```
create table applestore_description_ano AS
 select * from appleStore_description1
 UNION
 select * from appleStore_description2
 UNION
 select * from appleStore_description3
 UNION
 select * from appleStore_description4

 ** EXPLORATORY DATA ANALYSIS **

 --Check the number of unique apps--

 select count(DISTINCT id) unique_id from AppleStore;

 select count(distinct id) unique_id from applestore_description_ano;

 -- check for any missing value in key feilds --

 select count(*) missing_value from AppleStore
 where track_name is NULL or user_rating is NULL or prime_genre is NULL;

 select count(*) missing_value from applestore_description_ano
 where app_desc is NULL;

 --find out  number of apps per  genre --

 select prime_genre ,count(*) number_of_apps FROM AppleStore
 group by prime_genre
 order by number_of_apps desc;

 -- Get an overview of the apps ratings --

 select min(user_rating) min_rating,
 max(user_rating) max_rating,
 avg(user_rating) average_rating
 from AppleStore;

 ** DATA ANALYSIS **

 -- Determine whether paid apps have higher ratings than free apps --

 select CASE
         WHEN price>0 then 'paid'
         else 'free'
         end as App_type,
         avg(user_rating)  Avg_rating
 from AppleStore
 group by App_type;

 -- check if apps with more supported languages have higher ratings --

 select case
           when lang_num < 10 THEN  '<10 langauges'
           when lang_num BETWEEN 10 and 30 then '10-30 languages'
           else '>30 langauges'
           end as language_bucket,
           avg(user_rating) Avg_rating

from AppleStore
group by language_bucket
order by Avg_rating desc;

-- check genres with low ratings --

select prime_genre,
```

```
          avg(user_rating) Avg_rating
from AppleStore
group by prime_genre
order by Avg_rating
desc;

-- check the top rated apps for each genre --
select prime_genre,
track_name,
user_rating
from(
select prime_genre,
track_name,
user_rating,
rank() over(order by user_rating desc) rank
from AppleStore) a
where a. rank =1
```