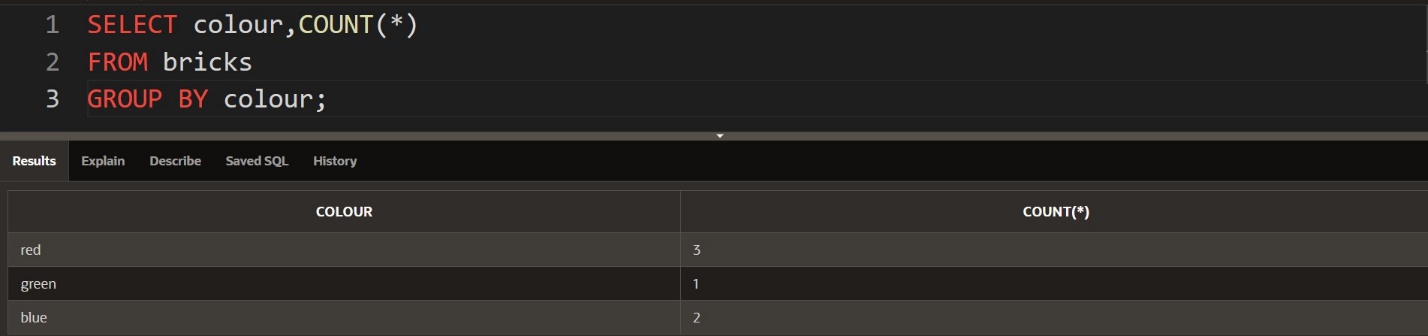
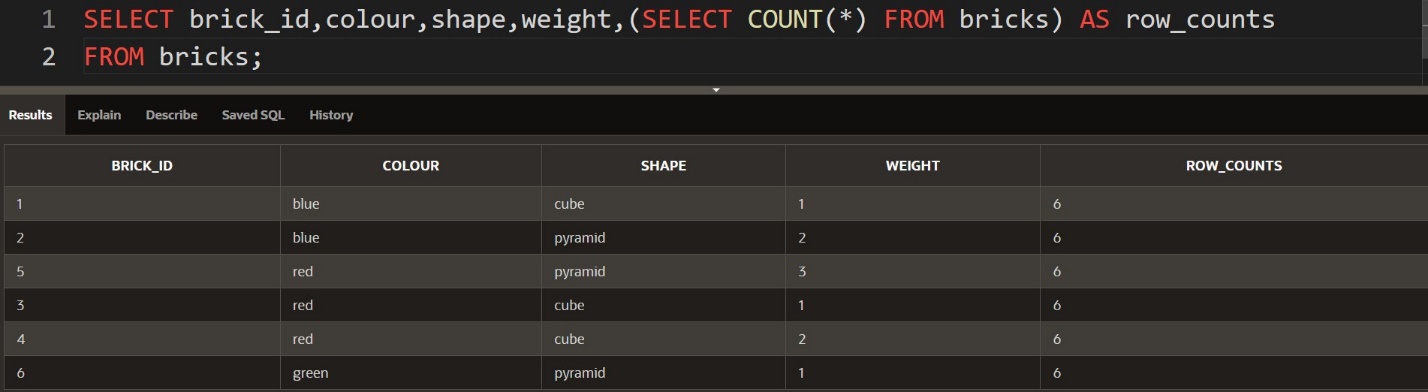
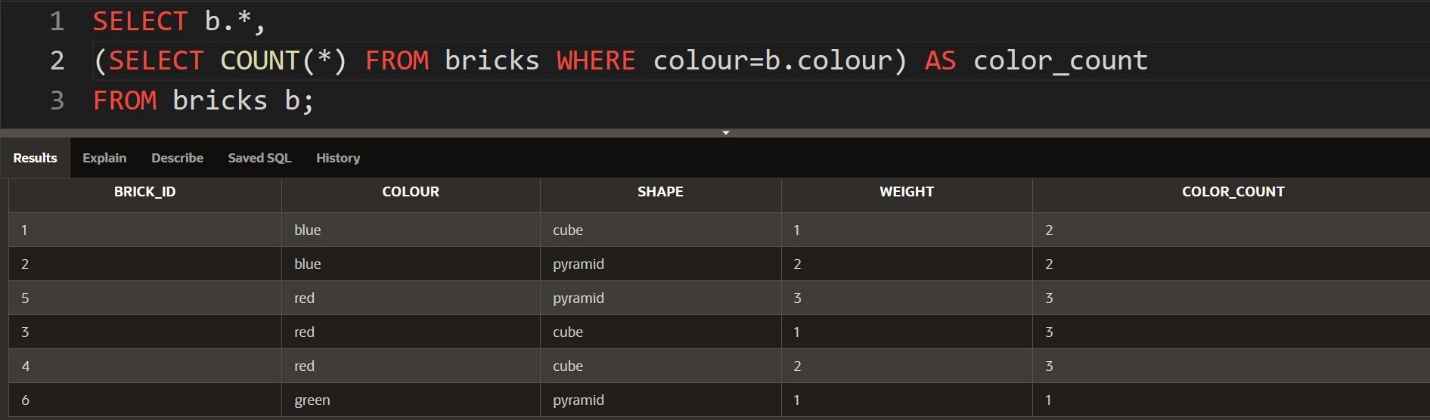
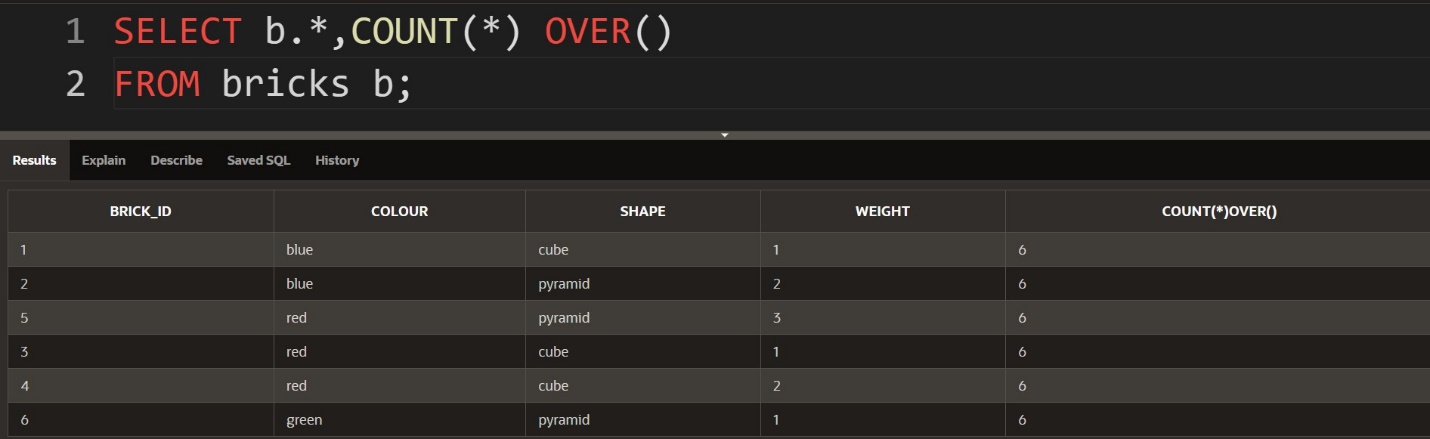
1. Bu mövzudan etibarən biz **SQL** də **analytic funksiyaları** öyrənmiş olacayıq.
2. **Analytic funksiyalara** keçid etməmişdən öncə bir öncəki mövzuda keçdiyimiz **GROUP BY** ilə bir nümunəyə baxaq, eynən aşağıdakı şəkildə olduğu kimi.

yuxarıdakı query də biz **bricks** cədvəlindən rənglərə görə qruplaşdıraraq hər rəngdən neçə ədəd var **bricks** cədvəlində onu tapmış olduq.

1. İndi isə mən elə birşey etmək istəyirəm ki, həm **bricks** cədvəlində olan bütün sütunları və həmçinində oraya əlavə sütun artıraraq orada **bricks** cədvəlində neçə sətir var məhz həmən sütuna həmən dəyəri vermək istəyirəm, eynən aşağıdakı şəkildə olduğu kimi.
2. Bəs yaxşı belə birşey etmək istəsəm o zaman necə? Məsələn istəyirəmki yenədə **bricks** cədvəlində olan bütün sütunlar çıxsın və həmçinin onlar ilə birlikdə əlavə artırdığım sütunda hər rəngdən neçə ədəd var onunda dəyərini göstərmiş olum.



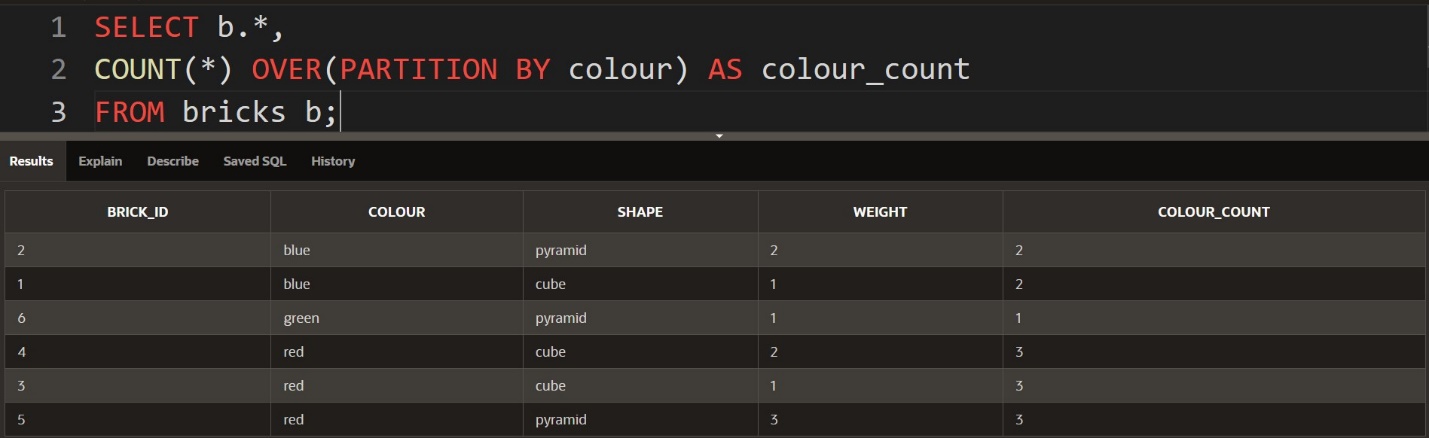
gördüyünüz kimi bu problemidə **correlated subquery** vasitəsi ilə həll etmiş olduq.

1. İndi isə gəlin yuxarıda yazdığımız query-lərin ən rahat və əlverişli üsulu olan **OVER()** clause-ı vasitəsi ilə həyata keçirtmiş olaq.
2. Məsələn gəlin, yuxarıdakı 2-ci şəkildə göstərmiş olduğum query və oradan əldə etdiyimiz nəticəni **OVER()** vasitəsi ilə həyata keçirdək.

gördüyünüz kimi yuxarıdakı şəkildə biz bir öncə etdiyimiz əməliyyatın birə-bir eynisini lakin budəfə çox rahat bir şəkildə həll yolunu yazmış olduq. Elə isə gəlin **OVER()** nədir onu anlamağa çalışaq.

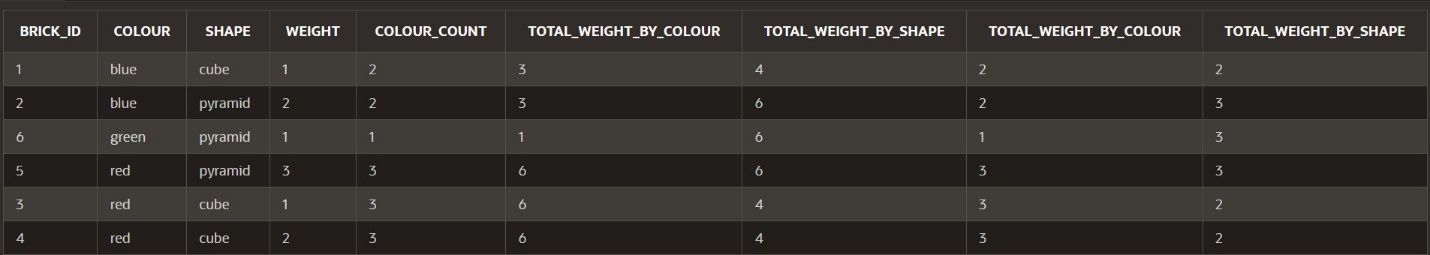
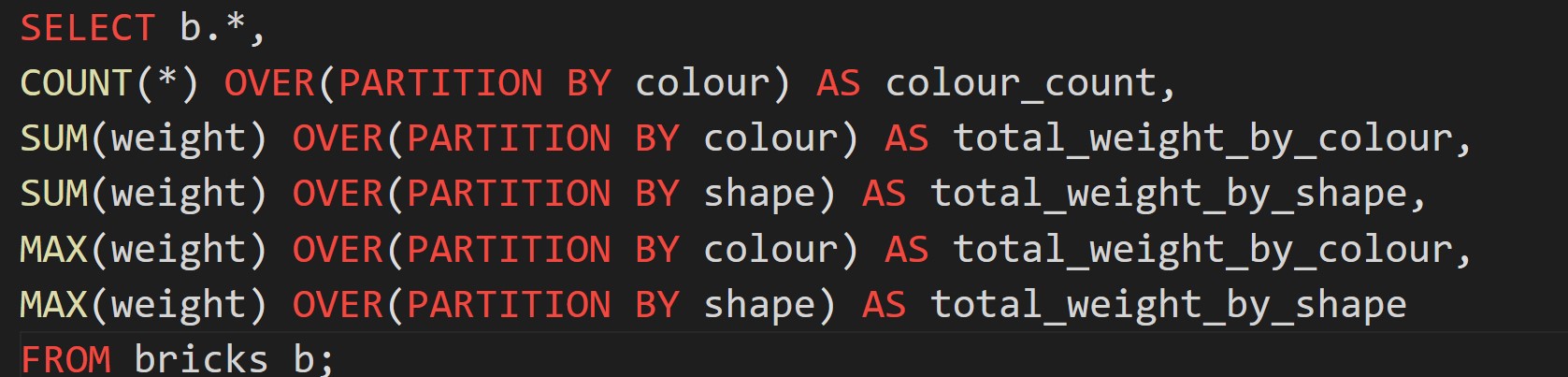
**OVER()**

1. **SQL** da **OVER()** bir **analytic funksiyadır** və bunlara həmçinin irəlidə ətraflı şəkildə öyrənəcəyimiz **windows funksiyalarıda** deyilməkdədir. Əgər query də **OVER()** funksiyasından istifadə edilibsə məhz bu funksiya ondan öncə gələn **aggregate,value** və **ranking funksiyalarını** windows funksiyalarına çevirir. Burada fikir verdinizsə tez-tez **windows funksiyaları** sözünü işlətməkdəyəm. Bu termin o mənanı kəsb edir ki, **windows** demək bir növ cədvəldə olan hər bir sətiri bir növ qrup halına salaraq həmən sətir üzərində ayrı-ayrılıqda biri-birindən asılı olmamaq şərti ilə **analytic funksiyaları** həyata keçirtməsinə şərait yaradır və bunu həyata keçirdən məhz **OVER()** funksiyasıdır. **GROUP BY** mövzusunu keçəndə görmüşdük ki, orada da **aggregate funksiyalarından** istifadə etmişdik lakin orada işlər tamamilə fərqli idi. Orada cədvəldə olan sətirlərin işlənmə prinsipi bu cürnə idiki, məsələn **GROUP BY** operatoruna vermiş olduğumuz sütununun dəyərinə uyğun gələn sətirlər toplu bir qrup halına salınaraq ən nəhayətində **SELECT** hissəsində hansı **aggregate funksiyasından** istifadə etmişiksə həmən toplu qrup içərisindən yalnızca bir ədəd sətirdəki dəyər çıxarılırdı, lakin **OVER()** funksiyasında işlər biraz fərqlidir. Burada işlənmə prinsipi **GROUP BY** dan fərqli olaraq bütün sətirləri tək bir qrup altına deyildə hər bir sətiri ayrı-ayrılıqda bir qrup kimi başa düşür və hər biri üçün bir **windows** yaradır yəni qrup və hər bir sətir üzərində siz bir növ **GROUP BY**-ın etdiyi işi etmiş olursunuz.
2. Həmçinin **OVER()** daxilində biz spesific **windows** yarada bilirik və bunuda **OVER()** daxilində elan etdiyimiz **partition by** operatorunun vasitəsi ilə həyata keçirdirik. İndi isə gəlin bu nəzəriyyəni anlamaq üçün nümunəyə baxaq.
3. Məsələn, gəlin yuxarıda **3-cü** şəkildə göstərmiş olduğum query-ni və ondan əldə etdiyimiz nəticənin birə-bir eynisini **OVER(**partition by)-dan istifadə edərək həyata keçirtməyə çalışaq.

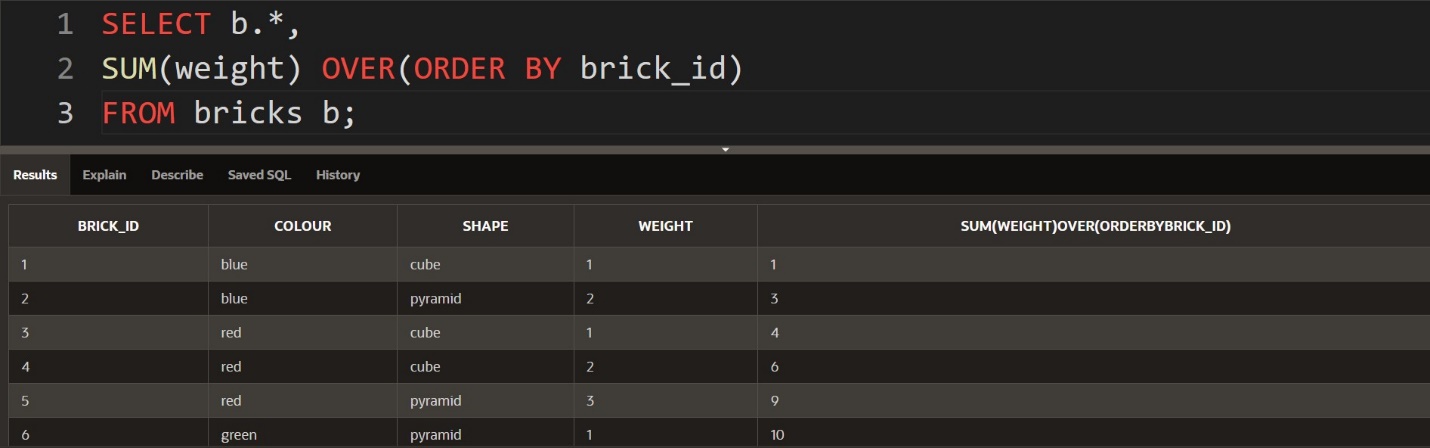


gördüyünüz kimi budəfə isə biz **OVER(PARTITION BY column\_name)** operatorundan istifadə edərək problemi həll etmiş olduq.

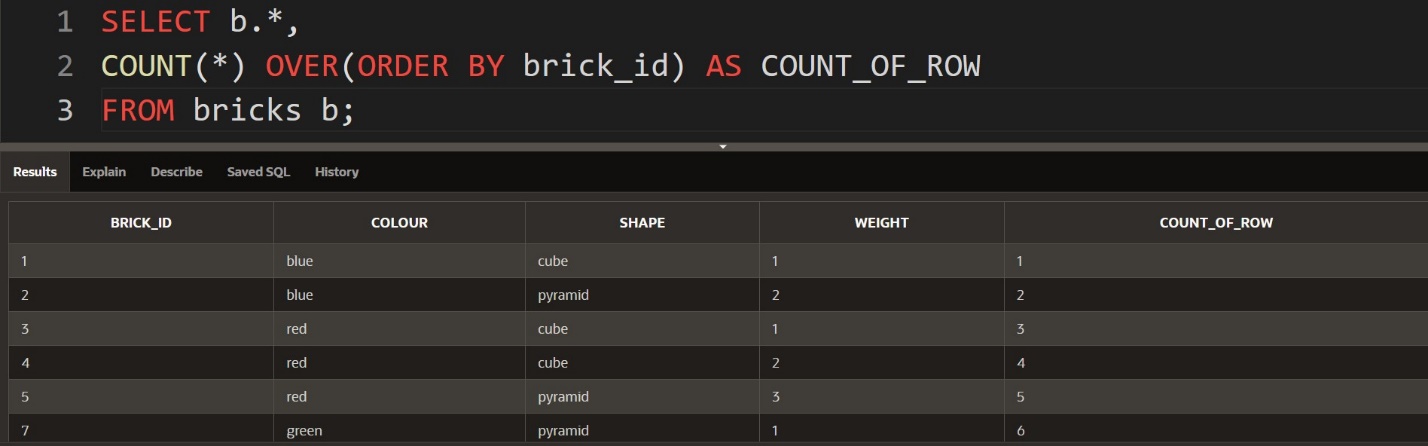
1. **OVER()** daxilində **PARTITION BY colum\_name** yazmaq bir növ **GROUP BY** column\_name yazmaq deməkdir, sadəcə dediyim kimi burada işlər toplu bir qrup halına deyildə hər bir sətiri qrup kimi başa düşə bilməsidir və **SQL-in** hər bir sətiri qrup kimi başa düşməsi **OVER()** hesabına həyata keçirilir. Və biz burada hər bir sətir üzərə qrupu **colour** sütununa görə etmiş olduq və **OVER()** funksiyasından öncə istifadə olunan **aggregate funksiyası** məhz **OVER()** daxilində elan etdiyimiz **PARTITION BY**-a verdiyimiz sütun adına görə həyata keçirilir yəni ondan asılıdır.
2. İndi isə gəlin çoxlu **OVER(PARTITION BY column\_name)-dən** istifadə edərək query yazmağa çalışaq.

****

yuxarıdakı query də biz fərqli-fərqli **aggregate funksiyalarından** istifadə edərək çoxlu sayıda **OVER(PARTITION BY column\_name)** analytic funksiyalarından istifadə etmiş olduq.

1. Həmçinin **OVER()** daxilində **ORDER BY** operatorundanda istifadə etmək mümkündür, eynən aşağıdakı şəkildə olduğu kimi.

gördüyünüz kimi yuxarıdakı query də biz **OVER()** daxilində **ORDER BY** operatorundan istifadə edərək həmən qrup halına salınmış hər bir sətiri **brick\_id** sütununa görə **ASC** şəklində sıralamış olduq və həmən bu sütuna görə biz onların cəmini hər bir sətirdə tapmış olduq. Burada fikir verdinizsə **brick\_id** sütununa görə ardıcıl şəkildə toplanılaraq cəmin tapmış olduq və bu hesablanma mövcud sətirdəki **weight** sütununda olan dəyər ilə bir öncəki sətirdəki **weight** sütunundakı dəyər toplanılaraq hərdəfəsində bir sonrakı sətirdə cəm dəyərin yazdırdı.

1. Daha sonra gəlin belə bir maraqlı query yazaq, məsələn mən istəyirəm ki, hər sətirin sıra sayını **resultset** də göstərmiş olum, eynən aşağıdakı şəkildə olduğu kimi.

gördüyünüz kimi yuxarıdakı query də biz **ORDER BY** brick\_id sütununa görə **OVER()** funksiyasından istifadə edərək **COUNT(\*)** funksiyasını yazaraq hər bir sətirin sıra sayını **resultset** də ən sonda göstərmiş olduq. Diqqətli fikir versəniz **brick\_id** 7 dəyərinə sahib olan sətir 6 göstərdi, yəni **ORDER BY** operatorundan **brick\_id** sütunu sizi çaşdırmasın ki, niyə görə ən sonda 7 deyildə 6 yazıldı. Bu bizdə oldu hər bir sətirin sıra sayını ekranda göstərmək. Bunu biz irəlidə **rank funksiyalarında** görmüş olacayıq.