Mobile Game App Analysis Using ❄ Snowflake SQL, Cortex Analyst and Semantic Model Generator

If you want to "talk" to your data, use Snowflake Cortex Analyst for delivering high text-to-SQL accuracy powered by state-of-the-art LLM Llama and semantic-model-generator repository to generate and curate a semantic model for Snowflake Cortex Analyst.

Project Overview:

This project focuses on analyzing mobile game applications using advanced data analytics tools, including Snowflake Cortex Analyst and an OSS semantic model generator tool. The goal is to derive actionable insights from player (user) behavior, revenue and game performance metrics.

Objectives:

- **User Behavior Analysis:** Understand how users interact with the game, identify patterns, and predict user engagement.
- **Revenue Optimization:** Analyze in-app purchase data to identify opportunities for increasing revenue.
- **Performance Metrics:** Monitor and optimize game performance to ensure a seamless user experience

Technologies Used:

- **Snowsight:** Snowflake User interface (notebooks, worksheets, apps, dashboards)
- **Snowflake Cortex Analyst:** Leveraging the power of Snowflake for scalable data storage, processing, and advanced analytics.
- **Semantic Model Generator:** Creating semantic models to interpret and analyze unstructured data.
- **VS Data Wrangler:** For data preprocessing.
- **Snow SQL:** For querying data within Snowflake.
- **Snow CLI:** For managing Snowflake resources and executing SQL commands from the command line.

Prerequisites:

- **Snowflake Account:** I used Snowflake Trial Account in AWS eu-central-1 region and ACCOUNTADMIN role to create, alter, drop
- **Python 3.11**
- **Git:** installed
- semantic-model-generator repository (and its dependencies) for creating the needed warehouse, database and stage
- **Snowflake Extension for Visual Studio Code –** setting Snowflake default connection and using. toml config file for loading configuration settings

Data Sources as local .txt files (initially provided locally for Power BI analysis):

1. **Player Activity Data**:
    - user_id: Every player has a unique ID assigned upon installing the game.
    - login_date: The relevant date on which a user was active in the game.
    - sessions: The number of monthly sessions a player made that day.
    - total_time_sec: The total number of seconds spent by the user in the game that day.
    - missions: The number of missions played by the user in the game that day.

- ad_revenue: Sum of revenues generated from Ads in the game.
- iap_revenue: Sum of revenues generated from in-Apps.
- transactions: The number of In-App purchases a player made.

2. **Player Install Data**:
    - user_id: Every player has a unique ID assigned upon installing the game.
    - install_date: The date of game install, marking the day the player installed the game.
    - ua_channel: User Acquisition channel (e.g., organic = the player installed the game directly, not through an advertisement).
    - cpi: Cost per Install, which is the cost of acquiring the player. If the value is 0, the player's UA ch annel is organic.
    - platform: The device the player uses, either iOS (Apple) or GP (Android).

Methodology:

1. **Data Cleaning (Wrangling):** Ensure data quality by handling missing values, errors, mixed data types, date formats, outliers, drop columns for storage and performance optimization. When analyzing cleaned data in Snowflake later on, I couldn't but notice that either I should have kept some of the columns (ex. total revenue for not having to calculate it all over again in CTE's or month index columns for sorting by) or I should have completely recreated the date columns in both files (as I later came across a difficulty to trunk the date month). The **Data Wrangling** file is provided for further details.

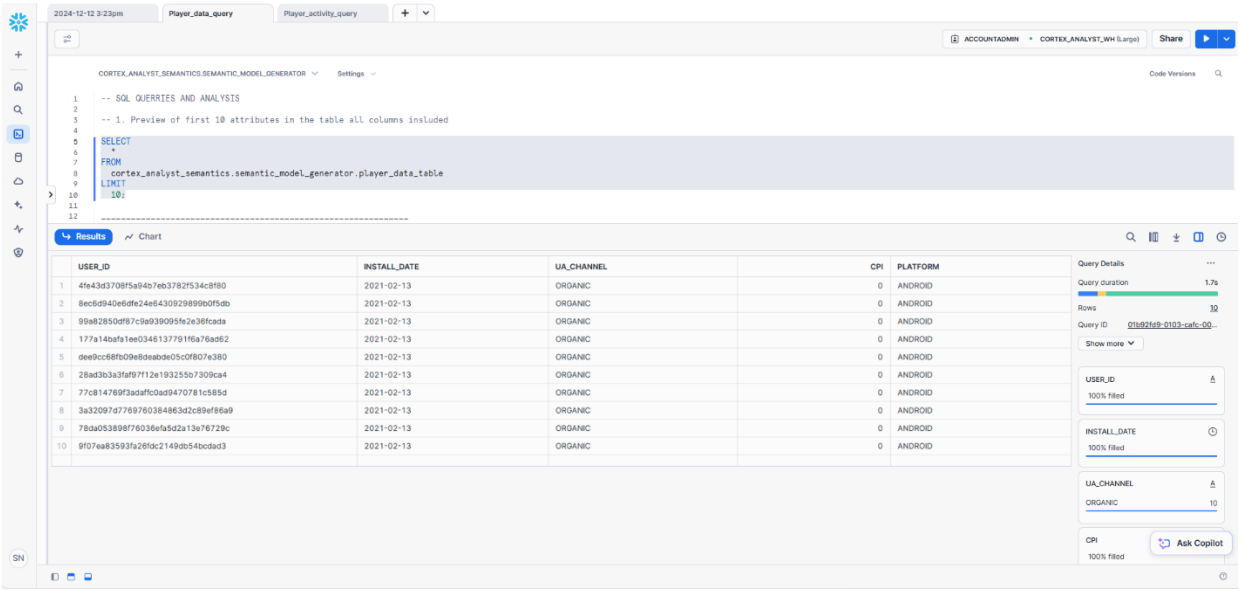
2. **Running the tool to generate the semantic model:** Run the Streamlit application provided in the git repository by running the **app.py as main file for starting Semantic Model Generator** through VS terminal:

    ```
    streamlit run app.py
    ```

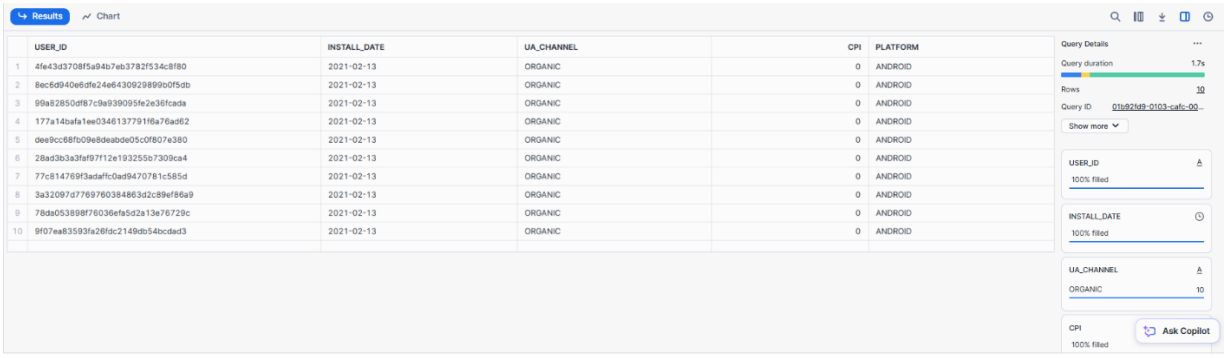
    The semantic model file in. yml format (**player_data.yml**) is the key that unlocks Cortex Analyst's power. This YAML file dictates the tables, columns, etc. that you can use in order to run queries that answer natural-language questions. Previously, I had worked with Power BI and Excel only and was quite amazed by the fact that all the dimensions, measures, time measures and relationships between tables can be defined in a .yml file. Initially, the generator is building it, but you have to fill out all the empty fields as filters, descriptions of the logical columns, synonyms as well as to create relationships between tables and logical columns as measures to ensure validation of the model.

3. **Data Ingestion in Snowsight:** Uploading data sources as .csv files to Semantic Model Generator Stage (and later regret it since Data Wrangler provides option for exporting and saving pandas dataframes in parquet, which can reduce the size of the file for up to 10 times, especially needed when handling large datasets) and loading the data into Snowflake tables with the inherited schema from the data source files (some of it is viewable in Snowsight_Sem_model_generator_Warehouse, Database, Stage video

4. **Data Analysis with SQL:** summarize the main characteristics of the data in SnowSQL worksheets writing SQL queries. The queries and results from them are provided in **SQL Queries and Analysis** file.



(If I've had run them in VS, I would now have them exported in HTML file, then I wouldn't have to create the pdf file to show the results of the SQL queries, provided in this repo. But, that's how you learn, right?)



5. **Data Analysis with natural language:** Prompting Cortex Analyst to provide insights of the same data:

- Prompt:

What's the biggest difference in days between the install date and the login date? On which device was made? By which user?

This is our interpretation of your question:

What's the biggest difference in days between the install date and the login date? On which device was it made? By which user?

```
WITH __player_activity_table AS (
  SELECT
```

What is your question?

- Cortex Analyst Answer:

What's the biggest difference in days between the install date and the login date? On which device was it made? By which user?

```
WITH __player_activity_table AS (
  SELECT
    user_id,
    login_date
  FROM cortex_analyst_semantics.semantic_model_generator.player_
), __player_data_table AS (
  SELECT
    user_id,
    platform,
    install_date
```

USER_ID	PLATFORM	DAYS_DIFFERENCE
64bfa653cfa1929c012b2b99c0889192	IOS	360

(more details about the prompts and answers are provided in **Cortex Analyst Semantics – Semantic Model Generator Prompts and Responses** file)

Challenges Faced:

- **Data Preprocessing:** The upmost challenging task. Columns with mixed data types
- **Streamlit App running:** Code missing for catching responses. After generating the .yml file, I tried to ask the Cortex Analyst a question, but it failed. I had to add the snippet to iteration.py file in:

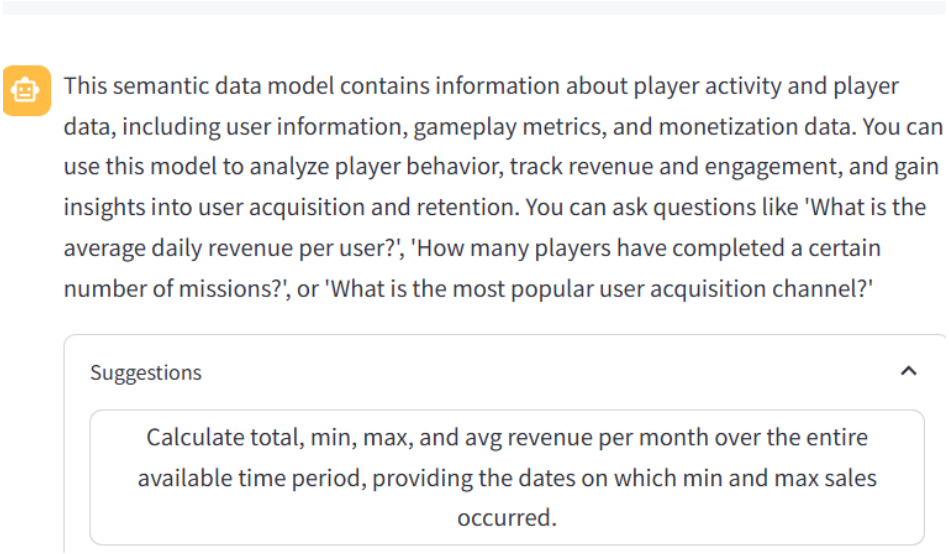
  ```
  def process_message(_conn: SnowflakeConnection, prompt: str) -> None:
  ```

  to catch the responses correctly.

```
# Check if response is a string and convert if necessary
            if isinstance(response, str):
                try: response = json.loads(response) # Convert JSON string to dictionary
                except json.JSONDecodeError: st.error("Error decoding JSON: " + response)
                return

            # Ensure response is a dictionary
            if isinstance(response, dict) and "message" in response and "content" in
response["message"]:
                content = response["message"]["content"]
            else:
                st.error(f"Unexpected response structure: {response}")
                return
```

- **Cortex Analyst** failures to respond: When prompted complex and vague questions, Cortex couldn't respond correctly, avoiding or exiting with errors. However, when I tried and wrote the solution in a SQL worksheet and offer the solution in editing mode, the LLM already knew half the answer in the first attempt reassigning the initial prompt and **totally nailed it on the second run using my solution in the notebook** (see attached video Snowflake_Generator + failure overcome), then used the prompt as suggestion



Insights:

- ✓ **User Engagement**:

- **Insight**: Players show varying engagement levels across different months and platforms. High engagement is seen in specific months, such as January and March, with Android being the dominant platform.
- **Actionable Takeaway**: Leverage peak engagement periods by launching new features, events, or updates during these times. Tailor marketing and development efforts primarily toward Android users, while still capitalizing on the high spending potential of iOS players.

- ✓ **Revenue Growth Strategies**:

- **Insight**: Revenue trends and user spending behaviors vary significantly across acquisition channels and time periods.
- **Actionable Takeaway**: Optimize revenue by focusing on high-performing acquisition channels like unity ads and leveraging the high spending of iOS users. Implement targeted promotions and in-game events during low-revenue periods to boost earnings.

- ✓ **Performance Improvements**:

- **Insight**: The game's performance in terms of user activity and revenue generation is influenced by multiple factors, including engagement metrics, platform usage, and acquisition channels.
- **Actionable Takeaway**: Continuously monitor and analyze player behavior to identify opportunities for improving game performance. Focus on enhancing the user experience for the dominant Android user base, while also catering to the high-value iOS segment.

Conclusion:

This project demonstrates the power of combining Snowflake Cortex AI and semantic modeling to derive mean ingful insights from mobile game data. By understanding user behavior and optimizing revenue and performan ce, game developers can make data-driven decisions to improve their products.

Lesson learnt when dealing with AI tools:

– be careful when you're giving prompts, by making mistakes you are helping LLM to learn upon your own mistakes, even when using different UI features (which was beneficial for this analysis),

- context is lost after about four or five side questions,

- the LLM created complex queries step-by-step, with first CTE's used unnecessarily just for selecting data and without aggregations, making them large and sometimes inexecutable.

Note: Copilot was used to get the initial guidelines for creating this introduction