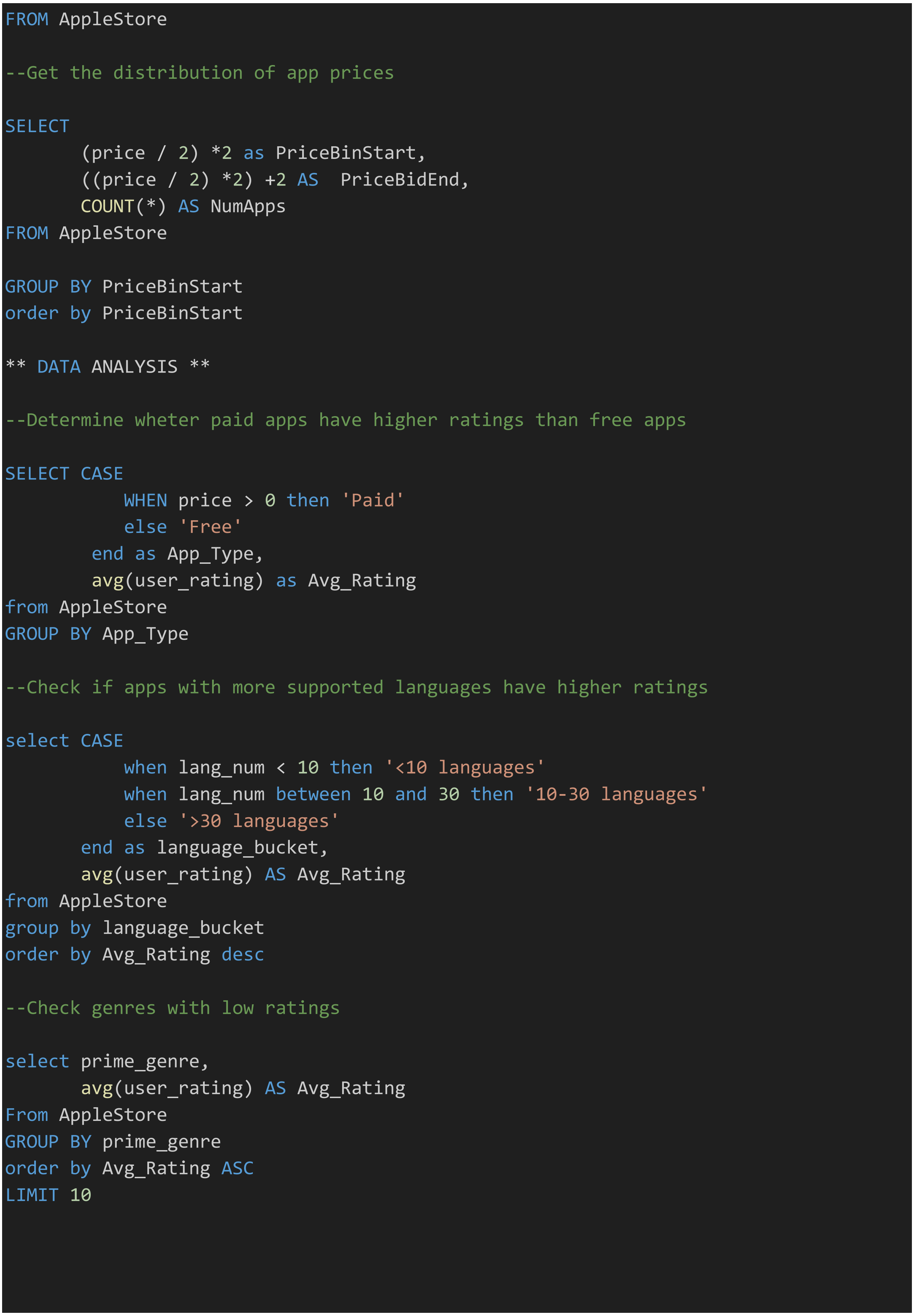
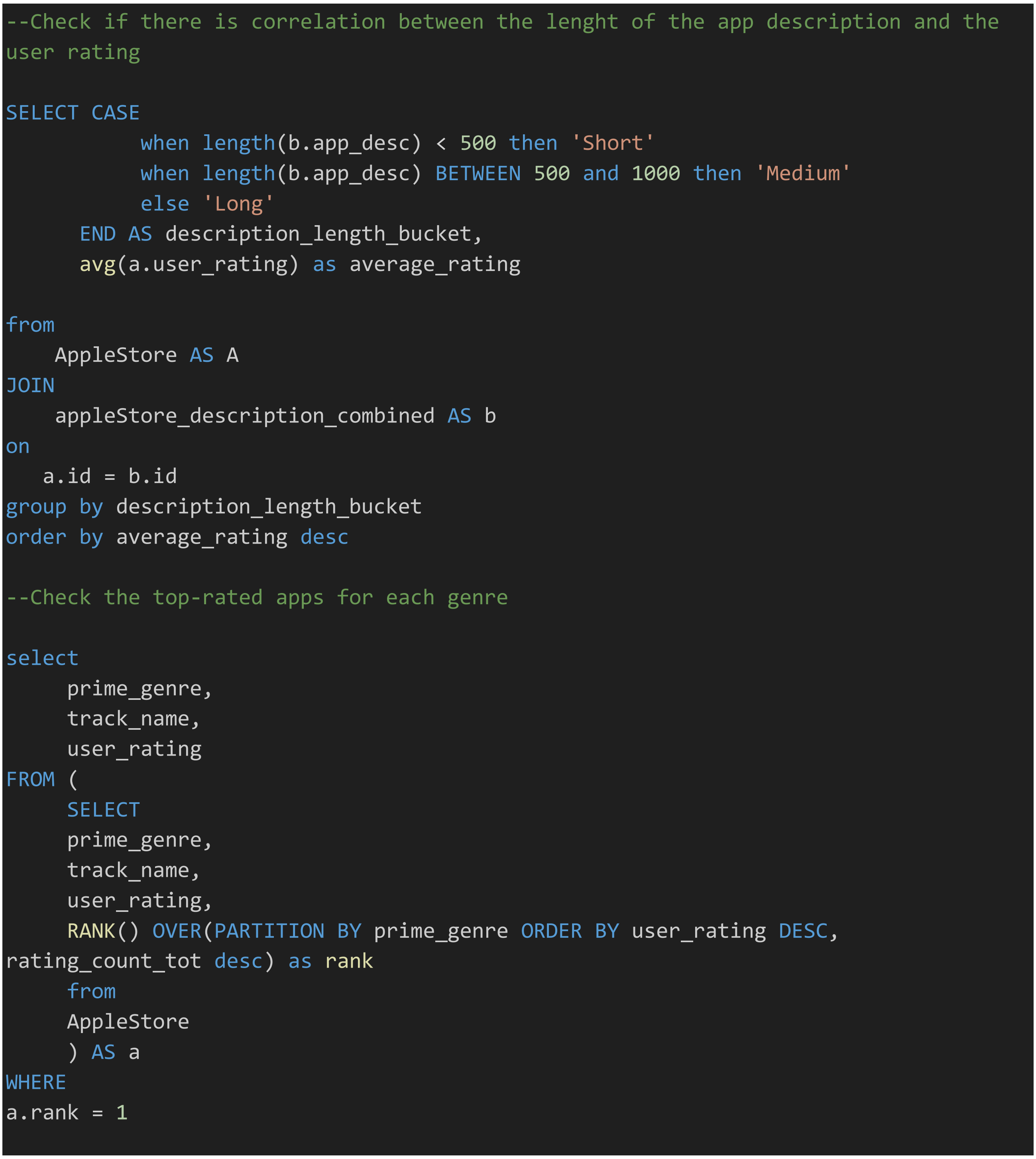
```
** ANALYZING APPLE APP STORE**

CREATE TABLE appleStore_description_combined AS

SELECT * FROM appleStore_description1

UNION ALL

SELECT * FROM appleStore_description2

UNION ALL

SELECT * FROM appleStore_description3

UNION ALL

SELECT * FROM appleStore_description4

** EXPLORATORY DATA ANALYSIS**

--Check the number of unique apps in both tablesAppleStore

SELECT COUNT (DISTINCT id) AS UniqueAppIDs
from AppleStore

SELECT COUNT (DISTINCT id) AS UniqueAppIDs
from appleStore_description_combined

--Check for any missing values in key fields

SELECT COUNT(*) AS MissingValues
from AppleStore
where track_name is null or user_rating is null or prime_genre is null

SELECT COUNT(*) AS MissingValues
from appleStore_description_combined
where app_desc is null

--Find out the number of apps per genre

select prime_genre, count(*) as NumApps
from AppleStore
group by prime_genre
order by NumApps DESC

--Get an overview of the apps' rating

select min(user_rating) as MinRating,
       max(user_rating) as MaxRating,
       avg(user_rating) AS AvgRating
```



```
FROM AppleStore

--Get the distribution of app prices

SELECT
       (price / 2) *2 as PriceBinStart,
       ((price / 2) *2) +2 AS  PriceBidEnd,
       COUNT(*) AS NumApps
FROM AppleStore

GROUP BY PriceBinStart
order by PriceBinStart

** DATA ANALYSIS **

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

SELECT CASE
           WHEN price > 0 then 'Paid'
           else 'Free'
        end as App_Type,
        avg(user_rating) as 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 languages'
           when lang_num between 10 and 30 then '10-30 languages'
           else '>30 languages'
       end as language_bucket,
       avg(user_rating) AS Avg_Rating
from AppleStore
group by language_bucket
order by Avg_Rating desc

--Check genres with low ratings

select prime_genre,
       avg(user_rating) AS Avg_Rating
From AppleStore
GROUP BY prime_genre
order by Avg_Rating ASC
LIMIT 10
```



```sql
--Check if there is correlation between the lenght of the app description and the
user rating

SELECT CASE
           when length(b.app_desc) < 500 then 'Short'
           when length(b.app_desc) BETWEEN 500 and 1000 then 'Medium'
           else 'Long'
      END AS description_length_bucket,
      avg(a.user_rating) as average_rating

from
    AppleStore AS A
JOIN
    appleStore_description_combined AS b
on
   a.id = b.id
group by description_length_bucket
order by average_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(PARTITION BY prime_genre ORDER BY user_rating DESC,
rating_count_tot desc) as rank
     from
     AppleStore
     ) AS a
WHERE
a.rank = 1
```