EAS 550 Group Project

1. Introduction & Vision

1.1. Project Goal

The goal of this project is to apply the database system and query language concepts learned in class to build a practical database application. You will design, implement, and deploy a complete system that transforms raw data into a functional product. This project is your opportunity to demonstrate your ability to handle the entire data lifecycle, from initial design to final application.

1.2. Timeline and Structure

This is an intensive project designed to be completed within a condensed timeline.

- **Final Submission:** December 14

The project is broken down into three distinct phases, with deliverables due roughly every two weeks. Time management and consistent teamwork will be critical to your success.

1.3. General Guidelines

- **Team Size:** 3-4 students.
- **Collaboration:** All work must be managed in a shared **GitHub** repository. Your repository must include a README.md file with a project overview, setup instructions, and clear steps on how to run your application.
- **Deliverables:** For each phase, you will submit your GitHub repo URL for that stage.

2. Getting Started: Your Project, Your Domain

Your first task as a team is to decide on a project theme and find a suitable dataset that interests you.

2.1. Step 1: Choose a Project Theme

Select an application domain. Here are some examples of industrial database applications for your reference:

- **E-commerce & Retail Analytics:** Analyze customer behavior, manage inventory, and build recommendation engines.

- **Financial Services & Fraud Detection:** Model transactions, detect anomalies, and analyze financial risk.
- **Healthcare Management:** Manage patient records and analyze clinical data (ensure you use anonymized data).
- **Supply Chain & Logistics:** Track shipments, optimize routes, and manage inventory.
- **Real Estate Platform:** Analyze property listings and market trends.
- **Sports or Music Analytics:** Analyze player performance, game statistics, or music trends.

2.2. Step 2: Find Your Dataset

Once you have a theme, find a public dataset that fits your application. The ideal dataset is complex enough for interesting analysis (multiple related files/tables are a plus) but manageable in size.

To get you started quickly, here is a list of 8 recommended, available, and appropriate datasets:

#	Dataset Name & Link	Domain
1	(https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce) ***Description:*** *A multi-table dataset with 9 CSV files covering orders, products, customers, and reviews. Good for practicing relational modeling and building a full analytics pipeline.*	E-commerce
2	(https://www.kaggle.com/datasets/eoinamoore/historical-nba-data-and-player-box-scores) ***Description:*** *Contains multiple files for game stats, player info, and team details. Good for a project analyzing player performance, team trends, and game outcomes.*	Sports Analytics
3	(https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset) ***Description:*** *A large, single CSV file with over 2 million property listings. This dataset is good if you want take challenge for data cleaning and normalization into a proper relational schema.*	Real Estate
4	(https://www.kaggle.com/datasets/computingvictor/transactions-fraud-datasets) ***Description:*** *A multi-file dataset containing transaction records, customer info, and fraud labels. Good for a project focused on fraud detection, security, and anomaly analysis.*	Finance / Security
5	(https://www.kaggle.com/datasets/prasad22/healthcare-dataset)	Healthcare

#	Dataset Name & Link	Domain
	Description: *A large, synthetic dataset mimicking patient records, including demographics, medical conditions, and billing. Appropriate for building a healthcare management or analytics system without privacy concerns.*	
6	(https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset) ***Description:*** *A large dataset with over 100,000 songs and their audio features (danceability, energy, etc.). Good for building a music analysis platform or a recommendation engine.*	Music / Media
7	(https://www.kaggle.com/datasets/thedevastator/video-game-sales-and-ratings) ***Description:*** *A dataset of video game sales, critic scores, and user ratings. Good for analyzing trends in the gaming industry and the correlation between sales and ratings.*	Entertainment
8	(https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes) ***Description:*** *Contains detailed records of motor vehicle collisions in NYC. This dataset is rich with temporal and geospatial data. It is ideal for an analytical application with map- based visualizations.*	Government / Civic

You are free to choose a dataset from other sources like **Google Dataset Search**, **Data.gov**, **AWS Open Data**, or the **UCI Machine Learning Repository, etc.**..

2.3. Step 3: Project Proposal (Due: November 6th)

Submit a one-page proposal that includes:

- Your team members.
- Your chosen project theme and a brief description of the application you intend to build.
- A link to your chosen dataset and a brief analysis of its suitability.
- The core technologies you plan to use.

3. Core Technical Stack & Learning Resources

All tools needed are free and open-source. Your entire environment must be containerized using **Docker** to ensure reproducibility.

Tool	Purpose	Tutorial / Resource
Git & GitHub	Version control and collaboration	GitHub Guide
Docker	Containerization for all services	(https://docs.docker.com/get-started/) / Video Intro

Tool	Purpose	Tutorial / Resource
PostgreSQL	Primary relational database	(https://www.pgtutorial.com/) / Python Integration Video
Python	Data ingestion and application logic	(https://docs.python.org/3/tutorial/)
Pandas	Data cleaning and manipulation	Pandas Intro / Manipulation Guide
SQLAlchemy	Python SQL Toolkit	(https://www.datacamp.com/tutorial/sqlalchemy-tutorial-examples) / GeeksForGeeks Guide
DBT Core	Data transformation (Optional)	Official dbt Guide / Modeling Guide
Streamlit / FastAPI	Application front-end or API	(https://docs.streamlit.io/get-started/tutorials/create-an-app) /(https://medium.com/codex/fastapi-tutorial-part-1-your-coffee-shop-opens-for-business-bdd428a86e5a)

4. Project Phases & Detailed Instructions

Phase 1: The Database Foundation (OLTP)

Timeline: Nov 2 - Nov 18 (Approx. 2.5 weeks)

Goal: Ingest and model your chosen raw dataset into a well-designed, normalized, and secure relational database (OLTP).

- **Step 1.1: Conceptual & Logical Design:**
  - Thoroughly analyze your dataset to identify the core entities, attributes, and relationships.
  - Create a detailed **Entity-Relationship Diagram (ERD)** using standard notation (you may also use Crow's Foot notation instead the ones used in our lectures). This is the blueprint for your database.
  - Design your schema to be in at least **Third Normal Form (3NF)**. Write a brief report justifying your design choices and explaining how your schema avoids data anomalies.
- **Step 1.2: Physical Implementation:**
  - Write a single, well-commented SQL script (schema.sql) with all the CREATE TABLE statements to implement your schema in PostgreSQL.
  - Use the most appropriate data types for each column and enforce data integrity with PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, and CHECK constraints.
- **Step 1.3: Data Ingestion Pipeline:**

    - Write Python script (ingest_data.py) that uses **Pandas** and **SQLAlchemy** to clean, transform, and load the data from your source files into the PostgreSQL tables.
    - Your script must handle potential data quality issues (e.g., missing values, incorrect data types) and load data in the correct order to satisfy foreign key constraints.
- **Step 1.4 (Optional): Initial Security Setup:**
    - Write a SQL script (security.sql) that implements a basic Role-Based Access Control (RBAC) model. Create at least two roles (e.g., a read-only analyst role and a read-write app_user role) with appropriate privileges.

Deliverables for Phase 1:

1. A GitHub repository containing your ERD, all SQL and Python scripts, and your 3NF justification report.
2. A docker-compose.yml file that allows your PostgreSQL database to be launched with a single command.
3. A demo video walking through your data model, showing your ingestion script running, and querying the populated database to prove it works. Please upload your video to Youtube.com as 'unlisted', then share the link. Each team only submit one link by one of the members.

Phase 1 Rubrics:

Category	Criteria	Description	Weight (%)	Performance Indicators
Phase 1 — Database Foundation (OLTP)	Conceptual & Logical Design	Clear ERD, correct entities/relationships, proper use of Crow's Foot notation, 3NF achieved and justified.	10	Comprehensive ERD, 3NF explained clearly.
	Physical Schema Implementation	SQL schema correctness, data types, constraints (PK/FK/NOT NULL/UNIQUE/CHECK).	5	Correct and meaningful constraints
	Data Ingestion Pipeline	Working Python + SQLAlchemy + Pandas ingestion; handles errors & integrity constraints.	10	modular, documented, robust to missing data.
	Security Setup (Optional Bonus)	Implemented role-based access control.	+5 Bonus	At least two roles created with different permissions.
	Subtotal Phase 1		**25%**	

Phase 2: Advanced Querying & The Analytical Layer (OLAP)

Timeline: Nov 19 - Dec 2 (Approx. 2 weeks)

Main Goal: Analyze your data with advanced queries.

Optional: build a formal data warehouse.

- **Step 2.1: Advanced Analytical Queries:**
    - Write at least three complex SQL queries against your **OLTP database** to answer key business questions relevant to your project theme. These queries must use advanced features like multi-table joins, subqueries, window functions, or complex aggregations.
- **Step 2.2: Query Performance Tuning:**
    - Select your most complex analytical query from Step 2.1. Use EXPLAIN ANALYZE to get its query plan and execution time.
    - Design and implement an indexing strategy on your OLTP tables to improve the query's performance. Document the "before" and "after" performance and explain why your indexes were effective.
- **Step 2.3 (Optional): Dimensional Modeling & ETL:**
    - Design a **star schema** for a data warehouse. Create a diagram and a report defining the "grain" of your fact table and the purpose of each dimension.
    - Use **dbt (Data Build Tool)** to transform the data from your OLTP schema and load it into your new data warehouse schema.

Deliverables for Phase 2:

1. Your GitHub repo updated with your advanced SQL queries and your **performance tuning report**.
2. If you completed the optional step, include your star schema diagram and your complete dbt project folder.

Phase 2 Rubrics:

Category	Criteria	Description	Weight (%)	Performance Indicators
Phase 2 — Analytical Layer (OLAP)	Advanced Analytical Queries	At least 3 meaningful, complex SQL queries addressing project questions using joins, subqueries, window functions, CTEs, or aggregation.	15	Queries are correct, optimized, and relevant to business goals.
	Query Optimization & Performance Tuning	Demonstrated query profiling with EXPLAIN ANALYZE; indexing strategy tested with before/after results.	15	Detailed report, measurable improvement, and justified index design.
	Data Transformation & Modeling (Optional)	Optional OLAP schema or dbt modeling for analytical workflow.	+10 bonus	Properly designed star/snowflake schema and brief rationale.

	Subtotal Phase 2		**30%**	

Phase 3: The Application Layer (API or Dashboard)

Timeline: Dec 2 - Dec 14 (Approx. 1.5 weeks)

Goal: Create a user-facing application that makes your data accessible and insightful.

- **Step 3.1: Choose Your Application Type:**
  - **Option A: Interactive BI Dashboard:** Use **Streamlit** to build a user-friendly dashboard that connects to your database (either OLTP or the optional OLAP warehouse). This is ideal for analytics-focused projects.
  - **Option B: REST API:** Use **FastAPI** to build a web API that serves data from your OLTP database. This is ideal for projects focused on building a transactional application.
- **Step 3.2: Build the Application:**
  - **If building a dashboard:** Create at least two distinct visualizations (e.g., charts, maps, tables) and at least one interactive widget (e.g., a date slider, a dropdown filter) that allows users to explore the data.
  - **If building an API:** Implement at least three GET endpoints to retrieve data (e.g., a list of items and a single item by ID) and at least one POST or PUT endpoint to demonstrate data modification. Your API must have automatically generated documentation (a key feature of FastAPI).
- **Step 3.3: Final Integration and Documentation:**
  - Ensure your entire project is runnable from your docker-compose.yml file.
  - Thoroughly document the setup and execution steps in your README.md file.

Deliverables for Phase 3 (Final Submission):

1. Your final, complete GitHub repository containing all code, scripts, and documentation from all three phases.
2. A final end-to-end demo video that showcases the entire application, from data ingestion to the final user interface (dashboard or API).

Phase 3 Rubrics:

Category	**Criteria**	**Description**	**Weight (%)**	**Performance Indicators**
Phase 3 — Application	Functional Implementation	Application (Streamlit or FastAPI) runs successfully with database connection.	10	Fully functional, error-free,

Layer (API or Dashboard)		Must include dynamic interaction or endpoints.		integrated end-to-end.
	Interactivity, Usability & Data Visualization	For dashboards: at least 2 insightful visualizations and 1 interactive filter. For APIs: at least 3 GET and 1 POST/PUT endpoint with proper documentation.	10	Responsive, user-friendly interface or clearly documented API.
	Documentation & Deployment	Project runs via docker-compose; includes detailed README with setup steps, screenshots, and demo video.	5	One-command setup, complete documentation, reproducible.
	Subtotal Phase 3		**25%**	

Teamwork Performa Rubrics:

Category	Criteria	Description	Weight (%)	Performance Indicators
Teamwork & Professionalism	Collaboration & Version Control	Consistent team contributions tracked via GitHub; meaningful commits and branches.	5	All members contribute regularly with clear **commit messages.**
	Communication & Project Management	Clear task division, use of milestones/issues in GitHub or task board.	5	Organized with clear milestones.
	Presentation & Demo	Final presentation/video is clear, professional, demonstrates full workflow, and reflects teamwork.	10	Smooth, engaging, and technically accurate.
	Subtotal		**20%**	