Objectives

- Top 10 most efficient apps to buy
    - We can focus our efforts on apps that are in both tables since the client finds it preferable to buy apps that are in both app stores due to a lower cost of maintenance.
    - Find ratings between free and paid apps?
    - If there's a lot of correlation between the top 10, we can combine them to represent a broader spread?
- Top 4 Capitalistically-Efficient Spooky Apps

- Report of costs/profits

Trends to Find

- Genre
    - Which genres had the most downloads on both stores?
- Content Rating
    - Lowest age
- Price Range
    - Which prices had the most downloads?
- Rating
    - Find a way to average the ratings between the two

Analysis

Develop some general recommendations about the price range, genre, content rating, or any other app characteristics that the company should target

We recommend

- Apps that are in both the app store and play store
- Apps that are listed as “free” in the app store
- Apps that are in the game genre
- Apps with a minimum of 4.5 rating
- Apps with an avg review count greater than or equal to 228000

Develop a Top 10 List of the apps that App Trader should buy based on profitability/return on investment as the sole priority.

- We focused our data analysis on Apps that are listed on both the App Store and Google Play Store, as preferred by App Trader due to its cost-effectiveness in the long run. We compiled a list of the top 10 apps recommended for purchase by App Trader, taking into consideration factors such as the app's average rating, average review count, genre, longevity, and profitability. We also used an avg review count based on the data to create a baseline for comparing the apps. We noticed during our analysis that the review count has a direct affect on the app’s longevity and success.

- PewDiePie's Tuber Simulator
- Egg, Inc
- Domino's Pizza USA
- Fernanfloo
- "Geometry Dash Lite"
- Wish - Shopping Made Fun
- Fruit Ninja
- Flow Free
- Toy Blast
- Angry Birds Rio

HALLOWEEN APP ANALYSIS

OUR RECOMMENDED APPS

Zombie Catchers
Plants Vs. Zombies 2
Plants Vs. Zombies Heroes
Zombie Tsunami

THEMING

For our top 4 Halloween apps, our team chose to “flesh” out our options with a Zombie theme. A central interest promotes the apps individually, but also together as consumers who are interested in one app, are likely to be interested in the others. Mwahahaha they will fall into our irresistible app choices!

COST

All of our chosen apps are free to download which means they would initially cost $25,000 to obtain. However, as $25,000 is the minimum to purchase an app, we’re keeping expenses as low as possible within our boundaries. In addition to this cost, they will each require $1000 a month to maintain. As we all know, no good purchase goes unhaunted!

RETURN ON INVESTMENT

All four apps are estimated to generate $5000 per month, resulting in $4000 after maintenance. However, if we take their ratings into account, each app has a projected longevity of 10 years. Keep in mind that they will pay their initial cost back and more after 7 months, then continue to generate $4000 in profit for another 9+ years, allowing their total profits to accumulate to over $450,000 for each app.

CONCLUSION

Our ghoulish initiative has an estimated 10 years of revenue generation while remaining at the lowest cost of investment possible. All four apps fall under the same theme and are able to promote each other to consumers who may already be interested in one. In conclusion, we are presenting App Trader a way to profit for years to come with as little room for risk as possible. In other words…

NO TRICKS, ONLY TREATS!

Code

```
SELECT a.name,ROUND(AVG(a.rating + p.rating)/2,2) AS total_rating,
ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) AS rounded_total_rating,
MAX(a.price) AS price,
MAX(p.review_count) AS play_store_reviews,
MAX(a.review_count) AS app_store_reviews,
p.genres,a.primary_genre,MAX(a.price) + 25000 AS price_to_buy_app,
ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) *24)+12)),2) AS
longevity_in_months,
ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) *24)+12)/12),2) AS
longevity_in_years,
ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) *24)+12)*5000),2) AS
money_made_from_app,
ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) *24)+12)*1000),2) AS
app_maintenance,
(ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2)
*24)+12)*5000),2)-ROUND((((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2)
*24)+12)*1000)+25000,2)) AS profit
FROM app_store_apps AS a
INNER JOIN play_store_apps AS p
ON a.name = p.name
WHERE p.review_count >= 444153 AND CAST(a.review_count AS int) >= 12892
GROUP BY a.name,p.genres,a.primary_genre
ORDER BY total_rating DESC
LIMIT 10;

Updated
WITH data AS (SELECT a.name,ROUND(AVG(a.rating + p.rating)/2,2) AS total_rating,
                        ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) AS
rounded_total_rating,
                        MAX(a.price) AS price,
                        MAX(p.review_count) AS play_store_reviews,
                        MAX(a.review_count) AS app_store_reviews,
                        p.genres,a.primary_genre,MAX(a.price) + 25000 AS price_to_buy_app,
                        (((ROUND((FLOOR(AVG(a.rating + p.rating)/2/.25)*.25),2) *24)+12)) AS
variable
```

```
                        FROM app_store_apps AS a
                        INNER JOIN play_store_apps AS p
                        ON a.name = p.name
                        WHERE p.review_count >= 444153 AND CAST(a.review_count AS int)
>= 12892
                        GROUP BY a.name,p.genres,a.primary_genre
                        ORDER BY total_rating DESC
                        LIMIT 10)
SELECT name,total_rating,rounded_total_rating,price,play_store_reviews, app_store_reviews,
                        genres,primary_genre, price_to_buy_app,
                        ROUND(variable,2) AS longevity_in_months,
                        ROUND((variable/12),2) AS longevity_in_years,
                        ROUND(((variable+12)*5000),2)::money AS money_made_from_app,
                        ROUND(((variable+12)*1000),2)::money AS app_maintenance,

(ROUND(((variable+12)*5000),2)-ROUND(((variable+12)*1000),2))::money AS profit
FROM data;
```

```
WITH app_stats AS (SELECT name,
p.rating,
s.rating,
ROUND(ROUND(((p.rating + s.rating)/2)*4)/4,2) AS avg_rating,
(p.review_count + s.review_count::INT)/2 AS avg_review_count,
MAX(p.price) AS app_store_price,
MAX(s.price),
p.genres,
s.primary_genre AS game_genre,
(s.price+25000)::money AS app_sale_cost
FROM play_store_apps AS p INNER JOIN app_store_apps AS s USING(name)
WHERE s.price = 0
GROUP BY name,
p.rating,
s.rating,
s.price,
p.genres,
s.primary_genre,
p.review_count,
s.review_count
ORDER BY avg_rating DESC)
```

```
SELECT  name,
game_genre,
app_store_price,
avg_rating,
avg_review_count,
app_sale_cost,
(avg_rating * 2)+1 AS longevity_in_yrs,
((avg_rating * 2 + 1) *12 *1000)::money AS maintenance_cost,
((avg_rating * 2 + 1) *12 *5000)::money AS app_revenue,
((avg_rating * 2 + 1) *12 *5000)::money - ((avg_rating * 2 + 1) *12 *1000)::money AS profit
FROM app_stats
WHERE avg_review_Count >=228522
ORDER BY avg_rating DESC
WITH Halloween_apps AS (SELECT DISTINCT name,
ROUND((p.rating+ s.rating)/2,1) AS avg_rating,
s.price AS price,
(s.price+25000)::money AS cost_to_buy_app
FROM play_store_apps AS p INNER JOIN app_store_apps AS s USING(name)
WHERE name ILIKE '%zombie%'
ORDER BY avg_rating DESC)


SELECT name,
avg_rating,
price,
(avg_rating * 2)+1 AS longevity_in_yrs,
((avg_rating * 2 + 1) *12 *1000)::money AS maintenance_cost,
((avg_rating * 2 + 1) *12 *5000)::money AS app_revenue,
((avg_rating * 2 + 1) *12 *5000)::money - ((avg_rating * 2 + 1) *12 *1000)::money AS profit
FROM Halloween_apps
```

Halloween app data

```
WITH Halloween_apps AS (SELECT DISTINCT name,
```

```
ROUND((p.rating+ s.rating)/2,1) AS avg_rating,
s.price AS price,
(s.price+25000)::money AS cost_to_buy_app
FROM play_store_apps AS p INNER JOIN app_store_apps AS s USING(name)
WHERE name ILIKE '%zombie%'
ORDER BY avg_rating DESC)


SELECT name,
avg_rating,
price,
(avg_rating * 2)+1 AS longevity_in_yrs,
((avg_rating * 2 + 1) *12 *1000)::money AS maintenance_cost,
((avg_rating * 2 + 1) *12 *5000)::money AS app_revenue,
((avg_rating * 2 + 1) *12 *5000)::money - ((avg_rating * 2 + 1) *12 *1000)::money AS profit
FROM Halloween_apps
```

Updated profit

```
WITH Halloween_apps AS (SELECT DISTINCT name,
ROUND((p.rating+ s.rating)/2,1) AS avg_rating,
s.price AS price,
(s.price+25000)::money AS cost_to_buy_app
FROM play_store_apps AS p INNER JOIN app_store_apps AS s USING(name)
WHERE name ILIKE '%zombie%'
ORDER BY avg_rating DESC)


SELECT name,
avg_rating,
price,
(avg_rating * 2)+1 AS longevity_in_yrs,
((avg_rating * 2 + 1) *12 *1000)::money AS maintenance_cost,
((avg_rating * 2 + 1) *12 *5000)::money AS app_revenue,
((avg_rating * 2 + 1) *12 *5000)::money - (((avg_rating * 2 + 1) *12 *1000)+25000)::money
AS profit
FROM Halloween_apps
```