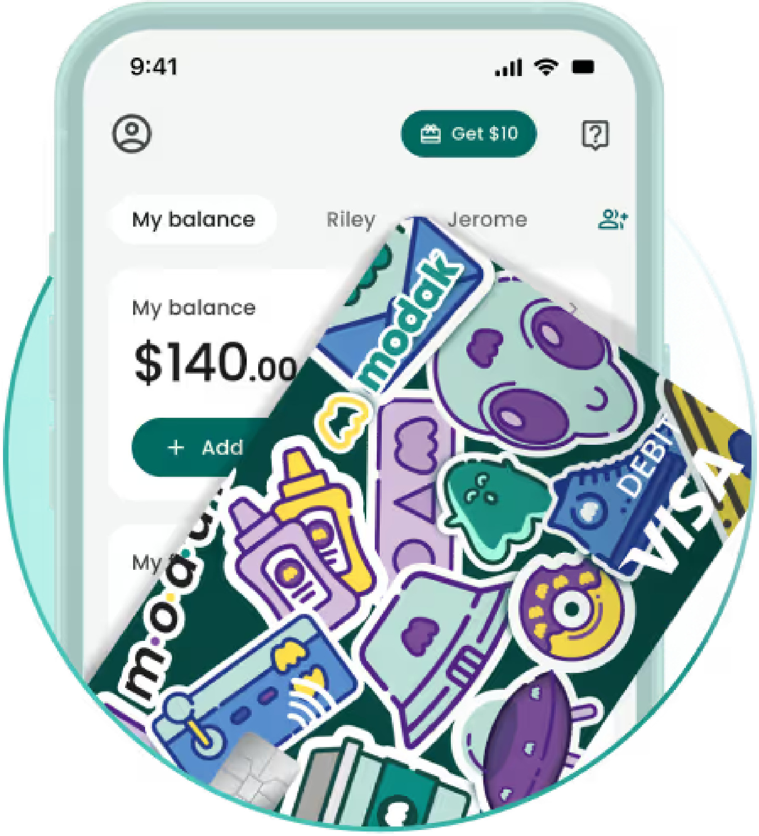
modak

Data Engineering Challenge: Allowance Day Discrepancies



Index

Introduction

The goal of this assessment is to help identify and understand discrepancies in the next_payment_day and payment_date fields across the backend datasets. These fields play a key role in ensuring that recurring allowances are scheduled correctly, and addressing the issues found will be critical for improving the reliability of this feature.

To tackle this challenge, I chose to break down the main problem into smaller, more manageable subproblems. This approach allowed me to focus on specific aspects of the data and analyze each one thoroughly, ensuring no detail was overlooked.

One key step in my process involved making some enhancements to the base datasets. For example, I added a new column to the allowance_events JSON file. By using the event timestamp, allowance frequency, and scheduled day, I calculated the next payment date for each user. This adjustment was essential to align the events data with the backend tables and uncover any inconsistencies in how payment schedules were being generated.

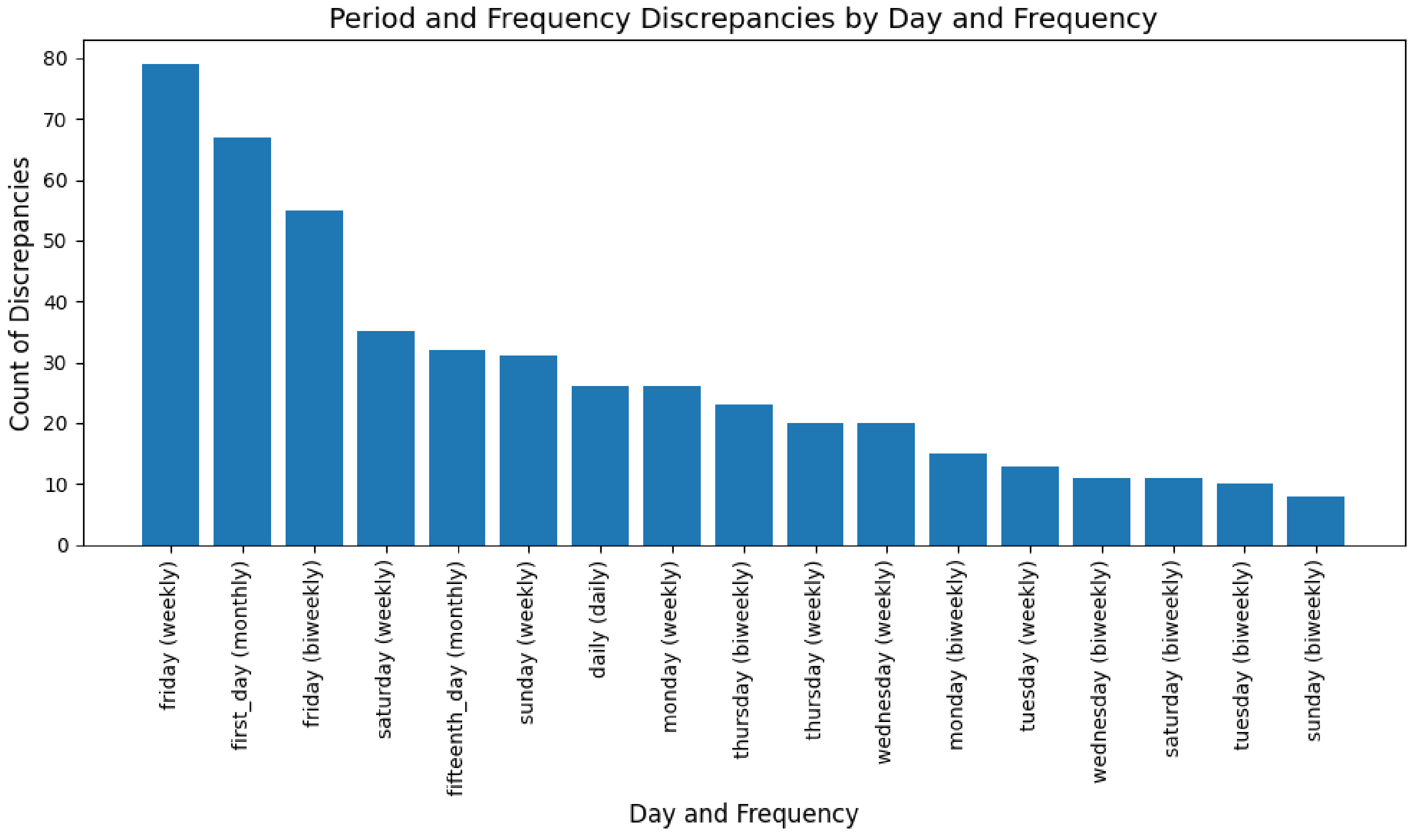
To streamline this entire process, everything presented here was automated through Python scripts. This means that if we ever need to repeat this analysis in the future, we can simply run the Airflow DAG, and the entire workflow, including the generation of this PDF, will be executed automatically. This ensures not only consistency in the results but also significant time savings for any future investigations.

Findings and Insights

Now we are going to review the insights found in the discrepancies datasets that I created in the previous step!

To do so, we will be analysing each dataset on its own:

1. Period and Frequency Discrepancies



Key Findings:

- 453 records where the frequency and day from backend data, does not align with the one in Allowance Events.

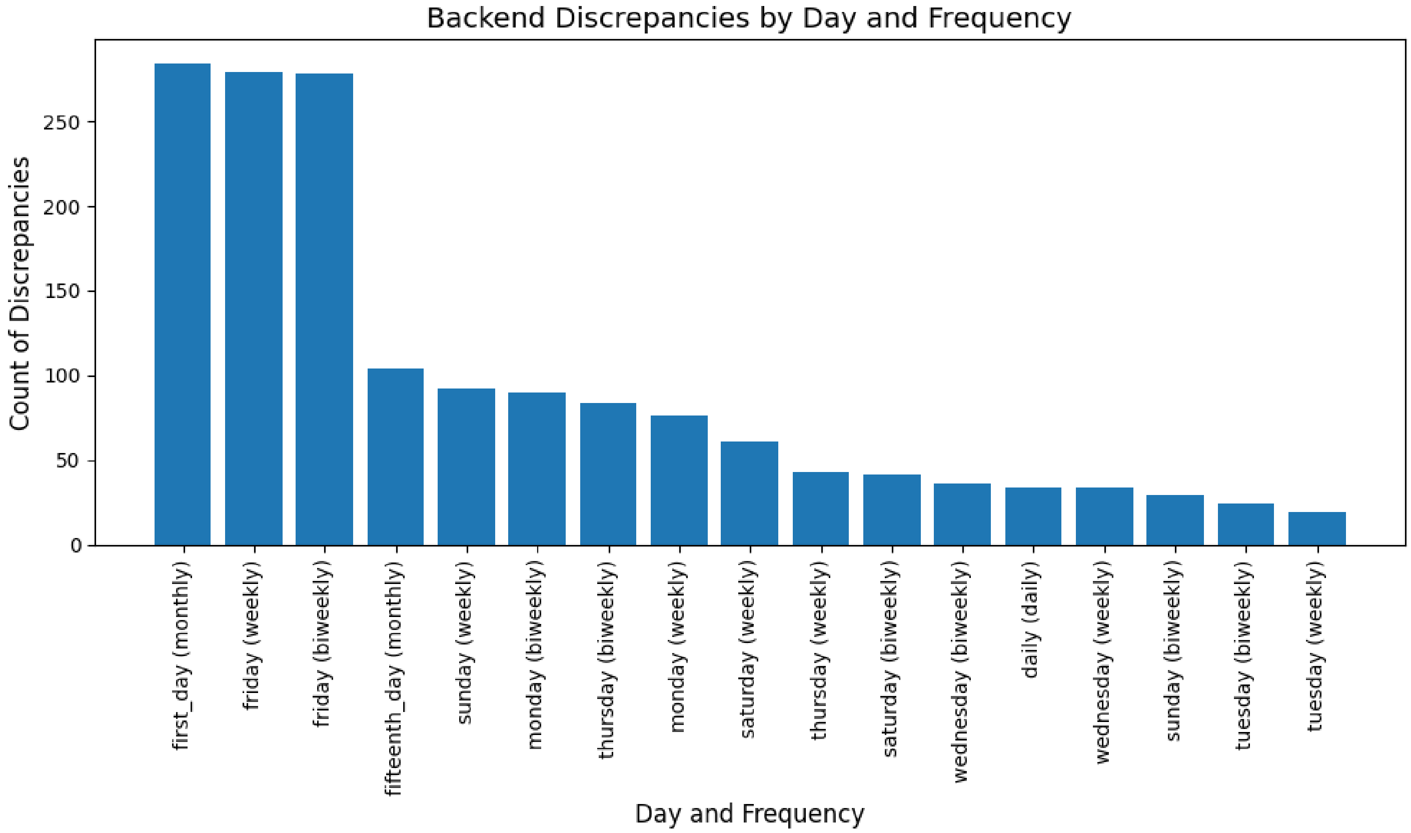
- Most frequent discrepancies occur in:

  - Weekly allowances on Fridays (69 cases).

  - Monthly allowances on the first day (65 cases).

  - Biweekly allowances on Fridays (54 cases).

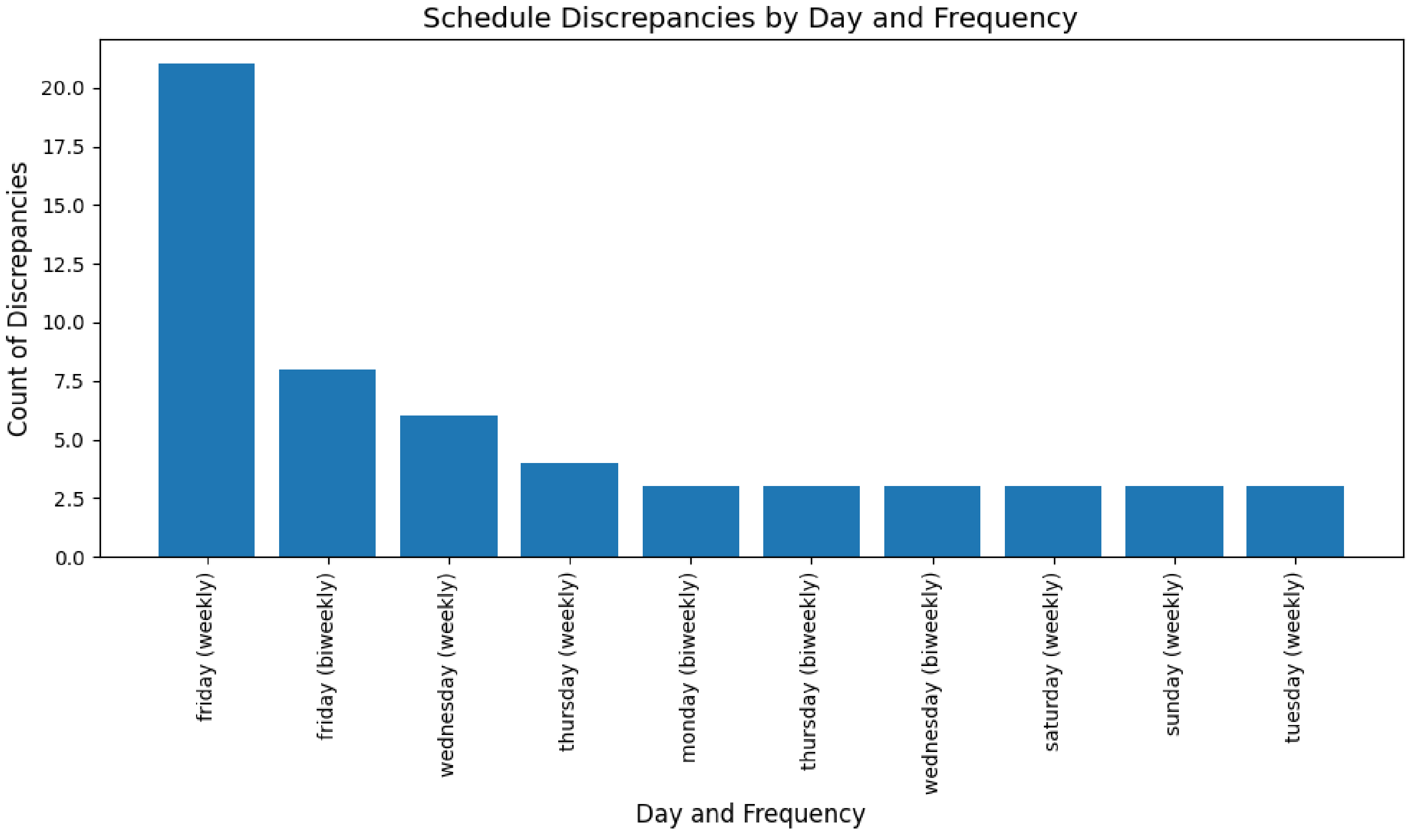
2. Backend Discrepancies



Key Findings:

- 1,608 records with mismatches between `next_payment_day` and calculated values.
- Similar patterns to Period and Frequency Discrepancies but with larger numbers:
  - Monthly allowances on the first day (284 cases).
  - Weekly allowances on Fridays (279 cases).
  - Biweekly allowances on Fridays (278 cases).

3. Schedule Discrepancies



Key Findings:

- 53 records where `payment_date` differs from `next_payment_day`.

- Similiar patterns to the other datasets, with friday being the most common case

Hypotheses and Explanations

We can clearly see that in every single graphic, the most common discrepancies are on Fridays and the first day of the month, so we could consider the following hypotheses:

1. Backend Logic Errors on Recurring Schedules:

- Pattern Observed: Fridays in weekly and biweekly schedules show disproportionately high discrepancies.

- Hypothesis: The backend logic for calculating the `next_payment_day` might have an off-by-one error or issues with day alignment. Weekends (Friday or nearby) could exacerbate this due to special processing for weekends or holidays.

2. Fixed Date Errors for Monthly Allowances:

- Pattern Observed: Monthly schedules on the 'first day' of the month have significant discrepancies.

- Hypothesis: The 'first day' logic might fail when processing near month transitions, especially if the logic doesnt account for varying month lengths or time zones.

3. Mismatches Between Event Logs and Backend Updates:

- Pattern Observed: Backend discrepancies are significantly high

- Hypothesis: Event logs, considered the source of truth, might not synchronize properly with backend updates. Concurrent updates could overwrite correct values, or delayed updates might result in stale `next_payment_day` values.

4. Load and Scaling Issues:

- Pattern Observed: Concentration of discrepancies on popular days like Fridays.

- Hypothesis: Backend processes might face load or scaling issues on these high-frequency update days, leading to timing mismatches or skipped updates.

Recommendations

This recommendations are aimed at addressing the root causes of discrepancies and improving system reliability:

- Investigate Backend Logic: Dive into the code handling of day and frequency calculations, with particular attention to Fridays and `first day` configurations. Refactoring or adding test cases could help catch edge cases.

- Audit Synchronization Processes: Review how event logs update backend tables to ensure proper alignment, for example, implementing stronger checks during synchronization could prevent delays and overwrite issues.

- Conduct Load Testing: Simulate high-frequency updates on peak days (like Fridays) to identify performance bottlenecks and ensure the system can handle real-world load conditions.

- Refine Day Transition Logic: Enhance algorithms for monthly day adjustments, especially for transitions involving months with different day counts.

Maximiliano Dacko