Journal Entries & Sent Questions - A not-so-brief Data History

> *There are two historical splits in how we have stored the relationship between sent questions and journal entries. At the beginning of Qeepsake's history, these were stored in two separate tables: sent_questions, and journal_entries. The sent_questions table contained a column called journal_entry_id that served as a direct link between a question prompt and the journal entry that was created by responding to that prompt.*

> *The first split happened in January 2021, when we began to store question sends that were never answered in the journal_entries table as journal entries with a question ID but NULL response & attachment. There were some changes in the meaning of different date fields, which are explained in this archived Slack message:*

Summary posted to Slack 5/26/22:

Okay y'all, thanks for everyone's help identifying this potential issue with date fields in the journal_entries table and getting to the bottom of it. Bruno helped me disentangle everything that's happening here, and I wrote up a (not-so) short summary.

Topline takeaway: BACKDATING journal entries does not change the "created_at" field, and most of our past analysis "holds", with the understanding that "created_at" signifies when the question was ANSWERED, not when it was PROMPTED.

There are two key time periods here: BEFORE 1/7/2021 (the sent question refactor), and AFTER 1/7/2021.

There is some nuance as to exactly when certain changes took place in the weeks leading up to and following that date, but for simplicity's sake this is the important historical split.

Two changes happened on that date: We started populating rows in the journal_entries table when the question was SENT instead of when it was ANSWERED, and we also started updating **created_at** when the user ANSWERED the question.

When exactly (if ever) does the "created_at" column get updated?

- Before 2021: Never — But it is first populated when the user answers the question
- After 2021: Whenever user answers the question (it is originally populated when we send the question)

In other words... *"Created_at", through all history, means when the user FIRST created a journal entry in their account by answering the question (If the user never answered the question, "created_at" will be equal to "sent_at", but this should not be mistaken for signifying any user action)*

What about backdating?

- Backdating an entry ONLY updates the "received_at" column and the "updated at" column. It doesn't affect sent or created at.
- This is good news for data's past analyses — We don't have to worry about entries that were "created" before they were "sent". All this means is that we have to interpret the results a little differently — They are built on the column that indicates when the user ANSWERED the question, not when they were PROMPTED. But this is pretty much compatible with our past learnings and understanding of these results.

What are these sent_ats = null, question_id not null-s?

- Those are backfills, from before 2021 when we didn't have a sent_at, and created_at ONLY indicated when the row was created

Is there a column that designates, for all journal entries (prompted, unprompted, backdated, etc), when the *row* was created in the table?

- There is no singular column that signifies when the row was created in the table. This is because...
- BEFORE Jan 7, 2021, a row was created when the user created a journal entry. The correct column to signify this is "created_at" - However, the data for when the entry was PROMPTED lives in the "sent_questions" table.
- AFTER Jan 7, 2021, a row was created whenever we sent a question. The correct column to signify this is "sent_at". "created_at" can be updated at a later date.

However... since we still have the sent_questions table, it *should* be possible for me to assign "sent_at" values to all pre-2021 journal entries, as well as populate the table with rows for each send without a response. That way, in the *data* version of this table, the following should be true for all "history"...

- sent_at will signify when a journal entry is PROMPTED (but may or may not have been ultimately created)
- created_at will signify when a journal entry is CREATED (either unprompted, or by the user responding to a prompted question)
- received_at will signify the date the user has ASSIGNED to the entry (by default, the day the entry was created, but editable through backdating)
- updated_at will signify the latest time the entry was EDITED in any way

(I will take this editing of the BigQuery version of the journal_entries_table as a project for next sprint) Finally, back to our original question...Is there a way to tell how many entries are CREATED long after they are PROMPTED? Yes! Now that we have our terms well-defined, we know that the two fields to compare for post-2021 data (the data that exists today without Ryan's cleaning)
are *sent_at* and *created_at* (AKA, "Prompted" and "Created", Duncan's proposed method). Using these two fields...

- 4.1% of *prompted, answered, non-milestone* journal entries were answered >7D after they were prompted in 2022
- 4.9% of *prompted, answered, non-milestone* journal entries were answered >7D after they were prompted in 2021

> *To reconcile this split into a table for historical analysis, data team built the journal_entries_with_sent_question_history table (the query to rebuild this table from journal_entries_archive and sent_question_archive so is saved as a saved query and also as a view in the zzz_Archive dataset in BigQuery), and an archived version of that table up until the second historical split also lives there.*

> *The second split happened on 10/12/2022 with the "Journal entry table refactor" that was done in order to significantly free up load on the production database. We moved away from the single-table model back to a dual-table model very similar to the original one... though with some key differences.*

Summary of the second split (10/19/2022):

The two tables we now need to use to build a "journal entries with sent question history" table are now called user_questions (analog to sent_questions) and journal_entries.

We no longer store non-responses in the journal_entries table, and the history of those non-responses has been cleared from the production DB (again, archived version lives in zzz_Archive).

Instead of a direct join from a "journal_entry_id" column in the user_questions table, we now need to join on a combination of three columns:

user_id = user_id

question_id = question_id

journal_entries.created_at = user_questions.answered at *(down to the granularity of minutes, but no more — These may differ by a second for two rows that we do in fact want to link together.)*

To have a single table that reflects the full history of question prompts + journal entries, we need to both...

1. Stitch together the new versions of the tables for all activity after the change was made (in a way which results in a table with the exact same format as the old version)
2. "Append" this onto the old, archived version of the same table.

This is done in the daily BigQuery scheduled query "journal_entries_with_sent_question_history", which I have tried to add maximal documentation in for posterity's sake, as the way we put these now 3 distinct histories of Qeepsake's database together has grown quite complex.

There is also one additional wrinkle, which is that the production DB going forward will ONLY be storing the past 12 months of user_question table history. This shouldn't be a problem because of the way our Stitch data service sends data to BigQuery — We "build up" a history by adding the net-new rows (as determined by the updated_at field) in BigQuery, and old data won't get erased even if it is no longer in the Production DB. ***However, it is vitally important that if for any reason we have to do a "full historical sync" again between the BigQuery DB and production DB, we'll need to archive a copy of the BigQuery version of the user_questions table before the sync.***