

# Creating High-Quality Seed Data for Asana RL Environment

January 6, 2026

**Author:** Raghav Borikar

**Affiliation:** Indian Institute of Technology, Bhilai

**GitHub Repository:** <https://github.com/Raghav-Borikar/Asana-Simulation>

**Date:** January 6, 2026

## Overview

This document describes the design and generation of a high-quality synthetic dataset simulating a large enterprise Asana workspace. The dataset is intended to serve as seed data for a reinforcement learning environment, enabling realistic evaluation and fine-tuning of computer-use AI agents on project management workflows.

The simulation represents a B2B SaaS company with approximately 8,000 employees, spanning engineering, product, marketing, sales, and operations teams, with six months of historical activity. Particular emphasis is placed on data realism, temporal consistency, and relational integrity, as these properties are critical to avoiding shortcut learning in downstream reinforcement learning agents.

## 1 Database Schema

### 1.1 Relational Schema Overview

The database schema models the core entities and relationships present in a real Asana workspace. It includes the following tables:

- Workspaces
- Teams
- Users
- Team Memberships

- Projects
- Sections
- Tasks
- Task Hierarchy (Subtasks)
- Comments
- Custom Field Definitions
- Custom Field Values
- Tags
- Task–Tag Associations

The schema is implemented in SQLite and enforces referential integrity using primary and foreign keys.

## 1.2 Table Definitions

### Workspaces

Represents the top-level Asana workspace.

Column	Type	Description
id	TEXT (PK)	Workspace identifier (UUID)
name	TEXT	Workspace name
domain	TEXT	Verified email domain
created_at	TIMESTAMP	Workspace creation time

### Teams

Represents functional groups within the workspace.

Column	Type	Description
id	TEXT (PK)	Team identifier
workspace.id	TEXT (FK)	Parent workspace
name	TEXT	Team name
department	TEXT	Functional area (Engineering, Marketing, etc.)

### Users

Represents members of the workspace.

Column	Type	Description
id	TEXT (PK)	User identifier
workspace_id	TEXT (FK)	Workspace membership
email	TEXT	Corporate email
full_name	TEXT	User name
role	TEXT	Job role
status	TEXT	active / inactive / invited

## Team Memberships

Many-to-many relationship between users and teams.

Column	Type	Description
team_id	TEXT (FK)	Team
user_id	TEXT (FK)	User
role	TEXT	Member role within team

Primary Key: (team\_id, user\_id)

## Projects

Represents initiatives or workstreams owned by teams.

Column	Type	Description
id	TEXT (PK)	Project identifier
workspace_id	TEXT (FK)	Workspace
team_id	TEXT (FK)	Owning team
name	TEXT	Project name
status	TEXT	On Track / At Risk / Off Track / On Hold / Complete
created_at	TIMESTAMP	Project start time
due_date	DATE	Target completion date

## Sections

Represents workflow stages within a project.

Column	Type	Description
id	TEXT (PK)	Section identifier
project_id	TEXT (FK)	Parent project
name	TEXT	Section name
order_index	INTEGER	Display order

## Tasks

Fundamental unit of work.

Column	Type	Description
id	TEXT (PK)	Task identifier
project_id	TEXT (FK)	Parent project
section_id	TEXT (FK)	Current section
parent_task_id	TEXT (FK)	Parent task (for subtasks)
assignee_id	TEXT (FK)	Assigned user
name	TEXT	Task title
description	TEXT	Task description
completed	BOOLEAN	Completion flag
completed_at	TIMESTAMP	Completion time
due_date	DATE	Due date
created_at	TIMESTAMP	Creation time

## Comments

Represents discussion and activity on tasks.

Column	Type	Description
id	TEXT (PK)	Comment identifier
task_id	TEXT (FK)	Task
user_id	TEXT (FK)	Comment author
text	TEXT	Comment content
created_at	TIMESTAMP	Comment timestamp

## Custom Field Definitions

Defines extensible metadata fields.

Column	Type	Description
id	TEXT (PK)	Field identifier
workspace_id	TEXT (FK)	Workspace
name	TEXT	Field name
type	TEXT	text / number / enum

## Custom Field Values

Stores task-specific custom field values.

Column	Type	Description
task_id	TEXT (FK)	Task
field_id	TEXT (FK)	Field definition
value_text	TEXT	Text value
value_number	REAL	Numeric value

## Tags and Task–Tag Associations

Supports cross-project categorization.

Table	Description
tags	Tag definitions
task_tags	Many-to-many task-tag mapping

### 1.3 Entity–Relationship Diagram

An ER diagram was created using dbdiagram.io and is included in this document. The diagram highlights:

- Workspace-centric organization
- Many-to-many team memberships
- Project → Section → Task hierarchy
- Self-referential task hierarchy for subtasks
- Extensible metadata via custom fields
- Cross-project tagging

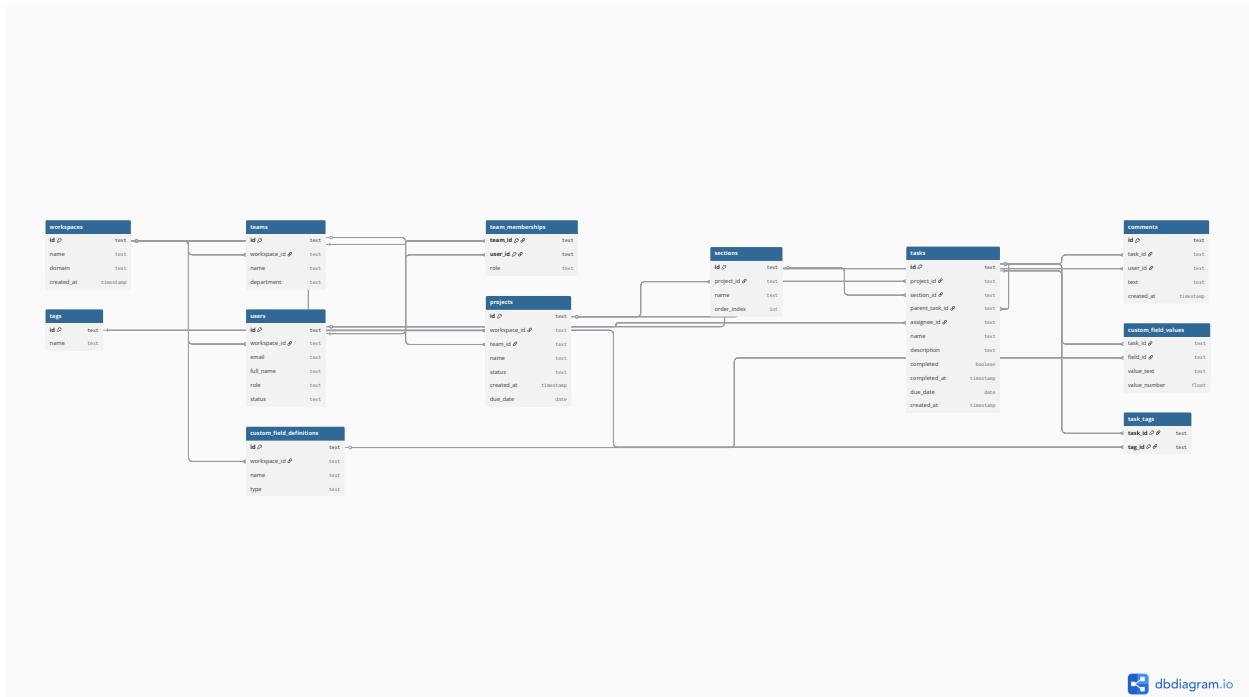


Figure 1: Entity–Relationship Diagram for the Asana Workspace Schema

As shown in Figure 1, the schema centers around the workspace entity with downstream project and task hierarchies.

## 1.4 Key Design Decisions

### Custom Fields

Custom fields are modeled using a two-table design (definitions and values). This allows:

- Different projects to reuse the same fields
- Sparse population of fields across tasks
- Future extensibility without schema changes

This mirrors how Asana supports project-specific metadata while maintaining a global field registry.

### Task Hierarchy (Tasks vs. Subtasks)

Tasks and subtasks are represented using a self-referential foreign key (`parent_task_id`) rather than a separate subtasks table.

This approach:

- Reflects Asana's internal task model
- Allows arbitrary nesting depth
- Avoids schema duplication
- Preserves relational integrity

Subtasks are distinguished at the data level by a non-null `parent_task_id` and constrained generation logic.

## 2 Seed Data Methodology

This section describes the column-by-column data generation strategy for each table, with emphasis on realism, distributions, and consistency.

### Table: users

Column	Data Type	Source Strategy	Methodology & Justification
id	TEXT (UUID)	Generated	UUIDv4 to simulate Asana GIDs
full_name	TEXT	Synthetic	Generated using census-style name distributions
email	TEXT	Derived	{first}.{last}@company.com
role	TEXT	Heuristic	Role titles aligned with team department
status	TEXT	Synthetic	90% active, 8% inactive, 2% invited

Inactive users represent attrition, leave, or contractors commonly observed in enterprise workspaces.

**Table: teams**

Column	Data Type	Strategy
id	TEXT	UUID
name	TEXT	Department-specific templates
department	TEXT	Engineering, Product, Marketing, Sales, Operations

Team sizes follow a non-uniform distribution, with most teams having 6–10 members and a long tail of larger teams.

**Table: projects**

Column	Data Type	Strategy
name	TEXT	Template-based
status	TEXT	Weighted enum
created_at	TIMESTAMP	Spread over 6-month window
due_date	DATE	Optional, project-type dependent

Project naming patterns are inspired by public Asana templates, GitHub project boards, and common enterprise conventions (e.g., “Q3 Platform Stability”, “Fall Campaign Launch”).

**Table: tasks**

Column	Data Type	Source Strategy	Methodology & Justification
id	TEXT	Generated	UUIDv4
name	TEXT	Heuristics	Department-specific templates
description	TEXT	Templates	20% empty, 50% short, 30% detailed
assignee_id	TEXT (FK)	Derived	15% unassigned; otherwise team-constrained
due_date	DATE	Synthetic	10% none, 25% $\leq 7d$ , 40% $\leq 30d$ , 25% long-horizon
created_at	TIMESTAMP	Synthetic	Business-day weighted
completed	BOOLEAN	Heuristic	Completion rate varies by department
completed_at	TIMESTAMP	Derived	After creation, before due date

### Completion Rates (by Department)

Department	Completion Rate
Engineering	~80%
Product	~75%
Marketing	~70%
Sales	~65%
Operations	~50%

This reflects differences between sprint-based work, campaign-based work, and ongoing operational tasks.

## Table: comments

Column	Strategy
text	Template-based
created_at	After task creation

Most tasks have no comments; a minority have multiple comments, reflecting realistic collaboration sparsity.

## Table: custom fields

Custom fields such as Priority, Effort, and Status are defined at the workspace level. Values are populated for only a subset of tasks ( $\approx 40\text{--}60\%$ ) to avoid unrealistic uniformity.

## Scraped / Real-World Inspiration Sources

While no proprietary data is used, generation strategies are informed by:

- Public Asana templates and community examples
- GitHub issue trackers (task naming patterns)
- Industry reports on team size and productivity
- Asana “Anatomy of Work” reports for completion behavior

## LLM Usage Decision

Large Language Models can generate realistic text but were intentionally not used during seed data generation.

This decision was made to prioritize:

- **Determinism** – identical inputs always produce identical datasets
- **Reproducibility** – critical for controlled RL experiments
- **Performance** – fast regeneration at scale without API latency
- **Stability** – avoidance of stochastic drift across dataset versions

Instead, realistic linguistic patterns were encoded using curated templates, probabilistic variation, and department-specific heuristics, preserving realism while ensuring full control over distributions.

## Temporal Consistency Guarantees

The generation pipeline enforces:

- Tasks are never completed before creation
- Completion timestamps are always after creation and before “now”

- Comments occur after task creation
- Due dates are consistent with creation time

## Relational Consistency Guarantees

The pipeline ensures:

- Tasks belong to valid projects and sections
- Assignees belong to the same workspace
- Subtasks reference existing parent tasks
- Custom field values reference valid definitions
- Tag associations reference existing tasks and tags

Foreign key constraints are enabled in SQLite to enforce integrity.

## Conclusion

The resulting dataset closely mirrors a real enterprise Asana workspace in structure, scale, and behavioral patterns. By emphasizing realism, consistency, and reproducibility, this dataset is well-suited for training and evaluating reinforcement learning agents on realistic computer-use workflows, while avoiding shortcut learning and distributional artifacts.