Applications of Big Data

Subject: Data Visualization

Group: Shawn Duhamel, Gabriel George, Noah Goulin

Documentation

Summary

The use case

Global architecture

Compilation process

ETL with Dataiku DSS

Data Extraction

Data Preparation

Data Loading

KPIs Computation

Data Visualization with Power BI

Importing Data

Data Model Enhancement

Dashboard Creation

About code quality

About generative AI

The use case

In this use case, our main goal is to craft an all-encompassing report that taps into the potential of Key Performance Indicators (KPIs) extracted from a designated dataset. This report is designed to go beyond the surface, delving into rich analytics and visualizations. Our focus is on uncovering patterns and insights related to the most listened-to tracks and albums, understanding listener behavior intricacies, and establishing rankings based on both historical trends and weekly variations. Through this analytical lens, our objective is to distill meaningful information that aids in a deeper understanding of the dataset and facilitates data-driven decision-making.

Global architecture

In the initial phase of our workflow, we laid a solid foundation by locally installing a PostgreSQL database within a Docker container, leveraging the efficiency and flexibility of this technology. This foundational step established a well-structured and easily accessible database environment.

Following the local installation of the PostgreSQL database, we seamlessly integrated Dataiku DSS, the free edition of the Data Science Studio, into our workflow as the ETL (Extract, Transform, Load) tool. This strategic integration allowed us to extract, prepare, and load data with precision, with Dataiku DSS effectively communicating with the locally hosted PostgreSQL database within the Docker container. KPI calculations were then performed directly within the PostgreSQL database, enhancing both computational efficiency and data integrity.

To bring the key performance indicators to life, Power BI was chosen for its robust visualization capabilities. Establishing a direct link between Power BI and the PostgreSQL database, we created a dynamic and interactive report. This report serves as a comprehensive tool to explore and interpret the computed KPIs with clarity and efficiency.

The integration of these tools within the global architecture ensures a seamless and powerful data analytics pipeline from database creation to visualization.

Compilation process

Install Dataiku DSS and Setup PostgreSQL Container

- Download and install Dataiku DSS (Data Science Studio) on your machine.
- Create a Docker container for PostgreSQL with the following specifications:
    - Container Name: "postgres"
    - Database Name: "postgres"
    - Username: "postgres"
    - Port: 5432
- Run the Docker container.

Setup DSS Project

- Download the [DSS Project archive]() and import it in DSS as a new project.
- Build the resulting pipeline.

Setup Power BI Report

- Download the [Power BI Report]().
- Launch the downloaded report.
- Update Data Source Settings:
  - Navigate to "Data source settings" in Power BI.
  - Ensure the server and database settings match the configuration of the PostgreSQL database.
  - Update settings if necessary to establish the connection.
- With the updated data source settings, the Power BI report is now connected to the PostgreSQL database, and the project setup is complete.

ETL with Dataiku DSS

The ETL process within Dataiku DSS is meticulously structured, comprising four key steps to ensure a seamless transition from raw data to actionable insights.

Data Extraction

Raw data, initially in the form of multiple CSV files, is imported into Dataiku DSS as a consolidated folder. Each CSV file encapsulates the listening history of a user, featuring the following details: artist, album, track and datetime. To accomplish this, a Python recipe named *compute_extracted_data* is employed.

Recipe : **compute_extracted_data**

The recipe reads input data from a specified folder, “raw_data”, containing multiple CSV files representing user listening histories. The script iterates through each file, reading its contents into a Pandas DataFrame, where columns are labeled as “artist”, “album”, “track”, and "datetime." The user's name is extracted from the file path and added as a new “usr” column. The individual listening histories are then concatenated into a single DataFrame, “extracted_data_df”. Exception handling is implemented to manage potential errors during the extraction process. Finally, the aggregated data is written to the Dataiku platform as a dataset named ”extracted_data” with the corresponding schema. This recipe essentially consolidates diverse user listening histories into a unified dataset, facilitating subsequent data preparation and analysis steps in the ETL pipeline.

Data Preparation

Following the extraction phase, the data undergoes meticulous preparation, encompassing cleaning and normalization processes. Executed through a dedicated Python recipe named *compute_cleaned_data*, this step ensures that the dataset is refined and standardized for optimal analysis. The focus here is on enhancing data quality and consistency before further integration into the database.

Recipe : **compute_cleaned_data**

The recipe reads input data from the "extracted_data" dataset, which contains user listening histories. It focuses on transforming and cleaning the data, starting with normalizing the "datetime" column to a consistent format. It then identifies and removes instances where the date is incorrect, replacing them with NaN values. It introduces a boolean column, "is_artist_unknown", indicating whether the artist is known or not, based on a custom utility function. Unknown artists labeled as "<Unknown>" are subsequently removed, and the rows are sorted based on user and datetime, with an index column added for reference. The final cleaned dataset is written back to the Dataiku platform as "cleaned_data", maintaining the specified column order. This recipe prepares the data for further analysis and ensures consistency and accuracy in the dataset.

Data Loading

With the prepared data in hand, the next step involves storing it as a structured table within the PostgreSQL database. The Python recipe named *compute_listenings_table* orchestrates this data loading process, seamlessly transferring the refined dataset into the designated database table. This step is pivotal for creating a foundation of organized and accessible data for subsequent analytical procedures.

Recipe : **compute_listenings_table**

The recipe reads input data from the "cleaned_data" dataset within the Dataiku platform to transfer it into a structured table named "listenings" within the PostgreSQL database, ensuring that the meticulously cleaned and formatted information is readily accessible and organized for subsequent analysis and reporting within the broader database environment.

KPIs Computation

Once the data is securely stored in the PostgreSQL database, the final phase revolves around computing Key Performance Indicators (KPIs). This is achieved through the execution of various SQL recipes, each tailored to calculate specific KPIs. The outcome is the generation of new tables within the database, housing the computed KPIs. Here is the list of all the KPIs:

Most diversified listeners of all time

The SQL query aims to generate a ranking of the top 10 most diversified listeners based on the variety of artists they have listened to over time. The query selects the "usr" column as the listener identifier and calculates the count of distinct artists for each listener using the COUNT(DISTINCT artist) aggregation function. The WHERE clause ensures that only records with a non-null value in the "artist" column are considered. The results are grouped by the listener, and the grouping is ordered in descending order based on the count of listened artists. The LIMIT

10 at the end restricts the output to the top 10 listeners with the most diverse range of artists in their listening history. In essence, this query provides valuable insights into the audience's music preferences, highlighting those listeners who have explored a broad spectrum of artists in their listening habits.

Most listened artist per week

The SQL query is designed to identify the most listened artist for each week. It uses a Common Table Expression (CTE) named "ranked_artists" to first rank artists by the count of listenings within each week. The CTE extracts the year and week from the "datetime" column, considers non-null values in the "artist" column, and employs the ROW_NUMBER() window function to assign a rank to each artist within the specified time frame. The main query then selects the year, week, artist, and the count of listenings from the CTE, filtering only those rows where the artist has the top rank (artist_rank = 1) for each week. The final output is an ordered list that reveals, for each week, the artist with the highest number of listenings, providing valuable insights into weekly music preferences and trends.

Number of listened tracks by listener and by artist

The SQL query aims to create a cross-tabulation of listened tracks by both listener ("usr") and artist. It achieves this by counting the occurrences of each unique combination of listener and artist in the "listenings" table. The SELECT statement includes the "usr" and "artist" columns, along with the COUNT(*) function to calculate the number of listened tracks for each combination. The results are grouped by both listener and artist, and the output is ordered first by listener ("usr") and then by the count of listened tracks in descending order. Essentially, the query provides a structured view of how many tracks each listener has listened to from each artist, offering valuable insights into individual listening preferences and artist popularity among the user base.

Biggest listeners of all time

Biggest listeners per week

Most listened album of all time

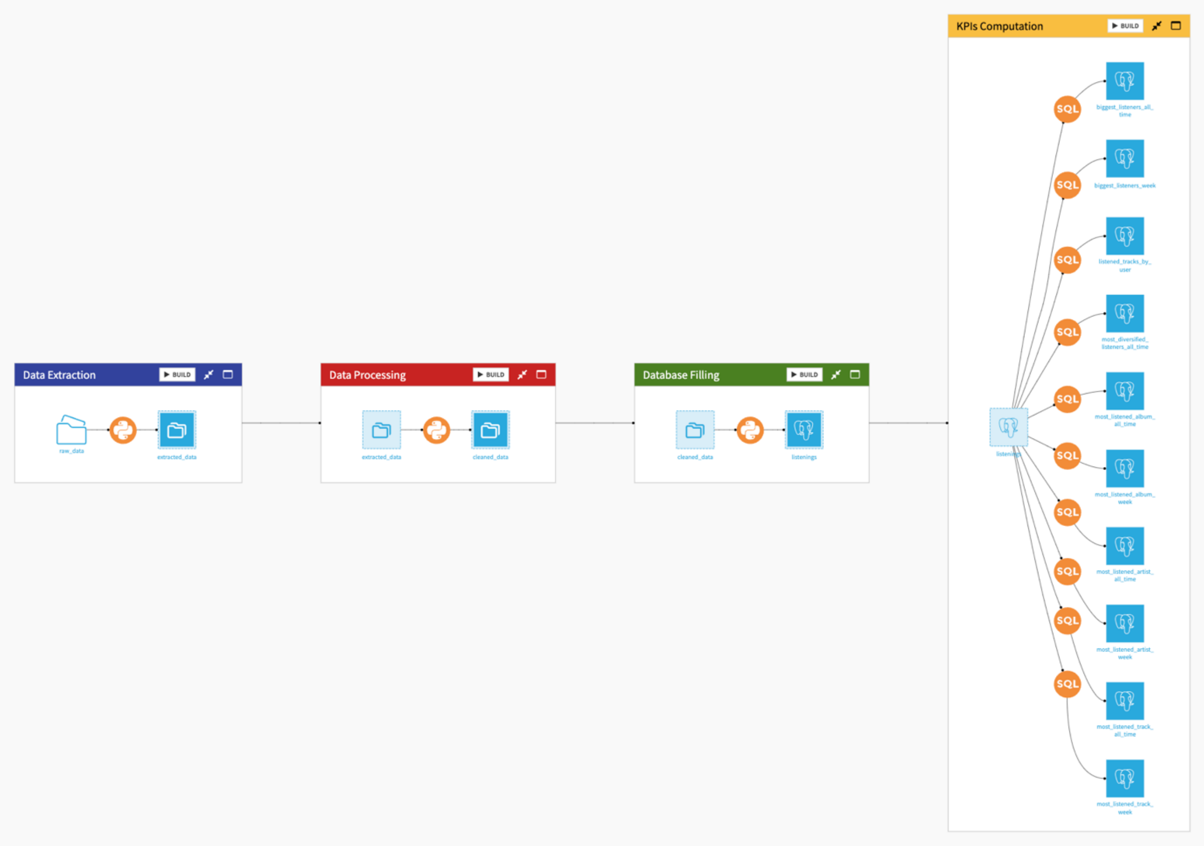
Most listened album per week

Most listened artist of all time

Most listened track of all time

Most listened track per week

This transformation pipeline from raw data to KPIs provides the essential insights needed for the subsequent visualization and reporting stages.



DATAIKU: FLOW

Data Visualization with Power BI

Importing Data

The initial step involves establishing a connection between Power BI and the PostgreSQL database, facilitating the retrieval of tables containing Key Performance Indicators (KPIs). This seamless integration allows for a dynamic and real-time representation of the essential metrics within Power BI.

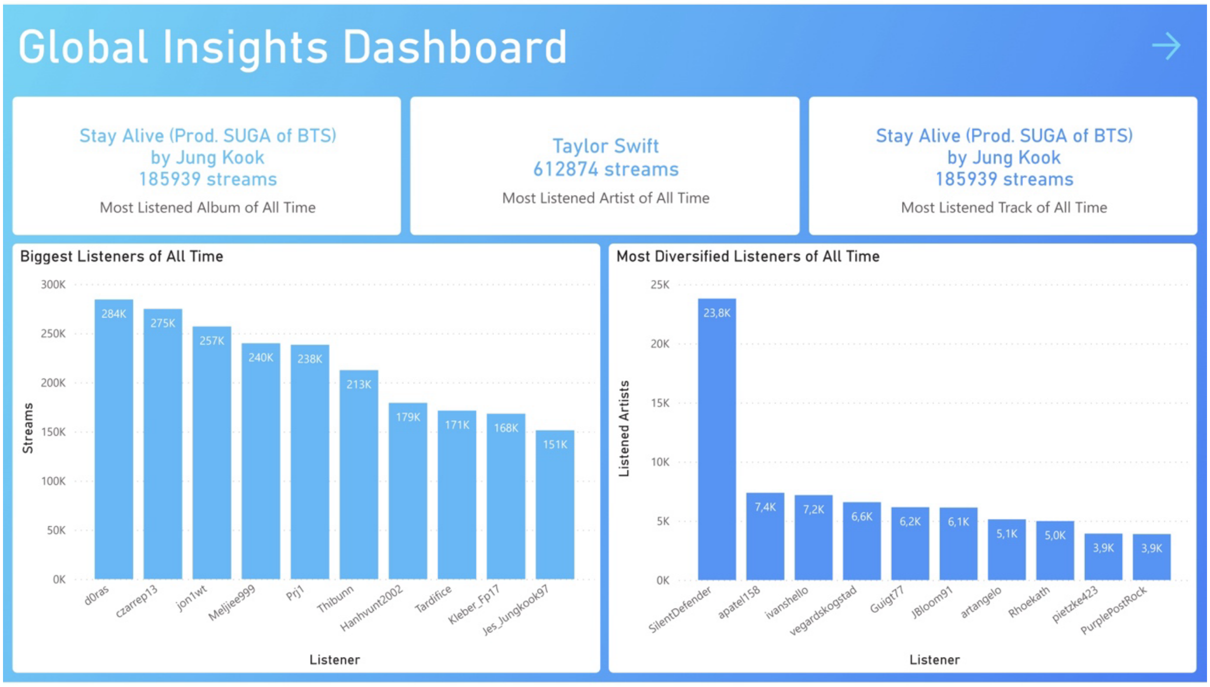
Data Model Enhancement

To enhance the data model, we introduce a new calendar table featuring a specialized year-week column. This strategic addition facilitates the creation of relationships between the calendar table and KPI tables that house weekly statistics. While the KPI tables with time-independent metrics do not require relationships, the calendar table serves as a crucial anchor for time-based analyses, enabling comprehensive insights into weekly trends.

Dashboard Creation

The culmination of the data visualization process manifests in the development of two robust dashboards within Power BI, each designed to provide a unique lens into our Key Performance Indicators. These dashboards go beyond mere visual representation by offering a comprehensive exploration of charts and metrics, delivering a visually intuitive and informative experience.

In particular, we introduce two distinctive dashboards: a Global Insights Dashboard and an Advanced Analytics Dashboard. The Global Insights Dashboard provides a high-level overview, showcasing global trends and aggregated metrics. On the other hand, the Advanced Analytics Dashboard offers a more in-depth analysis, allowing users to filter data dynamically based on weeks and individual listeners. This strategic segmentation ensures that users can tailor their exploration, delving into specific time frames and listener behaviors for a more nuanced understanding of the dataset. Through thoughtful design and strategic layout, both dashboards empower users to effortlessly explore and comprehend the nuances of each KPI, fostering informed decision-making and a deeper understanding of performance metrics.



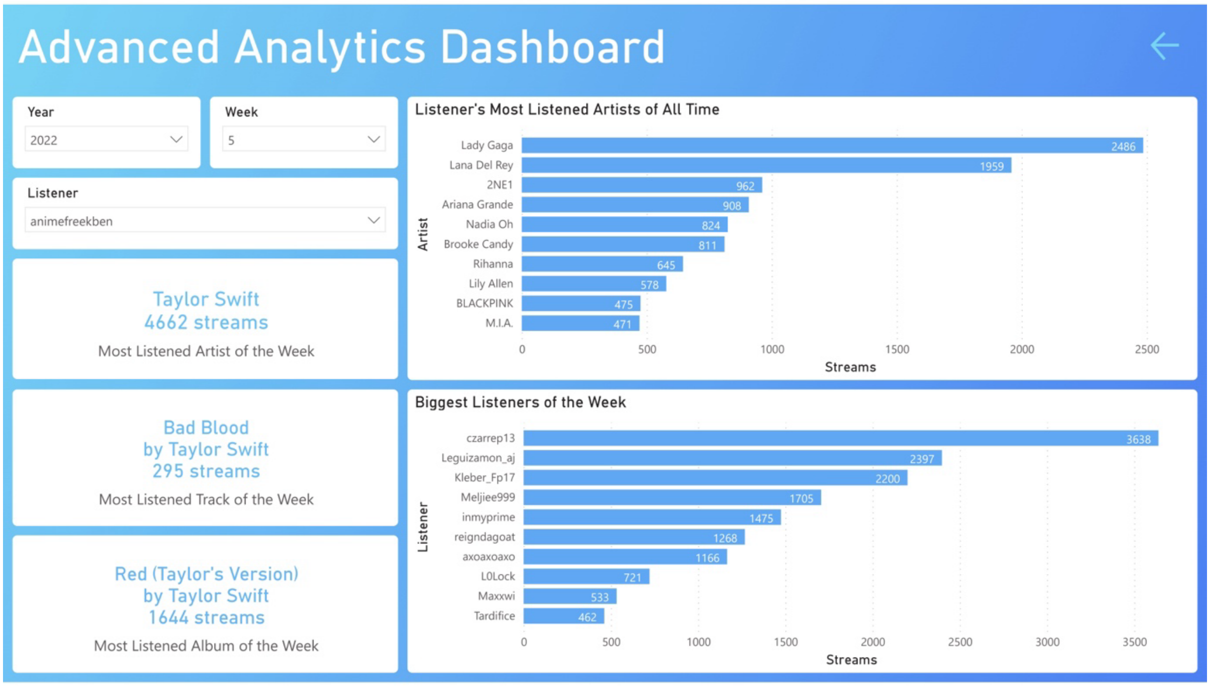
Global Insights Dashboard

Stay Alive (Prod. SUGA of BTS)
by Jung Kook
185939 streams
Most Listened Album of All Time

Taylor Swift
612874 streams
Most Listened Artist of All Time

Stay Alive (Prod. SUGA of BTS)
by Jung Kook
185939 streams
Most Listened Track of All Time

POWER BI REPORT: GLOBAL INSIGHTS DASHBOARD



Advanced Analytics Dashboard

Year: 2022

Week: 5

Listener: animefreekben

Taylor Swift
4662 streams
Most Listened Artist of the Week

Bad Blood
by Taylor Swift
295 streams
Most Listened Track of the Week

Red (Taylor's Version)
by Taylor Swift
1644 streams
Most Listened Album of the Week

Power BI Report: Advanced Analytics Dashboard

About code quality

The quality of the code is commendable across multiple dimensions. Firstly, its repeatability is evident in the script's structured approach to data processing within the Dataiku platform. The code is designed to execute consistently, ensuring reliability and reproducibility of the entire ETL process.

Secondly, the script demonstrates resilience by incorporating robust error handling mechanisms. It effectively manages potential issues during data extraction, handling exceptions to prevent disruptions caused by bad data. This resilience enhances the script's dependability in real-world scenarios where data inconsistencies may arise.

Lastly, the code maintains a high standard of cleanliness and documentation. Clear variable names, modular functions, and comprehensive comments enhance code readability and understanding. This emphasis on clarity not only aids in the current development but also promotes ease of maintenance and collaboration. The documentation further serves as a valuable resource for future users, ensuring that the logic and functionality of the code remain transparent and accessible.

About generative AI

Generative AI was used to make our documentation clearer, help with compatibility issues between tools like Dataiku, GitHub, and PostgreSQL, and provide support for understanding DAX and DSS features. However, we didn't use it for writing actual code.