SQL File 3* | SQL File 5* | SQL File 6*

Limit to 10000 rows

```sql

-- OVERVIEW OF THE DATASETS

-- Check the number of unique apps in both tables
SELECT COUNT(DISTINCT (track_name)) AS unique_apps
FROM applestore_description;

SELECT COUNT(DISTINCT (track_name)) AS unique_apps
FROM applestore;


-- Check for any missing values in key fields
SELECT COUNT(*) AS 'Missing Values'
FROM applestore
WHERE tracK_name IS NULL OR user_rating IS NULL OR prime_genre IS NULL;

SELECT COUNT(*) AS 'Missing Values'
FROM applestore_description
WHERE app_desc IS NULL;


-- Find out the number of apps per genre
SELECT prime_genre as 'Genre', COUNT(prime_genre) AS 'Genre Count'
from applestore
GROUP BY 1
ORDER BY 2 DESC;

-- Get an overview of the apps ratings
SELECT min(user_rating) as min_rating,
       max(user_rating) as max_rating,
       round(avg(user_rating), 2) as avg_rating
FROM applestore;


```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

min_rating	max_rating	avg_rating
0	5	3.53

```sql
-- ANALYSIS

-- Determine whether paid apps have a better rating than free apps
SELECT CASE
            WHEN price > 0 THEN 'paid'
            ELSE 'free'
        END AS App_Type,
        ROUND(AVG(user_rating), 2) as App_Rating
FROM applestore
GROUP BY 1
ORDER BY 2 DESC;

-- Check if apps with more supported languages have a better rating
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,
        ROUND(AVG(user_rating), 2) as average_rating
FROM applestore
GROUP BY 1
ORDER BY 2 DESC;

-- Check genre with low ratings
CREATE TEMPORARY TABLE test_table
    SELECT prime_genre,
           COUNT(*) AS num_apps,
           ROUND(AVG(user_rating), 2) AS average_rating
    FROM applestore
    GROUP BY 1
    HAVING average_rating < 3
    ORDER BY 2 ASC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total apps with <3 rating
772

```sql
-- FURTHER ANALYSIS

-- Check if there is a correlation between app description length and the user rating
SELECT CASE
            WHEN length(app_desc) < 500 THEN 'short'
            WHEN length(app_desc) BETWEEN 500 AND 1000 THEN 'medium'
            ELSE 'long'
        END AS description_length_bucket,
        ROUND(AVG(user_rating), 2) AS average_rating
FROM applestore a
JOIN applestore_description b
    ON a.id = b.id
GROUP BY 1
ORDER BY 2 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 top_rank
    FROM applestore) AS a
WHERE a.top_rank = 1;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

prime_genre	track_name	user_rating
Book	Color Therapy Adult Coloring Book for Adults	5
Business	TurboScan™ Pro - document & receipt scanner:...	5
Catalogs	CPlus for Craigslist app - mobile classifieds	5
Education	Elevate - Brain Training and Games	5
Entertainment	Bruh-Button	5
Finance	Credit Karma: Free Credit Scores, Reports & Al...	5
Food & Drink	Domino's Pizza USA	5
Games	Head Soccer	5
Health & Fitness	Yoga Studio	5
Lifestyle	ipsy - Makeup, subscription and beauty tips	5

Result Grid | Form Editor | Field Types