Design and Implementation of a Data Warehouse for Product Engagement Analysis: A Case Study of Notion

Nick Zerjeski
December 2025

Contents

Chapter 1

Domain Analysis and Description

As the title suggests, in this case study we look at Notion. Notion is a Software-as-a-Service (SaaS) company that provides an all-in-one productivity and collaboration platform. It enables individuals and teams to create, organize, and share knowledge through a unified interface that combines note-taking, database management, task tracking, and wiki functionality. The platform is structured around *workspaces* (multi-tenant units), which contain *pages* composed of modular *blocks* such as text, images, tables, and databases. These blocks are the atomic units of content and interaction. Workspaces can have multiple users, permission hierarchies, and integrations with external tools via API's.

The Notion ecosystem generates many voluminous data streams across several domains such as data about the Content, the User, Billing Data, Marketing and the Product Usage. In this case study we will focus on the Product Usage and Engagement. That is Event-level logs that captures user actions such as page creation, edits, views, comments, shares and API interactions. This data allows for temporal and behavioral analytics at high granularity.

A data warehouse is essential for Notion because its operational data is fragmented across multiple systems which are not optimized for analysis. Since the data is scattered it must be integrated into a consistent, subject-oriented repository to enable meaningful insights. A data warehouse provides stable, time-variant storage that preserves historical information necessary for analyzing trends in engagement and feature usage. This allows for fast, flexible querying of complex behavioral data. It also improves the data quality and consistency by cleaning and using heterogeneous sources. Therefore, a Data Warehouse enables us to transform raw event data into actionable intelligence. Ultimately, it provides the foundation for evidence-based decision-making and continuous product optimization.

1.1 Key Analytical Questions

The following five questions should later able to be answered by running queries on or data warehouse.

- **How does user engagement evolve over time across subscription tiers?** This question reveals differences in activity between free and paid plans, supporting decisions on pricing and feature differentiation.
- **Which content types (pages, databases, wikis, task boards) generate the highest sustained interaction rates?** Identifying the most engaging content structures helps prioritize feature development and template curation.
- **What are the activation rates of new users within their first seven days?** Activation is a leading indicator of retention; understanding early engagement patterns enables targeted onboarding improvements.
- **How does device usage (web, mobile, desktop) affect user activity and session duration?** Device-level analysis supports product optimization and informs cross-platform development priorities.
- **What proportion of total activity originates from collaborative versus individual work?** Measuring the balance between solo and shared content usage reveals how effectively Notion fosters team adoption within workspaces.

1.2 Key Performance Indicators (KPIs)

Each analytical question corresponds to one or more measurable KPIs central to product health evaluation:

- **Daily/Weekly/Monthly Active Users (DAU/WAU/MAU):** Core activity metrics measuring recurring usage patterns.
- **Stickiness Ratio (DAU/MAU):** Indicates habitual engagement and long-term adoption strength.
- **Activation Rate:** Percentage of new users performing a predefined minimum number of meaningful interactions (e.g., seven content edits within seven days).
- **Feature Adoption Rate:** Share of users engaging with advanced functionalities such as databases, templates, or integrations.
- **Average Session Depth:** Mean number of interactions per user session, used to evaluate content richness and user intent.

1.3 Granularity Discussion

Since we are trying to analyze **Product Usage and Engagement** the appropriate granularity in this context are **Event-level facts**. Event-level facts capture each individual user action that occurs within a workspace, including page creations, edits, views, comments, shares, and API-driven interactions tied to a timestamp. Therefore, this level of detail is fine enough to compute all engagement-related KPIs. Furthermore, it ensures that every analytical question can be answered, since each fact retains its full context regarding each dimension. Using this level of granularity ensures analytical flexibility and avoids premature aggregation that would cause problems later one if a finer granularity would be required.

Chapter 2

Conceptual Design

To build the Dimensional Fact Model, we first think about which dimensions and measures our fact should have. Since we are describing the Product Usage and Engagement our fact will represent a single atomic user event in Notion such as these described in Chapter 1. To capture the full behavioral context, we need to capture also the user, the content object, the workspace, the device, the event type, the session and the timestamp, which we do via dimensions. We therefore have the following dimensions:

Attribute	**Description**
`user_id`	Unique identifier of the user
`signup_date`	Date the user registered
`subscription_tier`	Free, Plus, Business, Enterprise
`user_type`	Individual, member, guest
`region`	User's geographic region
`lifecycle_stage`	Onboarding, active, dormant, churn-risk

Table 2.1: User Dimension

Attribute	**Description**
`content_id`	Identifier of the page or block
`content_type`	Page, database, wiki, task board, etc.
`is_template_based`	Indicates template-derived content
`is_shared`	Whether content is shared with others
`ownership_type`	Personal, workspace, or team space

Table 2.2: Content Dimension

Attribute	Description
workspace_id	Unique identifier of the workspace
workspace_plan	Subscription plan of the workspace
workspace_size_bucket	Size category (1, 2–10, 11–50, 50+)
industry_segment	Workspace industry classification
workspace_region	Geographic region of workspace

Table 2.3: Workspace Dimension

Attribute	Description
device_id	Identifier for the device type
platform	Web, desktop, mobile
operating_system	OS used for the event
app_version	Client version
device_form_factor	Phone, tablet, desktop

Table 2.4: Device Dimension

Attribute	Description
event_type	View, edit, create, comment, share, etc.
feature_category	Databases, templates, integrations, collaboration
interaction_intent	Consumption, creation, collaboration

Table 2.5: Event Dimension

Attribute	Description
session_id	Unique session identifier
session_start_time	Start timestamp of the session
session_end_time	End timestamp of the session
session_origin	Entry source (direct, link, notification)

Table 2.6: Session Dimension

Attribute	Description
date_key	Surrogate key for the date
calendar_date	Full calendar date
day_of_week	Numeric day of week
is_weekend	Weekend indicator
week_of_year	ISO week number
month	Month number
quarter	Calendar quarter
year	Calendar year
day_since_signup_bucket	Bucket for activation/cohort logic

Table 2.7: Time Dimension

To evaluate product usage and engagement effectively, the fact table must also include measures that quantify how users interact with Notion. Therefore, the following measures allow us to aggregate individual events into meaningful indicators:

Event Count

- **Description:** Always 1 per event.
- **Additivity:** Fully additive across all dimensions.

Active User Flag

- **Description:** 1 on a user's first event of a day, 0 otherwise.
- **Additivity:** Additive across users for a fixed day; semi-additive across time.

Activation Event Flag

- **Description:** 1 if the event is a meaningful interaction within the first seven days after signup.
- **Additivity:** Additive within user-time windows; non-additive when converted into activation rates.

Feature Usage Flag

- **Description:** 1 if the event uses an advanced feature.
- **Additivity:** Additive for counts; non-additive when used in ratios.

Collaboration Event Flag

- **Description:** 1 for events occurring in a collaborative context.
- **Additivity:** Additive for counts; non-additive for proportions.

Session Event Count

- **Description:** Number of events per session (sum of Event Count grouped by session).
- **Additivity:** Additive within a session; semi-additive across time.

Session Duration Seconds

- **Description:** Stored on the final event of a session.
- **Additivity:** Additive across sessions; semi-additive across time.

2.1 Dimensional Fact Model

Since we now discussed and defined the dimensions and measures of our Product Usage and Engagement fact, we can start building the Dimensional Fact Model (DFM).

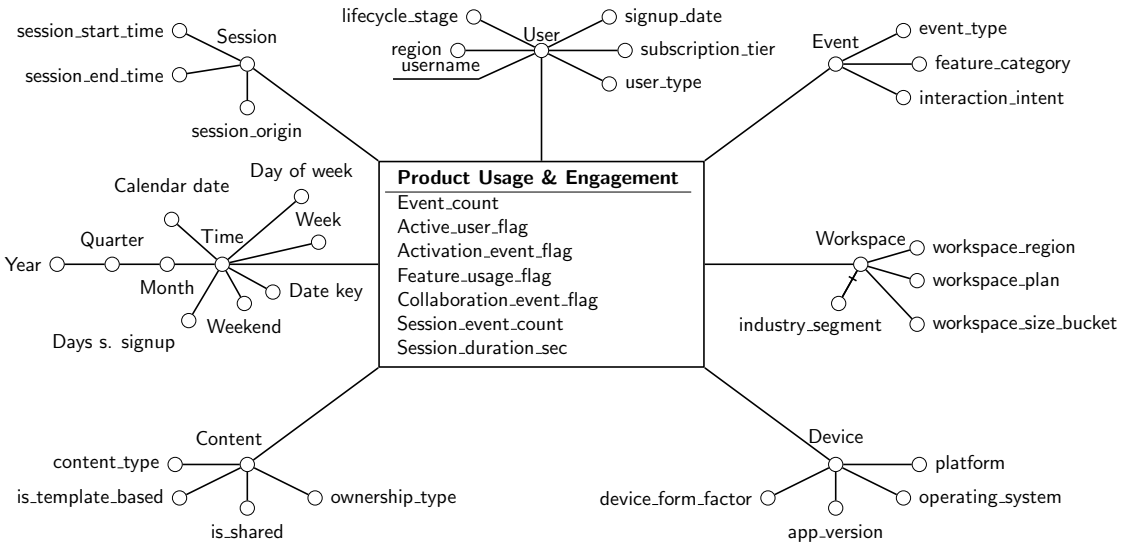

Figure 2.1: Dimensional Fact Model for Product Usage and Engagement.

For the user dimension, we choose `username` as a descriptive attribute to easily identify users in reports. It is not used in analytical queries. Additionally, we defined `industry_segment` as an optional attribute, since it is not always applicable to put workspaces into an industry context, for example for personal workspaces. Finally, we decided to model `Month`, `Quater` and `Year` as a hierarchy in the time dimension to allow for roll-up analysis over different time granularities.

Chapter 3

Logical Design

Since all hierarchies are shallow and local to their dimensions and no hierarchy is reused across multiple dimensions, there is no need to normalize those hierarchies into additional tables. To keep the number of joins to a minimum and the queries as simple as possible, especially with no significant gain in storage efficiency, we are using a star schema for our *Product Usage and Engagement* fact. Therefore, we have the following fact table, followed by the dimension tables.

Attribute	**Key**
`usage_id`	Primary Key
`time_key`	FK→ `dim_time(time_key)`
`user_key`	FK→ `dim_user(user_key)`
`workspace_key`	FK→ `dim_workspace(workspace_key)`
`content_key`	FK→ `dim_content(content_key)`
`device_key`	FK→ `dim_device(device_key)`
`event_key`	FK→ `dim_event(event_key)`
`session_key`	FK→ `dim_session(session_key)`
`event_count`	
`active_user_flag`	
`activation_event_flag`	
`feature_usage_flag`	
`collaboration_event_flag`	
`session_event_count`	
`session_duration_sec`	

Table 3.1: Fact table `fact_product_usage_engagement`.

Attribute
time_key
calendar_date
day_of_week
is_weekend
week_of_year
month
quarter
year
day_since_signup_bucket

Table 3.2: Time table

Attribute
user_key
user_id_nat
username
signup_date
subscription_tier
user_type
region
lifecycle_stage

Table 3.3: User table

Attribute
workspace_key
workspace_id_nat
workspace_plan
workspace_size_bucket
industry_segment
workspace_region

Table 3.4: Workspace table

Attribute
content_key
content_id_nat
content_type
is_template_based
is_shared
ownership_type

Table 3.5: Content table

Attribute
device_key
device_id_nat
platform
operating_system
app_version
device_form_factor

Table 3.6: Device table

Attribute
event_key
event_type
feature_category
interaction_intent

Table 3.7: Event table

Attribute
session_key
session_id_nat
session_start_time
session_end_time
session_origin

Table 3.8: Session table

To get an overview on how much space we will be using, we estimate the cardinality of our tables. For that, we assume that we save our data for two years. Furthermore, we assume that we have a user base of around five million users from which roughly 20% are active on a given day and produce around 10 interactions. Therefore, we have the following size of our fact table:

$$5\text{Mio User} \cdot 20\% \cdot 10 \text{ Interactions} \cdot 730 \text{ days} = 7.3 \cdot 10^9 \text{ rows}$$

We further assume, that each workspace has on average ten users, which yields a total of 500k rows for the workspace table. If each of the workspaces has around 50 content objects in it, the content table has 25Mio rows. For the device dimension, we simply assume that there are 100 different devices available and we also have around 100 different types of events. Finally, for the session, we assume a day is a session. Therefore, we have

$$5\text{Mio User} \cdot 20\% \cdot 730 \text{ days} = 7.3 \cdot 10^8 \text{ rows}$$

for the session table. To simplify the calculation of the size of the table, we assume that each column takes 8 Bytes. With that our tables have the sizes specified in Table 3.9. Keep in mind that some values are rounded.

Table	Size
Product Usage Engagement	800GB
Session	30GB
Content	1GB
User	26MB
Workspace	23MB
Time	50KB
Device	5KB
Event	3KB

Table 3.9: Sizes of the tables.

3.1 Demonstration

Now that we have the logical model in place, it is ready to support analytical workflows. To demonstrate how our schema answers to the defined business questions in section 1.1, we select two of them and express them directly in SQL to extract the data from our newly created fact and dimension tables. To be able to do this, we first have to populate our tables with some mock data. Note that only attributes are specified, that are needed for the analytical query.

usage_id	time_key	user_key	content_key	event_count
1	1	10	100	1
2	1	10	100	1
3	1	11	101	1
4	1	12	102	1
5	2	10	100	1
6	2	11	101	1
7	2	12	101	1
8	3	10	100	1
9	3	11	102	1
10	3	12	100	1

Table 3.10: Mock data for `fact_product_usage_engagement`.

time_key	calendar_date
1	2025-01-01
2	2025-01-02
3	2025-01-03

Table 3.11: Mock data for `dim_time`.

user_key	username	subscription_tier
10	user_0010	Free
11	user_0011	Plus
12	user_0012	Free

Table 3.12: Mock data for `dim_user`.

content_key	content_type
100	page
101	database
102	wiki

Table 3.13: Mock data for `dim_content`.

With the mock data in place, we can go to our business questions. As first question we choose

How does user engagement evolve over time across subscription tiers?

This is an important question for subscription-based SaaS products, since different user tiers often exhibit different engagement patterns. Understanding how frequently users in each tier interact with the platform over time allows to identify trends in adoption, retention, and value perception. In Listing 3.1 you can see the query that extracts the information needed for answering this question.

```
SELECT
    t.calendar_date,
    u.subscription_tier,
    SUM(f.event_count) AS total_events
FROM fact_product_usage_engagement AS f
JOIN dim_time AS t
    ON f.time_key = t.time_key
JOIN dim_user AS u
    ON f.user_key = u.user_key
GROUP BY
    t.calendar_date,
    u.subscription_tier
ORDER BY
    t.calendar_date,
    u.subscription_tier;
```

Listing 3.1: Query for answering: How does user engagement evolve over time across subsciption tiers?

The result of running this query can be seen in Table 3.14.

calendar_date	subscription_tier	total_events
2025-01-01	Free	3
2025-01-01	Plus	1
2025-01-02	Free	2
2025-01-02	Plus	1
2025-01-03	Free	2
2025-01-03	Plus	1

Table 3.14: Resulting table after running the query in Listing 3.1.

The second business question for which we run the query on is

Which content types generate the highest interaction rates?

This is an important business question because different content types are related to different workflows, and their usage patterns are strong indicators of feature adoption. Analyzing it provides insights on how future features can be developed. In Listing 3.2 you can see the query that extracts the information needed for answering this question.

```
SELECT
    c.content_type,
    SUM(f.event_count) AS total_events
FROM fact_product_usage_engagement AS f
JOIN dim_content AS c
    ON f.content_key = c.content_key
GROUP BY
    c.content_type
ORDER BY
    total_events DESC;
```

Listing 3.2: Query for answering: Which content types generate the highest interaction rates?

The result of this query can be seen in Table 3.15.

content_type	total_events
page	5
database	3
wiki	2

Table 3.15: Resulting table after running the query in Listing 3.2.

Chapter 4

Physical Design

Now that the logical design is in place, we can start to implement it in a real database system. For that, we create a script that creates all the tables with their corresponding surrogate keys and relations we defined in Chapter 3. A snippet of this script can be seen in Listing 4.1. Keep in mind that the full scripts are provided as appendix.

```
-- Creation of the user dimension
CREATE TABLE IF NOT EXISTS dim_user (
  user_key     INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  user_id_nat  VARCHAR(64) NOT NULL UNIQUE,
  username     VARCHAR(64),
  signup_date  DATE,
  ...
);

-- Snippet of the fact table
CREATE TABLE IF NOT EXISTS fact_product_usage_engagement (
  usage_id     BIGINT GENERATED ALWAYS AS IDENTITY,
  time_key     INT NOT NULL,
  ...
  PRIMARY KEY (usage_id, time_key),
  CONSTRAINT fk_fact_user
        FOREIGN KEY (user_key) REFERENCES dim_user(user_key),
) PARTITION BY RANGE (time_key);
```

Listing 4.1: Snippet of the script for creating the Data Warehouse.

Here we can see that every attribute with a _key postfix is a surrogate key. As you can see in Table 3.9, the fact table can get rather big. It is therefore useful to split it

into smaller tables. For our Key Analytical Questions we want to answer later on, it is beneficial to partition the table on the time dimension.

To populate our data warehouse, we first have to define in which time span we want to have data in it. Since data warehouses normally store data for up to two years, we choose January 2024 to December 2025 as the time span in which we want to store data in it. Of course, we can not simply add random data to it, because then we couldn't get meaningful data to answer our analytical questions. Therefore, we did the following assumptions:

First, we created monthly partitions for the fact table so that the data fits into the chosen time span and queries stay fast. Then we filled the time dimension with one row per day from 01.01.2024 to 31.12.2025 so that every event can be linked to a calendar day. We also set up small, fixed device, content, and event dimensions with realistic options - for example desktop vs. mobile, wiki vs. or database vs. page - to make sure we have meaningful categories instead of random strings.

For the user dimension, we generated 20 new users per month across the whole time span, which yields 480 total users. We varied subscription tier, user type, region, and life cycle stage in a deterministic way so each month is stable in that regard and we can compare Free vs. Plus vs. Business vs. Enterprise behavior. Workspaces were created separately with at least 50 entries that mix plans, size buckets, industries, and regions to keep workspaces varied without being chaotic.

Sessions were added at about five per day, with durations between roughly 20 and 70 minutes. This gives enough session coverage so that every event can find a plausible session on the same day.

Finally, the fact table generation ties everything together. We compute an activation probability per user based on their tier and only give activation events to users who pass that threshold in their first week. On top of that, we assign monthly event volumes per user that scale with their tier, then sprinkled some randomness in it to avoid overly uniform behavior.

For each event we pick a date within the valid window, that is an event can only occur after a user created their account, attach biased mixes for content types, devices, and event types, and link the event to a workspace and a same-day session. The measure flags are set according to the type of event and content so that activation, feature usage, and collaboration metrics come out believable. This way the warehouse ends up with structured, meaningful mock data so we are ready to answer our key analytical questions.

Chapter 5

Results

In the following, we present the results we obtained by answering the Key Analytical Questions defined in Chapter 2 using the data warehouse designed and implemented in the previous chapters. For that, we executed the queries defined for each question against the data warehouse and visualized the results using appropriate charts.

How does user engagement evolve over multiple subscription tiers over time? For our first question, we wanted to know how the user engagement evolves over multiple subscription tiers. But it is also helpful to see how engagement evolves over time in general, so we can spot trends and seasonality. Therefore, we created a query that aggregates the number of active users per month and subscription tier. Furthermore, we also wanted to see how many active users are in each tier in total over the time span, so we can see, which tier has the most engagement. Finally, we wanted to know what the total engagement was over the total time span to see how many users were active in total. To extract those information, we used the query described in Listing 5.1.

```
SELECT t.year, t.month, u.subscription_tier,
  SUM(f.active_user_flag) AS dau,
FROM fact_product_usage_engagement f
JOIN dim_time t ON t.time_key = f.time_key
JOIN dim_user u ON u.user_key = f.user_key
GROUP BY GROUPING SETS (
  (t.year, t.month, u.subscription_tier),
  (t.year, t.month),
  (u.subscription_tier),
  ()
)
```

```
ORDER BY t.year NULLS LAST, t.month NULLS LAST,
                u.subscription_tier NULLS LAST;
```

Listing 5.1: Query for answering: How does user engagement evolve over time across subsciption tiers?

To retrieve the data we wanted to know, it was benefitial to use the `GROUPING SETS` feature of SQL, which allows us to group by multiple dimensions in a single query. This way, we could get the monthly active users per subscription tier, the total active users per subscription tier, and the total active users overall in a single query. After executing the query against our data warehouse, we visualized the results of the evolution of user engagement over time in Figure 5.1.

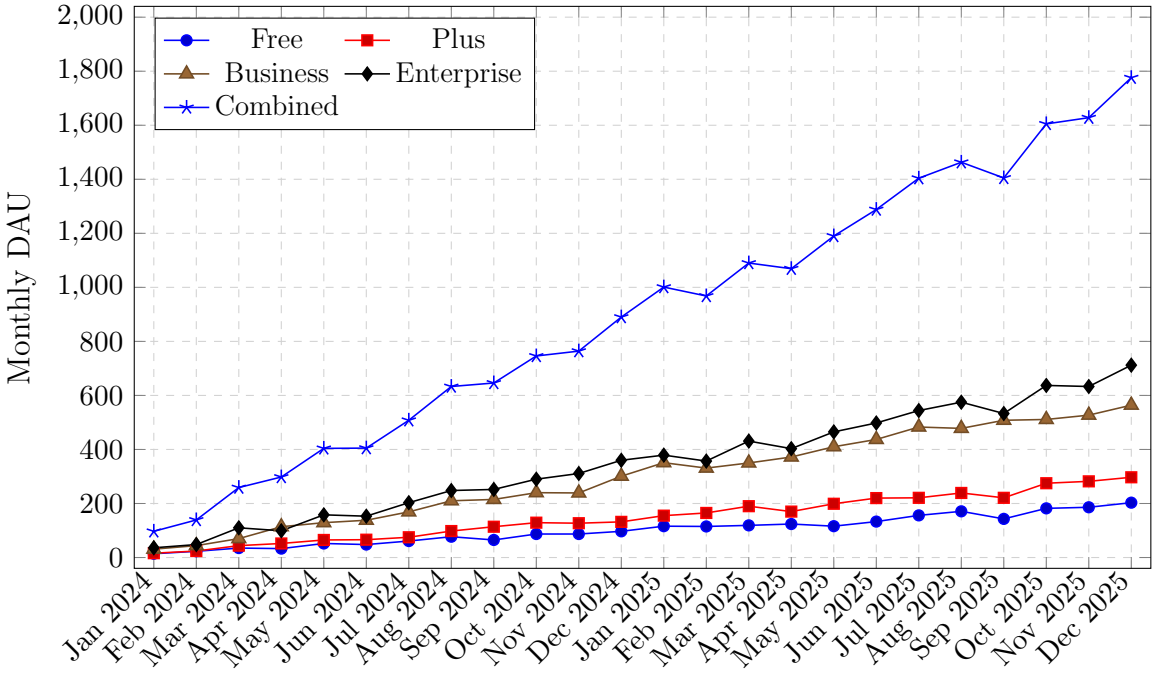

Figure 5.1: Monthly DAU by subscription tier for Jan 2024 - Dec 2025.

To compare the total engagement per subscription tier, we created a bar chart as shown in Figure 5.2.

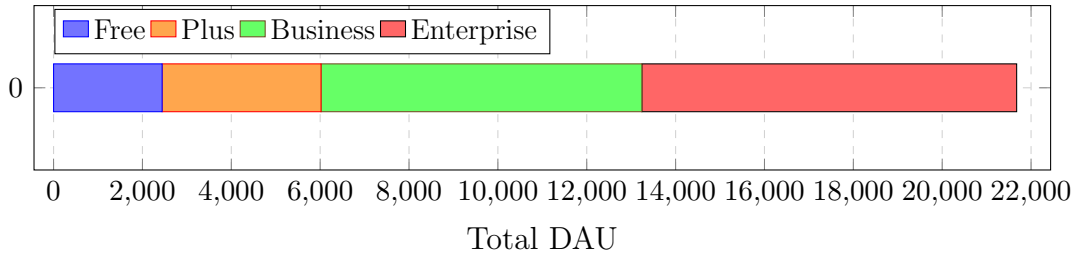

Figure 5.2: Total DAU by subscription tier for Jan 2024 - Dec 2025.

Here we can see that the *Enterprise* tier has the highest total daily active users (DAU) with 8,435 users, followed by the *Business* tier with 7,222 users. The *Plus* and *Free* tiers have significantly lower total DAU, with 3,576 and 2,443 users respectively. This indicates that higher subscription tiers tend to have more engaged users, which could be due to the higher involvement they have do to higher billing plans. To get a better understanding of the user engagement we take a look at the second question.

Which content types generate the highest sustained interaction rates? To answer our second question, we wanted to know which content types generate the highest sustained interaction rates. Therefore, we created a query that aggregates the number of interactions per content type and month. This way, we can see which content types are interacted with the most over time. To extract those information, we used the query described in Listing 5.2.

```
SELECT t.year, t.month, c.content_type,
       SUM(f.active_user_flag) AS dau
FROM fact_product_usage_engagement f
JOIN dim_time t    ON t.time_key = f.time_key
JOIN dim_content c ON c.content_key = f.content_key
GROUP BY CUBE (t.year, t.month, c.content_type)
ORDER BY t.year NULLS LAST, t.month NULLS LAST,
                    c.content_type NULLS LAST;
```

Listing 5.2: Query for answering: How does user engagement evolve over time across different content types?

This time, we used `CUBE` to retrieve all combinations of content types, months and years. This way, we could get all levels of granularity in a single query. Therefore, we can get a little bit more insights into the data which is especially benefitial when investigating content types. First, we take a look at the usage of the diffrent content types over time as shown in Figure 5.3.

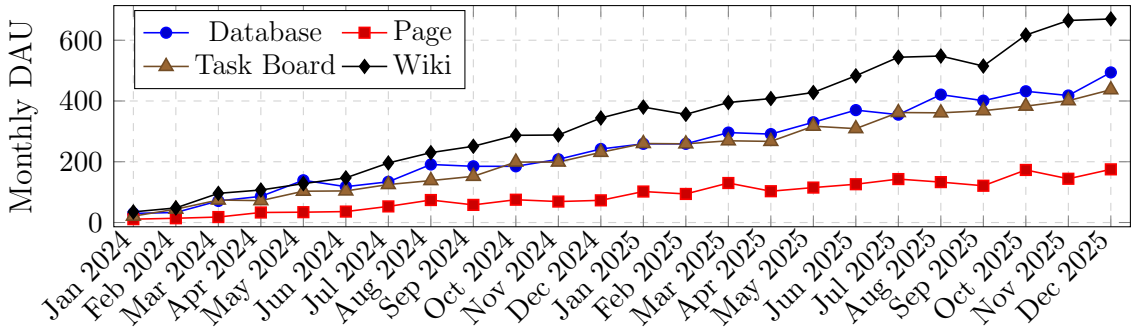

Figure 5.3: Monthly DAU by content type for Jan 2024 - Dec 2025.

Here we can see that the *Wiki* content type has the highest monthly DAU, followed by *Task Board*, *Database*, and *Page*. This indicates that users tend to interact more with *Wiki* content, which could be due to its collaborative nature and the fact that it is often used for documentation and knowledge sharing. To highlight seasonal patterns independent of year-to-year growth, we aggregated monthly DAU across years for each content type. In Figure 5.4, we can see the seasonality of each content type.

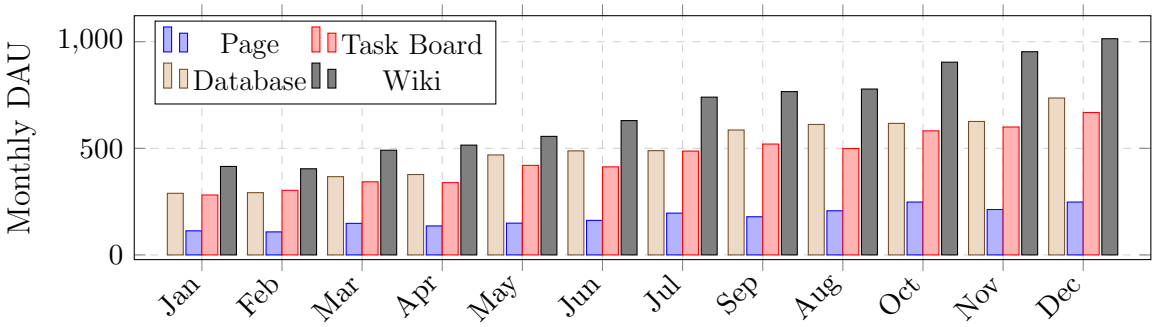

Figure 5.4: Seasonality by month across years (aggregated over 2024 - 2025).

As we can see, no content type shows a significant seasonal pattern. Since we aggregated the data across years, the dip we saw earlier in September is not visible anymore. Overall, we can conclude that *Wiki* and *Task Board* content types generate the highest sustained interaction rates, while *Page* content has the lowest engagement. To see if the content types with the highest daily active users also have the highest event count we run the query shown in Listing 5.3.

```
SELECT
  c.content_type,
```

```
  SUM(f.event_count) AS events,
  DENSE_RANK() OVER (ORDER BY SUM(f.event_count) DESC)
                                  AS content_type_rank
FROM fact_product_usage_engagement f
JOIN dim_content c ON c.content_key = f.content_key
GROUP BY c.content_type
ORDER BY content_type_rank, c.content_type;
```

Listing 5.3: Query for answering: How does user engagement evolve over time across different content types?

After executing the query against our data warehouse, we visualized the results in Table **??**.

Content Type	Events	Rank
wiki	40719	1
database	30393	2
task_board	26982	3
page	10544	4

Table 5.1: Resulting table after running the query in Listing 5.3.

Here we can see that the *Wiki* content type has the highest event count with 40,719 events, followed by *Database* with 30,393 events, *Task Board* with 26,982 events, and *Page* with 10,544 events. This indicates that the content types with the highest daily active users also tend to have the highest event counts, suggesting a strong correlation between user engagement and interaction frequency. Overall, we can conclude that *Wiki* and *Database* content types not only generate the highest sustained interaction rates but also have the highest event counts. This information can be used to inform product development and marketing strategies, as well as to identify areas for improvement in user engagement. Now we take a look at the activation rates of new users.

What are the activation rates of new users within their first seven days? To answer our third question, we wanted to know what the activation rates of new users are within their first seven days. Therefore, we created a query that calculates the percentage of new users who become active within their first seven days after sign-up. To extract those information, we used the query described in Listing 5.4.

```
WITH user_events AS (
  SELECT
    u.user_key,
```

```
    u.signup_date,
    t.calendar_date,
    f.activation_event_flag,
    ROW_NUMBER() OVER w AS rn_after_signup
  FROM fact_product_usage_engagement f
  JOIN dim_user u ON u.user_key = f.user_key
  JOIN dim_time t ON t.time_key = f.time_key
  WHERE t.calendar_date >= u.signup_date
    AND t.calendar_date <  u.signup_date + INTERVAL '7 days'
  WINDOW w AS
    (PARTITION BY u.user_key ORDER BY t.calendar_date)
),
activation_by_user AS (
  SELECT
    user_key,
    MIN(signup_date) AS signup_date,
    MAX(activation_event_flag) AS activated_within_7d
  FROM user_events
  GROUP BY user_key
)
SELECT
  DATE_TRUNC('month', signup_date)::date AS signup_month,
  COUNT(*)                               AS new_users,
  SUM(activated_within_7d)               AS activated_users,
  ROUND(
    SUM(activated_within_7d)::numeric / NULLIF(COUNT(*), 0),
    4
  )                                      AS activation_rate
FROM activation_by_user
GROUP BY DATE_TRUNC('month', signup_date)
ORDER BY signup_month;
```

Listing 5.4: Query for answering: What is the activation rate over time

In this query, we first created a Common Table Expression (CTE) called `user_events` that retrieves all user events within the first seven days after sign-up. We then created another CTE called `activation_by_user` that aggregates the data by user and determines whether each user activated within the seven-day window. Finally, we calculated the activation rate by dividing the number of activated users by the total number of new users for each month. After executing the query against our data warehouse, we visualized the results in Figure 5.5.

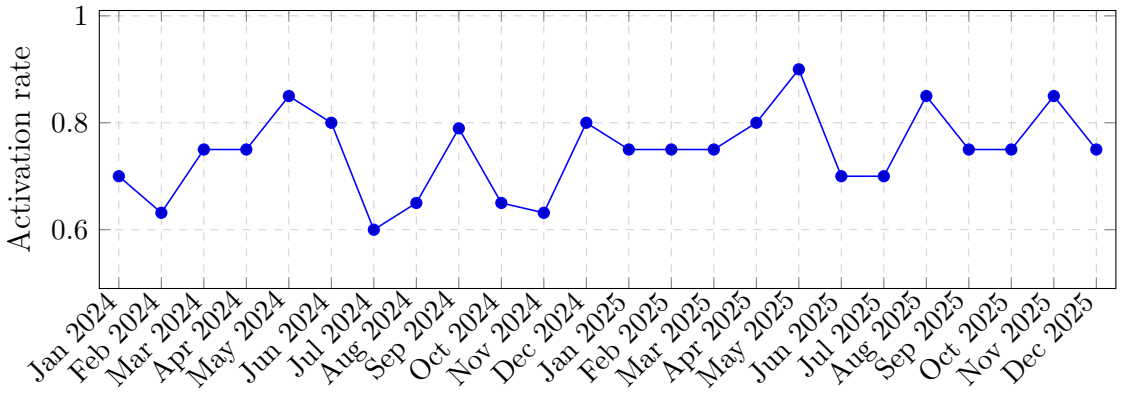

Figure 5.5: Activation rate of new users within seven days by signup month for Jan 2024 - Dec 2025.

Here we can see that the activation rate of new users within their first seven days fluctuates between approximately 0.6 and 0.9 over the observed period. There is a noticable dip in mid 2024 with the exception of September 2024, where the activation rate drops to around 0.6. After that, the activation rate gradually increases again, reaching a peak of around 0.9 in May 2025 and after that it stabelizes between 0.7 and 0.85. Overall, we can conclude that the activation rates of new users within their first seven days show some fluctuations over time, but generally tend to be relatively high. This information can be used to inform user onboarding strategies and identify areas for improvement in the activation process. Next, we take a look on how devices influences the user activity and session duration.

How does device usage affect user activity and session duration? Now, to answer our fourth question, we wanted to know how device usage affects user activity and session duration. Therefore, we created a query that aggregates the number of active users and average session duration per device type and month. This way, we can see how different devices impact user engagement over time. To extract those information, we used the query described in Listing 5.5.

```
SELECT
  t.year,
  t.month,
  d.platform,
  SUM(f.event_count)          AS events,
  SUM(f.active_user_flag)     AS dau,
  ROUND(AVG(NULLIF(f.session_duration_sec,0))::numeric, 1)
                              AS avg_session_duration_sec
FROM fact_product_usage_engagement f
```

```
JOIN dim_time t     ON t.time_key = f.time_key
JOIN dim_device d   ON d.device_key = f.device_key
GROUP BY ROLLUP (t.year, t.month, d.platform)
ORDER BY t.year NULLS LAST, t.month NULLS LAST,
                            d.platform NULLS LAST;
```

Listing 5.5: Query for answering: How does device type influence user activity

In this query, we joined the fact table with the time and device dimension tables to retrieve the necessary information. We then grouped the data by year, month, and device platform using the `ROLLUP` feature to get all levels of granularity in a single query. After executing the query against our data warehouse, we visualized the results of user activity over time by device type in Figure 5.6.

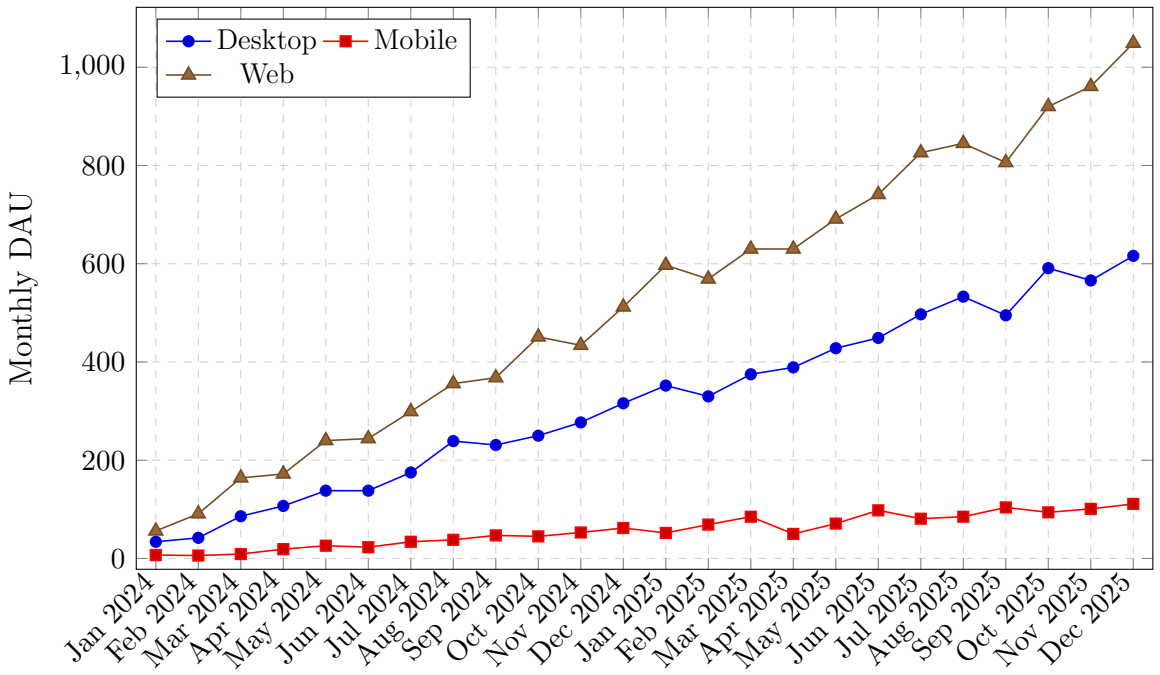

Figure 5.6: Monthly DAU by device type for Jan 2024 - Dec 2025.

Here we can see that web users have the highest monthly DAU, followed by desktop and mobile users. This indicates that users tend to be more active on web platforms, which could be due to the convenience and accessibility of web applications. Desktop users also show a significant level of activity, while mobile users have the lowest monthly DAU. This could be due to the limitations of mobile devices, such as smaller screens and reduced functionality compared to desktop and web platforms. To get a

better understanding of how device usage affects session duration, we visualized the average session duration per device type in Figure 5.7.

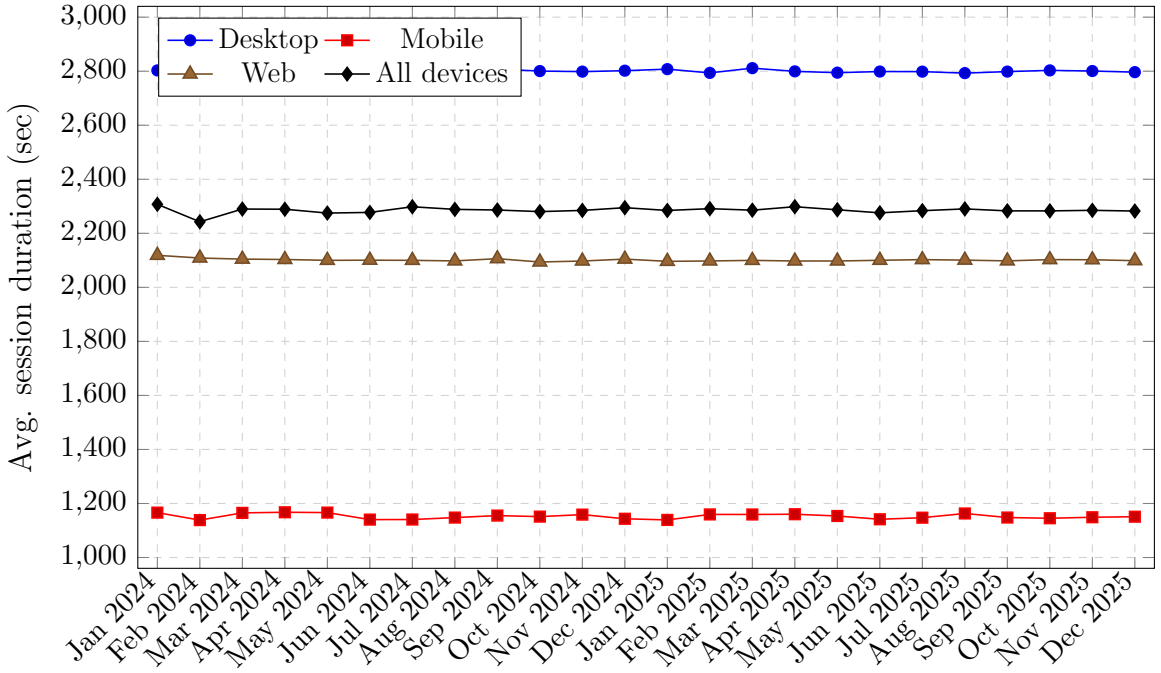

Figure 5.7: Average session duration by device type for Jan 2024 - Dec 2025.

This plots also shows that web users have the longest average session duration, followed by desktop and mobile users. This further supports the idea that web platforms provide a more engaging user experience, leading to longer sessions. Desktop users also have a relatively high average session duration, while mobile users have the shortest sessions. This could be due to the fact that mobile users often use apps for quick tasks and may not spend as much time on them compared to desktop and web platforms. But also that the desktop and web version is more handy. Overall, we can conclude that device usage has a significant impact on user activity and session duration, with web platforms showing the highest engagement levels. This information can be used to inform product development and marketing strategies, as well as to identify areas for improvement in user engagement across different devices. We now proceed to our last key analytical question.

What proportion of total activity originates from collaborative versus individual work? To answer our fifth and final question, we wanted to know what proportion of total activity originates from collaborative versus individual work.

Therefore, we created a query that aggregates the number of active users and events based on whether the activity was collaborative or individual. To extract those information, we used the query described in Listing 5.6.

```
WITH base AS (
  SELECT
    t.year,
    t.month,
    CASE WHEN f.collaboration_event_flag = 1
         THEN 'collaborative' ELSE 'individual' END
                                  AS work_mode,
    SUM(f.event_count) AS events
  FROM fact_product_usage_engagement f
  JOIN dim_time t ON t.time_key = f.time_key
  GROUP BY t.year, t.month,
    CASE WHEN f.collaboration_event_flag = 1
         THEN 'collaborative' ELSE 'individual' END
), totals AS (
  SELECT
    year, month,
    SUM(events) AS total_events
  FROM base
  GROUP BY year, month
)
SELECT
  b.year,
  b.month,
  b.work_mode,
  b.events,
  ROUND(b.events::numeric / NULLIF(t.total_events,0), 4)
              AS proportion
FROM base b
JOIN totals t USING (year, month)
ORDER BY b.year, b.month, b.work_mode;
```

Listing 5.6: Query for answering: What proportion of total activity originates from collaborative versus individual work

In this query, we first created a Common Table Expression (CTE) called `base` that retrieves the number of events per month and work mode (collaborative or individual). We then created another CTE called `totals` that calculates the total number of events per month. Finally, we joined the two CTEs to calculate the proportion of events for each work mode by dividing the number of events by the total number of

events for each month. After executing the query against our data warehouse, we visualized the results in Figure 5.8.

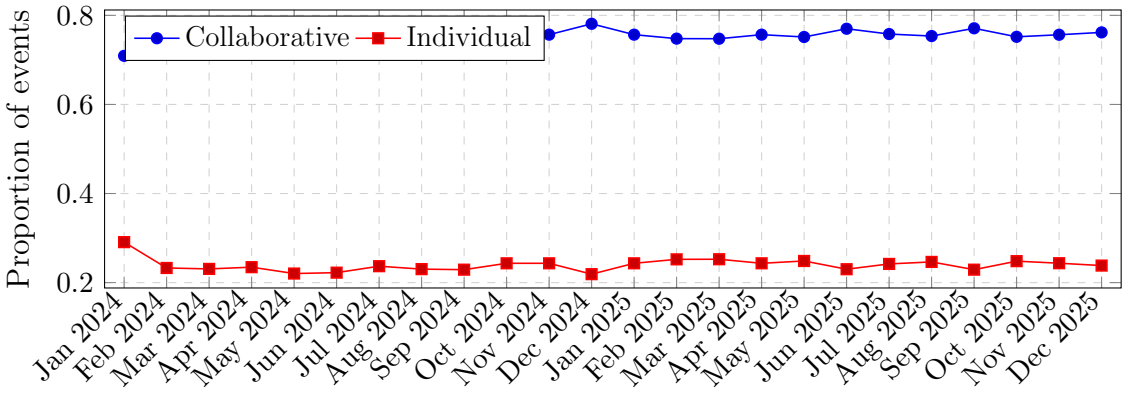

Figure 5.8: Monthly proportion of collaborative vs. individual activity for Jan 2024 - Dec 2025.

Here we can see that collaborative work consistently accounts for a higher proportion of total activity compared to individual work. The proportion of collaborative work ranges from approximately 70% to 80%, while individual work ranges from approximately 20% to 30%. There are almost none fluctuations over time and the overall trend indicates that collaborative work is the dominant mode of activity. This suggests that users tend to engage more in collaborative tasks, which could be due to the nature of the platform and its focus on teamwork and collaboration. Overall, we can conclude that a significant proportion of total activity originates from collaborative work, highlighting the importance of collaboration in driving user engagement on the platform. This information can be used to inform product development and marketing strategies, as well as to identify areas for improvement in fostering collaboration among users.

Chapter 6

Conclusion

After analyzing the domain of our case study in notion, we have identified several key analytical questions that will generate valuable insights. These questions are designed to explore various aspects of the data and provide a comprehensive understanding of the underlying patterns and trends in the company.

After that, we formulated a robust dimensional fact model that effectively captures the relationships and attributes necessary to address these questions. The data model serves as a blueprint for organizing and structuring the data in a way that facilitates efficient analysis.

From the dimensional fact model, we derived a detailed logical schema that outlines the specific tables, fields, and relationships required to implement the data model in a database system. This schema provides a clear roadmap for the physical implementation of the data warehouse.

Next, we developed a comprehensive ETL (Extract, Transform, Load) process to populate the data warehouse with relevant data from various sources. The ETL process ensures that the data is accurately extracted, transformed into the appropriate format, and loaded into the data warehouse for analysis.

Finally, we implemented a series of analytical queries to extract meaningful insights from the data warehouse. These queries are designed to answer the key analytical questions identified earlier and provide actionable information to support decision-making within the company.

In conclusion, the systematic approach we followed—from identifying analytical questions to implementing ETL processes and analytical queries—has laid a solid foundation for effective data analysis in our case study. This structured methodology

not only enhances our understanding of the data but also equips us with the tools necessary to derive actionable insights that can drive informed business decisions.

Appendix

Script for creating the schema

```
-- ============================================================
-- NOTION DW: Product Usage & Engagement
-- ============================================================

CREATE SCHEMA IF NOT EXISTS notion_dw;
SET search_path = notion_dw;

-- -------------------------
-- DIMENSIONS
-- -------------------------

CREATE TABLE IF NOT EXISTS dim_time (
  time_key                INT PRIMARY KEY,
  calendar_date           DATE NOT NULL,
  day_of_week             SMALLINT,
  is_weekend              BOOLEAN,
  week_of_year            SMALLINT,
  month                   SMALLINT,
  quarter                 SMALLINT,
  year                    SMALLINT,
  day_since_signup_bucket SMALLINT
);

CREATE TABLE IF NOT EXISTS dim_user (
  user_key           INT GENERATED ALWAYS AS IDENTITY PRIMARY
      KEY,
  user_id_nat        VARCHAR(64) NOT NULL UNIQUE,
  username           VARCHAR(64),
  signup_date        DATE,
  subscription_tier  VARCHAR(32),
```

```
  user_type           VARCHAR(32),
  region              VARCHAR(64),
  lifecycle_stage     VARCHAR(32)
);

CREATE TABLE IF NOT EXISTS dim_workspace (
  workspace_key          INT GENERATED ALWAYS AS IDENTITY
     PRIMARY KEY,
  workspace_id_nat       VARCHAR(64) NOT NULL UNIQUE,
  workspace_plan         VARCHAR(32),
  workspace_size_bucket  VARCHAR(32),
  industry_segment       VARCHAR(64),
  workspace_region       VARCHAR(64)
);

CREATE TABLE IF NOT EXISTS dim_content (
  content_key         INT GENERATED ALWAYS AS IDENTITY PRIMARY
      KEY,
  content_id_nat      VARCHAR(64) NOT NULL UNIQUE,
  content_type        VARCHAR(32),
  is_template_based   BOOLEAN,
  is_shared           BOOLEAN,
  ownership_type      VARCHAR(32)
);

CREATE TABLE IF NOT EXISTS dim_device (
  device_key          INT GENERATED ALWAYS AS IDENTITY PRIMARY
      KEY,
  device_id_nat       VARCHAR(64) NOT NULL UNIQUE,
  platform            VARCHAR(32),
  operating_system    VARCHAR(32),
  app_version         VARCHAR(32),
  device_form_factor  VARCHAR(32)
);

CREATE TABLE IF NOT EXISTS dim_event (
  event_key           INT GENERATED ALWAYS AS IDENTITY PRIMARY
      KEY,
  event_type          VARCHAR(32),
  feature_category    VARCHAR(64),
  interaction_intent  VARCHAR(64),
  UNIQUE (event_type, feature_category, interaction_intent)
);
```

```
CREATE TABLE IF NOT EXISTS dim_session (
  session_key        INT GENERATED ALWAYS AS IDENTITY PRIMARY
      KEY,
  session_id_nat     VARCHAR(64) NOT NULL UNIQUE,
  session_start_time TIMESTAMP,
  session_end_time   TIMESTAMP,
  session_origin     VARCHAR(64)
);

-- -------------------------
-- FACT
-- -------------------------

CREATE TABLE IF NOT EXISTS fact_product_usage_engagement (
  usage_id                 BIGINT GENERATED ALWAYS AS
     IDENTITY,
  time_key                 INT NOT NULL,
  user_key                 INT NOT NULL,
  workspace_key            INT NOT NULL,
  content_key              INT NOT NULL,
  device_key               INT NOT NULL,
  event_key                INT NOT NULL,
  session_key              INT NOT NULL,
  event_count              SMALLINT NOT NULL,
  active_user_flag         SMALLINT NOT NULL,
  activation_event_flag    SMALLINT NOT NULL,
  feature_usage_flag       SMALLINT NOT NULL,
  collaboration_event_flag SMALLINT NOT NULL,
  session_event_count      INT,
  session_duration_sec     INT,
  PRIMARY KEY (usage_id, time_key),
  CONSTRAINT fk_fact_time      FOREIGN KEY (time_key)
     REFERENCES dim_time(time_key),
  CONSTRAINT fk_fact_user      FOREIGN KEY (user_key)
     REFERENCES dim_user(user_key),
  CONSTRAINT fk_fact_workspace FOREIGN KEY (workspace_key)
     REFERENCES dim_workspace(workspace_key),
  CONSTRAINT fk_fact_content   FOREIGN KEY (content_key)
     REFERENCES dim_content(content_key),
  CONSTRAINT fk_fact_device    FOREIGN KEY (device_key)
     REFERENCES dim_device(device_key),
```

```
  CONSTRAINT fk_fact_event     FOREIGN KEY (event_key)
     REFERENCES dim_event(event_key),
  CONSTRAINT fk_fact_session   FOREIGN KEY (session_key)
     REFERENCES dim_session(session_key),
  CONSTRAINT ck_event_count CHECK (event_count IN (0,1)),
  CONSTRAINT ck_flags CHECK (
    active_user_flag IN (0,1)
    AND activation_event_flag IN (0,1)
    AND feature_usage_flag IN (0,1)
    AND collaboration_event_flag IN (0,1)
  )
) PARTITION BY RANGE (time_key);

-- -------------------------
-- INDEXES
-- -------------------------

CREATE INDEX IF NOT EXISTS idx_fact_user      ON
   fact_product_usage_engagement (user_key);
CREATE INDEX IF NOT EXISTS idx_fact_workspace ON
   fact_product_usage_engagement (workspace_key);
CREATE INDEX IF NOT EXISTS idx_fact_content   ON
   fact_product_usage_engagement (content_key);
CREATE INDEX IF NOT EXISTS idx_fact_device    ON
   fact_product_usage_engagement (device_key);
CREATE INDEX IF NOT EXISTS idx_fact_event     ON
   fact_product_usage_engagement (event_key);
CREATE INDEX IF NOT EXISTS idx_fact_session   ON
   fact_product_usage_engagement (session_key);
CREATE INDEX IF NOT EXISTS idx_fact_time      ON
   fact_product_usage_engagement (time_key);

CREATE INDEX IF NOT EXISTS idx_user_nat       ON dim_user (
   user_id_nat);
CREATE INDEX IF NOT EXISTS idx_ws_nat         ON
   dim_workspace (workspace_id_nat);
CREATE INDEX IF NOT EXISTS idx_content_nat    ON dim_content
   (content_id_nat);
CREATE INDEX IF NOT EXISTS idx_device_nat     ON dim_device (
   device_id_nat);
CREATE INDEX IF NOT EXISTS idx_session_nat    ON dim_session
   (session_id_nat);
```

Script for populating the data warehouse

```
SET search_path = notion_dw;

-- ============================================================
-- (1) Ensure partitions for 2024-01 .. 2025-12 (monthly)
--     Required because fact_product_usage_engagement is
   PARTITION BY RANGE(time_key)
-- ============================================================
DO $$
DECLARE
  m_start DATE;
  m_end   DATE;
  p_from  INT;
  p_to    INT;
  p_name  TEXT;
BEGIN
  m_start := DATE '2024-01-01';

  WHILE m_start <= DATE '2025-12-01' LOOP
    m_end := (m_start + INTERVAL '1 month')::DATE;

    p_from := (EXTRACT(YEAR FROM m_start)::INT * 10000
               + EXTRACT(MONTH FROM m_start)::INT * 100
               + EXTRACT(DAY FROM m_start)::INT);

    p_to := (EXTRACT(YEAR FROM m_end)::INT * 10000
             + EXTRACT(MONTH FROM m_end)::INT * 100
             + EXTRACT(DAY FROM m_end)::INT);

    p_name := format('fact_p_%s', to_char(m_start, 'YYYYMM'))
       ;

    EXECUTE format(
      'CREATE TABLE IF NOT EXISTS %I PARTITION OF
         fact_product_usage_engagement FOR VALUES FROM (%s) TO
          (%s);',
      p_name, p_from, p_to
    );

    m_start := m_end;
  END LOOP;
END $$;
```

```
-- ============================================================
-- (2) TIME DIMENSION: 2024-01-01 .. 2025-12-31
-- ============================================================
INSERT INTO dim_time (
  time_key, calendar_date, day_of_week, is_weekend,
  week_of_year, month, quarter, year, day_since_signup_bucket
)
SELECT
  (EXTRACT(YEAR FROM d)::INT * 10000
   + EXTRACT(MONTH FROM d)::INT * 100
   + EXTRACT(DAY FROM d)::INT) AS time_key,
  d::DATE AS calendar_date,
  EXTRACT(ISODOW FROM d)::SMALLINT AS day_of_week,
  (EXTRACT(ISODOW FROM d) IN (6,7)) AS is_weekend,
  EXTRACT(WEEK FROM d)::SMALLINT AS week_of_year,
  EXTRACT(MONTH FROM d)::SMALLINT AS month,
   EXTRACT(QUARTER FROM d)::SMALLINT AS quarter,
   EXTRACT(YEAR FROM d)::SMALLINT AS year,
   NULL::SMALLINT AS day_since_signup_bucket
FROM generate_series(DATE '2024-01-01', DATE '2025-12-31',
   INTERVAL '1 day') AS g(d)
ON CONFLICT (time_key) DO NOTHING;

-- ============================================================
-- (3) DEVICE DIMENSION: small fixed set
-- ============================================================
INSERT INTO dim_device (device_id_nat, platform,
   operating_system, app_version, device_form_factor)
VALUES
  ('dev_web_win', 'web',     'Windows', '3.08.0', 'desktop'),
  ('dev_web_mac', 'web',     'macOS',   '3.08.1', 'desktop'),
  ('dev_desktop', 'desktop', 'macOS',   '3.09.0', 'desktop'),
  ('dev_ios',     'mobile',  'iOS',     '3.07.9', 'phone'),
  ('dev_android', 'mobile',  'Android', '3.08.2', 'phone')
ON CONFLICT (device_id_nat) DO NOTHING;

-- ============================================================
-- (4) CONTENT DIMENSION: small fixed set
-- ============================================================
INSERT INTO dim_content (content_id_nat, content_type,
   is_template_based, is_shared, ownership_type)
VALUES
```

```
  ('cnt_page',      'page',       FALSE, FALSE, 'personal'),
  ('cnt_database',  'database',   TRUE,  TRUE,  'workspace'),
  ('cnt_wiki',      'wiki',       TRUE,  TRUE,  'team_space')
    ,
  ('cnt_taskboard', 'task_board', FALSE, TRUE,  'workspace'),
  ('cnt_template',  'page',       TRUE,  FALSE, 'personal')
ON CONFLICT (content_id_nat) DO NOTHING;

-- ============================================================
-- (5) EVENT DIMENSION: small fixed set
-- ============================================================
INSERT INTO dim_event (event_type, feature_category,
   interaction_intent)
VALUES
  ('view',         'Core',          'consumption'),
  ('create',       'Core',          'creation'),
  ('edit',         'Core',          'creation'),
  ('comment',      'Collaboration', 'collaboration'),
  ('share',        'Collaboration', 'collaboration'),
  ('db_query',     'Databases',     'consumption'),
  ('api_call',     'Integrations',  'creation'),
  ('template_use', 'Templates',     'creation')
ON CONFLICT (event_type, feature_category, interaction_intent
   ) DO NOTHING;

-- ============================================================
-- (6) USER DIMENSION: monthly cohorts (480 users)
--     - 20 users per month from Jan 2024 through Dec 2025
   (24 months).
--     - Tier mix spread evenly; lifecycle varied.
-- ============================================================
INSERT INTO dim_user (
  user_id_nat, username, signup_date, subscription_tier,
  user_type, region, lifecycle_stage
)
SELECT
  'usr_' || LPAD(i::TEXT, 4, '0') AS user_id_nat,
  'user_' || LPAD(i::TEXT, 4, '0') AS username,
  (DATE '2024-01-05'
   + (((i-1) / 20) * INTERVAL '1 month')
   + ((i-1) % 20) * INTERVAL '1 day')::date AS signup_date,
  (ARRAY['Free','Plus','Business','Enterprise'])[(i % 4) + 1]
      AS subscription_tier,
```

```
  (ARRAY['individual','member','guest'])[(i % 3) + 1] AS
     user_type,
  (ARRAY['EU','NA','APAC'])[(i % 3) + 1] AS region,
  (ARRAY['onboarding','active','active','dormant'])[(i % 4) +
      1] AS lifecycle_stage
FROM generate_series(1,480) i
ON CONFLICT (user_id_nat) DO NOTHING;

-- ============================================================
-- (7) WORKSPACE DIMENSION: >= 50 workspaces
-- ============================================================
INSERT INTO dim_workspace (
  workspace_id_nat, workspace_plan,
  workspace_size_bucket, industry_segment, workspace_region
)
SELECT
  'ws_' || LPAD(i::TEXT, 3, '0') AS workspace_id_nat,
  (ARRAY['Free','Plus','Business','Enterprise'])[1 + (random
     ()*3)::INT] AS workspace_plan,
  (ARRAY['1','2--10','11--50','50+'])[1 + (random()*3)::INT]
     AS workspace_size_bucket,
  (ARRAY['Education','Software','Finance','Consulting'])[1 +
     (random()*3)::INT] AS industry_segment,
  (ARRAY['EU','NA'])[1 + (random())::INT] AS workspace_region
FROM generate_series(1,50) i
ON CONFLICT (workspace_id_nat) DO NOTHING;

-- ============================================================
-- (8) SESSION DIMENSION: dense coverage (5 sessions per day)
--     - session_start_time within 2024-01-01 .. 2025-12-31
--     - duration 20..70 minutes
-- ============================================================
INSERT INTO dim_session (
  session_id_nat, session_start_time, session_end_time,
     session_origin
)
SELECT
  'sess_' || LPAD(i::TEXT, 5, '0') AS session_id_nat,
  ts AS session_start_time,
  LEAST(
    ts + ((1200 + random()*3000)::INT || ' seconds')::
       INTERVAL,
    TIMESTAMP '2025-12-31 23:59:59'
```

```
  ) AS session_end_time,
  (ARRAY['direct','link','notification'])[1 + (random()*2)::
     INT] AS session_origin
FROM (
  SELECT
    i,
    (TIMESTAMP '2024-01-01'
     + ((i / 5)::INT || ' days')::INTERVAL  -- walk the
        calendar
     + (random() * INTERVAL '12 hours')) AS ts
  FROM generate_series(0, (731*5)) i  -- ~5 sessions per day
     across full range
) s
ON CONFLICT (session_id_nat) DO NOTHING;

-- ============================================================
-- (9) FACT TABLE: Structured activity (activation + ongoing
   engagement)
--     - Activation events: up to 2 per activating user
   within 7 days of signup.
--     - Ongoing events: tier-weighted monthly volume with
   content/device bias.
-- ============================================================
WITH months AS (
  SELECT date_trunc('month', d)::date AS month_start
  FROM generate_series(DATE '2024-01-01', DATE '2025-12-01',
     INTERVAL '1 month') d
),
month_spans AS (
  SELECT
    month_start,
    (month_start + INTERVAL '1 month' - INTERVAL '1 day')::
       date AS month_end
  FROM months
),
user_profile AS (
  SELECT
    u.user_key,
    u.signup_date,
    u.subscription_tier,
    CASE u.subscription_tier
      WHEN 'Free'        THEN 6
      WHEN 'Plus'        THEN 10
```

```
      WHEN 'Business'   THEN 16
      ELSE                   20
    END AS monthly_base,
    CASE u.subscription_tier
      WHEN 'Free'       THEN 0.25
      WHEN 'Plus'       THEN 0.40
      WHEN 'Business'   THEN 0.55
      ELSE                   0.70
    END AS activation_prob,
    -- deterministic activation choice per user based on md5(
       user_id_nat)
    (
      (('x' || substr(md5(u.user_id_nat), 1, 8))::bit(32)::
         int % 100)::numeric / 100.0
      < CASE u.subscription_tier
          WHEN 'Free'       THEN 0.25
          WHEN 'Plus'       THEN 0.40
          WHEN 'Business'   THEN 0.55
          ELSE                   0.70
        END
    ) AS activates
  FROM dim_user u
),
activation_events AS (
  SELECT
    up.user_key,
    GREATEST(up.signup_date, DATE '2024-01-01') + (g-1) AS
       event_date,
    'activation'::TEXT AS stage
  FROM user_profile up
  CROSS JOIN generate_series(1,2) g
  WHERE up.activates
    AND up.signup_date + (g-1) <= DATE '2025-12-31'
),
recurring_volume AS (
  SELECT
    up.user_key,
    up.signup_date,
    up.subscription_tier,
    ms.month_start,
    ms.month_end,
    up.monthly_base
```

```
      + (CASE WHEN up.subscription_tier IN ('Business','
         Enterprise') THEN 6 ELSE 0 END)
      + (random()*4)::INT  AS monthly_events
  FROM user_profile up
  JOIN month_spans ms
    ON ms.month_start >= date_trunc('month', up.signup_date)
   AND ms.month_start <= DATE '2025-12-01'
),
recurring_events AS (
  SELECT
    rv.user_key,
    rv.subscription_tier,
    rv.signup_date,
    rv.month_start,
    rv.month_end,
    rv.monthly_events,
    generate_series(1, rv.monthly_events) AS idx
  FROM recurring_volume rv
),
event_pool AS (
  -- Merge activation and ongoing events into one set with
     event_date and attributes.
  SELECT
    e.user_key,
    GREATEST(e.signup_date, e.month_start)
      + ((random() * (e.month_end - GREATEST(e.signup_date, e
         .month_start)))::INT) AS event_date,
    e.subscription_tier,
    FALSE AS is_activation_stage
  FROM recurring_events e
  WHERE GREATEST(e.signup_date, e.month_start) <= e.month_end

  UNION ALL

  SELECT
    a.user_key,
    a.event_date,
    up.subscription_tier,
    TRUE AS is_activation_stage
  FROM activation_events a
  JOIN dim_user up ON up.user_key = a.user_key
),
event_enriched AS (
```

```
  SELECT
    ep.*,
    (ep.event_date
     + (random() * INTERVAL '20 hours')) AS event_ts,  --
        spread across the day
    -- Content mix: team spaces heavier on wikis/databases/
       task boards.
    CASE
      WHEN random() < 0.28 THEN 'cnt_database'
      WHEN random() < 0.52 THEN 'cnt_wiki'
      WHEN random() < 0.72 THEN 'cnt_taskboard'
      WHEN random() < 0.88 THEN 'cnt_page'
      ELSE                      'cnt_template'
    END AS content_id_nat,
    -- Device mix: desktop/web more common for work, mobile
       for lighter use.
    CASE
      WHEN random() < 0.35 THEN 'dev_desktop'
      WHEN random() < 0.60 THEN 'dev_web_win'
      WHEN random() < 0.75 THEN 'dev_web_mac'
      WHEN random() < 0.88 THEN 'dev_ios'
      ELSE                      'dev_android'
    END AS device_id_nat,
    -- Event mix: creation/collaboration weighted for richer
       engagement.
    CASE
      WHEN random() < 0.18 THEN 'create'
      WHEN random() < 0.36 THEN 'edit'
      WHEN random() < 0.48 THEN 'comment'
      WHEN random() < 0.60 THEN 'share'
      WHEN random() < 0.72 THEN 'db_query'
      WHEN random() < 0.84 THEN 'template_use'
      WHEN random() < 0.92 THEN 'api_call'
      ELSE                      'view'
    END AS event_type
  FROM event_pool ep
)
INSERT INTO fact_product_usage_engagement (
  time_key,
  user_key,
  workspace_key,
  content_key,
  device_key,
```

```
  event_key,
  session_key,
  event_count,
  active_user_flag,
  activation_event_flag,
  feature_usage_flag,
  collaboration_event_flag,
  session_event_count,
  session_duration_sec
)
SELECT
  t.time_key,
  ev.user_key,
  w.workspace_key,
  c.content_key,
  d.device_key,
  e.event_key,
  s.session_key,
  1 AS event_count,
  CASE
    WHEN ev.is_activation_stage THEN 1
    WHEN ((EXTRACT(DAY FROM ev.event_date))::INT % 5) = 0
       THEN 1
    ELSE 0
  END AS active_user_flag,
  CASE
    WHEN ev.is_activation_stage THEN 1
    ELSE 0
  END AS activation_event_flag,
  CASE
    WHEN e.event_type IN ('create','edit','db_query','
       template_use','api_call') THEN 1
    WHEN random() < 0.20 THEN 1 ELSE 0
  END AS feature_usage_flag,
  CASE
    WHEN e.event_type IN ('comment','share') THEN 1
    WHEN c.content_type IN ('wiki','task_board','database')
       AND random() < 0.65 THEN 1
    ELSE 0
  END AS collaboration_event_flag,
  GREATEST(2, 2 + (random()*12)::INT) AS session_event_count,
  CASE d.platform
    WHEN 'desktop' THEN 2400 + (random()*800)::INT
```

```
    WHEN 'web'     THEN 1800 + (random()*600)::INT
    ELSE                900 + (random()*500)::INT
  END AS session_duration_sec
FROM event_enriched ev
JOIN user_profile up ON up.user_key = ev.user_key
JOIN dim_time t      ON t.calendar_date = ev.event_date
JOIN dim_content c   ON c.content_id_nat = ev.content_id_nat
JOIN dim_device d    ON d.device_id_nat = ev.device_id_nat
JOIN dim_event e     ON e.event_type = ev.event_type
CROSS JOIN LATERAL (
  SELECT workspace_key FROM dim_workspace ORDER BY random()
    LIMIT 1
) w
CROSS JOIN LATERAL (
  SELECT session_key
  FROM dim_session s
  WHERE ev.event_date BETWEEN s.session_start_time::date AND
    s.session_end_time::date
  ORDER BY random()
  LIMIT 1
) s;
```