AutoDBA Web – Audit Module Design

1. Overview of the Audit Engine

The AutoDBA audit engine is built using Python 3.11+ and is designed to run read-only analyses on the connected MySQL database. The engine is composed of three specialized modules, each focusing on a different aspect of database health: schema structure, indexing efficiency, and constraint integrity.

These scripts are executed by the backend using Node's child_process module and communicate via standard output in JSON format.

2. Module 1 – Structural & Schema Analysis (audit_schema.py)

This module analyzes the general organization and cleanliness of the database schema.

It detects:

- Missing primary keys on tables

- Orphaned tables (no relationships)

- Empty or unused tables

- Excessive column counts (indicating poor normalization)

- Inconsistent naming conventions (plural/singular, casing)

- Type mismatches (e.g., VARCHAR vs INT used for similar fields)

3. Module 2 – Index Efficiency Analyzer (audit_indexes.py)

This module evaluates the indexing strategy of each table and identifies inefficiencies.

It detects:

- Missing indexes on columns used in joins or WHERE clauses

- Redundant or overlapping indexes

- Low-cardinality indexes (e.g., on gender, status)

- Over-indexed tables causing unnecessary overhead

4. Module 3 – Constraint Integrity Checker (audit_constraints.py)

This module verifies the integrity and correctness of relational constraints (mainly foreign keys).

It detects:

- Broken foreign key references (violated joins)

- Mismatched types between PK/FK fields (signed/unsigned)

- Missing ON DELETE / ON UPDATE rules

- Orphaned child records due to missing parent rows

5. Shared Utilities (utils.py)

The `utils.py` file is shared across all modules to reduce duplication.

It includes:

- connect_to_db(): establishes secure MySQL connection using pymysql

- Helpers to query metadata from information_schema

- Type-matching and cardinality check functions

- Formatting helpers to standardize issue output

6. Audit Output Format (JSON)

Each Python script prints a JSON object to stdout, which is captured by the backend and sent to the frontend.

Example format:

{

"module": "schema" | "indexes" | "constraints",

"score": integer (0–100),

```
 "issues": [
  {
    "type": string (issue code),
    "table": string,
    "column": string | null,
    "description": string,
    "severity": "low" | "medium" | "high",
    "suggestion": optional string
  }
 ]
}
```

7. Execution Flow

All modules receive environment variables for DB connection credentials (host, port, user, password, db).

They are invoked by the backend (`pythonBridge.js`) using child_process.spawn or exec.

Each module runs independently. The backend merges their results into a combined object returned to the frontend.