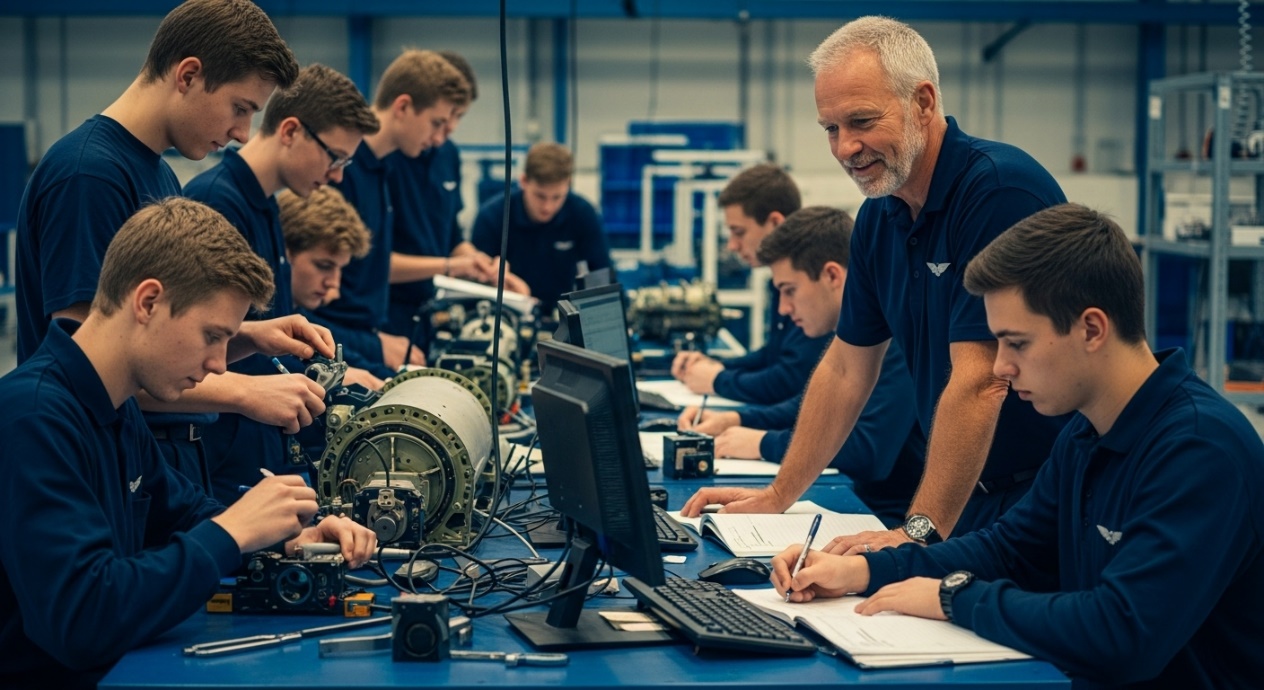
**Executive Summary**

Executive Summary This executive summary evaluates the relational database system developed for an Aviation Maintenance College. It addresses the limitations of previous fragmented recordkeeping practices and outlines how the new system supports scalability, compliance, and operational efficiency. Key decisions regarding database architecture, including the use of PostgreSQL via Supabase, are critically assessed. The summary also examines GDPR compliance and offers recommendations for future improvement.



**Summary of Work Carried Out**

The primary objective was to eliminate inconsistent and siloed recordkeeping systems and replace them with a centralized, validated, and scalable database. Previously, data management relied heavily on departmental Excel spreadsheets and legacy tools, resulting in redundant records, inconsistent formatting, inefficient reporting, and frequent human error.

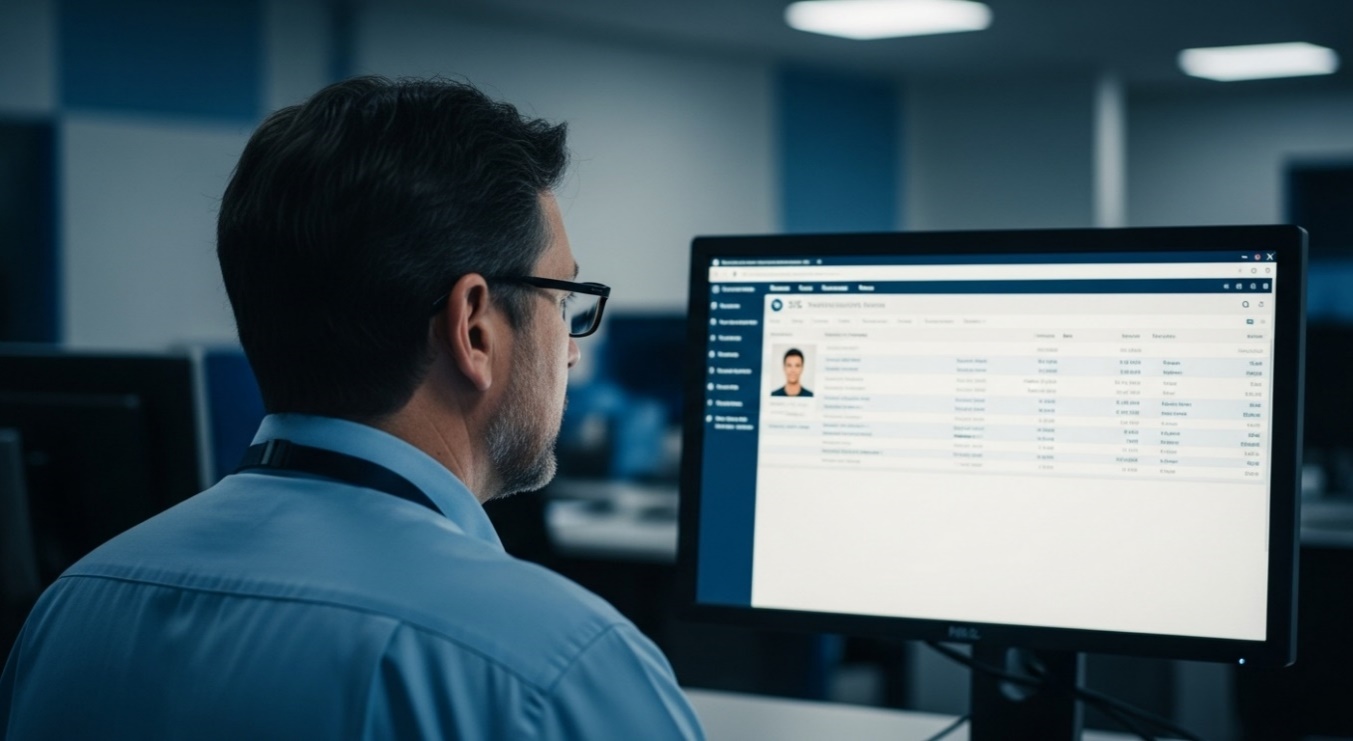
The new system was designed following Master Data Management (MDM) principles—emphasizing consistency, data quality, and a single source of truth for student and academic records (Loshin, 2010; Otto, 2015). To support this, the project focused on integrating two categories of data: system-generated and manually curated.

**System-generated Data**

System-generated Data Structured data from the Student Information System (SIS) and Learning Management System (LMS), such as attendance logs, student demographics, and theory exam results, were exported in CSV format. These files were processed using a custom-built Streamlit application developed in Python. The application performed several validation steps:

* Ensuring column structure consistency
* Validating information such as student IDs
* Highlighting duplicates
* Verifying formatting accuracy

After passing validation, the data was uploaded into the PostgreSQL database using the python library SQLAlchemy.



System-generated data taken from the college’s SIS (Student Information System)

**Manually Entered Data**

Manual records, such as practical assessments and compliance logs, introduced considerable challenges due to inconsistent formats and the absence of automated data capture. While the adoption of standardized Excel templates with dropdown menus and conditional formatting significantly reduced input error, this solution remains a stopgap. In the long term, continued reliance on manual entry increases operational risk, particularly as the volume of assessments grows. Alternative strategies such as digitizing the task booklets into the LMS, where input and grading are managed digitally, could reduce dependency on human input and enhance traceability. LMS-integrated assessment tools have been shown to reduce manual data entry time by over 40% and improve traceability through timestamped logs and digital records (Gamage, Ayres & Behrend, 2022).

****

Manually entered data taken from student practical task booklets

The schema **(Figure 1)** was designed using relational database principles to support normalization and relational integrity. Key entities such as students, instructors, attendance, assessments, and exams were stored in separate tables, linked through primary and foreign keys. UUIDs (Universally Unique Identifiers) were used as primary keys in cases where no natural unique key existed, such as when a student could retake an exam on the same day. While UUIDs may slightly impact query performance compared to integers, their ability to ensure global uniqueness across merged datasets made them a practical choice (Leach et al., 2005; Peterson, 2018).

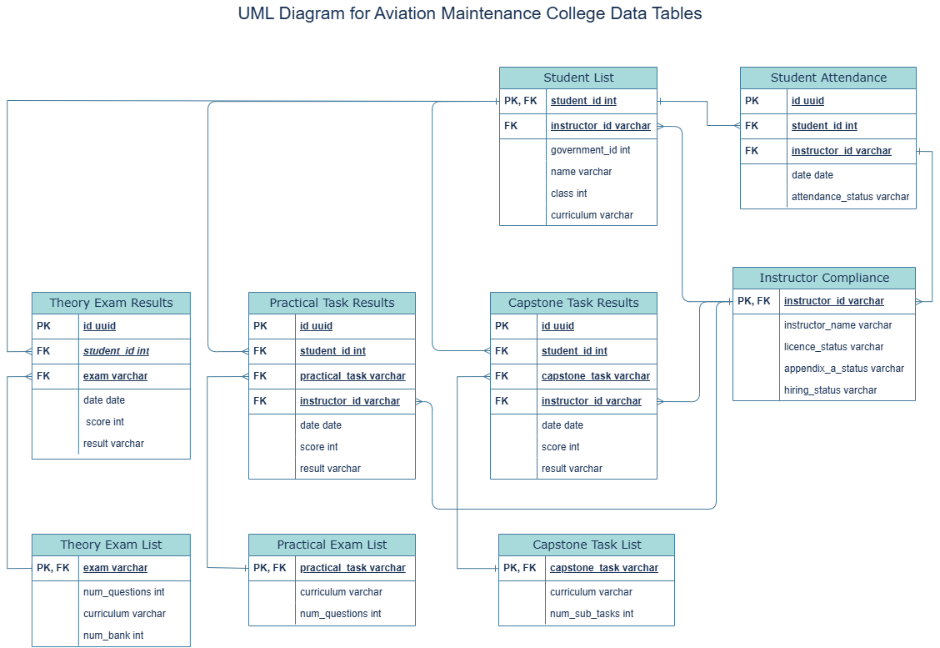


Figure 1: UML diagram showing the database’s tables and the relationships between them

**Critical Evaluation of Database Schema**

A core design objective was compliance with third normal form (3NF), a key relational modeling principle intended to reduce redundancy and avoid update anomalies. As outlined by Connolly and Begg (2014), a table satisfies 3NF if all non-key attributes are fully functionally dependent on the primary key and no transitive dependencies exist.

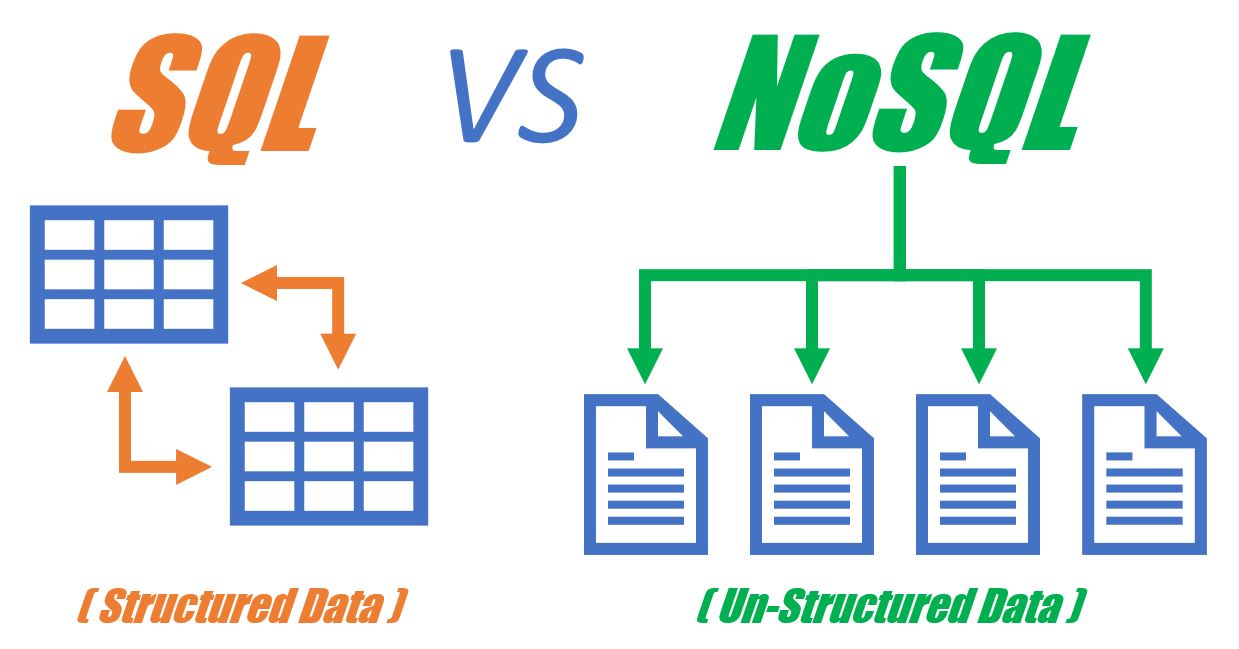
Most tables in the schema meet these criteria. For example, the assessment results table links student IDs and exam IDs to scores and dates using foreign keys, avoiding duplication. Similarly, the attendance table connects students and instructors without repeating descriptive fields.

One area for improvement is the Instructor Compliance table, which combines employment details, licensing data, and training records in one structure. Although each instructor appears only once, real-world changes (e.g. re-employment with updated certifications) can lead to conflicting or redundant entries. Splitting this table into separate employment and compliance entities would improve normalization, support historical tracking, and enable more flexible querying.

Overall, the schema aligns well with relational design best practices, offering a robust and adaptable foundation for future data needs.

**Analysis of DBMS Choice**

**SQL vs NoSQL**



A relational (SQL) database was selected because the college's academic data is structured, consistent, and subject to strict accuracy and validation requirements. SQL databases enforce schema rules, support foreign key constraints, and provide full transactional integrity, features that are essential when managing student records, assessment scores, and compliance logs with high accuracy (Elmasri & Navathe, 2015). The predictability of SQL’s relational structure ensures that data anomalies, such as duplicate entries or orphaned records, are minimized through constraints and validation rules.

In contrast, NoSQL alternatives such as MongoDB, Cassandra, and Couchbase are better suited to scenarios involving dynamic schemas, real-time event processing, or storage of multimedia and semi-structured data. These systems often rely on eventual consistency rather than immediate consistency, introducing latency and uncertainty in critical academic workflows where precision is paramount (Cattell, 2011; Sadalage & Fowler, 2012). Given the regulatory environment and the institution’s reliance on repeatable, auditable reports, a relational database was clearly the more appropriate solution.

**Justification for PostgreSQL**

Among the available relational database engines, PostgreSQL was chosen for its open-source licensing, standards compliance, and robust feature set. Unlike lightweight engines such as SQLite, PostgreSQL is a fully-featured RDBMS that supports ACID transactions, advanced indexing (e.g., GIN and GiST), stored procedures, full-text search, and materialized views. Additionally, its support for JSONB allows for semi-structured data storage without abandoning relational principles, offering flexibility where needed (Han et al., 2011).

Other SQL-based engines were also considered. MySQL, while popular and efficient for web applications, has historically lacked full support for advanced features such as window functions and recursive queries, although this has improved in recent versions. Microsoft SQL Server provides excellent performance and tooling but requires licensing fees and a Windows-based infrastructure, increasing long-term total cost of ownership. Oracle Database, while powerful, was immediately ruled out due to its prohibitive cost and complexity. PostgreSQL stood out as a community-backed, enterprise-grade solution with cross-platform compatibility, making it particularly attractive for educational institutions with constrained budgets but demanding data integrity requirements.

PostgreSQL’s seamless integration with Python tools such as SQLAlchemy and Pandas further streamlined the development pipeline. This compatibility enabled efficient querying, visualization, and automation workflows across the data lifecycle, from ingestion to reporting.

**Backend Platform: Why Supabase Was Chosen**

Once PostgreSQL was selected, attention turned to the deployment platform. Several cloud-based hosting options were reviewed, including Amazon RDS, Google Cloud SQL, Heroku Postgres, and Supabase. The selection criteria included ease of setup, cost-effectiveness, support for modern authentication and API features, and regulatory compliance capabilities.

* **Amazon RDS** offers robust scalability, automated backups, and fine-grained monitoring but introduces administrative overhead, particularly for setting up networking, user roles, and backups. Its cost structure can become complex and prohibitive for non-commercial institutions.
* **Google Cloud SQL** integrates well with other Google services and supports PostgreSQL instances but lacks native support for Row-Level Security (RLS), which is critical for access control in a GDPR-compliant academic system.
* **Heroku Postgres** offers a simplified developer experience with easy CI/CD integration, but the free tier is limited in terms of database connections, and there is reduced visibility into underlying infrastructure.

Supabase offers a developer-friendly interface, instant API generation, built-in RLS, authentication services, and seamless PostgreSQL integration. These features significantly reduce backend development time and complexity, making it a strong fit for academic environments with limited technical staffing.

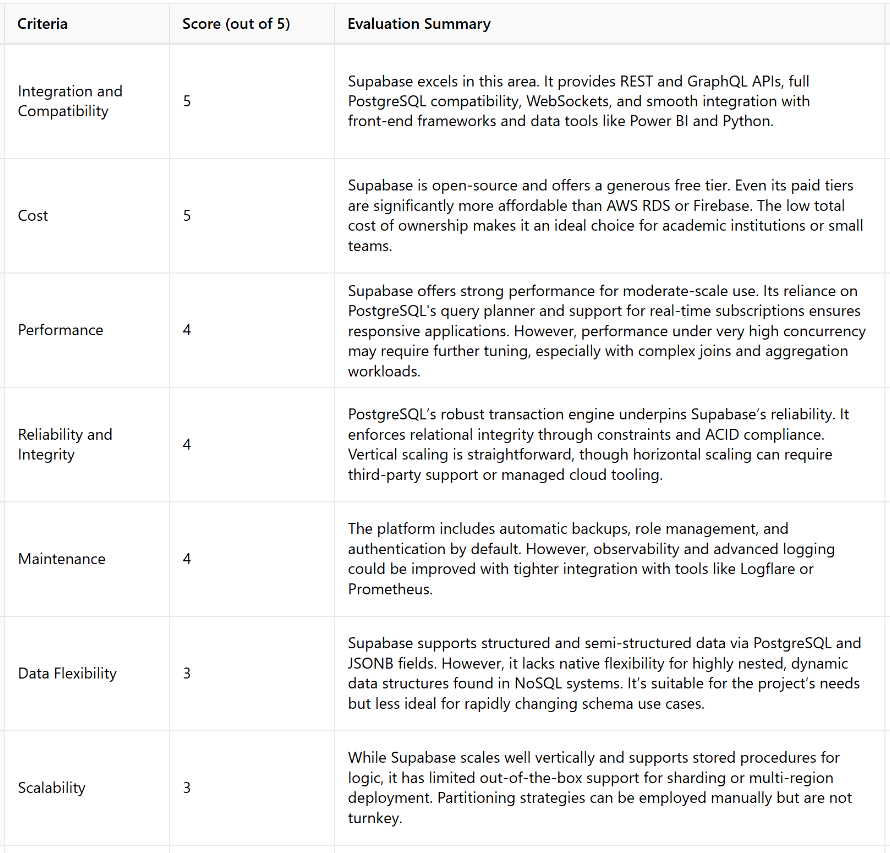


Figure 2: Supabase platform evaluation by usability, scalability, cost, security, and API support.

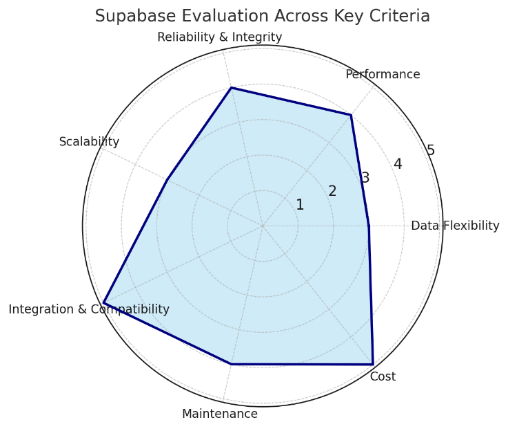


Figure 3: Radar chart showing Supabase’s evaluation against the grading criteria

As shown in **Figure 2,** Supabase was evaluated across usability, scalability, cost, security, and API support. Its strengths are also summarized in Figure 3 as a radar chart.

Supabase performed particularly well in:

* **Integration and Compatibility (5/5):** REST and GraphQL support, native PostgreSQL, compatibility with Power BI and Python.
* **Cost (5/5):** Transparent pricing and a generous free tier.
* **Maintenance (4/5):** Built-in tools for role-based access and backup; observability tools could be improved.

Its built-in RLS **(Figure 4)** supports GDPR compliance by restricting data access at the row level according to user roles (GDPR.eu, n.d.). While full audit logging is limited, third-party tools like Logflare can be integrated to enhance traceability.

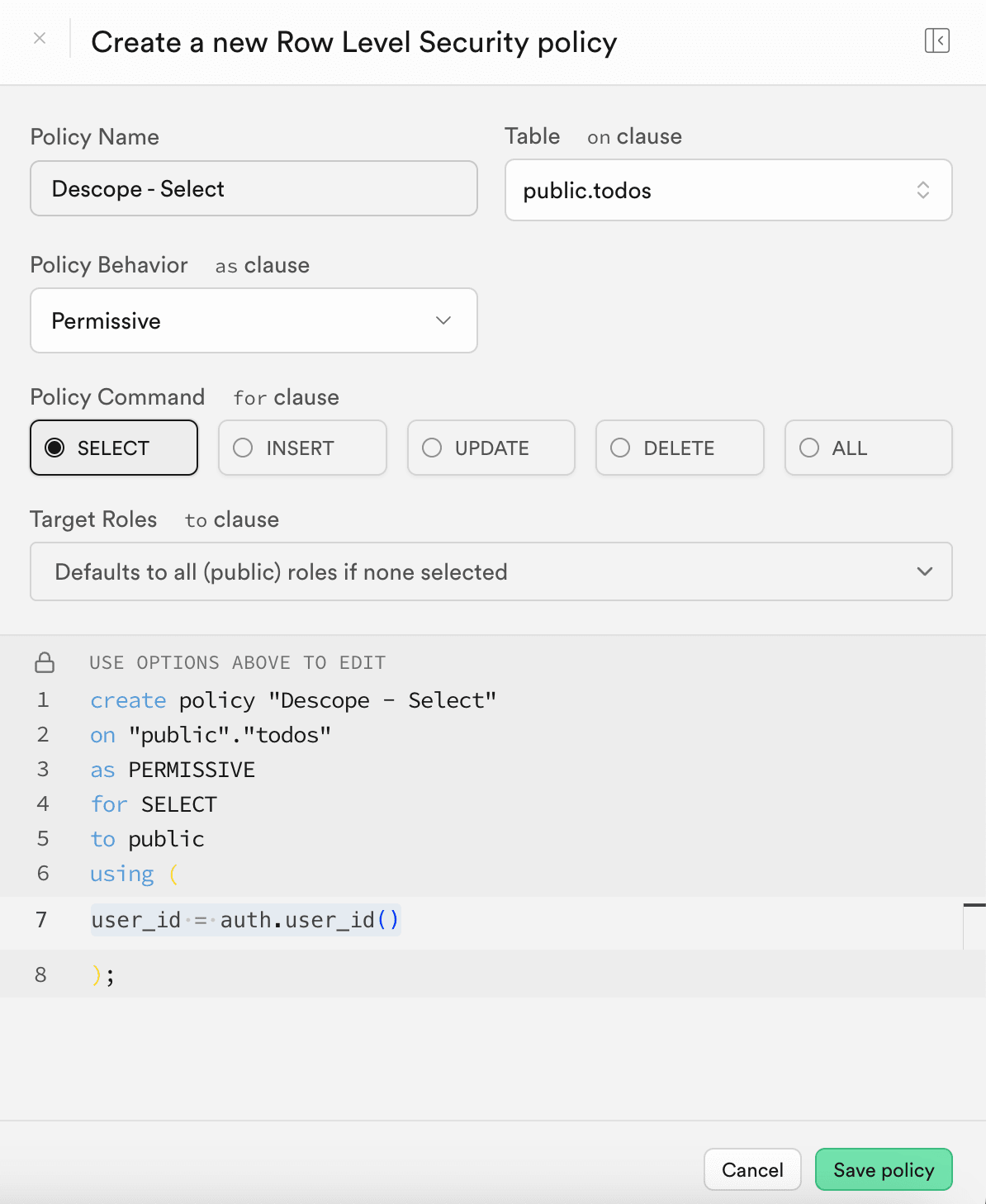
**

Figure 4: Supabase’s RLS (Row Level Security) configuration screen

**GDPR Compliance and Legal Requirements**

Given the sensitive nature of educational data, compliance with the General Data Protection Regulation (GDPR) was a fundamental design requirement. This included safeguarding student identifiers, assessment records, and instructor compliance data in accordance with guidelines from the European Data Protection Board and the UK Information Commissioner’s Office.

**Data Minimization**

The new system embraces the **principle of data minimization** by ensuring that only essential and relevant data is collected and stored. Through careful schema design in third normal form (3NF), redundant fields were eliminated, and only core data points, such as student identifiers, attendance logs, and assessment results, were retained. This lean structure aligns with GDPR’s requirement to avoid over-collection and ensures data is purpose-specific and necessary for institutional needs.

**Integrity and Confidentiality**

To uphold **data integrity and confidentiality**, Supabase’s Row-Level Security (RLS) was implemented to restrict database access at the row level based on user roles. This prevents unauthorized users from viewing or modifying records they are not permitted to access. In combination with built-in authentication and role-based access controls, this framework ensures secure handling of sensitive student data, protecting it from exposure, tampering, or misuse.

**Accuracy**

In support of the **accuracy** principle, all incoming data undergoes validation before entering the database. This is achieved through the custom-built Streamlit application, which checks for duplicate entries, ID mismatches, and format errors. For manually entered records, such as practical exam scores and instructor compliance logs, standardized Excel templates with dropdown menus and conditional formatting further reduce the risk of input errors and ensure consistency.

**Accountability**

Finally, the system addresses **accountability** by providing a transparent and traceable record of data handling activities. The use of PostgreSQL with Supabase allows for automatic audit trail logging through tools like Logflare, capturing access and modification events. Additionally, validation logs generated during the ingestion process create a secondary layer of traceability, offering demonstrable evidence that the college adheres to GDPR requirements in both data processing and protection.

Supabase’s built-in authentication and role-based access controls support GDPR compliance by restricting data access based on user roles (e.g. instructor, assessor, administrator). These controls, combined with Row-Level Security (RLS), ensure only authorized users can access relevant records.

While full audit trails are not native to Supabase, integration with tools like Logflare enables automatic logging of access and modification events. This supports accountability by providing traceable records of who accessed what data and when, useful for audits and appeals.

The system was designed with privacy by design and by default, embedding compliance into its core architecture. Ongoing review of access policies and logging practices is recommended to maintain alignment with evolving legal standards.

**Conclusions and Recommendations**

The implemented system delivers a centralized and scalable data infrastructure tailored to the academic and regulatory needs of the Aviation Maintenance College. Its adoption of PostgreSQL and Supabase has improved data integrity, streamlined reporting, and enhanced compliance with GDPR. To maintain this momentum and address areas for refinement, the following targeted recommendations are proposed:

**High-Priority Recommendations**

* **Refactor the Instructor Compliance Table**  
  Currently, employment details and compliance data are combined in a single table. Separating these into distinct entities would align with third normal form, reduce data redundancy, and support accurate historical tracking as instructor roles evolve.
* **Improve Manual Data Capture**   
  The current use of Excel templates reduces errors but is not scalable. Long-term, digitizing task booklets via OCR or integrating data entry into the LMS would reduce manual workload, improve traceability, and enhance data quality, despite requiring upfront investment.

**Lower-Priority Recommendations**

* **Establish a Formal Data Governance Policy**  
  While validation logic exists, formalizing a governance framework will ensure data quality standards are upheld as the system scales and new users are onboarded.
* **Monitor and Evaluate Supabase Usage Tiers**  
  The current usage falls within the free tier, but periodic monitoring will help anticipate scaling needs and avoid disruptions as more data is ingested.

These recommendations are aimed at sustaining the system’s current strengths while proactively addressing scalability, traceability, and institutional governance needs. Regular evaluation and incremental improvements will ensure the database remains aligned with operational demands and evolving compliance frameworks.

**References**

Cattell, R. (2011) ‘Scalable SQL and NoSQL data stores’, *ACM SIGMOD Record*, 39(4), pp. 12–27. Available at: <https://dl.acm.org/doi/10.1145/1978915.1978919>

Connolly, T.M. and Begg, C.E. (2014) *Database systems: a practical approach to design, implementation, and management*. 6th ed. Harlow: Pearson.

Davoudian, A. and Liu, M. (2020) ‘A comparative study of SQL and NoSQL database systems’, *Journal of Information Systems Education*, 31(4), pp. 246–259.

Elmasri, R. and Navathe, S.B. (2015) *Fundamentals of database systems*. 7th ed. Boston: Pearson.

European Data Protection Board (n.d.) *Guidelines, recommendations and best practices*. Available at: <https://edpb.europa.eu/edpb_en>

GDPR.eu (n.d.) *A guide to GDPR data privacy requirements*. Available at: <https://gdpr.eu/>

Gamage, S.H.P.W., Ayres, J.R. and Behrend, M.B. (2022) ‘Optimizing manual data entry in practical assessments: A review of LMS-integrated grading tools’, *Journal of Learning Analytics*, 9(1), pp. 22–39.

Gonçalves, C.T., Gonçalves, M.J.A. and Campante, M.I. (2023) ‘Developing integrated performance dashboards visualisations using Power BI as a platform’, *Information*, 14(11), p. 614. Available at: <https://www.mdpi.com/2078-2489/14/11/614>

Han, J., Kamber, M. and Pei, J. (2011) *Data mining: concepts and techniques*. 3rd ed. Boston: Morgan Kaufmann.

ICO (Information Commissioner’s Office) (n.d.) *Guide to the UK General Data Protection Regulation (UK GDPR)*. Available at: <https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/>

Leach, P.J., Mealling, M. and Salz, R. (2005) ‘A Universally Unique Identifier (UUID) URN Namespace’, *RFC 4122*. Available at: <https://tools.ietf.org/html/rfc4122>

Loshin, D. (2010) *Master data management*. Boston: Morgan Kaufmann.

Otto, B. (2015) *Fundamentals of master data management: bringing data governance into practice*. Springer. Available at: <https://www.researchgate.net/publication/220618579_Data_Governance>

Peterson, J. (2018) *SQL performance explained*. 2nd ed. Vienna: Markus Winand.

RelationalDBDesign (n.d.) *Relationships and referential integrity*. Available at: <https://www.relationaldbdesign.com/database-analysis/module7/relationships-referential-integrity.php>

Sadalage, P.J. and Fowler, M. (2012) *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Boston: Addison-Wesley. Available at: <https://martinfowler.com/books/nosql.html>

Sakr, S. and Gaber, M.M. (2014) *Large scale and big data: processing and management*. Boca Raton, FL: CRC Press. Available at: <https://www.routledge.com/Large-Scale-and-Big-Data-Processing-and-Management/Sakr-Gaber/p/book/9780367736500>

Sciore, E. (2007) *Database management: a systems approach using Java*. Boston: Jones & Bartlett. Available at: <https://www.researchgate.net/publication/225084162_Database_Design_And_Implementation>

Stonebraker, M. (2010) ‘SQL databases v. NoSQL databases’, *Communications of the ACM*, 53(4), pp. 10–11. Available at: <https://doi.org/10.1145/1721654.1721659>

Stonebraker, M. and Cetintemel, U. (2005) ‘One size fits all: an idea whose time has come and gone’, *Proceedings of the 21st International Conference on Data Engineering*, pp. 2–11.

van der Veen, J.S., Van der Waaij, B., Lazovik, A., Nieuwenhuis, L.J.M. and Rajkumar, R.R. (2012) ‘Performance evaluation of cloud databases for cloud applications’, in *Cloud computing and services science*. Springer, pp. 1–21. Available at: <https://link.springer.com/chapter/10.1007/978-3-642-36675-1_1>