# Project Summary: KSA Brand Licensing Analysis Tool

Here’s the detailed report on my project:

1. Project Goal & Intention

My main goal was to build a data-driven tool to help evaluate the licensing potential of various brands within the Saudi Arabian (KSA) market. The intention was to go beyond simple popularity and provide insights into:

* **Public Hype:** How much are people talking about the brand? (Volume)
* **Market Presence:** How established is the brand in the local e-commerce space (Amazon.sa)? (Saturation)
* **Perceived Quality:** What do consumers think of the products already available? (Ratings)
* **Product Popularity:** How much traction do existing products have? (Review Counts)

We aimed to combine these factors into a quantifiable **Suitability Score** and present the findings in an easy-to-understand format, initially targeting a command-line tool and later evolving into a GUI application.

2. Data Collection & Tools Used

We targeted a strategic mix of 11 brands relevant to KSA and the licensing conference context:

* **Conference Related:** Fanatics, Lazurde, Vacheron Constantin, PIF, Saudi Aramco, Riyadh Season
* **KSA Giants (Baseline):** Al-Hilal, Al-Nassr, STC
* **Potential "Gems":** KSA Anime, KSA One Piece

My data collection involved several steps and iterations:

* **Initial Approach (Manual Scraping - Failed):**
  + Tried using ntscraper/snscrape for Twitter and pytrends for Google Trends. **Failed** due to blocking and instability.
  + Tried using requests and BeautifulSoup for Noon/Amazon. **Failed** due to blocking and rapidly changing website structures.
* **Revised Approach (Apify API - Successful):**
  + **Twitter (X):** Successfully used the xtdata/twitter-x-scraper Apify Actor via the apify-client library in Python (1\_scrape\_hype.py). Collected **5852 tweets** (across English/Arabic runs) for my target brands.
    - **Challenge:** Discovered later that the **tweet text and engagement counts were missing/empty** in the saved data, and **dates were unparseable**. This severely limited downstream analysis.
  + **Google Trends:** Attempted using apify/google-trends-scraper. **Failed** as the actor required a paid subscription beyond the free trial/platform credits. Data source was abandoned.
  + **Amazon.sa:** After several attempts with incorrect or non-functional Actor IDs (apify/amazon-scraper, apify/amazon-crawler), we successfully used junglee/Amazon-crawler via apify-client (2\_scrape\_ecommerce\_apify.py). Collected **209 product listings** (up to 25 per brand searched).
  + **Noon.sa:** Attempted using apify/web-scraper and buseta/noon-advanced-scraper. **Failed** as both actors were either blocked or unable to find products (likely outdated selectors). Data source was abandoned.
* **Database:** All successfully collected raw data (tweets without text/stats, Amazon product listings) was stored in an **SQLite** database (licensing\_data.db) managed via Python's sqlite3 library.

3. Data Cleaning & Feature Engineering (ETL)

This phase was performed in a Jupyter Notebook (EDA\_and\_Modeling.ipynb) using **Pandas**:

* **Loading:** Data loaded from SQLite into DataFrames (df\_tweets, df\_products).
* **Cleaning:**
  + Converted product price, avg\_rating, num\_reviews to numeric types.
  + Filled missing avg\_rating and num\_reviews with 0.
  + **Challenge:** Attempted to parse tweet\_date. Discovered all dates were invalid/empty strings (''). This prevented time-series analysis. Rows with invalid dates were *kept* to preserve tweet volume counts.
* **Feature Engineering:**
  + **Attempted engagement\_score & daily\_hype\_score:** **Failed** because raw like/retweet counts were missing from df\_tweets, and dates were invalid for resampling.
  + **Calculated tweet\_volume:** Used df\_tweets.groupby('brand\_name').size() as a proxy for Hype Level.
  + **Calculated market\_saturation:** Used df\_products.groupby('brand\_name').size().
  + **Calculated avg\_perceived\_quality:** Averaged avg\_rating per brand.
  + **Calculated avg\_num\_reviews:** Averaged num\_reviews per brand.
  + **Combined Metrics:** Merged all calculated features into df\_combined\_metrics.
  + **Normalization:** Created normalized versions (0-100 scale) of these metrics for scoring and visualization, including log transformation for review counts and inverse scaling for saturation.
  + **Calculated suitability\_score:** Applied weights to the normalized metrics to get a final score per brand.

4. Visualization & Analysis

We used **Matplotlib** and **Seaborn** within the Jupyter Notebook to generate insights:

* **Bar Charts:** Compared brands based on raw tweet\_volume and market\_saturation.
* **Scatter Plot (Hype vs. Saturation):** Visualized brands based on tweet volume vs. product count, using color for average quality and size for average review count. This helped identify potential opportunities (like Riyadh Season) and saturated markets (like Al-Hilal, Lazurde).
* **Radar Chart (in GUI):** Compared a selected brand's normalized metrics (Hype, Quality, Popularity, Low Saturation) against the average across all brands.

5. Final Application: GUI Tool (consultant\_tool.py)

We built a desktop application using **Tkinter**:

* **Functionality:**
  + Loads the final processed data from brand\_metrics\_final.csv.
  + Provides an input field for the user to enter a brand name.
  + Displays a text report showing raw metrics, the calculated suitability\_score, and a qualitative recommendation (High/Moderate/Low Potential).
  + Generates and displays a radar chart comparing the selected brand's profile to the average.
* **Purpose:** To provide a simple, interactive way for a non-technical user (like a CEO at the conference) to quickly get insights on a specific brand based on our analysis.

6. Limitations & Challenges Encountered

* **Data Quality Issues:** The biggest limitation was the missing tweet text, engagement counts, and valid dates in the collected Twitter data. This prevented sentiment analysis, topic modeling, and time-series analysis of hype. The root cause likely lies in how the xtdata/twitter-x-scraper Actor returned data or how our initial script saved it.
* **Scraping Difficulties:** We faced significant challenges scraping Google Trends (paid actor), Noon.sa (blocking/outdated actors), and even Amazon.sa (blocking, input schema complexity). This highlights the difficulty and unreliability of web scraping, especially against sophisticated targets.
* **Simplified Metrics:** Due to data limitations, our "Hype" metric is just tweet volume, not engagement or sentiment. Our "Quality" is just the average star rating, not nuanced review analysis.
* **Scope:** We only looked at Twitter and Amazon.sa. A real-world analysis would include many more platforms (Instagram, TikTok, other e-commerce sites, news mentions).
* **Static Data:** The analysis is based on a snapshot in time. Real brand hype fluctuates constantly.

7. Future Expansion Opportunities

This project provides a solid foundation. To make it truly stand out, we could:

1. **Fix Twitter Data Collection:** Re-run the Twitter scrape (1\_scrape\_hype.py) ensuring the tweet\_content, createdAt, and engagement stats (likeCount, etc.) are correctly retrieved from Apify and saved to the database. This would unlock Sentiment Analysis, Topic Modeling, and Daily Hype tracking.
2. **Scrape Amazon Reviews:** Implement the 3\_scrape\_reviews\_apify.py script (using an appropriate Apify actor like apify/amazon-reviews-scraper) to gather review text. This would enable calculation of the Counterfeit Risk Score and Review Sentiment.
3. **Find a Working Noon Scraper:** Periodically check the Apify Store or other platforms for updated/working Noon.sa scrapers to add that data source.
4. **Integrate TikTok Data:** Investigate robust methods (likely paid APIs or advanced scraping techniques) to incorporate TikTok hype signals.
5. **Refine Suitability Score:** Make the scoring more sophisticated, potentially using machine learning or more complex weighting based on product category.
6. **Add Financial Data Layer:** For brands like Aramco/PIF, integrate basic financial indicators or investment news sentiment as alternative metrics.
7. **Automate & Deploy:** Set up the scrapers to run automatically on a schedule (e.g., using Apify Scheduler or GitHub Actions) and potentially deploy the analysis/GUI tool as a web application.

*Overall, despite the data challenges, we successfully built a pipeline to collect data from difficult sources, performed meaningful feature engineering, generated insightful visualizations, and created a functional GUI tool prototype.*