stats_167_final

2026-06-09

View Data

```
dbListTables(con)
```

```
##  [1] "BATCH_JOB_EXECUTION"          "BATCH_JOB_EXECUTION_CONTEXT"
##  [3] "BATCH_JOB_EXECUTION_PARAMS"   "BATCH_JOB_EXECUTION_SEQ"
##  [5] "BATCH_JOB_INSTANCE"           "BATCH_JOB_SEQ"
##  [7] "BATCH_STEP_EXECUTION"         "BATCH_STEP_EXECUTION_CONTEXT"
##  [9] "BATCH_STEP_EXECUTION_SEQ"     "episode"
## [11] "movie"                        "season"
## [13] "sqlite_sequence"              "tv_show"
## [15] "view_summary"
```

```
dbListFields(con, "episode")
```

```
## [1] "episode_number" "release_date"   "runtime"        "created_date"
## [5] "id"             "modified_date"  "season_id"      "original_title"
## [9] "title"
```

```
dbListFields(con, "movie")
```

```
## [1] "available_globally" "release_date"       "created_date"
## [4] "id"                 "modified_date"      "runtime"
## [7] "locale"             "original_title"     "title"
```

```
dbListFields(con, "season")
```

```
## [1] "release_date"   "season_number"  "created_date"   "id"
## [5] "modified_date"  "runtime"        "tv_show_id"     "original_title"
## [9] "title"
```

```
dbListFields(con, "tv_show")
```

```
## [1] "available_globally" "release_date"       "created_date"
## [4] "id"                 "modified_date"      "locale"
## [7] "original_title"     "title"
```

```
dbListFields(con, "view_summary")
```

```
##  [1] "cumulative_weeks_in_top10" "end_date"
##  [3] "hours_viewed"              "start_date"
##  [5] "view_rank"                 "views"
##  [7] "created_date"              "id"
##  [9] "modified_date"             "movie_id"
## [11] "season_id"                 "duration"
```

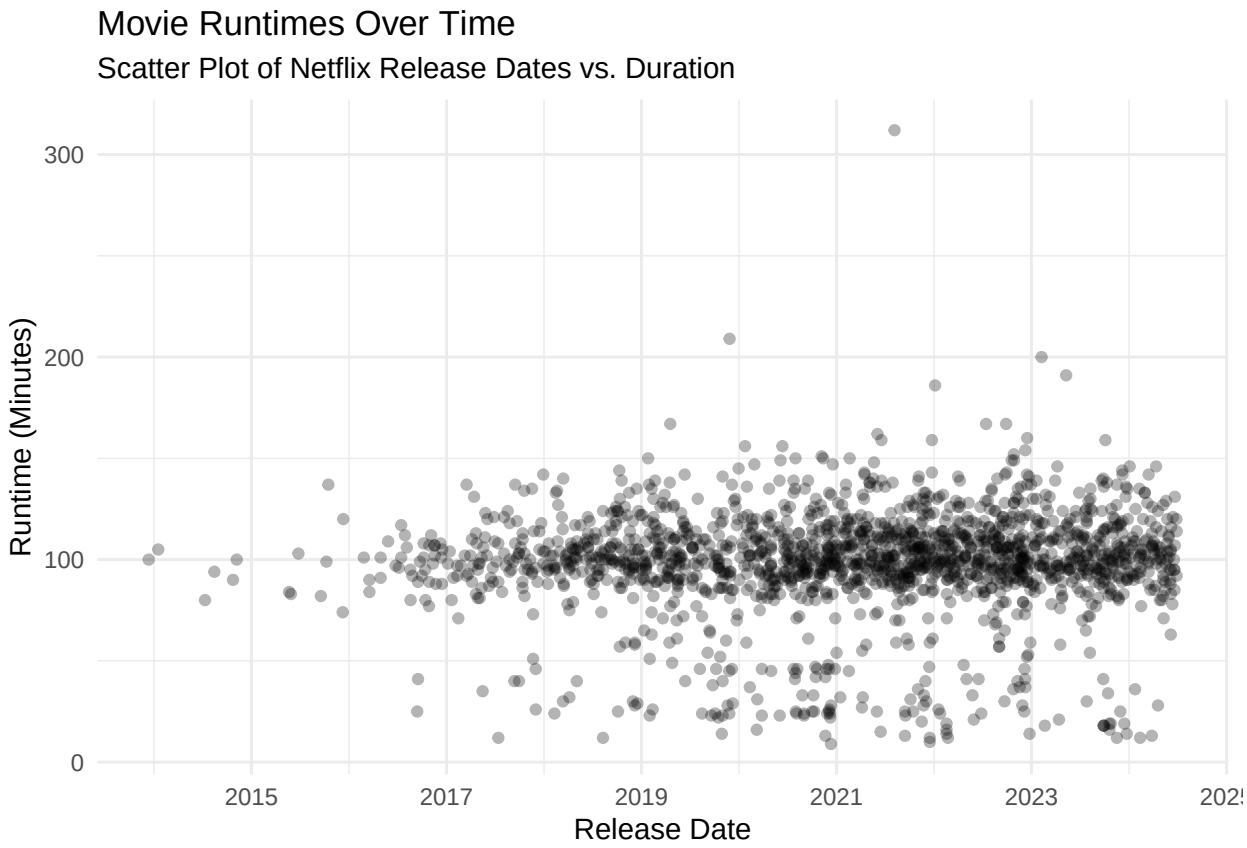
Q1 (Movies/Release)

```
SELECT
      id AS movie_id,
      title,
      release_date,
      runtime
  FROM movie
  WHERE release_date IS NOT NULL
    AND runtime IS NOT NULL
LIMIT 5;
```

Table 1: 5 records

movie_id	title	release_date	runtime
1	Damsel	1709856000000	110
2	Lift	1705017600000	107
3	Society of the Snow	1704326400000	146
4	Under Paris	1717545600000	104
6	Mother of the Bride	1715212800000	90

```
library(ggplot2)
sql_query <- "
  SELECT
      id AS movie_id,
      title,
      release_date,
      runtime
  FROM movie
  WHERE release_date IS NOT NULL
    AND runtime IS NOT NULL;
"
movies_df <- dbGetQuery(con, sql_query)
## clean dates
movies_df$release_date <- as.numeric(movies_df$release_date)
movies_df$release_date <- as.POSIXct(movies_df$release_date / 1000, origin = "1970-01-01", tz = "UTC")
movies_df$release_date <- as.Date(movies_df$release_date)
## plot
ggplot(movies_df, aes(x = release_date, y = runtime)) +
  geom_point(alpha = 0.3) +
  scale_x_date(date_labels = "%Y", date_breaks = "2 years") +
  labs(
    title = "Movie Runtimes Over Time",
    subtitle = "Scatter Plot of Netflix Release Dates vs. Duration",
    x = "Release Date",
    y = "Runtime (Minutes)"
  ) +
  theme_minimal()
```




```
ggsave("rq1.png", width = 6, height = 4, dpi = 300)

## fit model
model <- lm(runtime ~ release_date, data = movies_df)
summary(model)
```

```
##
## Call:
## lm(formula = runtime ~ release_date, data = movies_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -89.096  -7.558   2.166  13.257 213.528
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.870e+01  1.575e+01   4.362 1.36e-05 ***
## release_date 1.580e-03  8.421e-04   1.876   0.0608 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.53 on 1772 degrees of freedom
## Multiple R-squared:  0.001982,   Adjusted R-squared:  0.001419
## F-statistic: 3.519 on 1 and 1772 DF,  p-value: 0.06084
```

```
## r^2 low comfirms no correlation, p-value high confirms release_date has no impact on movie length
```

Q2 Top 10 shows

```sql
SELECT
    t.title,
    SUM(v.views) AS overall_total_views
FROM tv_show t
JOIN season s ON t.id = s.tv_show_id
JOIN view_summary v ON s.id = v.season_id
WHERE s.season_number IS NOT NULL
GROUP BY t.title
ORDER BY overall_total_views DESC
LIMIT 10;
```

Table 2: Displaying records 1 - 10

title	overall_total_views
CoComelon	315000000
Young Sheldon	194100000
Bridgerton	178600000
Gabby's Dollhouse	172600000
Suits (2011)	161000000
Peppa Pig	139700000
Lottie Dottie Chicken	114900000
Brooklyn Nine-Nine	114500000
Little Angel	111000000
Lupin	109600000

```sql
WITH RankedShows AS (
    SELECT
        t.id AS tv_show_id,
        t.title,
        SUM(v.views) AS overall_total_views,
        RANK() OVER (ORDER BY SUM(v.views) DESC) AS show_rank
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    WHERE s.season_number IS NOT NULL
    GROUP BY t.id, t.title
)
SELECT
    rs.title,
    s.season_number,
    SUM(v.views) AS total_season_views
FROM RankedShows rs
JOIN season s ON rs.tv_show_id = s.tv_show_id
JOIN view_summary v ON s.id = v.season_id
WHERE rs.show_rank = 1
  AND s.season_number IS NOT NULL
GROUP BY rs.title, s.season_number
ORDER BY s.season_number;
```

Table 3: Displaying records 1 - 10

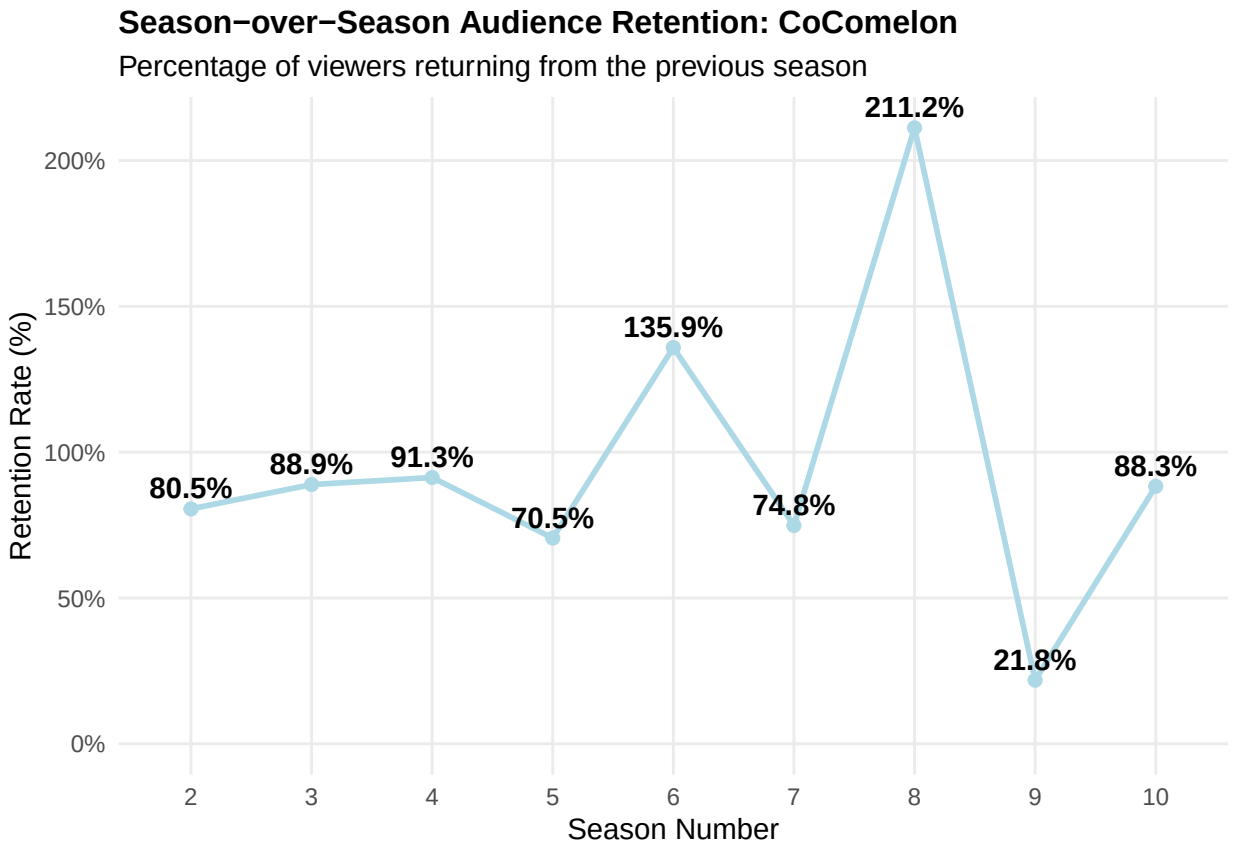
title	season_number	total_season_views
CoComelon	1	51400000
CoComelon	2	41400000
CoComelon	3	36800000
CoComelon	4	33600000
CoComelon	5	23700000
CoComelon	6	32200000
CoComelon	7	24100000
CoComelon	8	50900000
CoComelon	9	11100000
CoComelon	10	9800000

```
library(dplyr)
query2 <- "
WITH RankedShows AS (
    SELECT
        t.id AS tv_show_id,
        t.title,
        SUM(v.views) AS overall_total_views,
        RANK() OVER (ORDER BY SUM(v.views) DESC) AS show_rank
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    WHERE s.season_number IS NOT NULL
    GROUP BY t.id, t.title
)
SELECT
    rs.title,
    s.season_number,
    SUM(v.views) AS total_season_views
FROM RankedShows rs
JOIN season s ON rs.tv_show_id = s.tv_show_id
JOIN view_summary v ON s.id = v.season_id
WHERE rs.show_rank = 1
  AND s.season_number IS NOT NULL
GROUP BY rs.title, s.season_number
ORDER BY s.season_number;
"
top_show <- dbGetQuery(con, query2)

retention_df <- top_show %>%
  arrange(season_number) %>%
  mutate(
    previous_season_views = lag(total_season_views),
    retention_rate = (total_season_views / previous_season_views),
    retention_pct = paste0(round(retention_rate * 100, 1), "%")
  ) %>%
  filter(!is.na(retention_rate))

print(retention_df)
```

```
##       title season_number total_season_views previous_season_views
## 1 CoComelon             2           41400000               51400000
## 2 CoComelon             3           36800000               41400000
## 3 CoComelon             4           33600000               36800000
## 4 CoComelon             5           23700000               33600000
## 5 CoComelon             6           32200000               23700000
## 6 CoComelon             7           24100000               32200000
## 7 CoComelon             8           50900000               24100000
## 8 CoComelon             9           11100000               50900000
## 9 CoComelon            10            9800000               11100000
##   retention_rate retention_pct
## 1      0.8054475         80.5%
## 2      0.8888889         88.9%
## 3      0.9130435         91.3%
## 4      0.7053571         70.5%
## 5      1.3586498        135.9%
## 6      0.7484472         74.8%
## 7      2.1120332        211.2%
## 8      0.2180747         21.8%
## 9      0.8828829         88.3%
```

```
library(scales)

ggplot(retention_df, aes(x = as.factor(season_number), y = retention_rate, group = 1)) +
  geom_line(color = "lightblue", size = 1) +
  geom_point(color = "lightblue", size = 2) +
  geom_text(aes(label = retention_pct), vjust = -.5, fontface = "bold") +
  scale_y_continuous(labels = percent_format(), limits = c(0, NA)) +
  labs(
    title = paste("Season-over-Season Audience Retention:", unique(retention_df$title)),
    subtitle = "Percentage of viewers returning from the previous season",
    x = "Season Number",
    y = "Retention Rate (%)"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 12),
    panel.grid.minor = element_blank()
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
ggsave("rq2.png", width = 6, height = 6, dpi = 300)
```

Q3 tv vs movies

```
WITH MoviePerformance AS (
    SELECT
        m.title,
        'Movie' AS format_type,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    GROUP BY m.title
),
ShowPerformance AS (
    SELECT
        t.title,
        'TV Show' AS format_type,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    GROUP BY t.title
)
SELECT * FROM MoviePerformance
UNION ALL
SELECT * FROM ShowPerformance
```

```
ORDER BY max_weeks DESC
LIMIT 5;
```

Table 4: 5 records

title	format_type	max_weeks
KPop Demon Hunters	Movie	50
Squid Game	TV Show	32
Stranger Things	TV Show	29
Wednesday	TV Show	28
Ms. Rachel	TV Show	25

```
query3 <- "
WITH MoviePerformance AS (
    -- Extract the maximum weeks charted for each movie
    SELECT
        m.title,
        'Movie' AS format_type,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    GROUP BY m.title
),
ShowPerformance AS (
    -- Extract the maximum weeks charted for each TV show overall
    SELECT
        t.title,
        'TV Show' AS format_type,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    GROUP BY t.title
)
-- Stack the results together into one unified dataset
SELECT * FROM MoviePerformance
UNION ALL
SELECT * FROM ShowPerformance;
"
movie_v_tv <- dbGetQuery(con, query3)
movie_v_tv_clean <- movie_v_tv %>%
  filter(!is.na(max_weeks))
wilcox_result <- wilcox.test(max_weeks ~ format_type, data = movie_v_tv_clean)
print(wilcox_result)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  max_weeks by format_type
## W = 465937, p-value = 7.685e-12
## alternative hypothesis: true location shift is not equal to 0
```

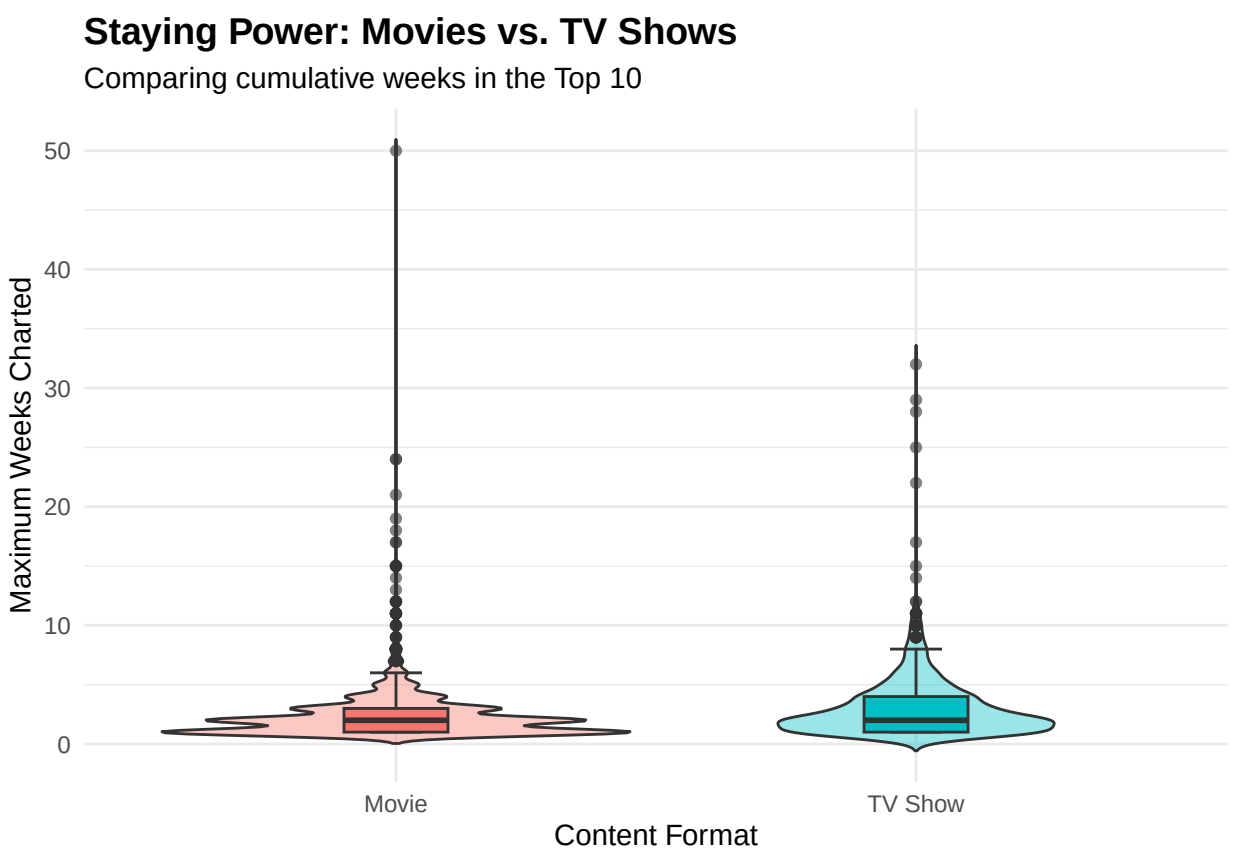
```r
print(aggregate(max_weeks ~ format_type, data=movie_v_tv_clean, FUN=median))
```

```
##   format_type max_weeks
## 1       Movie         2
## 2     TV Show         2
```

```r
print(aggregate(max_weeks ~ format_type, data = movie_v_tv_clean, FUN = mean))
```

```
##   format_type max_weeks
## 1       Movie  2.557879
## 2     TV Show  3.188769
```

```r
ggplot(movie_v_tv_clean, aes(x = format_type, y = max_weeks, fill = format_type)) +
  geom_violin(alpha = 0.4, trim = FALSE) +
  geom_boxplot(width = 0.2, outlier.alpha = 0.6, staplewidth = 0.5) +
  labs(
    title = "Staying Power: Movies vs. TV Shows",
    subtitle = "Comparing cumulative weeks in the Top 10",
    x = "Content Format",
    y = "Maximum Weeks Charted",
    fill = "Format"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "none"
  )
```



```
ggsave("rq3.png", width = 6, height = 4, dpi = 300)
```

Q4 movie length

```
WITH ChartedMovies AS (
    SELECT
        m.title,
        m.runtime,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    WHERE m.runtime IS NOT NULL AND m.runtime > 0
    GROUP BY m.title, m.runtime
)
SELECT
    title,
    max_weeks,
    CASE
        WHEN runtime < 90 THEN '1. Short (<90m)'
        WHEN runtime BETWEEN 90 AND 119 THEN '2. Standard (90-119m)'
        WHEN runtime BETWEEN 120 AND 149 THEN '3. Long (120-149m)'
        ELSE '4. Very Long (150m+)'
    END AS runtime_bucket
FROM ChartedMovies
LIMIT 5;
```

Table 5: 5 records

title	max_weeks	runtime_bucket
“Sr.”	NA	1. Short (<90m)
#Alive	NA	2. Standard (90-119m)
#AtFirstSight	NA	2. Standard (90-119m)
#FriendButMarried	NA	2. Standard (90-119m)
#FriendButMarried 2	NA	2. Standard (90-119m)

```
query4 <- "
WITH ChartedMovies AS (
    -- Get the maximum weeks charted for valid movies
    SELECT
        m.title,
        m.runtime,
        MAX(v.cumulative_weeks_in_top10) AS max_weeks
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    WHERE m.runtime IS NOT NULL AND m.runtime > 0
    GROUP BY m.title, m.runtime
)
-- Assign each movie to a categorical runtime bucket
SELECT
    title,
    max_weeks,
    CASE
```

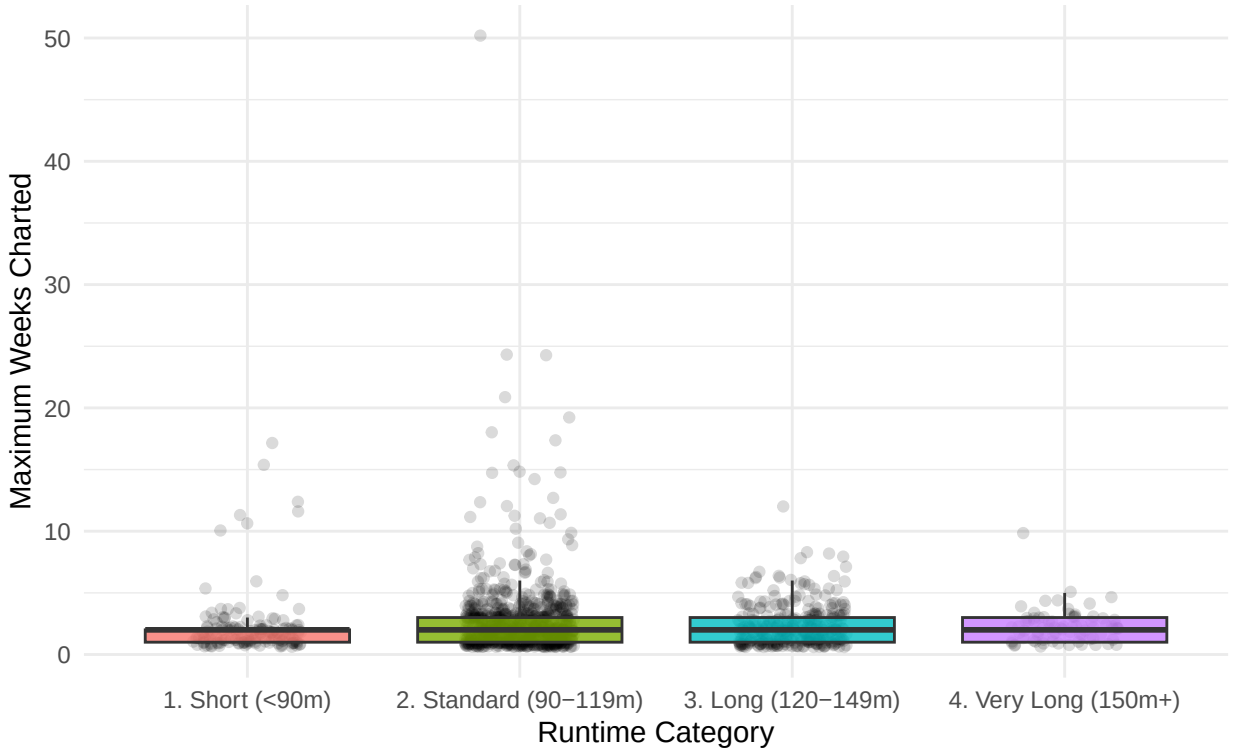
```
        WHEN runtime < 90 THEN '1. Short (<90m)'
        WHEN runtime BETWEEN 90 AND 119 THEN '2. Standard (90-119m)'
        WHEN runtime BETWEEN 120 AND 149 THEN '3. Long (120-149m)'
        ELSE '4. Very Long (150m+)'
    END AS runtime_bucket
FROM ChartedMovies;
"
runtime <- dbGetQuery(con, query4) %>%
  filter(!is.na(max_weeks))
kw_result <- kruskal.test(max_weeks ~ as.factor(runtime_bucket), data = runtime)
print(kw_result)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  max_weeks by as.factor(runtime_bucket)
## Kruskal-Wallis chi-squared = 18.402, df = 3, p-value = 0.0003634
```

```
pairwise.wilcox.test(runtime$max_weeks, runtime$runtime_bucket, p.adjust.method = "BH")
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity correction
##
## data:  runtime$max_weeks and runtime$runtime_bucket
##
##                       1. Short (<90m) 2. Standard (90-119m) 3. Long (120-149m)
## 2. Standard (90-119m) 0.00019         -                     -
## 3. Long (120-149m)    0.01324         0.21240               -
## 4. Very Long (150m+)  0.00465         0.95981               0.50390
##
## P value adjustment method: BH
```

```
ggplot(runtime, aes(x = runtime_bucket, y = max_weeks, fill = runtime_bucket)) +
  geom_jitter(color = "black", alpha = 0.15, width = 0.2) +
  geom_boxplot(alpha = 0.8, outlier.shape = NA) +
  labs(
    title = "The Golden Page Count: Chart Staying Power by Runtime",
    subtitle = "Comparing Top 10 longevity across industry-standard movie lengths",
    x = "Runtime Category",
    y = "Maximum Weeks Charted",
    fill = "Runtime Bucket"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    legend.position = "none" # Hide the legend to save space
  )
```




```
ggsave("rq4.png", width = 6, height = 4, dpi = 300)
```

Q5 Season 1 v 2

```
WITH StrictTwoSeasonShows AS (
    SELECT
        tv_show_id
    FROM season
    GROUP BY tv_show_id
    HAVING MAX(season_number) = 2
),
SeasonViews AS (
    SELECT
        t.title,
        s.season_number,
        SUM(v.views) AS total_views
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    JOIN StrictTwoSeasonShows st ON t.id = st.tv_show_id
    GROUP BY t.title, s.season_number
),
ShowComparison AS (
    SELECT
        title,
        MAX(CASE WHEN season_number = 1 THEN total_views END) AS season_1_views,
```

```
        MAX(CASE WHEN season_number = 2 THEN total_views END) AS season_2_views
    FROM SeasonViews
    GROUP BY title
)
SELECT * FROM ShowComparison
WHERE season_1_views IS NOT NULL
  AND season_2_views IS NOT NULL
LIMIT 5;
```

Table 6: 5 records

title	season_1_views	season_2_views
100 Days to Indy	300000	100000
3Below: Tales of Arcadia	2600000	1800000
7SEEDS	500000	300000
A Clean Sweep	100000	100000
A Killer's Mistake	200000	300000

```
query5 <- "
WITH StrictTwoSeasonShows AS (
    -- Step 1: Find shows where the highest season number is exactly 2
    SELECT
        tv_show_id
    FROM season
    GROUP BY tv_show_id
    HAVING MAX(season_number) = 2
),
SeasonViews AS (
    -- Step 2: Get the views, but ONLY for the shows we isolated in Step 1
    SELECT
        t.title,
        s.season_number,
        SUM(v.views) AS total_views
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    JOIN StrictTwoSeasonShows st ON t.id = st.tv_show_id
    GROUP BY t.title, s.season_number
),
ShowComparison AS (
    -- Step 3: Pivot the data into columns
    SELECT
        title,
        MAX(CASE WHEN season_number = 1 THEN total_views END) AS season_1_views,
        MAX(CASE WHEN season_number = 2 THEN total_views END) AS season_2_views
    FROM SeasonViews
    GROUP BY title
)
-- Step 4: Ensure both seasons actually charted and have data
SELECT * FROM ShowComparison
WHERE season_1_views IS NOT NULL
  AND season_2_views IS NOT NULL;
```

```
"
s1s2 <- dbGetQuery(con, query5)

paired_test <- wilcox.test(s1s2$season_1_views, s1s2$season_2_views, paired = TRUE)
print(paired_test)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  s1s2$season_1_views and s1s2$season_2_views
## V = 61136, p-value = 2.02e-11
## alternative hypothesis: true location shift is not equal to 0
```

```
median(s1s2$season_1_views)
```

```
## [1] 900000
```

```
median(s1s2$season_2_views)
```

```
## [1] 700000
```

Q6

```
WITH MovieViews AS (
    SELECT
        m.title,
        'Movie' AS format_type,
        SUM(CASE WHEN v.start_date < 1704067200000 THEN v.views ELSE 0 END) AS h2_2023_views,
        SUM(CASE WHEN v.start_date >= 1704067200000 THEN v.views ELSE 0 END) AS h1_2024_views
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    GROUP BY m.title
),
ShowViews AS (
    SELECT
        t.title,
        'TV Show' AS format_type,
        SUM(CASE WHEN v.start_date < 1704067200000 THEN v.views ELSE 0 END) AS h2_2023_views,
        SUM(CASE WHEN v.start_date >= 1704067200000 THEN v.views ELSE 0 END) AS h1_2024_views
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    GROUP BY t.title
),
CombinedViews AS (
    SELECT * FROM MovieViews
    UNION ALL
    SELECT * FROM ShowViews
)
SELECT * FROM CombinedViews
WHERE h2_2023_views > 0 AND h1_2024_views > 0
LIMIT 5;
```

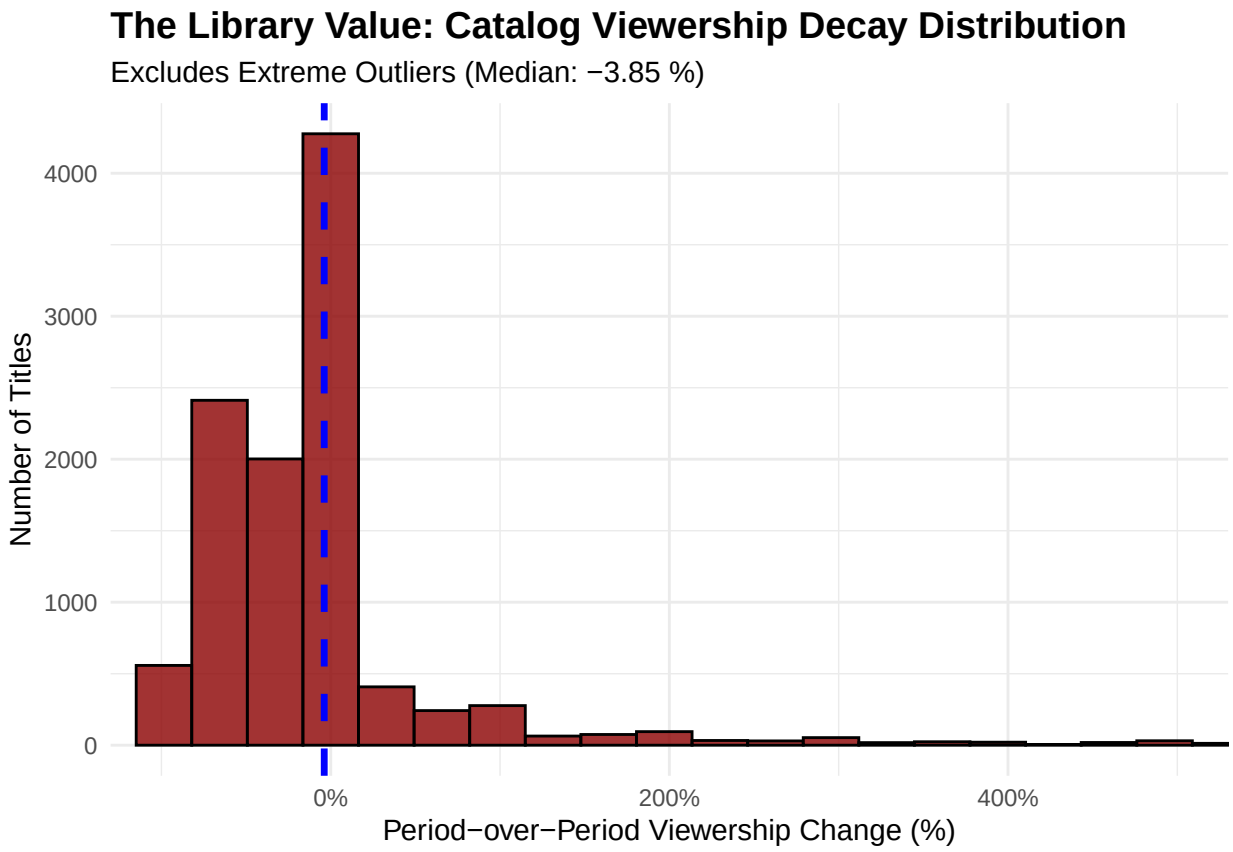
Table 7: 5 records

title	format_type	h2_2023_views	h1_2024_views
“Sr.”	Movie	200000	100000
#Alive	Movie	4600000	4300000
#FriendButMarried	Movie	100000	100000
#FriendButMarried 2	Movie	100000	100000
#Manhole	Movie	300000	200000

```
query6 <- "
WITH MovieViews AS (
    -- Aggregate movie views using Unix start_date to split the periods
    SELECT
        m.title,
        'Movie' AS format_type,
        SUM(CASE WHEN v.start_date < 1704067200000 THEN v.views ELSE 0 END) AS h2_2023_views,
        SUM(CASE WHEN v.start_date >= 1704067200000 THEN v.views ELSE 0 END) AS h1_2024_views
    FROM movie m
    JOIN view_summary v ON m.id = v.movie_id
    GROUP BY m.title
),
ShowViews AS (
    -- Aggregate TV show views using Unix start_date to split the periods
    SELECT
        t.title,
        'TV Show' AS format_type,
        SUM(CASE WHEN v.start_date < 1704067200000 THEN v.views ELSE 0 END) AS h2_2023_views,
        SUM(CASE WHEN v.start_date >= 1704067200000 THEN v.views ELSE 0 END) AS h1_2024_views
    FROM tv_show t
    JOIN season s ON t.id = s.tv_show_id
    JOIN view_summary v ON s.id = v.season_id
    GROUP BY t.title
),
CombinedViews AS (
    -- Stack the two datasets together
    SELECT * FROM MovieViews
    UNION ALL
    SELECT * FROM ShowViews
)
-- Filter strictly for titles that generated views in BOTH periods
SELECT * FROM CombinedViews
WHERE h2_2023_views > 0 AND h1_2024_views > 0;
"
drop <- dbGetQuery(con, query6)

decay <- drop %>%
  mutate(
    decay_rate = ((h1_2024_views - h2_2023_views) / h2_2023_views) * 100
  )
netflix_median_decay <- median(decay$decay_rate)
```

```
ggplot(decay, aes(x = decay_rate)) +
  geom_histogram(fill = "darkred", color = "black", bins = 1000, alpha = 0.8) +
```

```
  geom_vline(aes(xintercept = median(decay_rate)), linetype = "dashed", color = "blue", size = 1.2) +
  scale_x_continuous(labels = function(x) paste0(x, "%")) +

  coord_cartesian(xlim = c(-100, 500)) +

  labs(
    title = "The Library Value: Catalog Viewership Decay Distribution",
    subtitle = paste("Excludes Extreme Outliers (Median:", round(netflix_median_decay, 2), "%)"),
    x = "Period-over-Period Viewership Change (%)",
    y = "Number of Titles"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14)
  )
```




```
ggsave("rq6.png", width = 6, height = 4, dpi = 300)

outliers <- decay %>% filter(decay_rate > 500) %>% arrange(desc(decay_rate)) %>% select(title, h2_2023_
head(outliers)
```

```
##             title h2_2023_views h1_2024_views decay_rate
## 1        Shaitaan        100000      32800000      32700
## 2     City Hunter        100000      29500000      29400
## 3         Players        200000      58800000      29300
## 4      Spellbound        100000      29100000      29000
## 5 Despicable Me 3        500000     135100000      26920
```

```
## 6          Justice         100000       23900000       23800
```

Q7 global vs us

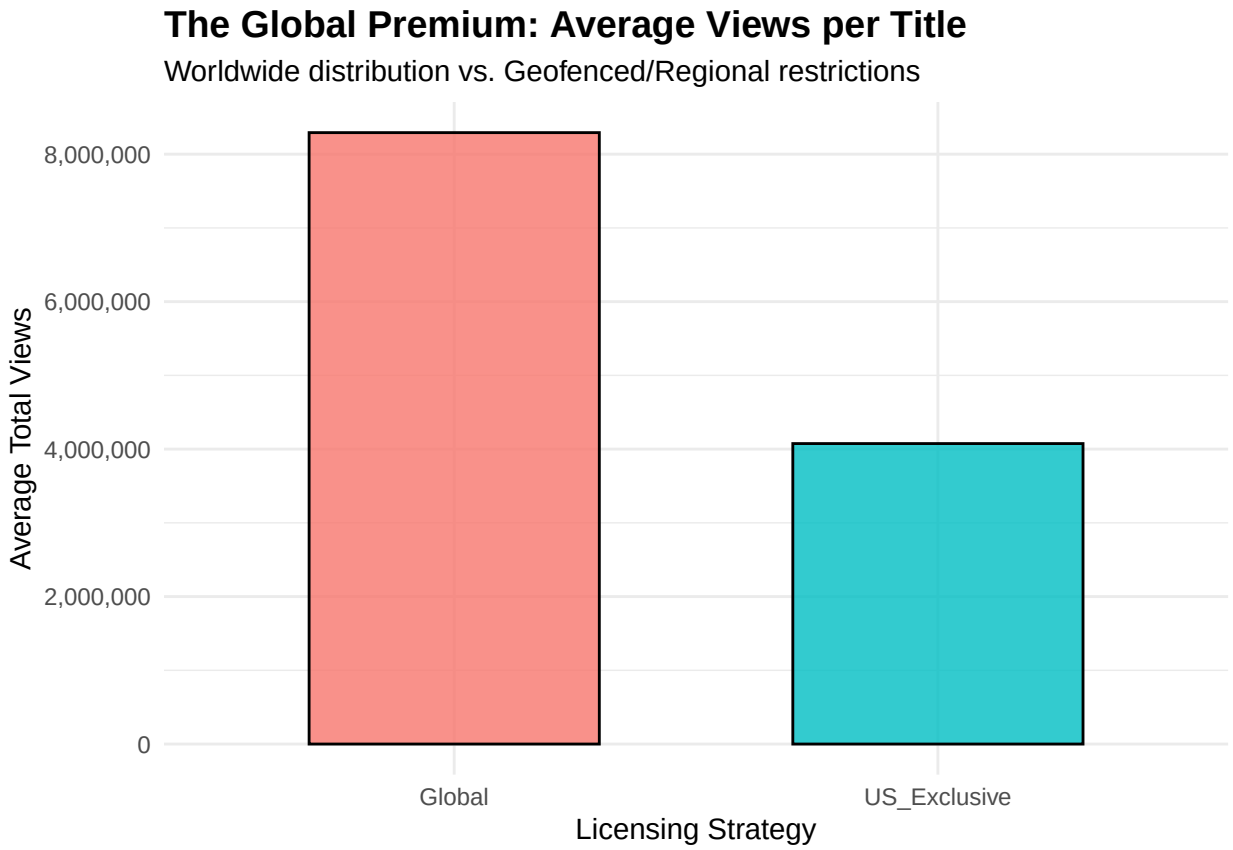
```sql
WITH AllTitles AS (
    SELECT id AS title_id, title, available_globally FROM movie
    UNION ALL
    SELECT id AS title_id, title, available_globally FROM tv_show
),
ChartedViews AS (
    SELECT
        v.movie_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    WHERE v.movie_id IS NOT NULL
    GROUP BY v.movie_id
    UNION ALL
    SELECT
        s.tv_show_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    JOIN season s ON v.season_id = s.id
    GROUP BY s.tv_show_id
),
TitleTotals AS (
    SELECT
        CASE WHEN a.available_globally = 1 THEN 'Global' ELSE 'US_Exclusive' END AS licensing_model,
        c.total_views
    FROM AllTitles a
    JOIN ChartedViews c ON a.title_id = c.title_id
)
SELECT
    licensing_model,
    COUNT(*) AS total_titles_charted,
    AVG(total_views) AS average_views
FROM TitleTotals
GROUP BY licensing_model;
```

Table 8: 2 records

licensing_model	total_titles_charted	average_views
Global	7409	8292496
US_Exclusive	19023	4075046

```sql
query7 <- "WITH AllTitles AS (
    -- Step 1: Combine all titles and their global status
    SELECT id AS title_id, title, available_globally FROM movie
    UNION ALL
    SELECT id AS title_id, title, available_globally FROM tv_show
),
ChartedViews AS (
    -- Step 2: Calculate the total views for each specific title
```

```
    SELECT
        v.movie_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    WHERE v.movie_id IS NOT NULL
    GROUP BY v.movie_id
    UNION ALL
    SELECT
        s.tv_show_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    JOIN season s ON v.season_id = s.id
    GROUP BY s.tv_show_id
),
TitleTotals AS (
    -- Step 3: Match the total views to the licensing model
    SELECT
        CASE WHEN a.available_globally = 1 THEN 'Global' ELSE 'US_Exclusive' END AS licensing_model,
        c.total_views
    FROM AllTitles a
    JOIN ChartedViews c ON a.title_id = c.title_id
)
-- Step 4: Calculate the final means directly in SQL
SELECT
    licensing_model,
    COUNT(*) AS total_titles_charted,
    AVG(total_views) AS average_views
    -- Adding ROUND(AVG(total_views), 0) can clean up the decimal places if you prefer
FROM TitleTotals
GROUP BY licensing_model;"
global_v_us <- dbGetQuery(con, query7)

ggplot(global_v_us, aes(x = licensing_model, y = average_views, fill = licensing_model)) +
  geom_bar(stat = "identity", color = "black", alpha = 0.8, width = 0.6) +
  scale_y_continuous(labels = comma) +
  labs(
    title = "The Global Premium: Average Views per Title",
    subtitle = "Worldwide distribution vs. Geofenced/Regional restrictions",
    x = "Licensing Strategy",
    y = "Average Total Views",
    fill = "Model"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 14), legend.position = "none")
```




```
ggsave("rq7.png", width = 6, height = 4, dpi = 300)
```

Q8 TV or movies preffered globally

```
WITH GlobalMovies AS (
    SELECT id AS title_id, title, 'Movie' AS format_type
    FROM movie
    WHERE available_globally = 1
),
GlobalShows AS (
    SELECT id AS title_id, title, 'TV Show' AS format_type
    FROM tv_show
    WHERE available_globally = 1
),
GlobalTitles AS (
    SELECT * FROM GlobalMovies
    UNION ALL
    SELECT * FROM GlobalShows
),
TitleViews AS (
    SELECT
        movie_id AS title_id,
        SUM(views) AS total_views
    FROM view_summary
    WHERE movie_id IS NOT NULL
    GROUP BY movie_id
```

```
    UNION ALL
    SELECT
        s.tv_show_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    JOIN season s ON v.season_id = s.id
    GROUP BY s.tv_show_id
)
SELECT
    g.title,
    g.format_type,
    t.total_views
FROM GlobalTitles g
JOIN TitleViews t ON g.title_id = t.title_id
ORDER BY total_views DESC
LIMIT 5;
```

Table 9: 5 records

title	format_type	total_views
Lift	Movie	529000000
Bridgerton	TV Show	529000000
Squid Game	TV Show	465600000
The Beautiful Game	Movie	393700000
Stranger Things	TV Show	393700000

```
query8 <- "
WITH GlobalMovies AS (
    -- Get only movies with worldwide rights
    SELECT id AS title_id, title, 'Movie' AS format_type
    FROM movie
    WHERE available_globally = 1
),
GlobalShows AS (
    -- Get only TV shows with worldwide rights
    SELECT id AS title_id, title, 'TV Show' AS format_type
    FROM tv_show
    WHERE available_globally = 1
),
GlobalTitles AS (
    -- Stack the global titles together
    SELECT * FROM GlobalMovies
    UNION ALL
    SELECT * FROM GlobalShows
),
TitleViews AS (
    -- Calculate total views for Movies
    SELECT
        movie_id AS title_id,
        SUM(views) AS total_views
    FROM view_summary
    WHERE movie_id IS NOT NULL
    GROUP BY movie_id
```

```
    UNION ALL
    -- Calculate total views for TV Shows (aggregating all seasons)
    SELECT
        s.tv_show_id AS title_id,
        SUM(v.views) AS total_views
    FROM view_summary v
    JOIN season s ON v.season_id = s.id
    GROUP BY s.tv_show_id
)
-- Join the global titles to their view counts
SELECT
    g.title,
    g.format_type,
    t.total_views
FROM GlobalTitles g
JOIN TitleViews t ON g.title_id = t.title_id;
"
preference <- dbGetQuery(con, query8)

wilcox_export <- wilcox.test(total_views ~ format_type, data = preference)
print(wilcox_export)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  total_views by format_type
## W = 5718760, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
summary_stats <- preference %>%
  group_by(format_type) %>%
  summarize(
    median_views = median(total_views),
    mean_views = mean(total_views)
  )
print(summary_stats)
```

```
## # A tibble: 2 x 3
##   format_type median_views mean_views
##   <chr>              <dbl>      <dbl>
## 1 Movie            1600000   7879781.
## 2 TV Show          2500000   8591228.
```

```
dbDisconnect(con)
```