslikdən isə etibarən, artıq biz elə bir query yazacayıq ki, həmən yazmış olduğumuz query bir öncəkilərdə olduğu kimi table daxilində çoxlu sətirlər üzərində yazmış olduğumuz query-ə əsasən əməliyyat aparacaq,lakin **resultset**-ə cəmi bir sətir data qaytaracaqdır, bax bunu həyata keçirdən bizdə olacaqdır **SQL** də **Grouping** və **Aggregates**-lər mövzusu.

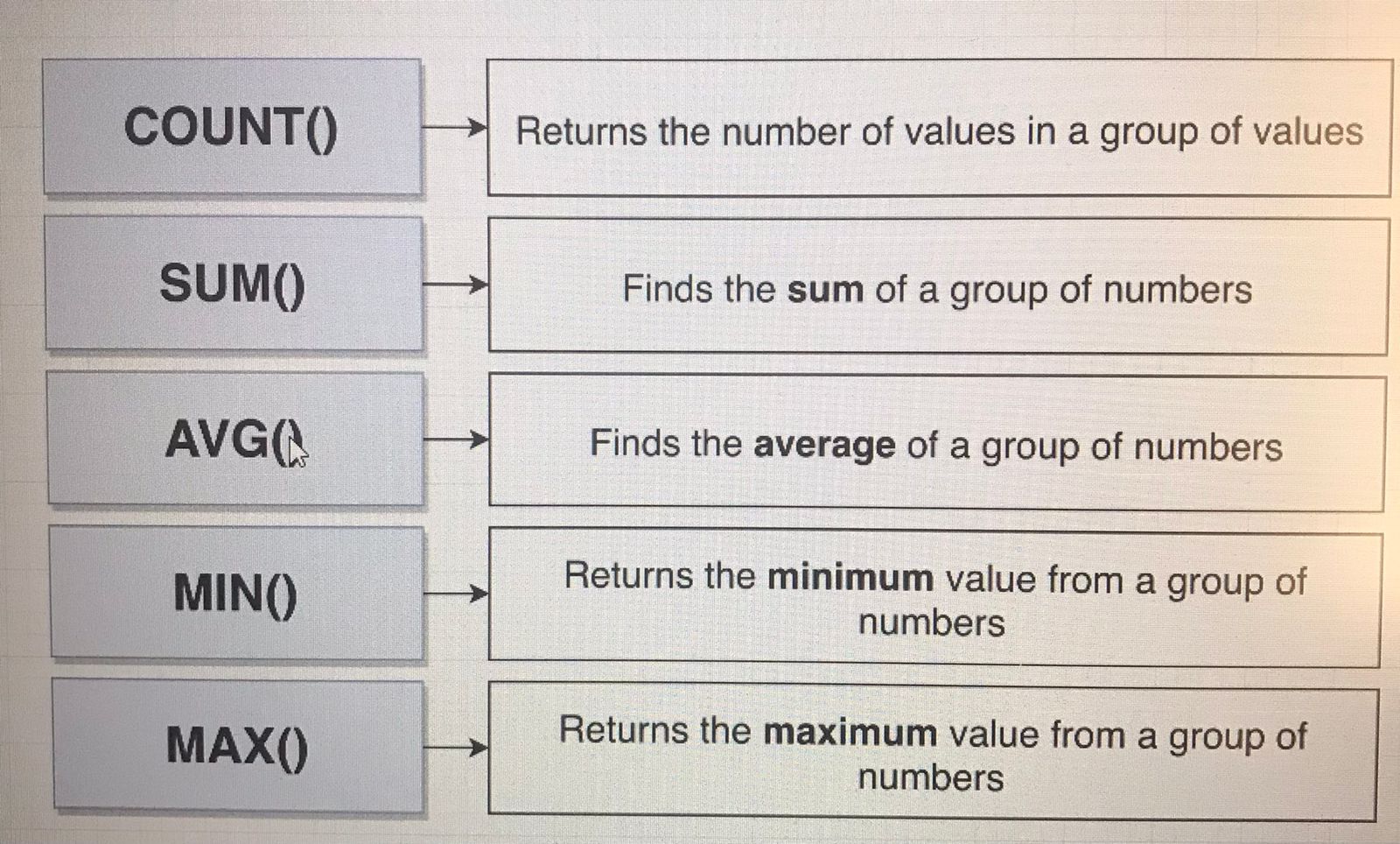
1. İndi isə gəlin, ilk olaraq, **GROUP BY**-ın istifadə qaydasına baxaq və onun nə olduğunu anlamağa çalışaq. Bunu anlamaq üçün ən yaxşı üsul bundan ibarətdir ki, simulyasiya etmək. Məsələn, gəlin ən çox **comment** yazan **users**-ların siyahısını görək.



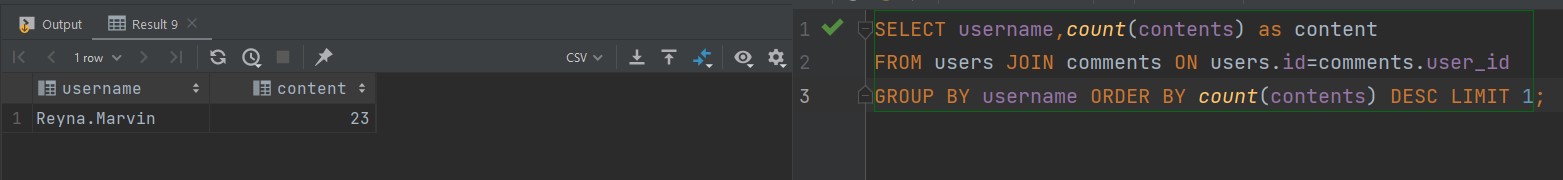
yuxarıdakı şəkildə olan query-də sintaksiz ilk baxışdan sizə çətin gələcəkdir, bu normaldır, lakin bunu anlamağ üçün gəlin aşağıdakı simulyasiya formasına baxaraq görək **SQL** də **GROUP BY** necə işləyir.



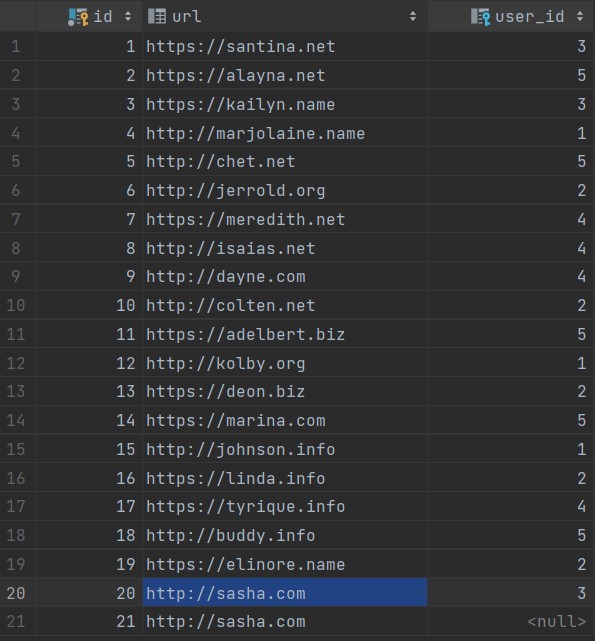
1. İlk öncə gördüyünüz kimi yuxarıdakı şəkildə olan query də **SQL** də GROUPPED BY **username** adında bir sütun yaranır və bir qrup sayılır və bu qrup özündə topluq olaraq bu qrupa uyğun gələn **sətirləri** saxlamaqdadır. Yəni users və comments table-larının birləşdirilməsindən alınan **resultset** də **GROUP BY** operatoruna vermiş olduğumuz column-a görə qruplaşdıracaqdır və bu username-ə uyğun gələn sətirlər uyğun olduğu qruplara daxil olunub. Yadınızda saxlayın nə zamanki **GROUP BY** operatorun **SQL** də istifadə edirsinizsə bilinki **resultset** table-ında **GROUP BY** operatorunda işlətdiyiniz **column** adına görə əlavə sütun yaranacaqdır.
2. Daha sonra fikir versəniz **username**-lərdən təkrarlandığı halda hər qrup üzərə bir **uniqueness** username götürülümdüşdür, buda normaldır, çünki eyni **username**-ə xas təkrar qrup ola bilməz.
3. Daha sonra unutmayın ki, **SELECT** hissəsində siz, **GROUP BY** operatorunda istifadə etdiyiniz columndan başqa onu istifadə edə bilməzsiniz, etsəniz xəta ilə qarşılaşacaqsınız, yox əgər etmək istəyirsiniz isə bu zama həmən sütun ya **GROUP BY** operatorunda istifadə olunmalıdır, yadakı **aggregate** funksiya daxilində olmalıdır.
4. İndi isə gəlin **SQL** da **aggregate** funksiyaların nə olduğunu öyrənməyə çalışaq. Aşağıdakı şəkildə **SQL** da hansı **aggregate** funksiyaları vardır onların bir qismi və açıqlaması göstərilmişdir.



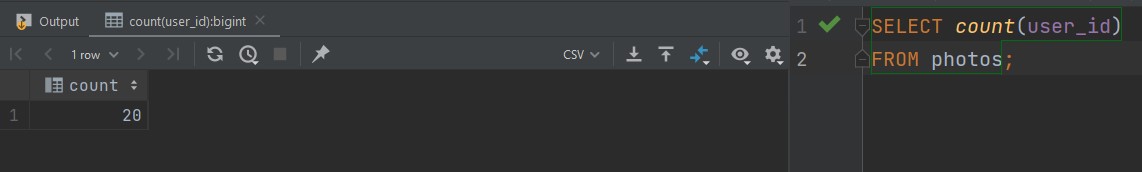
bunlar bizdə adlanır **aggregate** funksiyalar və bunların hər birinin tək bir məqsədi vardır və odaki budur. Bütün sətirlərdən geriyə yalnızca bir **result** qaytarmaqdadır. Elə isə gəlin query yazaq və daha yaxşı anlamağa başlıyaq. Məsələn gəlin, elə bir query yazaq ki, ən çox **comment** yazan **user** kimdir onu göstərək.

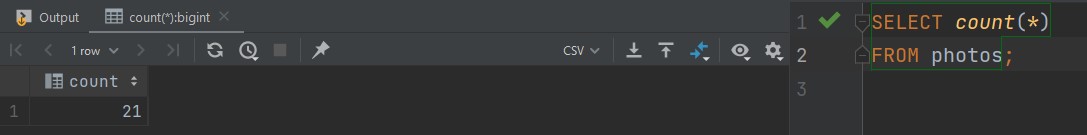


gördüyünüz kimi yuxarıdakı query də biz ən çox comment yazan **user**-ı və nə qədər comment sayısı var onu göstərmiş olduq. İndi isə gəlin bunu anlamağa çalışaq.

1. Yuxarıdakı query də istifadə olunan **count()** aggregate funksiyası geriyə rəqəm qaytarmaqdadır və bu funksiyanın içərisinə biz **comments** table-ında olan **contents** sütununu vermişik ki, **comment**-lərin sayı nə qədərdir onu görək deyə.
2. Daha sonra **GROUP BY** operatoruna **username** sütunun əlavə etmişik bu isə özlüyündə o deməkdir ki, **username**-ə görə biz qruplaşdırma əməliyyatın aparırıq və hər qrupda ayrı-ayrı **username**-ə xas olan user-ların **comment**-ləri yığılacaqdır.
3. Ən sonda isə onların sayını **count()** funksiyasının köməyi ilə sayıb **LIMIT** operatorundan istifadə edərək bizə **TOP** 1 kimdir onu versin istədik.
4. Daha sonra yadınızda saxlayın ki, əgər bir sütun özündə **NULL** dəyərini saxlayırsa və həmən sütuna görə **aggregate** funksiyalarından biri sayılan **count()** istifadə olunacaqdırsa, həmən **NULL** dəyərə sahib olan sütun **count** olunmur. Məsələn, aşağıdakı şəkildə olan **photos** table-ında 21 sətir məlumat vardır, lakin 21-ci sətirdə **user\_id** sütunundakı dəyər **NULL** dır.

və bu table üzərindən biz **aggregate** funksiyasını **user\_id** sütununa görə istifadə etmiş olsaq,aşağıdakı kimi **photos** table-ında 20 sətir məlumatın

olduğunu görmüş olarıq.

ancaq 21-ci sətirdə **user\_id** sütununun **NULL** dəyər saxlamasına baxmıyaraq, nəticə etibarı ilə digər sütunlarda dəyər vardır və bu isə o deməkdir ki, biz bu məlumatı itirmiş oluruq, bu problemi, yəni **photos** table-ında ümumilikdə neçə sətir məlumatın olduğunu öyrənmək istəyirik isə bu zaman **count()** funksiyasına **column\_name** yazmaq əvəzinə **best practise** olaraq **\*** yazmağımız kifayətdir.