Understood. I will create a complete, up-to-date and Django-aligned canonical definition of all custom database tables in your system, preserving all useful context and aligning with the v14 architecture and documentation.

This will include table names, Django model classes, purposes, key columns (including FK/PK/ENUMs), and references to your functional specs when relevant. I'll compile and structure everything so it can replace your current canonical table definition without ambiguity or omission.

I'll get started and will notify you when the compiled document is ready.

# **Konnaxion v14 – Custom Database Tables (Canonical List)**

*(Tables provided by Django or boilerplate (e.g. auth\_user, django\_admin\_log, socialaccount\_\*) are omitted. Each entry lists the* ***Display Name → Model Class****, followed by a brief purpose and key columns (primary keys, foreign keys, enums, JSON fields; standard timestamps omitted unless notable). Structure and naming follow the v14 modular architecture as implemented.)*

## **Kollective Intelligence**

### **EkoH (Expertise & Reputation Domain)**

* **Expertise Categories → ExpertiseCategory:** Catalog of knowledge domains used to classify expertise. **Key columns:** id (PK), name (unique domain name).
* **User Expertise Scores → UserExpertiseScore:** Each user’s current expertise score per domain. **Key columns:** id (PK), user (FK to User), category (FK to ExpertiseCategory), raw\_score, weighted\_score.
* **User Ethics Scores → UserEthicsScore:** Ethical weight multiplier applied to a user’s scores. **Key columns:** user (OneToOne FK to User, also PK), ethical\_score (numeric value).
* **Score Configurations → ScoreConfiguration:** Named weight parameters (global or field-specific) for score calculations. **Key columns:** id (PK), weight\_name (e.g. parameter name), weight\_value, field (nullable, identifies which field the weight applies to).
* **Context Analysis Logs → ContextAnalysisLog:** Log of AI-driven adjustments to scores in context. **Key columns:** id (PK), entity\_type (model or context identifier), entity\_id, field (name of score field adjusted), input\_metadata (JSON of context data), adjustments\_applied (JSON of changes made).
* **Confidentiality Settings → ConfidentialitySetting:** Per-user privacy level preference for displaying identity alongside scores. **Key columns:** user (OneToOne FK to User, also PK), level (ENUM – public / pseudonym / anonymous).
* **Score History → ScoreHistory:** Audit trail of all score changes for transparency. **Key columns:** id (PK), merit\_score (FK to UserExpertiseScore), old\_value, new\_value, change\_reason.

### **Smart Vote (Weighted Voting System)**

* **Votes → Vote:** Records each user vote with both raw and weighted values. **Key columns:** id (PK), user (FK), target\_type (string identifier of the content/user being voted on), target\_id (ID of target entity), raw\_value (e.g. vote value before weighting), weighted\_value (value after EkoH-based weighting).
* **Vote Modalities → VoteModality:** Defines parameters for various voting modes (approval, ranking, rating, etc.). **Key columns:** id (PK), name (unique modality name), parameters (JSON field storing settings for this voting mode).
* **Emerging Experts → EmergingExpert:** Flags users who are rapidly gaining expertise (merit) scores. **Key columns:** id (PK), user (FK), detection\_date (date flagged), score\_delta (recent increase in score).
* **Vote Results → VoteResult:** Aggregated result of votes per target (e.g. total weighted score). **Key columns:** id (PK), target\_type, target\_id, sum\_weighted\_value (cumulative weighted score), vote\_count (number of votes aggregated).
* **Integration Mappings → IntegrationMapping:** Links Smart Vote context to other modules’ objects for cross-module voting. **Key columns:** id (PK), module\_name (target module identifier), context\_type (target object type), mapping\_details (JSON with mapping info).

## **ethiKos**

### **Korum (Structured Debates Platform)**

* **Debate Categories → EthikosCategory:** Thematic categories for debates (e.g. Politics, Ethics, etc.). **Key columns:** id (PK), name (unique category name), description (optional).
* **Debates → EthikosTopic:** High-level debate topics or questions created by users (one per debate). **Key columns:** id (PK), title (debate question or title), status (ENUM – e.g. open/closed/archived), start\_date, end\_date (optional scheduling of debate period).
* **Stances → EthikosStance:** Each user’s stated stance or position on a given debate topic. **Key columns:** id (PK), topic (FK to EthikosTopic), user (FK), value (integer value representing stance, constrained between -3 and +3).
* **Debate Arguments → EthikosArgument:** User-submitted arguments/posts within a debate thread. **Key columns:** id (PK), topic (FK to EthikosTopic), author (FK to User), content (text of the argument), parent (FK to another EthikosArgument for threaded replies, nullable), side (optional ENUM flag like “pro”/“con”).

*(The following planned tables for AI-driven debate features were outlined but are not present in the current implementation and are omitted in this list: AI Clones, Comparative Analysis Logs, Debate Archives, Debate Summaries.)*

### **Konsultations (Public Consultations & Feedback)**

* **Consultations → Consultation:** Public consultation instance (e.g. a time-bound survey or call for feedback). **Key columns:** id (PK), title, open\_date, close\_date, status (ENUM – open/closed/archived).
* **Citizen Suggestions → CitizenSuggestion:** User-submitted ideas or proposals within a consultation. **Key columns:** id (PK), consultation (FK to Consultation), author (FK to User), content (suggestion text).
* **Consultation Votes → ConsultationVote:** Votes on consultation proposals, with raw and weighted values (if EkoH weighting applied). **Key columns:** id (PK), user (FK), consultation (FK), raw\_value, weighted\_value.
* **Consultation Results → ConsultationResult:** Stores aggregated voting outcomes for a consultation. **Key columns:** id (PK), consultation (FK), results\_data (JSONB snapshot of vote totals or statistics).
* **Impact Track → ImpactTrack:** Post-consultation action log to track implementation of accepted proposals. **Key columns:** id (PK), consultation (FK), action (description of follow-up action), status (status of the action), date (when logged).

*Here is a* ***ready-to-paste documentation table*** *for all actual keenkonnect models found in your backend code. This is structured for easy integration into your canonical reference file and aligns with Django best practices. Each model includes its* ***purpose****, main fields (with PK/FK/ENUM/constraints), and notes.*

## ***KeenKonnect Module – Canonical Database Table Reference (Fully Synced to Code)***

***Note:*** *Fields like created\_at/updated\_at are omitted unless nonstandard. All ForeignKeys are to users.User unless otherwise noted.*

### ***Project***

***Purpose:*** *Container for a collaborative project workspace (formerly “CollaborationSpace”).*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *title* | *CharField(255)* | *Project title* |
| *description* | *TextField* | *Optional project description* |
| *creator* | *FK(User)* | *Who created this project* |
| *category* | *CharField(120)* | *e.g. “Energy”, “Education”, etc.* |
| *status* | *CharField(16, ENUM)* | *e.g. draft, active, completed, archived* |
| *created\_at* | *DateTimeField* | *Auto-added* |
| *updated\_at* | *DateTimeField* | *Auto-updated* |

### ***ProjectResource***

***Purpose:*** *Links documents, files, or resources to a project.*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *project* | *FK(Project)* | *Parent project* |
| *title* | *CharField(255)* | *Resource name* |
| *url* | *URLField* | *Resource URL/path* |
| *added\_by* | *FK(User)* | *Uploader* |
| *created\_at* | *DateTimeField* |  |

### ***ProjectTask***

***Purpose:*** *A task, to-do, or milestone in a project (Kanban or similar).*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *project* | *FK(Project)* | *Parent project* |
| *title* | *CharField(255)* | *Task summary* |
| *description* | *TextField* | *Task details* |
| *assignee* | *FK(User, nullable)* | *Assigned user* |
| *status* | *CharField(16, ENUM)* | *e.g. todo, in\_progress, done, blocked* |
| *due\_date* | *DateField (nullable)* | *Optional due date* |
| *created\_at* | *DateTimeField* |  |
| *updated\_at* | *DateTimeField* |  |

### ***ProjectMessage***

***Purpose:*** *A message in a project chat/thread.*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *project* | *FK(Project)* | *Parent project* |
| *sender* | *FK(User)* | *Message author* |
| *content* | *TextField* | *Message text* |
| *created\_at* | *DateTimeField* |  |

### ***ProjectTeam***

***Purpose:*** *Defines project team membership and roles.*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *project* | *FK(Project)* | *Parent project* |
| *user* | *FK(User)* | *Team member* |
| *role* | *CharField(32)* | *e.g. owner, member, advisor, etc.* |
| *joined\_at* | *DateTimeField* |  |

### ***ProjectRating***

***Purpose:*** *Stores ratings/reviews for a project.*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *project* | *FK(Project)* | *Rated project* |
| *user* | *FK(User)* | *Reviewer* |
| *rating* | *IntegerField* | *e.g. 1–5* |
| *comment* | *TextField* | *Optional feedback* |
| *created\_at* | *DateTimeField* |  |

### ***Tag***

***Purpose:*** *Reusable keyword for projects or tasks.*

| ***Field*** | ***Type*** | ***Notes*** |
| --- | --- | --- |
| *id* | *BigAutoField (PK)* | *Primary key* |
| *name* | *CharField(64, uniq)* | *Tag label (unique)* |

## ***Notes***

* *Models like RealTimeDocument, ChatMessage, VideoSession, and AIInteractionLog are not present in the current codebase (under these names). If you require them for future features, add them to both code and doc.*
* *If you use ManyToMany relationships (e.g., tags on projects/tasks), document them in the canonical file.*
* *Each model’s “purpose” should be updated as your platform evolves.*

## **KonnectED**

### **CertifiKation (Skills & Certification)**

* **Certification Paths → CertificationPath:** Defines a structured learning or certification path (a sequence of skills/lessons to master). **Key columns:** id (PK), name (name of the certification or learning path), description (textual description of the path).
* **Evaluations → Evaluation:** Stores results of an automated or manual evaluation for a user on a certification path. **Key columns:** id (PK), user (FK), path (FK to CertificationPath), raw\_score (numeric score achieved), metadata (JSON field with additional details, e.g. answers or scoring breakdown).
* **Peer Validations → PeerValidation:** Records a peer mentor’s validation decision on a user’s submitted evidence for a skill (part of the certification). **Key columns:** id (PK), evaluation (FK to Evaluation being reviewed), peer (FK to User acting as validator), decision (ENUM – approved or rejected).

### **Module: KonnectED**

#### **Portfolios → Portfolio**

**Purpose:** User skill showcase. Each portfolio is a curated collection of KnowledgeResources that demonstrate a user’s skills, achievements, or project evidence.

**Fields:**

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| id | BigAutoField (PK) | Primary key |
| user | ForeignKey(User) | Owner of the portfolio |
| title | CharField(255) | Portfolio title |
| description | TextField (optional) | Description of the portfolio |
| items | ManyToManyField(KnowledgeResource, blank=True, related\_name="portfolios") | Collection of evidence artefacts (documents, videos, etc.) |
| created\_at | DateTimeField (auto\_now\_add) | Timestamp |
| updated\_at | DateTimeField (auto\_now) | Timestamp |

**Notes:**

* items links to existing KnowledgeResource objects.
* To support ad-hoc/loose artefacts in future, consider an auxiliary JSONField or Attachment model (not currently present).

**Example Table Structure (for documentation):**

| **id** | **user** | **title** | **description** | **items (M2M)** | **created\_at** | **updated\_at** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 12 | Data Viz | DS work | [3, 7, 12] | ... | ... |

* **Interop Mappings → InteropMapping:** Mapping of internal certifications to external systems’ identifiers (for LMS interoperability). **Key columns:** id (PK), local\_certification (FK to CertificationPath), external\_system (name of external platform), external\_id (identifier of equivalent certification on external system).

### **Knowledge (Collaborative Learning Library)**

* **Knowledge Resources → KnowledgeResource:** Metadata for a shared learning resource (article, video, course, etc.) in the KonnectED library. **Key columns:** id (PK), title, type (ENUM – e.g. video, doc, course, other), url (link or location of the resource), author (FK to User who added it, nullable).
* **Knowledge Recommendations → KnowledgeRecommendation:** Records that a particular resource was recommended to a user (often by an ML algorithm or expert). **Key columns:** id (PK), user (FK – recommendation recipient), resource (FK to KnowledgeResource), recommended\_at (timestamp of recommendation).
* **Learning Progress → LearningProgress:** Tracks a user’s progress/completion percentage for a given learning resource. **Key columns:** id (PK), user (FK), resource (FK to KnowledgeResource), progress\_percent (progress as a percentage) (unique per user-resource pair).

### **Co-Creation (Community Content Creation)**

* **Co‑Creation Projects → CoCreationProject:** A collaborative content creation project (e.g. creating a course or document together). **Key columns:** id (PK), title, status (ENUM – e.g. draft, active, archived).
* **Co‑Creation Contributions → CoCreationContribution:** An individual contribution or edit made by a user to a co-creation project. **Key columns:** id (PK), project (FK to CoCreationProject), user (FK to contributor), content (text/content of the contribution).

### **Forums (Discussion Boards for Learning)**

* **Forum Topics → ForumTopic:** A discussion topic/thread in the educational forums (usually tied to a subject or question). **Key columns:** id (PK), title, category (free-text or predefined category of the topic), creator (FK to User who started the topic).
* **Forum Posts → ForumPost:** Posts or replies within a forum topic thread. **Key columns:** id (PK), topic (FK to ForumTopic), author (FK to User), content (text of the post).

## **Kreative (+ Kontact)**

### **Konservation (Creative Content & Cultural Preservation)**

* **Tags → Tag:** Global tagging vocabulary for artworks (and other content). **Key columns:** id (PK), name (unique tag name). *(This tag list is shared and reused in multiple creative contexts.)*
* **Artworks → KreativeArtwork:** A single piece of art or creative work uploaded by a user (image, video, audio, etc.). **Key columns:** id (PK), artist (FK to User, creator of the artwork), title, description (optional), media\_file (file path for the content), media\_type (ENUM – image/video/audio/other), year (optional year of creation), medium (text field for medium/material, e.g. “oil on canvas”), style (text field for artistic style). *(Each artwork can have multiple tags; see ArtworkTag below.)*
* **Artwork Tags → ArtworkTag:** Join table linking Artworks with Tags (many-to-many). **Key columns:** id (PK), artwork (FK to KreativeArtwork), tag (FK to Tag). *(Enforces uniqueness per artwork-tag pair.)*
* **Galleries → Gallery:** A curated gallery or collection of artworks, often for virtual exhibition purposes. **Key columns:** id (PK), title (gallery name), description (optional), created\_by (FK to User curator, nullable), theme (optional theme name), created\_at (timestamp). *(Each Gallery can contain many artworks; see GalleryArtwork below.)*
* **Gallery Artworks → GalleryArtwork:** Through-table for artworks included in a gallery, preserving order and uniqueness. **Key columns:** id (PK), gallery (FK to Gallery), artwork (FK to KreativeArtwork), order (position of the artwork in the gallery).
* **Tradition Entries → TraditionEntry:** A submission of cultural heritage content (e.g. photos, videos, descriptions of traditions) for preservation in the “Konservation” archive. **Key columns:** id (PK), title (name of the tradition or entry), description (text description), region (region/culture identifier, text), media\_file (uploaded media file showcasing the tradition), submitted\_by (FK to User, nullable), submitted\_at (timestamp), approved (boolean flag if approved for archive), approved\_by (FK to User who approved, nullable), approved\_at (timestamp).

### **Kontact (Collaboration & Networking)**

* **Collaboration Sessions → CollabSession:** Real-time collaborative sessions for artists (e.g. joint painting, jam sessions). **Key columns:** id (PK), name (session title), host (FK to User who started the session), session\_type (ENUM – e.g. painting, music, mixed media), started\_at, ended\_at (timestamps for session duration), final\_artwork (FK to KreativeArtwork created as the outcome, nullable).

## **Insights (Reporting & Analytics Module)**

### **Dimension Tables (Analytical Reference Data)**

* **Date Dimension → dim\_date:** Canonical calendar dates for aggregations. **Key columns:** date\_id (PK, surrogate key), calendar\_date (actual date), year, month, week, day, iso\_week, etc. (Pre-populated with a range of dates for analysis).
* **Domain Dimension → dim\_domain:** Enumerates high-level domain contexts (module or functional domains) for analytics. **Key columns:** domain\_id (PK), domain\_code (ENUM of domain codes). *(Matches the module/domain identifiers used in the OLTP system.)*
* **API Endpoint Dimension → dim\_endpoint:** Lists backend API endpoints for performance analytics. **Key columns:** endpoint\_id (PK), path (URL path or endpoint name).

### **Fact Tables (Partitioned Event/Metric Data)**

* **Smart Vote Fact → smart\_vote\_fact:** Records of votes cast, for analytics of voting patterns. **Key columns:** id (PK, UUID), date\_id (FK to dim\_date), domain\_id (FK to dim\_domain, e.g. which module the vote is in), question\_id (UUID of the voted content/question), user\_id (UUID hash or anonymized user ID), vote\_value (numeric vote value), score\_normalised (normalized score used in vote). *(Partitioned monthly by date; indexed by domain and date.)*
* **Usage MAU Fact → usage\_mau\_fact:** Monthly active user metrics and content creation counts per domain. **Key columns:** id (PK, UUID), month\_id (FK to dim\_date, points to first day of month), domain\_id (FK to dim\_domain), mau (count of monthly active users), projects\_created, docs\_uploaded (counts of content created in that month/domain). *(Partitioned by year; B-tree indexed by month and domain.)*
* **API Performance Fact → api\_perf\_fact:** Daily API performance metrics (per endpoint, per hour). **Key columns:** id (PK, UUID), date\_id (FK to dim\_date), hour\_of\_day (0–23), endpoint\_id (FK to dim\_endpoint), p95\_latency\_ms (95th percentile latency in ms), error\_rate\_pct (error percentage), request\_count (total requests). *(Partitioned by month via date; indexed by endpoint, date, hour.)*

**Note:** The above list is derived from the Konnaxion v14 documentation and the latest Django models and migration definitions. It is structured by module and sub-module for clarity, and is intended to replace the outdated canonical tables list with an accurate, developer-friendly reference. Each table’s purpose and key schema details have been verified against the implementation and v14 specifications for completeness and correctness.