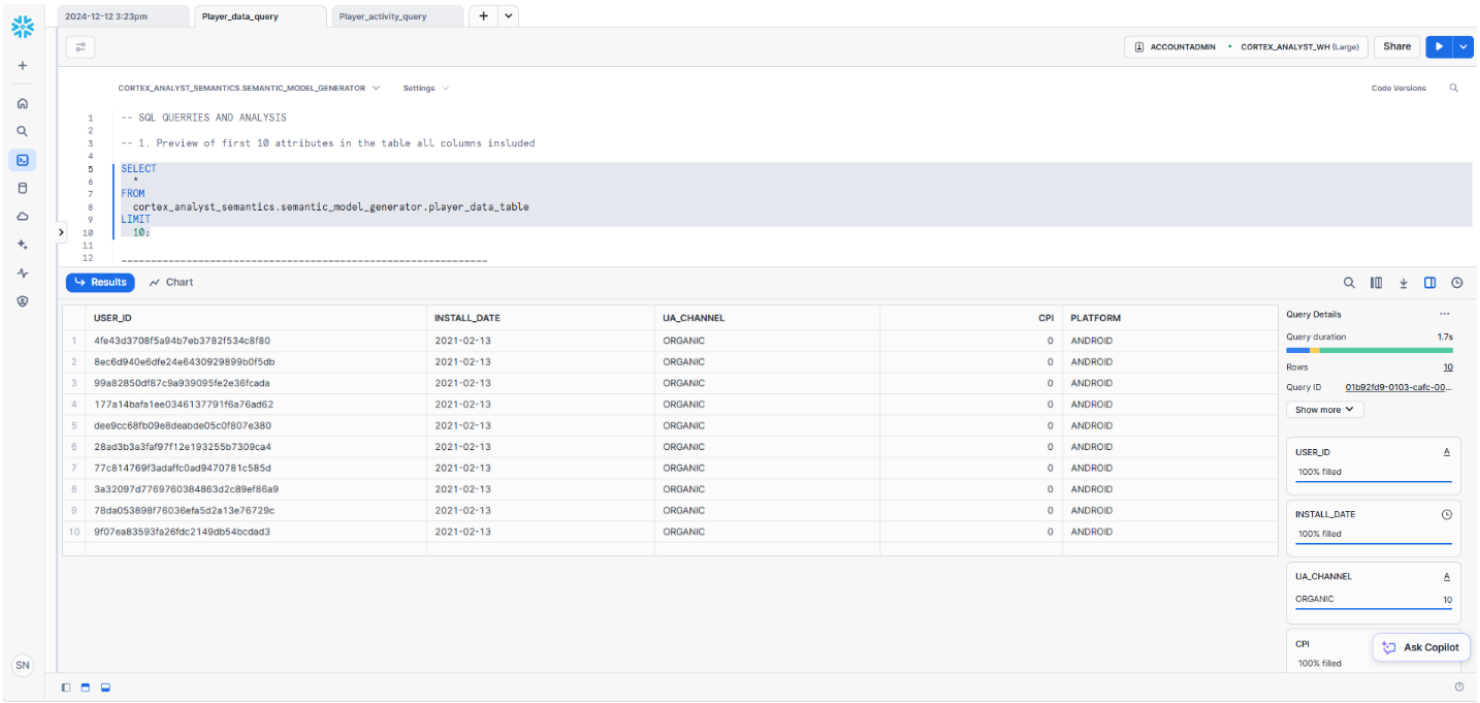
SQL Queries and Analysis in Snowsight SQL Worksheet

Query 1: Preview of first 10 attributes of the table player_data

```
SELECT
  *
FROM
  cortex_analyst_semantics.semantic_model_generator.player_data_table
LIMIT
  10;
```

Result: In the right pane of results preview you can go deeper in query details such as query duration, statistics about column data, as well as preview the chart in the second tab (starting date 2021-01-01, ending date 2021-12-31)



Query 2: Count of players by platform (device)

	NUMBER_OF_PLAYERS	PLATFORM
0	32,940	ANDROID
1	1,930	IOS

Result: 94% of the players installed the game using Android device

Chart the results

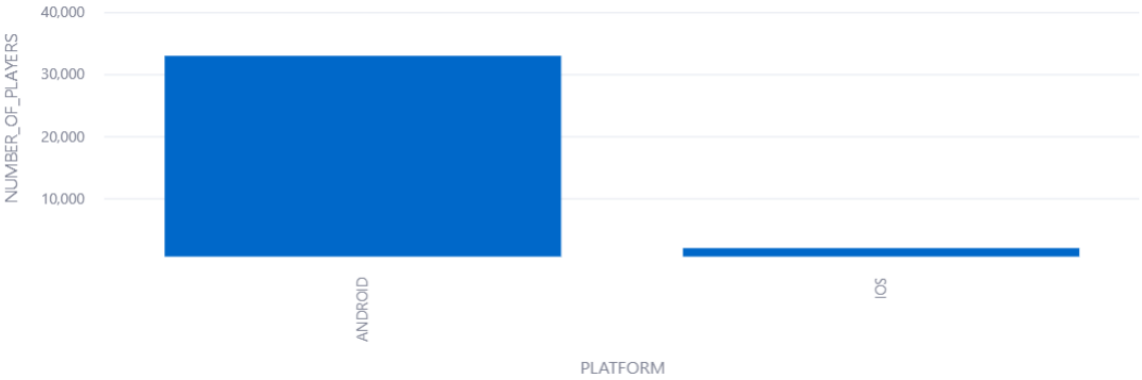
```
import pandas as pd
import streamlit as st

from snowflake.snowpark.context import get_active_session
```

```
session = get_active_session()

my_df = cell6.to_pandas()   #previuos cell into pandas
st.subheader(" ❄ chart - Players by platform ")
st.bar_chart(my_df, x='PLATFORM', y='NUMBER_OF_PLAYERS')
```

❄ chart - Players by platform



Query 3: How much the players pay for installation per channel? Organic means ‘free, unpaid” - no costs for installation

```
WITH cost_and_players_per_platform AS (
   SELECT
      COUNT (user_id) AS players_count, ua_channel, SUM (cpi) AS total_cpi
   FROM
      cortex_analyst_semantics.semantic_model_generator.player_data_table
   GROUP BY
      UA_CHANNEL
   )
SELECT
   ua_channel, players_count, total_cpi
FROM
   cost_and_players_per_paltform
ORDER BY
   total_cpi DESC;
```

Result: The players who used unity ads as acquisition channel for installing the game spent the most

	UA_CHANNEL	PLAYERS_COUNT	TOTAL_CPI
0	UNITY ADS	19,065	47,498
1	IRONSOURCE	3,262	5,023
2	GOOGLE	196	161
3	APPLE SEARCH ADS	2	66
4	ORGANIC	12,345	0

Query 4: How many players are “organic”?

```
WITH _players_type AS (
SELECT
   CASE
      WHEN cpi = 0 THEN 'organic'
      ELSE 'paid'
   END AS player_type,
   COUNT (user_id) AS players_count,
   SUM (cpi) AS total_cpi
FROM
   cortex_analyst_semantics.semantic_model_generator.player_data_table
GROUP BY
   player_type)
SELECT
   player_type, players_count, total_cpi
FROM _players_type;
```

Result: 35% of the players are “free” players (installed the game without any cost)

	PLAYER_TYPE	PLAYERS_COUNT	TOTAL_CPI
0	organic	12,434	0
1	paid	22,436	52,748

Query 5: How many installs are new per month?

```
WITH monthly_installs_and_diff AS
(
   SELECT
      LEFT(TO_CHAR(install_date, 'Month'), LEN(TO_CHAR(install_date, 'Month')) - 2) AS month_per_year,
      COUNT (user_id) as players
   FROM
      cortex_analyst_semantics.semantic_model_generator.player_data_table
   GROUP BY
      LEFT(TO_CHAR(install_date, 'Month'), LEN(TO_CHAR(install_date, 'Month')) - 2)
    )
SELECT
   month_per_year,
   SUM (players) AS total_players,
   total_players - AVG(total_players) OVER() AS diff_from_avg,
   (total_players - AVG(total_players) OVER())/ AVG(total_players) OVER() AS pct_diff_from_avg

FROM
   monthly_installs_and_diff
GROUP BY
   month_per_year
ORDER BY
   diff_from_avg;
```

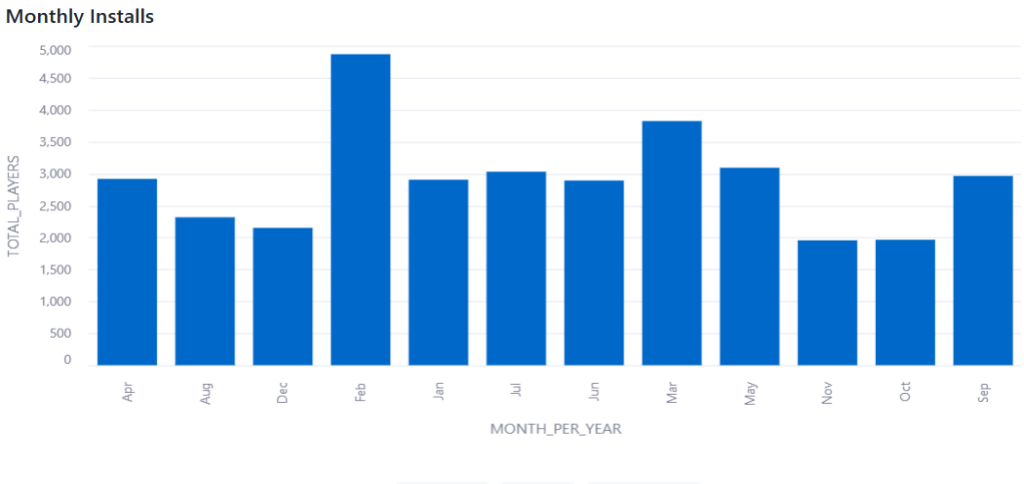
Result: February leads with highest number of installs differing from average installs by 992, in spite of November which has the lowest number of installs differing by 950

	MONTH_PER_YEAR	TOTAL_PLAYERS	DIFF_FROM_AVG	PCT_DIFF_FROM_AVG
0	Nov	1,955	-950.833	-0.3272
1	Oct	1,964	-941.833	-0.3241
2	Dec	2,150	-755.833	-0.2601
3	Aug	2,316	-589.833	-0.203
4	Jun	2,891	-14.833	-0.0051
5	Jan	2,904	-1.833	-0.0006
6	Apr	2,917	11.167	0.0038
7	Sep	2,963	57.167	0.0197
8	Jul	3,029	123.167	0.0424
9	May	3,091	185.167	0.0637

Chart the query

```
my_df = cell16.to_pandas()

import altair as alt
# Chart the data
st.subheader(" Monthly Installs ")
chart = alt.Chart(my_df).mark_bar().encode(
    x='MONTH_PER_YEAR', y='TOTAL_PLAYERS', tooltip=['MONTH_PER_YEAR', 'sum(TOTAL_PLAYERS)',
'PCT_DIFF_FROM_AVG']
).properties( width=600, height=400
).configure_axis( labelFontSize=12, titleFontSize=14
)
st.altair_chart(chart, use_container_width=True)
```



Query 6: Preview of first 10 attributes of the table player_activity

```
SELECT
  *
FROM
  "CORTEX_ANALYST_SEMANTICS"."SEMANTIC_MODEL_GENERATOR"."PLAYER_ACTIVITY_TABLE"
LIMIT
  10;
```

Result: In the right pane of results preview you can go deeper in query details such as query duration, statistics about column data, as well as preview the chart in the second tab (starting date 2021-01-01, ending date 2021-12-31)

	USER_ID	LOGIN_DATE	SESSIONS	TOTAL_TIME_SEC	MISSIONS	AD_REVENUE	IAP_REVENUE	TRANSACTIONS
0	7a7f48bde15393d3b9d843	2021-12-22	1	0	0	0	0	0
1	a7d45d3ed402551c403d0e	2021-10-12	1	0	0	0	0	0
2	1da71a2d596d77bdbb5c06	2021-02-05	1	0	0	0	0	0
3	1da71a2d596d77bdbb5c06	2021-09-05	1	0	0	0	0	0
4	9a5ea24b5fdef8a1946efd3	2021-05-14	1	0	0	0	0	0
5	48ddb6a63e2ebaf4e9de13	2021-03-19	1	0	0	0	0	0
6	c501da69054871761582a3	2021-06-10	1	0	0	0	0	0
7	c501da69054871761582a3	2021-05-12	1	0	0	0	0	0
8	c501da69054871761582a3	2021-10-12	1	0	0	0	0	0
9	0d4b41426aaefd5cb639c26	2021-09-03	1	0	0	0	0	0

Query 7: What's the total revenue by month?

```
WITH monthly_revenue AS (
  SELECT
    user_id,
    login_date,
    ad_revenue,
    iap_revenue
  FROM cortex_analyst_semantics.semantic_model_generator.player_activity_table
)
SELECT
    DATE_TRUNC('MONTH', login_date) AS month,
    SUM(ad_revenue) AS total_ad_revenue,
    SUM(iap_revenue) AS total_iap_revenue,
    total_ad_revenue + total_iap_revenue AS total_monthly_revenue
FROM monthly_revenue
GROUP BY
    DATE_TRUNC('MONTH', login_date)
ORDER BY
  month DESC;
```

Result: The total revenue generated from ad revenue and in-apps revenue has its peak in May. It significantly dropped in August. Ad revenue generates 63% of total revenue

	↑ MONTH	TOTAL_AD_REVENUE	TOTAL_IAP_REVENUE	TOTAL_MONTHLY_REVEI
11	2021-01-01	4,423	4,525	8,948
10	2021-02-01	5,279	7,443	12,722
9	2021-03-01	5,989	5,151	11,140
8	2021-04-01	11,864	1,487	13,351
7	2021-05-01	12,380	6,013	18,393
6	2021-06-01	13,333	1,922	15,255
5	2021-07-01	13,472	721	14,193
4	2021-08-01	9,998	995	10,993
3	2021-09-01	12,918	4,444	17,362

Query 8: In average per day, how much time do players spend playing and how many sessions and missions are played?

```
WITH avereges AS (
   SELECT
      login_date,
      (SUM (total_time_sec)/60) / COUNT (user_id) AS min_per_player,
      SUM (sessions) / COUNT (user_id) AS session_per_player,
      SUM (missions) / COUNT (user_id) AS mission_per_player,
      SUM (transactions) / COUNT (user_id) AS tnx_per_player
   FROM
      cortex_analyst_semantics.semantic_model_generator.player_activity_table
   GROUP BY
      login_date
      )
SELECT
   login_date, min_per_player, session_per_player, mission_per_player, tnx_per_player
FROM
  avereges
ORDER BY
   min_per_player DESC
LIMIT 10;
```

Result: Although the number of installs in January is smaller than monthly average, the most time spent (in minutes) playing the game was registered on 23 January

	LOGIN_DATE	MIN_PER_PLAYER	SESSION_PER_PLAYER	MISSION_PER_PLAYER	TNX_PER_PLAYER
0	2021-01-23	580,829.0121	1.9218	0.0318	0
1	2021-07-01	567,713.2476	2.0182	0.561	0
2	2021-08-01	529,461.7404	1.9145	0.1842	0
3	2021-01-20	507,248.3946	1.8849	0.1123	0
4	2021-10-01	499,625.0576	2.2568	0.2297	0
5	2021-03-01	490,109.2109	2.4512	0.2698	0
6	2021-01-14	486,097.5181	1.9816	0.1496	0
7	2021-04-01	482,516.0674	2.3545	0.2592	0
8	2021-06-01	466,697.1189	2.2702	0.5681	0
9	2021-02-01	464,910.2527	2.3155	0.5149	0

Query 9: Which players played the most missions and made the most transactions?

```
WITH stats_per_player AS (

    SELECT
        user_id, SUM(missions) AS total_missions, SUM(transactions) AS total_tnx
    FROM
        cortex_analyst_semantics.semantic_model_generator.player_activity_table
    GROUP BY
        user_id
    )
SELECT
    user_id, total_missions, total_tnx
FROM
    stats_per_player
ORDER BY
    (total_tnx, total_missions) DESC
LIMIT 10;
```

Result: We can see that max number of transactions made by user is 3 and is not depending or correlated with the number of missions played

	USER_ID	TOTAL_MISSIONS	TOTAL_TNX
0	a956bf0d69554116e28060eec46089a6	109	3
1	2be74d6551dbe2341f4e46205be29543	87	3
2	135c606f88a2f75913c3313ecbb4f982	141	2
3	0c6ae951d1a16abddf88d60663eab365	83	2
4	2cdd5450c43901cbe559ce6d47bf1c3b	63	2
5	98876eb703a16a60b46525de4367c478	15	2
6	d9f36334882f2f01e6646174e956aedd	0	2
7	1319e958a503025f67bb4d8a5a45dd83	0	2
8	2111c598dc95175bb6284cc142f23ae6	1,144	1
9	3b37185771cb4c8cea9c48241089418e	988	1

Query 10: Calculate the average revenue generated per in-app purchases

```
WITH profit_per_tnx AS (

    SELECT
        login_date,
        AVG(iap_revenue / transactions) AS avg_revenue_per_transaction
    FROM
        cortex_analyst_semantics.semantic_model_generator.player_activity_table
    WHERE
        transactions > 0
    GROUP BY
        login_date
```

```
    )

SELECT
    login_date, avg_revenue_per_transaction
FROM
    profit_per_tnx
ORDER BY
    avg_revenue_per_transaction DESC
LIMIT 20;
```

Result: Since the revenue generated from in-apps purchases refers to the actual attractiveness of the game, I used it to analyze how much profit does it make. Not to surprise, when the number of installs was the highest (in February), the highest iap-revenue occurred.

	LOGIN_DATE	AVG_REVENUE_PER_TRANSACTION
0	2021-02-21	2,284
1	2021-02-19	2,134
2	2021-04-18	786
3	2021-11-03	785
4	2021-01-31	780
5	2021-06-30	780
6	2021-03-14	775
7	2021-01-29	771
8	2021-11-09	769
9	2021-05-27	768

Joins

Query 11: How many hours, missions and sessions are played per channel?

```
WITH activity_per_channel AS (
    SELECT

        dt.ua_channel AS channel,
        COUNT(DISTINCT ac.user_id) AS players,
        SUM(ac.sessions) AS total_sess,
        SUM(ac.missions) AS total_miss,
        SUM(ac.total_time_sec) /86400 AS total_h
    FROM
        cortex_analyst_semantics.semantic_model_generator.player_data_table dt
    INNER JOIN
        cortex_analyst_semantics.semantic_model_generator.player_activity_table ac
    ON
        dt.user_id = ac.user_id
    GROUP BY
```

```
        dt.ua_channel
)
SELECT
    channel, players, total_sess, total_miss, total_h
FROM
    activity_per_channel

ORDER BY
    channel;
```

Result: As for total cost installation, the unity ads channel has the highest number of players, as well as sessions, missions played and total hours spent playing

	CHANNEL	PLAYERS	TOTAL_SESS	TOTAL_MISS	TOTAL_H
0	APPLE SEARCH ADS	2	16	5	949.1294
1	GOOGLE	196	942	2,588	25,491.3029
2	IRONSOURCE	3,262	24,728	63,541	2,218,319.1115
3	ORGANIC	12,345	115,054	521,117	6,537,488.8685
4	UNITY ADS	19,064	159,858	762,511	8,624,154.8579

Query 12: How much the total revenue from paid players differs from the free ones?

```
WITH rev_per_player_type AS (
  SELECT
    CASE
      WHEN dt.cpi = 0 THEN 'organic'
      ELSE 'paid'
    END AS player_type,
    ac.ad_revenue AS ad_revenue,
    ac.iap_revenue AS iap_revenue,
    ac.ad_revenue + ac.iap_revenue AS total_revenue
  FROM
    cortex_analyst_semantics.semantic_model_generator.player_activity_table AS ac
    JOIN
        cortex_analyst_semantics.semantic_model_generator.player_data_table AS dt
    ON dt.user_id = ac.user_id
)
SELECT
  player_type,
  SUM(ad_revenue) AS total_ad_rev,
  SUM(iap_revenue) AS total_iap_rev,
  SUM(total_revenue) AS total_revenue
FROM
  rev_per_player_type
GROUP BY
  player_type;
```

Result: The organic (free) players made 6% less total profit then the paid ones, but participate with significant 75% in total iap-revenue. The paid players impact on total revenue from ad-revenue with 63%

	PLAYER_TYPE	TOTAL_AD_REV	TOTAL_IAP_REV	TOTAL_REVENUE
0	paid	72,287	9,906	82,193
1	organic	42,241	30,729	72,970

Query 13: Calculate total, min, max and avg revenue per month, providing the dates on which min and max sales occurred

```
WITH monthly_revenue AS (
      SELECT
        DATE_PART(MONTH, login_date) AS month,
        SUM(ad_revenue + iap_revenue) AS total_revenue,
        MAX(ad_revenue + iap_revenue) AS max_revenue,
        MIN(ad_revenue + iap_revenue) AS min_revenue,
        AVG(ad_revenue + iap_revenue) AS avg_revenue
      FROM cortex_analyst_semantics.semantic_model_generator.player_activity_table
        GROUP BY
          DATE_PART(MONTH, login_date)
      )
      SELECT
        mr.month,
        mr.total_revenue, mr.max_revenue, mx.login_date AS max_revenue_date,
        mr.avg_revenue, mr.min_revenue, mn.login_date AS min_revenue_date

      FROM monthly_revenue AS mr
      JOIN cortex_analyst_semantics.semantic_model_generator.player_activity_table AS mx
        ON mr.month = DATE_PART(MONTH, mx.login_date)AND mr.max_revenue = mx.ad_revenue + mx.iap_revenue
        JOIN cortex_analyst_semantics.semantic_model_generator.player_activity_table AS mn
        ON mr.month = DATE_PART(MONTH, mn.login_date)AND mr.max_revenue = mn.ad_revenue + mn.iap_revenue

      ORDER BY mr.month DESC ;
```

Result: The highest revenue of 18393 was made on 03 May 2021 and the lowest income of 8948 on 13 January 2021

	MONTH	TOTAL_REVENUE	MAX_REVENUE	MAX_REVENUE_DATE	AVG_REVENUE	MIN_REVENUE	MIN_REVENUE_DATE
0	12	11,786	753	2021-12-27	1.0707	0	2021-12-27
1	11	10,523	785	2021-11-03	1.0916	0	2021-11-03
2	10	10,497	749	2021-10-06	0.9758	0	2021-10-06
3	9	17,362	753	2021-09-24	1.1963	0	2021-09-24
4	8	10,993	752	2021-08-14	0.8326	0	2021-08-14
5	7	14,193	721	2021-07-31	0.9604	0	2021-07-31
6	6	15,255	780	2021-06-30	1.1187	0	2021-06-30
7	5	18,393	784	2021-05-03	1.3159	0	2021-05-03
8	4	13,351	789	2021-04-18	0.9033	0	2021-04-18
9	3	11,140	775	2021-03-14	0.6356	0	2021-03-14

General Insights and Recommendations

User Engagement

1. **High Engagement Periods**:
    - **Insight**: Players tend to engage the most in March (high login activity) and January (longest session times).
    - **Actionable Takeaway**: Plan special events, promotions, or new content releases during these months to capitalize on high engagement periods and sustain player interest.
- **Device and Platform Preferences**:
    - **Insight**: A significant majority of players (94%) are using Android devices. However, iOS players, despite being a smaller segment, generate higher in-app purchase revenue per transaction.
    - **Actionable Takeaway**: Prioritize optimization and new feature development for Android users while creating exclusive offers for iOS users to leverage their higher spending potential.
- **Mission-Based Engagement**:
    - **Insight**: Higher income is generated after players complete the fifth mission, and longer play sessions occur in January.
    - **Actionable Takeaway**: Develop more engaging and rewarding missions beyond the fifth mission to maintain player interest. Encourage players to participate in long play sessions through events or challenges.
- **Revenue Growth Strategies**
- **Channel Optimization**:
    - **Insight**: Players acquired through unity ads spend the most, and organic players contribute significantly to in-app purchase revenue.
    - **Actionable Takeaway**: Increase investment in unity ads for acquiring high-value players. Also, develop strategies to convert organic players into paying customers through personalized promotions and incentives.
- **Seasonal Revenue Trends**:
    - **Insight**: Total revenue peaks in May and drops significantly in August. Revenue per install does not necessarily correlate with the number of installs.
    - **Actionable Takeaway**: Plan revenue-boosting activities, such as special in-game events and promotions, during low-revenue months like August. Ensure that high-install months are also optimized for revenue generation through attractive in-app purchase offers.
- **In-App Purchase Focus**:
    - **Insight**: Ad revenue contributes 63% of total revenue, and iOS players generate higher average in-app purchase revenue per transaction.
    - **Actionable Takeaway**: Optimize ad placements to maximize ad revenue. Additionally, create targeted in-app purchase campaigns for iOS players and other high-value segments to boost revenue.
- **Performance Improvements**
- **Platform-Specific Optimization**:
    - **Insight**: Android is the dominant platform, but iOS players have higher spending per transaction.
    - **Actionable Takeaway**: Ensure the game performs well across a wide range of Android devices. Implement specific optimizations for iOS to enhance user experience and capitalize on their higher spending potential.
- **Engagement and Retention**:
    - **Insight**: Despite high engagement in January and March, the number of installs is lower in January. No significant difference in engagement between paid and free users.
    - **Actionable Takeaway**: Focus on retention strategies that encourage players to return and stay engaged, such as regular content updates, loyalty rewards, and community-building activities. Use insights from high-engagement periods to replicate success in lower-engagement months.
- **Data-Driven Improvements**:
    - **Insight**: No correlation between time spent playing and revenue generation.
    - **Actionable Takeaway**: Focus on other metrics, such as in-app purchases and ad interactions, for performance improvements. Use data-driven approaches to identify and address specific areas that directly impact revenue and user satisfaction.