CASE STUDY: Research Project Database Management System

By Devam Tanna (202512010) &

Charmi Bhayani (202512028)

Group Number: 9

Date: 15/11/2025

Index

Lab:4 & 5 Software Requirement Specification 3
Lab-6: Research Project Database - Comprehensive Noun Analysis and ER Diagram 38
Lab-7: V-2 of ER Diagram 66
Lab-8: ERD Normalization 70
Initial Relational Model: Full Relations List 71
Functional Dependencies Analysis: 72
Anomaly Analysis 73
1NF Form 74
2NF Form 74
Lab-9: Final Normalization & DDL 77
3NF: 77
Relational Model: Full Relations List 79
Lab:10 Queries Execution on Database 99
Lab:11 User Interface And DataConnection 124
Changes made: 142

Lab:4 & 5 Software Requirement Specification

Introduction

1.1 Purpose

In today's academic environment, information plays an important role in how institutes manage their activities. Research projects, in particular, generate a lot of data—such as student contributions, faculty guidance, deadlines, budgets, and funding details. If this information is stored in separate files or emails, it becomes difficult to track, often leading to confusion, duplication of work, or even loss of important records.

A Database Management System (DBMS) provides a better way to handle this challenge. It allows data to be stored in an organized, secure, and consistent manner while making it easier for multiple users to access the same information without errors.

The Research Project Database Management System (RPDMS) is designed to revolutionize how DA-IICT manages its research ecosystem. Currently, research project management at our institute suffers from fragmentation, inefficiency, and lack of transparency. This system will serve as a centralized digital platform that transforms research administration from a chaotic, paper-based process into a streamlined, data-driven operation.

The primary purpose is to create an integrated environment where:

- **Students** can focus on research rather than administrative overhead
- **Faculty** can efficiently monitor and guide multiple projects simultaneously
- **Administrators** gain real-time visibility into institutional research performance
- **Sponsors** receive transparent, automated reporting on their investments
- **The Institute** builds a comprehensive research knowledge base for future reference

In the long run, this database will not only make daily management easier but also serve as a record of the institute's research journey. It will provide insights into achievements, publication outcomes, and funding utilization, aligning with DA-IICT's commitment to knowledge sharing and professional growth, as promoted by initiatives like the IEEE Student Branch.

In short, the purpose of this system is not just to store project data but to improve collaboration, reduce errors, and ensure that research work is managed smoothly and transparently.

[1] GeeksforGeeks, "Introduction of DBMS (Database Management System)," Available: https://www.geeksforgeeks.org/dbms/introduction-of-dbms-database-management-system-set-1/

[2] BMC Blogs, "What is a DBMS?," Available: https://www.bmc.com/blogs/dbms-database-management-systems/

[3] Wikipedia, “Centralized Database,” Available: https://en.wikipedia.org/wiki/Centralized_database

[4] Moldstud, “The Impact of Database Development in Higher Education,” Available: https://moldstud.com/articles/p-the-impact-of-database-development-in-higher-education

[5] IEEE Student Branch DA-IICT, Available: http://ieee.daiict.ac.in/ieee2018/index.html

[6] Tableau, “What is Data Management?,” Available: https://www.tableau.com/learn/articles/what-is-data-management

1.2 Intended Audience and Reading Suggestions

This SRS addresses the complete software system requirements, not just database design. The document covers functional requirements, user interfaces, system workflows, and innovative features that will differentiate our solution.

Primary Audience:

- **Development Team**: Complete technical specifications for implementation
- **Faculty Supervisors**: Understanding of system capabilities and workflows
- **Institute Administration**: Resource requirements and expected benefits
- **Student Users**: Clear expectations of system functionality
- **Project Stakeholders**: Comprehensive view of deliverables and timelines

1.3 Product Scope

The Research Project Database will act as a central platform where all details related to research projects in the institute can be stored and managed. At present, much of this information is kept in separate files or communicated informally, which makes it hard to track and update. This project is meant to solve that problem by bringing everything into one system.

The scope of the system covers the following:

1. **Project Information**
   Basic details such as project title, objectives, duration, and current status will be stored. This makes it easy for faculty, students, and administrators to quickly check the progress of any project.

2. **Faculty and Student Roles**
   Each project involves supervisors and student participants. The database will clearly show who is responsible for which project, and supervisors can track student contributions more easily.

3. **Funding and Budget Management**
   Many projects are sponsored by external agencies. The system will keep records of sponsors, funding amounts, and how funds are being used. This ensures transparency and helps sponsors trust the institute's research work [1].

4. **Progress Reports and Publications**
   Regular progress reports can be uploaded by students and reviewed by supervisors. Final outcomes such as publications or patents will also be stored, so the database becomes a record of the institute's research output [2].

5. **Future Use and Transparency**
   Even after a project ends, its data will remain in the system. This helps future researchers refer to past work and learn from it. It also allows administrators and sponsors to review the long-term impact of research projects.

1.4 References

[1] *Research Information Management: Defining RIM and the Library's Role,* OCLC Research, 2017. Available: https://www.oclc.org/research/publications/2017/oclcresearch-research-information-management.html

[2] *The Impact of Database Development in Higher Education,* Moldstud. Available: https://moldstud.com/articles/p-the-impact-of-database-development-in-higher-education

2. Overall Description

2.1 Product Perspective

The Research Project Database is envisioned as a **centralized management system** for DA-IICT's research projects. Currently, information is fragmented across spreadsheets, emails, and manual reports, leading to inefficiencies. This product will replace these scattered methods with a single platform that integrates students, faculty, administrators, and sponsors. It acts as both a **transactional system** (data entry, updates, report uploads) and a **decision-support tool** (dashboards, budget tracking, sponsorship reporting).

The Student's Story:

Scenario: Riya's Research Journey Riya, a final-year Computer Science student, is working on a Machine Learning project under Dr. Patel's guidance. Her monthly routine involves:

- Searching through email chains to find her previous reports
- Recreating progress summaries because files are scattered across drives
- Missing deadline notifications buried in overflowing inboxes
- Spending 4-6 hours monthly on administrative tasks instead of research
- Losing research momentum due to documentation overhead

The Faculty's Story:

Scenario: Dr. Patel's Management Dilemma Dr. Patel supervises 8 student projects across different domains. His current challenges:

- Maintaining separate spreadsheets for each project's progress
- Manually tracking submission deadlines and sending reminders
- Inconsistent report formats making comparison difficult
- No centralized view of student performance across projects

- Delayed identification of struggling students or stalled projects

The Admin's Story:

Scenario: Research Office Bottleneck The Research Office receives 15-20 daily queries about:

- Project funding status and utilization
- Student progress reports for sponsor updates
- Publication tracking for annual reports
- Budget approvals delayed due to scattered documentation

Impact: Administrative staff spend 60% of time on information retrieval rather than strategic planning.

The Sponsor's Story:

Scenario: Industry Partner Frustration TechCorp sponsors 3 student projects with ₹5 lakhs total funding. Their experience:

- Irregular updates despite contractual monthly reporting requirements
- Inconsistent report formats making assessment difficult
- No visibility into fund utilization between formal reports
- Reduced confidence leading to decreased future sponsorship

Impact: 30% reduction in industry sponsorship renewal due to communication gaps.

These examples show how students, faculty, administrators, and sponsors are all affected by the lack of a proper system.

The solution is simple yet powerful: a centralized Research Project Database that brings all project-related data into one place.

This system will allow:

Students to upload reports, track their timelines, and focus on research rather than administrative hassles.

Faculty supervisors to view all projects under them in one dashboard, monitor student contributions, and ensure deadlines are met.

Administrators to track funding, sponsors, and publications with just a few clicks.

Sponsors to receive timely and transparent updates on the projects they support.

Over time, this database will also act as a historical archive of DA-IICT's research journey, highlighting achievements, mapping trends, and showcasing the institute's contributions to knowledge creation.

2.2 Root Cause Analysis

The fundamental problems stem from:

1. **Information Silos**: Data scattered across emails, drives, and personal files
2. **Manual Processes**: No automation for routine tasks and notifications
3. **Lack of Standards**: No consistent formats or workflows
4. **Poor Visibility**: No real-time insight into project status or performance
5. **Communication Gaps**: No structured channels between stakeholders

3. Requirements Collection/ Fact Finding Phase

3.1 Background Reading/s

Book Reference: **"Database System Concepts"** by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan (Covers fundamentals of databases, ER diagrams, relational models, normalization, SQL — essential for your project database design.)

Website reference: *GeeksforGeeks, "Introduction of DBMS (Database Management System),"* Available: https://www.geeksforgeeks.org/dbms/introduction-of-dbms-database-management-system-set-1/

4. The combined Requirements gathered from Background Reading/s

4.1 Interview/s

We conducted an interview of our senior and go to know about the followings:

- That DAU research department does works on digital DBMS system.
- And is thus not a fluent task to manage the paper based database.
- Tracking of project progress for faculty as well as admin is thus a hectic task.
- For students as well it is also a task to manage and track the timelines along with project work is hectic.
- want automated deadline tracking and student notifications
- require standardized report templates
- want direct communication channels with students within the system
- Thus centralized DBMS would be a best solution which would eliminate the management task from every ends whether from students, faculty admin or sponsers.
- Requirement for role-based access control
- Integration with existing financial systems
- Automated compliance and audit trails

4.2 Questionnaire/s

General Questions (for everyone)

- Can you tell us about your role in research projects? (Student, Faculty, Admin, Sponsor, etc.)
- What difficulties do you currently face while keeping track of research projects?
- If we create this system, what would be the most useful features for you?

For Students

- What kind of details about student participants should the system store? (e.g., name, roll number, department, skills, etc.)
- Would you like to be able to update your project progress directly in the system?
- Should the system let you upload project-related documents like reports or presentations?
- How should the system handle cases where multiple students are working on the same project?

For Faculty / Guides

- What faculty information should the system keep? (e.g., designation, specialization, contact info, department)
- Would it help if you had a dashboard to view all the projects you are guiding?
- Should the system track your workload (number of projects you are supervising)?
- Do you think notifications or reminders about project milestones would be useful?

Project Details

- What basic information about a project should we capture? (title, domain, objectives, timeline, status, etc.)
- Should projects be classified as sponsored, non-sponsored, completed, or ongoing?
- Should research outputs like publications, patents, or conference papers be linked with projects?
- What indicators of progress would be useful to track (milestones, deadlines, outcomes, etc.)?

Funding & Sponsors

- What funding-related details do you think should be recorded? (funding source, amount, allocation, utilization, etc.)
- Would it be useful to generate financial reports for each project?
- Do you think sponsors should have access to the system? If yes, what exactly should they be able to see?
- Should the system allow more than one sponsor for a single project?

For Admin / Management

- What kinds of reports would be most helpful for you? (e.g., projects by department, total funds received, project status overview)
- Should different users have different levels of access (student, faculty, admin, sponsor)?
- Would you like this system to connect with existing university systems (like student records or HR)?
- How should completed or archived projects be stored and accessed?

Security & Technical Considerations

- What information should be kept private or restricted?
- Who should be allowed to add, update, or delete project details?
- Do you think backup and recovery features are necessary?
- Should the system be web-based, mobile-friendly, or both?

Google Form Link:

https://docs.google.com/forms/d/e/1FAIpQLSeO3vg3Lg6l8D4zCG6LCEzdsS9q5uTaqjEVlLEx-1dI_iC2Iw/viewform?usp=sharing&ouid=109654480350081818084

Google Form Survey Questions:

1. What is your role in the research project?

• Student

• Faculty/Guide

• Sponsor

- Other…

2. How many research projects have you been part of?

- 0
- 1
- 2
- More than 2

3. What type of project are you currently working on?

- Sponsored
- Unsponsored
- Collaborative
- Independent

4. How is your project funded?

- Self-funded
- DAU-funded
- Industry-funded
- Government-funded

5. Which features would you like in the Research Project DBMS?

- Student & Faculty Records Management
- Project & Funding Tracking
- Publication Database
- Sponsor Details Management
- All of the above
- Other…

6. How should the system be accessed?

- Website
- Mobile app
- Both
- No preference

7. What level of access should faculty have in the system?

- View only student/project details
- Add/edit project details

- Approve/validate project updates
- Full access

8. How should project progress be tracked?

- Milestone/Phase completion
- Percentage completion
- Document uploads/reports
- Regular status updates

9. How should sponsors be managed in the DBMS?

- Only record sponsor details
- Link sponsors to specific projects
- Allow sponsors to track project status
- All of the above

4.3 Observation/:

Survey Responses:

Timestamp	Email Address	1. What is your role in the research project?	2. How many research projects have you been part of?	3. What type of project are you currently working on?	3. How is your project funded?	4. Which features would you like in the Research Project DBMS?	5. How should the system be accessed?	6. What level of access should faculty have in the system?	7. How should project progress be tracked?	8. How should sponsors be managed in the DBMS?
9/2/2025 18:55:43	nevilnandasana@gmail.com	Student	0	Unsponsored	self-funded	All the above	both	Approve/validate project updates	Regular status updates	Link sponsors to specific projects
9/2/2025 18:59:52	202512079@dau.ac.in	Student	0	independent	self-funded	Project & Funding Tracking	website	Full access	Percentage completion	Link sponsors to specific projects
9/2/2025	devbarot1012@gmail.com	Student	0			All the above		Approve/validate	Document	Only record

19:03:15								project updates	uploads/reports	sponsor details
9/2/2025 19:38:41	doshikrisha781@gmail.com	Student	0			Project & Funding Tracking	website	Approve/validate project updates	Document uploads/reports	Allow sponsors to track project status
9/2/2025 20:24:09	nishkashah6@gmail.com	Student	0			Student & Faculty Records Management	mobile app	Add/edit project details	Milestone/Phase completion	Link sponsors to specific projects
9/2/2025 23:30:42	202512017@dau.ac.in	Student	1	independent	self-funded	All the above	both	Full access	Document uploads/reports	Link sponsors to specific projects
9/5/2025 22:20:28	priyanshusoni5440@gmail.com	Student	1	independent	self-funded	Student & Faculty Records Management	website	View only student/project details	Milestone/Phase completion	Only record sponsor details
9/6/2025 14:36:44	kbs384650@gmail.com	Student	more than 2	Unsponsored		Student & Faculty Records	both	Add/edit project details	Regular status updates	All of the above

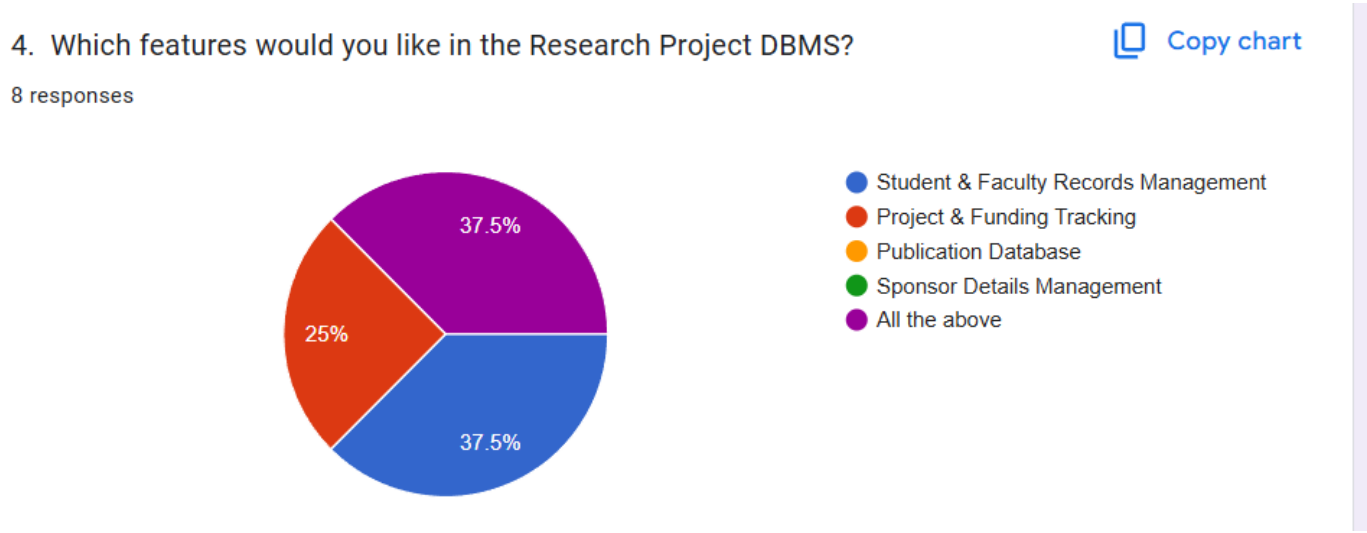
						Manag ement				

Observation:

- We managed to get the insights from students.
- Mostly projects are unsponsored and thus we think that extra attributes of sponsored projects can be linked to the project relation by creating one extra relation or entity that too weak entity which would be dependent on the project relation.

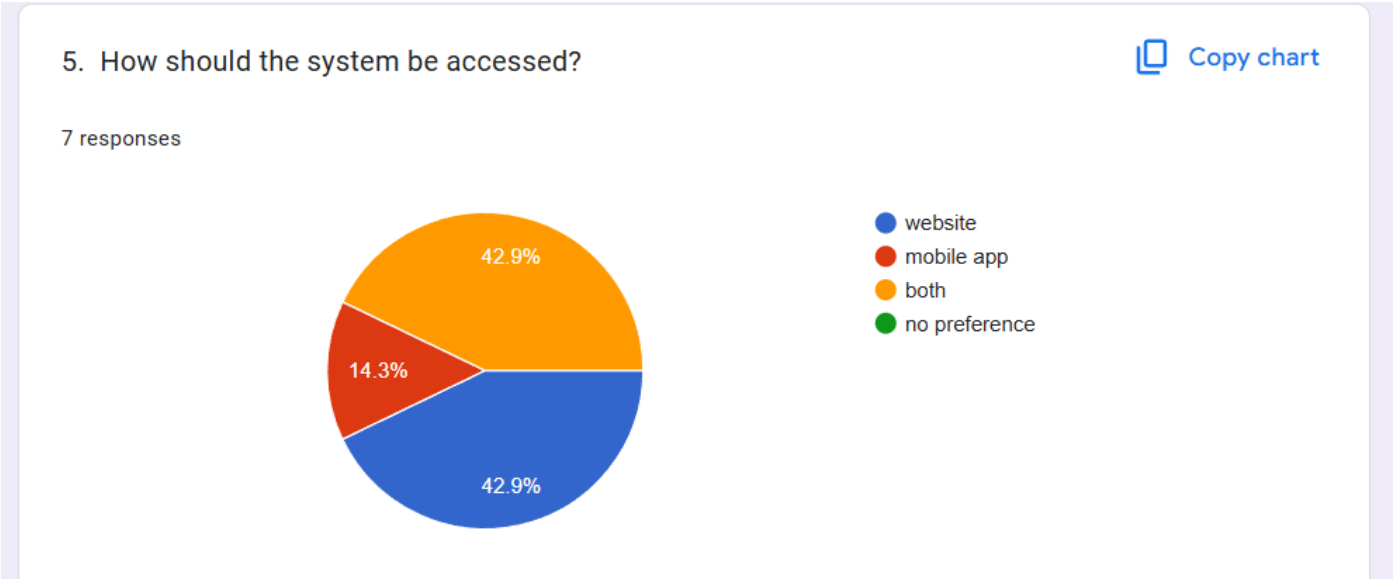
As every sponsored project must be an existing project.

- Most respondents' feels that this database should collect and manage only student and faculty related data or track funding related to project….. But as per the interview tracking timeline from students end is also a difficult task. Thus we are upto include students and faculty records management as well as tracking of projects.



- Most respondents feels that web site and mobile application is appropriate.

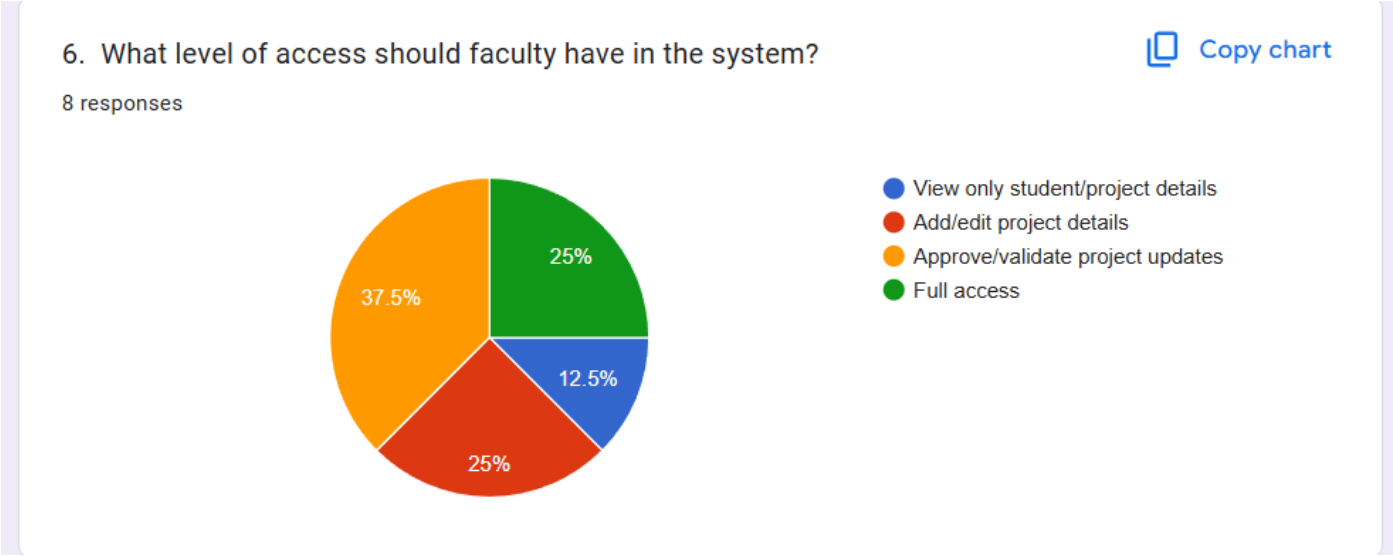
We think website would be most appropriate taking data space constrain in head.



- About the accessibility from faculty's end most respondents feels that faculty must be able to approve project updates, along with that we are thinking to allow the faculty to access the project details and project_progress details for only those project to which he/she is linked via project relation.

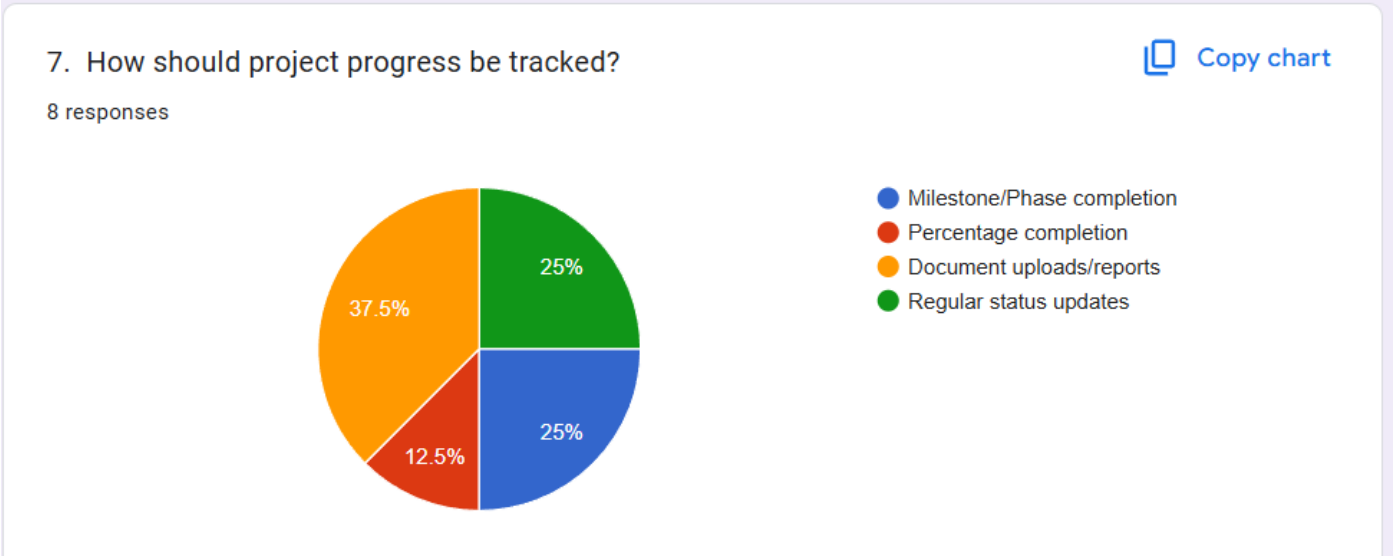
Otherwise, it would be hectic from their ends to manage the projects under them.

These constrain would allow them to check the progress for every projects under them, under just one dashboard.



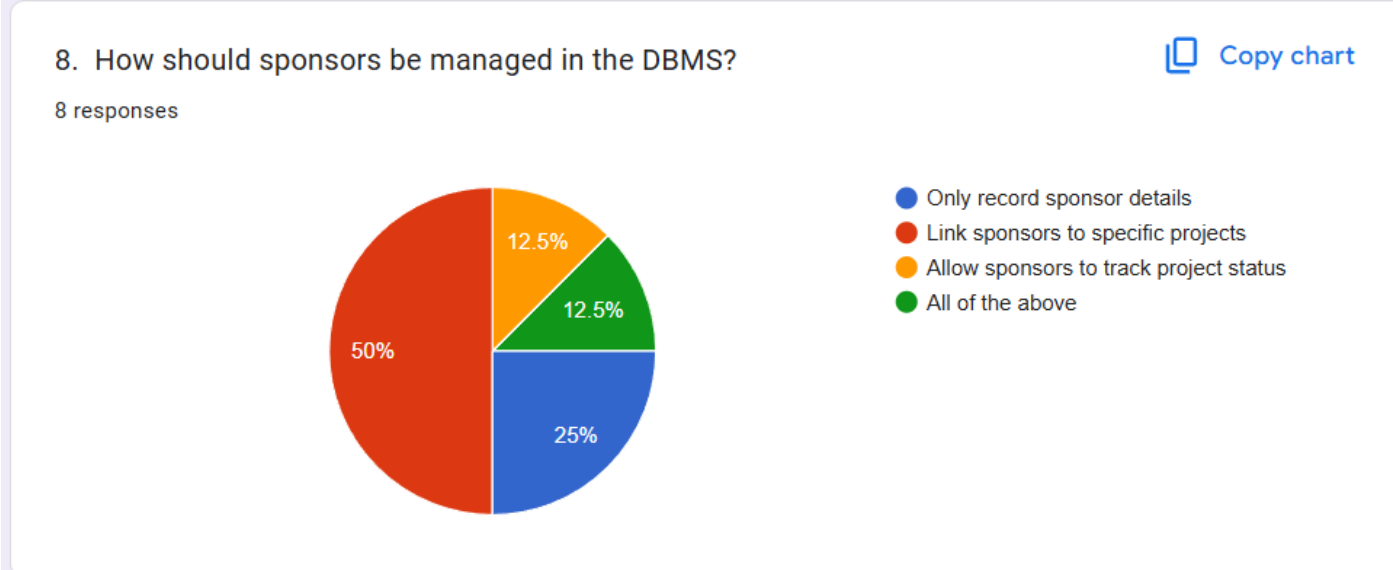
- To track the project progress we would design one relation which would act as weak entity dependent on project , carrying details like project id , progress id and document and report

Which could be uploaded by student and whose remark would thus be upto the faculty to mail to the respective students about the same.



- As per the response and what we feel about the sponsors connection with project we think to make a relation tbl_Researchproject , tbl_sponsor and realtion set tbl_sponsors which would link the sponsors with the project they are sponsoring.

This would limit the access of sponsors till accessing the details of the project they are sponsoring that's it. No more details will be exposed to the external sponsors regarding other projects.



5. List Requirements(based on database designing)

5.1 Functional Requirements

5.1.1 User Authentication and Authorization

1.1: Users must authenticate using institute credentials (LDAP integration)

1.2: System shall provide role-based access control (Student, Faculty, Admin, Sponsor)

1.3: Users can update their profile information except role assignments

1.4: Session management with automatic timeout after 2 hours of inactivity

5.1.2 Project Management Core Features

2.1: Faculty can create new research projects with complete metadata

2.2: Students can be assigned to projects with defined roles and permissions

2.3: Project status tracking through defined lifecycle stages (Proposed → Active → On Hold → Completed → Archived)

2.4: Automated deadline notifications via email and in-app alerts

5.1.3 Document Management System

3.1: Students can upload progress reports in standardized templates

3.2: Document version control with revision history

3.3: Faculty can review and provide feedback on submitted documents

3.4: Document approval workflow with status tracking

3.5: Secure file storage with access controls

3.6: Document search and filter functionality

5.1.4 Reporting and Analytics

4.1: Real-time project dashboards for all user roles

4.2: Customizable report generation (PDF, Excel formats)

4.3: Statistical analysis of project performance metrics

4.4: Publication and patent tracking linked to projects

4.5: Sponsor-specific reporting with automated generation

5.1.5 **Administrative Functions**

5.1: User management interface for administrators

5.2: System configuration and parameter settings

5.3: Data backup and recovery procedures

5.4: Integration APIs for external systems

5.5: Comprehensive audit trail and compliance reporting

5.2 User Interface Requirements

5.2.1 General UI Principles

- **UI-1**: Responsive design supporting desktop, tablet, and mobile devices
- **UI-2**: Intuitive navigation with maximum 3 clicks to reach any function
- **UI-3**: Consistent design language following modern web standards
- **UI-4**: Accessibility compliance (WCAG 2.1 Level AA)
- **UI-5**: Multi-language support (English and Hindi)

5.2.2 Role-Specific Interfaces

- **UI-6**: Student dashboard showing personal projects, deadlines, and recent activities
- **UI-7**: Faculty supervisor dashboard with all guided projects overview
- **UI-8**: Administrative control panel with system-wide metrics and management tools
- **UI-9**: Sponsor portal with investment-specific project insight.

5.3 Designing Requirements

//based on the reading and survey findings we feel the requirements for particular users could be:

The relations required for it could be:

 tbl_faculty – Stores details of faculty members guiding research projects, with attributes such as *faculty_id, name, course_id,* etc.

Would be handled mainly by admin and by particular entities itself for updating his/her details.

 tbl_student – Contains student information including *student_id, name, program_id,* and other academic details.

Would be handled mainly by admin and by particular entities itself for updating his/her details.

 tbl_ResearchProject – Acts as the core relation linking students and faculty, storing attributes such as *project_id, student_id, faculty_id, project_title, subject, start_date, submission_date, status,* etc.

Would be handles by admin to manage this table, this relation would allow the connecting entities to view each other like faculty under any such project could view the student under same project and vice-versa. Also faculty would be having rights to update its records like status and all.

 tbl_ResearchProjectProgress – this relation would be a weak entity set dependent on the **tbl_ResearchProject as it would keep track of every progress of every projects. It would store attributes like progress_id, project_id, project_doc, etc.**

Validation or remark or changes on which will be provided by faculty to respective student would be sent by faculty to student through mail.

This would be accessible to faculty and students only as it would include day to day work / report submissions subject to research project.

 tbl_fundings – Maintains financial data such as *funding_id, amount, date of allocation,* and other budget-related details.

Mainly for admin use and thus would be managed by them only.

Students and faculty associated project funding would be visible to them.

 tbl_sponsor – Stores sponsor-specific information like *sponsor_id, name, organization, contact details,* etc.

This would be managed by admin only.

 tbl_sponsors – Would act as connecting bridge between the sponsors and their respective sponsoring projects. It would be and relationship set between three entity sets i.e. tbl_ResearchProject and tbl_sponsor and tbl_funding. with attributes such as *sponsor_id, funding_id,* and related fields.

This would be managed by admin.

Which would allow the particular sponsors to track their sponsored project details.

 tbl_publication – this would store the publication details for administration and historic records.

This would be managed by admin itself.

Database Schema Requirements:

Entity Definitions and Relationships:

tbl_faculty - Faculty Member Management

- Primary Key: faculty_id (auto-increment integer)
- name (varchar 100, not null)
- email (varchar 100, unique, not null)
- department (varchar 50, not null)
- designation (varchar 50)
- specialization (varchar 200)
- contact_phone (varchar 20)
- office_location (varchar 100)
- hire_date (date)
- active_status (boolean, default true)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_student - Student Information Management

- Primary Key: student_id (auto-increment integer)
- enrollment_id (varchar 20, unique, not null)
- name (varchar 100, not null)
- email (varchar 100, unique, not null)
- program_type (enum: 'undergraduate', 'masters', 'phd', 'postdoc')
- department (varchar 50, not null)

- year_of_study (integer)
- contact_phone (varchar 20)
- address (text)
- enrollment_date (date)
- expected_graduation (date)
- active_status (boolean, default true)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_ResearchProject - Core Project Management

- Primary Key: project_id (auto-increment integer)
- project_title (varchar 200, not null)
- project_code (varchar 50, unique, not null)
- faculty_id (foreign key to tbl_faculty)
- student_id (foreign key to tbl_student)
- subject_domain (varchar 100, not null)
- project_description (text)
- objectives (text)
- methodology (text)
- start_date (date, not null)
- planned_end_date (date, not null)
- actual_end_date (date)
- status (enum: 'proposed', 'approved', 'active', 'on_hold', 'completed', 'cancelled')
- priority_level (enum: 'low', 'medium', 'high', 'critical')
- budget_allocated (decimal 12,2)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_ResearchProjectProgress - Progress Tracking (Weak Entity)

- Primary Key: progress_id (auto-increment integer)
- project_id (foreign key to tbl_ResearchProject, not null)
- submission_date (date, not null)
- progress_description (text, not null)

- milestone_achieved (varchar 200)
- percentage_complete (decimal 5,2)
- documents_submitted (json) -- Array of document references
- faculty_remarks (text)
- faculty_approval_status (enum: 'pending', 'approved', 'needs_revision', 'rejected')
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_sponsor - Sponsor Information Management

- Primary Key: sponsor_id (auto-increment integer)
- sponsor_name (varchar 150, not null)
- organization_type (enum: 'government', 'private', 'ngo', 'international', 'internal')
- contact_person (varchar 100)
- email (varchar 100, not null)
- phone (varchar 20)
- address (text)
- website_url (varchar 200)
- industry_sector (varchar 100)
- established_year (year)
- relationship_start_date (date)
- active_status (boolean, default true)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_funding - Financial Management

- Primary Key: funding_id (auto-increment integer)
- funding_source_name (varchar 100, not null)
- total_amount (decimal 15,2, not null)
- currency (varchar 3, default 'INR')
- funding_type (enum: 'grant', 'contract', 'donation', 'internal', 'self_funded')
- start_date (date, not null)
- end_date (date, not null)
- terms_conditions (text)

- reporting_requirements (text)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_sponsors - Project-Sponsor-Funding Relationship

- Primary Key: sponsors_id (auto-increment integer)
- project_id (foreign key to tbl_ResearchProject, not null)
- sponsor_id (foreign key to tbl_sponsor, not null)
- funding_id (foreign key to tbl_funding, not null)
- sponsorship_amount (decimal 12,2, not null)
- sponsorship_start_date (date, not null)
- sponsorship_end_date (date, not null)
- deliverables_required (text)
- reporting_frequency (enum: 'monthly', 'quarterly', 'semi_annual', 'annual', 'milestone_based')
- contract_document (varchar 500)
- active_status (boolean, default true)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

tbl_publication - Research Output Management

- Primary Key: publication_id (auto-increment integer)
- project_id (foreign key to tbl_ResearchProject, not null)
- title (varchar 300, not null)
- publication_type (enum: 'journal', 'conference', 'book_chapter', 'patent', 'thesis', 'report')
- authors (text, not null) -- JSON array of author information
- venue (varchar 200) -- Journal/Conference name
- publication_date (date)
- doi (varchar 100)
- isbn_issn (varchar 50)
- impact_factor (decimal 5,3)
- citation_count (integer, default 0)
- abstract (text)
- keywords (text) -- Comma-separated keywords

- file_path (varchar 500)
- open_access (boolean, default false)
- created_date (timestamp, default current_timestamp)
- modified_date (timestamp, default current_timestamp on update)

Database Constraints and Relationships:

```
-- Foreign Key Constraints
ALTER TABLE tbl_ResearchProject
ADD CONSTRAINT fk_project_faculty
FOREIGN KEY (faculty_id) REFERENCES tbl_faculty(faculty_id) ON DELETE RESTRICT;

ALTER TABLE tbl_ResearchProject
ADD CONSTRAINT fk_project_student
FOREIGN KEY (student_id) REFERENCES tbl_student(student_id) ON DELETE RESTRICT;

ALTER TABLE tbl_ResearchProjectProgress
ADD CONSTRAINT fk_progress_project
FOREIGN KEY (project_id) REFERENCES tbl_ResearchProject(project_id) ON DELETE CASCADE;

ALTER TABLE tbl_sponsors
ADD CONSTRAINT fk_sponsors_project
FOREIGN KEY (project_id) REFERENCES tbl_ResearchProject(project_id) ON DELETE CASCADE;

ALTER TABLE tbl_sponsors
ADD CONSTRAINT fk_sponsors_sponsor
FOREIGN KEY (sponsor_id) REFERENCES tbl_sponsor(sponsor_id) ON DELETE RESTRICT;

ALTER TABLE tbl_sponsors
ADD CONSTRAINT fk_sponsors_funding
FOREIGN KEY (funding_id) REFERENCES tbl_funding(funding_id) ON DELETE RESTRICT;

ALTER TABLE tbl_publication
```

```
ADD CONSTRAINT fk_publication_project
FOREIGN KEY (project_id) REFERENCES tbl_ResearchProject(project_id) ON DELETE RESTRICT;

-- Check Constraints
ALTER TABLE tbl_ResearchProjectProgress
ADD CONSTRAINT chk_percentage_range
CHECK (percentage_complete >= 0 AND percentage_complete <= 100);

ALTER TABLE tbl_ResearchProject
ADD CONSTRAINT chk_date_logic
CHECK (planned_end_date >= start_date);

ALTER TABLE tbl_funding
ADD CONSTRAINT chk_positive_amount
CHECK (total_amount > 0);
```

6. User Classes and Characteristics

- **Students** – Work on research projects, upload reports, and check deadlines. They need an easy-to-use system.
- **Faculty** – Guide multiple student projects. They need to see progress, remark or guide the students work and manage timelines.
- **Administrators** – Handle student, faculty, project, funding, sponsors, historic information regarding publications, and reports. They need quick and accurate information.
- **Sponsors** – Provide money for projects. They only need updates about the projects they fund.

6.1 Primary User Classes

6.1.1 Students

Characteristics:

- Tech-savvy but time-constrained
- Mobile-first users preferring app-like interfaces
- Need clear guidance and structured workflows
- Collaborative and communication-focused

Primary Use Cases:

1. **Upload Progress Report**
    - Student logs in and navigates to project dashboard
    - Selects project and creates new progress entry
    - Uploads documents using standardized template
    - Submits for faculty review
    - Receives confirmation and tracking number
2. **Track Project Timeline**
    - View project milestones and deadlines
    - Check completion status and pending tasks
    - Receive automated reminders for upcoming deadlines
    - Access historical progress data
3. **Communication with Supervisor**
    - Send direct messages within project context
    - Post questions on project discussion board
    - Receive feedback on submitted work
    - Schedule review meetings

6.1.2 Faculty

Characteristics:

- Experienced users comfortable with desktop interfaces
- Manage multiple projects simultaneously
- Need comprehensive oversight capabilities
- Focus on guidance and quality control

Primary Use Cases:

1. **Project Portfolio Management**
    - View dashboard of all supervised projects
    - Monitor student progress across multiple projects
    - Identify projects requiring immediate attention
    - Generate comparative performance reports
2. **Review and Feedback Process**
    - Access submitted student reports
    - Provide structured feedback using standard forms
    - Approve or request revisions on deliverables
    - Track student response to feedback
3. **Administrative Oversight**
    - Update project status and milestones
    - Manage project budgets and resources
    - Generate sponsor reports
    - Archive completed projects

6.1.3 Administrator

Characteristics:

- Power users requiring comprehensive system access
- Focus on institutional metrics and compliance
- Need detailed reporting capabilities
- Manage system configuration and user access

Primary Use Cases:

1. **Institutional Research Overview**
    - Monitor all active projects across departments
    - Generate institutional performance metrics
    - Track funding utilization and sources
    - Produce compliance and audit reports
2. **User and System Management**

- Create and manage user accounts
- Configure system parameters and workflows
- Monitor system usage and performance
- Manage data backup and security

6.1.4 Sponsors

Characteristics:

- Limited system access focused on funded projects
- Business-oriented requiring ROI visibility
- Need regular, standardized reporting
- Value transparency and communication

Primary Use Cases:

1. **Investment Monitoring**
    - Access real-time project progress for sponsored projects
    - Review budget utilization and financial reports
    - Track deliverable completion and timeline adherence
    - Evaluate project outcomes and publication.

6.2 Detailed Use Case Specifications

Use Case 1: Student Project Report Submission

Actors: Student (Primary), Faculty (Secondary) **Preconditions:** Student is registered and assigned to active project
Main Flow:

1. Student authenticates and accesses project dashboard
2. System displays current project status and pending requirements
3. Student selects "Submit Progress Report" option

4. System presents standardized report template
5. Student completes required fields and uploads supporting documents
6. System validates submission completeness and format
7. Student reviews and confirms submission
8. System generates submission confirmation and notifies supervisor
9. System updates project timeline and progress metrics

Alternative Flows:

- **A1**: Incomplete submission - System highlights missing fields and prevents submission
- **A2**: File format error - System provides format guidelines and requests correction
- **A3**: Late submission - System flags delay and notifies supervisor

Postconditions: Report is submitted, supervisor notified, project progress updated

7. Operating Environment

The system will run on standard desktops/laptops in labs, admin offices, and faculty rooms, with optional mobile access for students. It will use a relational DBMS (such as MySQL or PostgreSQL) and a web-based interface accessible through common browsers like Chrome, Edge, and Firefox. Secure internet/intranet connectivity is required for multi-user access.

The system should be able to integrate with other institute systems (such as student records or HR) through APIs if needed, and also support exporting reports in formats like PDF or Excel for sponsors. Interfaces must be simple and user-friendly so that students, faculty, administrators, and sponsors can use the system with minimal training.

From a business side, the system must stay within the institute's budget (favoring open-source tools), be ready before the next academic session, and meet reporting schedules (monthly for students, quarterly for sponsors). Access will remain strictly role-based, ensuring that each user only sees information relevant to their role.

8. Product Function

The system will provide the following key functions:

- **Students**: Upload progress reports, check deadlines, view project status.
- **Faculty**: Monitor projects under their supervision, track student contributions, set deadlines, update project details like status, and can provide remarks on students work submitted under tbl_ResearchProjectProgress by mailing them.
- **Administrators**: Manage students, faculty, sponsors, projects; funding allocations; track sponsors & publications; generate consolidated reports.
- **Sponsors**: Access transparent updates on projects they support, view fund utilization.
- **System-wide**: Store project metadata (titles, subjects, timelines, publications), manage role-based access, and maintain historical records of research activities.
- **DAIICT-WEB Site**: content of successful projects can be projected on main college web site.

9. Privileges

9.1 Functions & Users who can access those modules.

9.1.1 Upload Reports:

Only Students can upload report which would be stored under tbl_ResearchProjectProgress.

9.1.2 Add Submission date:

Faculty can add or update submission date of project based on the work done.

9.1.3 Update Project Status:

Faculty can update the project status based on the documents received under

tbl_ResearchProjectProgress.

9.1.4 View Project Progress:

Only associated faculty to particular project can view the documents submitted on tbl_ResearchProjectProgress by respective student.

9.1.5 View Funding details:

Admin can view all records while students , faculty, and sponsors can view it for project to which they are related or associated with.

9.1.6 Manage Publication Details:

Admin can add, update, delete the publication details.

It can also publish the data on website to showcase the success of project and publication.

9.1.7 Manage student:

Admin can manage student and the respective entity of entity set can update their details.

9.1.8 Manage Faculty:

Admin can manage faculty and the respective entity of entity set can update their details.

9.1.9 Manage Sponsor:

Admin can manage sponsor and the respective entity of entity set can update their details.

9.1.10 Mange Project details:

Admin can manage the project details, while faculty can have access to view and update the details of associated projects. Students and sponsors can have access to view the details of the associated projects.

10. Assumptions

Assumptions:

Users have access to basic computing resources (PCs, internet, institute accounts).

Faculty, students, and administrators will actively update the system with correct data.

Sponsors are willing to adopt online reporting instead of manual updates.

Institute IT infrastructure supports deployment and hosting.

Dependencies:

Database depends on institute's central IT systems (servers, storage, security).

Smooth functioning relies on consistent internet/intranet connectivity.

Future enhancements (AI analytics, dashboards, publication tracking) depend on integration with external APIs/digital libraries.

Data quality depends on timely and accurate updates from users.

11. Business Constrains:

While designing the Research Project Database, we also have to consider some rules and limitations that come from the institute's policies, available resources, and sponsor requirements. These are not technical limits but business-related conditions that shape how the system should work.

- **Budget and Cost**
  The system has to be built and maintained within the limited budget provided by the institute. Expensive software or licenses should be avoided, and open-source options are preferred wherever possible.
- **Time and Deadlines**
  The database must be ready before the start of the next academic session so that students and faculty can use it in their projects. Also, project updates (like student reports or sponsor updates) must follow the timelines set by the institute—for example, students submit monthly reports, and sponsors get quarterly updates.
- **Policies and Compliance**
  The system must follow institute rules regarding data privacy, research ethics, and intellectual property. Since it will also store financial details and sponsor information, it should follow government and funding agency guidelines.
- **Access Restrictions**
  Not every user should see everything. Students will only see their own project data, sponsors will only see details of projects they are funding, and faculty will only see the projects they are guiding. This ensures privacy and trust.
- **Growth and Storage**
  The system should be able to handle more projects and users in the future. It must also store documents, reports, and publications safely for at least 10 years so that future researchers and administrators can refer back to them.
- **Sponsor Reporting**
  Many sponsors expect updates in specific formats, like PDF or Excel reports. The system should be able to generate such reports automatically and make sure they are delivered on time as per sponsor agreements.
- **Operations and Backup**
  The system should not go down during important academic deadlines (like submission weeks). Regular backups should be taken every day and kept safely for at least 6 months to avoid data loss.
- **Unique Seiling Proposition**

  Specialization search engine on student's interface,

  It would be helpful for students to connect with faculty specialized in domain in which he/she wants to look for any research and so.

Lab-6: Research Project Database - Comprehensive Noun Analysis and ER Diagram

1. Overall Problem Description

1.1 Product Perspective

The Research Project Database is envisioned as a **centralized management system** for DA-IICT's research projects. Currently, information is fragmented across spreadsheets, emails, and manual reports, leading to inefficiencies.
This product will replace these scattered methods with a single platform that integrates students, faculty, administrators, and sponsors. It acts as both a **transactional system** (data entry, updates, report uploads) and a **decision-support tool** (dashboards, budget tracking, sponsorship reporting).

The Student's Story:

Scenario: Riya's Research Journey Riya, a final-year Computer Science student, is working on a Machine Learning project under Dr. Patel's guidance. Her monthly routine involves:

- Searching through email chains to find her previous reports
- Recreating progress summaries because files are scattered across drives
- Missing deadline notifications buried in overflowing inboxes
- Spending 4-6 hours monthly on administrative tasks instead of research
- Losing research momentum due to documentation overhead

The Faculty's Story:

Scenario: Dr. Patel's Management Dilemma Dr. Patel supervises 8 student projects across different domains. His current challenges:

- Maintaining separate spreadsheets for each project's progress
- Manually tracking submission deadlines and sending reminders
- Inconsistent report formats making comparison difficult
- No centralized view of student performance across projects
- Delayed identification of struggling students or stalled projects

The Admin’s Story:

Scenario: Research Office Bottleneck The Research Office receives 15-20 daily queries about:

- Project funding status and utilization
- Student progress reports for sponsor updates
- Publication tracking for annual reports
- Budget approvals delayed due to scattered documentation

Impact: Administrative staff spend 60% of time on information retrieval rather than strategic planning.

The Sponsor’s Story:

Scenario: Industry Partner Frustration TechCorp sponsors 3 student projects with ₹5 lakhs total funding. Their experience:

- Irregular updates despite contractual monthly reporting requirements
- Inconsistent report formats making assessment difficult
- No visibility into fund utilization between formal reports
- Reduced confidence leading to decreased future sponsorship

Impact: 30% reduction in industry sponsorship renewal due to communication gaps.

These examples show how students, faculty, administrators, and sponsors are all affected by the lack of a proper system.

The solution is simple yet powerful: a centralized Research Project Database that brings all project-related data into one place.

This system will allow:

Students to upload reports, track their timelines, and focus on research rather than administrative hassles.

Faculty supervisors to view all projects under them in one dashboard, monitor student contributions, and ensure deadlines are met.

Administrators to track funding, sponsors, and publications with just a few clicks.

Sponsors to receive timely and transparent updates on the projects they support.

Over time, this database will also act as a historical archive of DA-IICT's research journey, highlighting achievements, mapping trends, and showcasing the institute's contributions to knowledge creation.

1.2 Root Cause Analysis

The fundamental problems stem from:

6. **Information Silos**: Data scattered across emails, drives, and personal files
7. **Manual Processes**: No automation for routine tasks and notifications
8. **Lack of Standards**: No consistent formats or workflows
9. **Poor Visibility**: No real-time insight into project status or performance
10. **Communication Gaps**: No structured channels between stakeholders

Extended Problem Description: Research Project Database Management System

Current State Analysis and Problem Identification

The Research Landscape at Academic Institutions

Modern academic institutions operate in an increasingly complex research environment where the management of research projects has become a critical factor in institutional success. Universities

and research institutes worldwide are handling hundreds of concurrent research projects, involving thousands of students, faculty members, and external stakeholders. The traditional approaches to managing this complexity have proven inadequate, leading to significant operational inefficiencies and missed opportunities for knowledge creation and dissemination.

At DA-IICT, like many other premier technical institutions, research activities span multiple disciplines including computer science, electronics, information and communication technology, and emerging interdisciplinary fields. The institute currently manages approximately 200+ active research projects at any given time, involving undergraduate capstone projects, master's thesis work, PhD dissertations, and faculty-led research initiatives. These projects generate substantial data volumes, require complex coordination between multiple stakeholders, and demand rigorous tracking of progress, budgets, and outcomes.

Industry Context and Global Trends

The global research management landscape has undergone significant transformation in recent years. Leading universities such as MIT, Stanford, and IIT systems have implemented sophisticated research information management systems (RIMS) that provide comprehensive visibility into their research portfolios. These institutions report 40-60% improvements in research productivity and 70% reduction in administrative overhead after implementing centralized research databases.

International funding agencies including NSF, NIH, European Research Council, and Indian funding bodies like SERB and CSIR are increasingly requiring real-time project tracking and standardized reporting formats. Universities that cannot provide timely, accurate reporting face funding penalties and reduced competitiveness in securing future grants. The Department of Science and Technology (DST) India has mandated digital project management systems for all funded research, making such systems essential for institutional compliance.

Furthermore, the COVID-19 pandemic has accelerated the need for digital research collaboration tools. Remote supervision, virtual progress reviews, and digital documentation have become standard practice, making centralized database systems not just convenient but necessary for continued research operations.

Critical Problems in Current Research Management

1. Information Fragmentation and Data Silos

The most pressing issue facing research project management is the severe fragmentation of information across multiple platforms and storage systems. Currently, project-related data exists in disparate locations:

Email Communications: Critical project updates, deadline changes, and feedback are buried in email threads that become increasingly difficult to track and reference. Faculty supervisors often manage multiple projects through separate email chains, leading to confusion and missed communications.

Local File Systems: Students store project documents on personal computers or shared drives with inconsistent naming conventions and version control. This leads to frequent instances where the most current version of important documents cannot be located when needed.

Spreadsheet Management: Faculty members maintain individual Excel or Google Sheets files to track student progress, but these sheets are not interconnected, leading to duplicate data entry and inconsistent formatting across different supervisors.

Physical Documentation: Many administrative processes still rely on paper-based forms for project approvals, budget requests, and milestone confirmations, creating bottlenecks in information flow and making data analysis nearly impossible.

This fragmentation results in an estimated 15-20% loss in research productivity as students and faculty spend significant time on administrative tasks rather than actual research work. The lack of centralized information also makes it extremely difficult to generate institutional reports, track research trends, or demonstrate research impact to stakeholders.

2. Inefficient Progress Monitoring and Deadline Management

Research projects are inherently time-sensitive, with multiple interconnected deadlines including proposal submissions, progress reviews, interim presentations, thesis defenses, and publication deadlines. The current manual tracking system has several critical flaws:

Passive Deadline Tracking: Students and faculty rely on personal calendars and manual reminders, resulting in approximately 30% of projects experiencing delays due to missed intermediate deadlines.

Inconsistent Progress Reporting: Without standardized templates or submission processes, progress reports vary significantly in quality and content, making it difficult for supervisors to assess actual project status or compare progress across different projects.

Limited Visibility: Faculty supervisors often lack real-time visibility into student work between formal meetings, making it difficult to identify struggling projects early or provide timely intervention.

Administrative Overhead: Research office staff spend approximately 60% of their time collecting, organizing, and redistributing project status information rather than focusing on strategic research support activities.

3. Financial Management and Sponsor Relations Challenges

Research project funding involves complex relationships between the institution, external sponsors, and internal budget allocation systems. Current challenges include:

Opaque Budget Tracking: Sponsors receive irregular updates on fund utilization, leading to decreased trust and reduced likelihood of continued funding. Industry partners report that inconsistent financial reporting is a major factor in their decisions to reduce research investments.

Manual Report Generation: Creating sponsor reports requires manual data collection from multiple sources, often taking 2-3 days of administrative effort per report and resulting in delays that frustrate sponsor organizations.

Limited Financial Oversight: Project budgets are tracked in isolation, making it difficult to identify cost overruns early or reallocate resources between projects efficiently.

Compliance Risks: Without automated tracking, the institution faces potential compliance issues with funding agency requirements, particularly for government-funded research projects that have strict reporting mandates.

4. Publication and Outcome Tracking Deficiencies

Research projects generate valuable intellectual property through publications, patents, and other research outputs, but current tracking methods are inadequate:

Disconnected Publication Records: Published papers are often not systematically linked back to their originating research projects, making it impossible to demonstrate research impact or track return on investment for sponsored projects.

Lost Institutional Memory: Completed projects and their outcomes are not systematically archived, leading to duplication of effort when new researchers work on similar problems.

Impact Assessment Difficulties: Without comprehensive outcome tracking, the institution cannot effectively demonstrate its research impact to accreditation bodies, potential sponsors, or in competitive funding applications.

Stakeholder Impact Analysis

Student Experience Challenges

Research students currently face numerous obstacles that detract from their primary focus on learning and discovery:

Administrative Burden: Students report spending 4-6 hours monthly on administrative tasks related to progress reporting, document management, and communication coordination.

Uncertainty and Stress: Without clear visibility into project timelines and requirements, students experience unnecessary stress about meeting expectations and deadlines.

Limited Collaboration: The current system provides no mechanism for students working on related projects to discover each other or share resources effectively.

Documentation Difficulties: Students struggle to maintain comprehensive project documentation, often losing track of important decisions, methodology changes, or experimental results.

Faculty Supervisor Frustrations

Faculty members supervising research projects encounter significant management overhead:

Time Inefficiency: Faculty report that administrative tasks related to project management consume 20-25% of their research-related time, reducing their ability to focus on actual research guidance and mentorship.

Limited Oversight Capability: With no centralized dashboard, faculty supervising multiple projects find it challenging to maintain awareness of each project's status and identify projects requiring immediate attention.

Inconsistent Data Quality: Without standardized reporting templates, faculty receive progress updates in varying formats and levels of detail, making it difficult to assess actual progress or provide targeted feedback.

Communication Gaps: Faculty often discover problems with student projects weeks after they occur, when early intervention could have prevented more serious issues.

Administrative and Institutional Challenges

Institute administrators face significant challenges in managing the overall research portfolio:

Strategic Planning Difficulties: Without comprehensive data on research project performance, administrators cannot effectively plan resource allocation, identify successful research areas for expansion, or recognize underperforming domains requiring support.

Compliance Risks: Manual tracking systems create risks of non-compliance with funding agency requirements, accreditation standards, and institutional reporting obligations.

Resource Optimization Problems: Administrators cannot efficiently allocate shared resources (equipment, laboratory space, computational resources) across projects due to lack of visibility into actual project needs and timelines.

Performance Measurement Challenges: The institution cannot effectively measure research productivity, track improvement over time, or benchmark performance against peer institutions.

Sponsor and External Partner Concerns

External stakeholders who fund or collaborate on research projects face their own set of challenges:

Limited Transparency: Sponsors receive infrequent and inconsistent updates on project progress, making it difficult to assess whether their investments are being used effectively.

Communication Inefficiencies: Sponsor representatives often need to contact multiple individuals to get comprehensive project status updates, creating frustration and reducing their confidence in the institution's project management capabilities.

Outcome Tracking Difficulties: Sponsors struggle to connect their research investments to eventual outcomes like publications, patents, or commercial applications.

Technology Infrastructure Context

The current technological landscape at DA-IICT, while robust in many areas, lacks integration specifically designed for research project management:

Existing Systems: The institute maintains separate systems for student information management, financial tracking, and academic administration, but these systems do not communicate effectively with each other.

User Technology Expectations: Students and faculty are accustomed to modern, integrated digital experiences in other aspects of their academic and personal lives, making the current fragmented approach particularly frustrating.

Scalability Requirements: As the institute continues to grow and expand its research portfolio, the current manual systems will become increasingly inadequate.

Competitive and Strategic Context

The research project database initiative must be viewed within the broader context of institutional competitiveness and strategic positioning:

Institutional Reputation: Efficient research management directly impacts the institution's ability to attract top faculty, students, and research funding.

Industry Partnerships: Technology companies and research organizations increasingly expect their academic partners to have sophisticated project management capabilities.

Accreditation and Rankings: Research productivity and management efficiency are key factors in institutional accreditation and national ranking assessments.

Innovation Opportunities

The proposed Research Project Database Management System represents an opportunity to not just solve current problems but to create competitive advantages:

Data-Driven Research Management: By capturing comprehensive project data, the system will enable predictive analytics to identify potential project issues before they become critical.

Enhanced Collaboration: The system can facilitate discovery of synergies between different research projects and enable more effective resource sharing.

Automated Compliance: Built-in compliance tracking can ensure that all projects meet funding agency requirements without manual oversight.

Research Impact Demonstration: Comprehensive outcome tracking will enable the institution to better demonstrate its research impact to stakeholders and funding agencies.

Real-World Case Studies and Impact Analysis

Case Study 1: Stanford Research Management Platform

Stanford University implemented their Research Management System (RMS) in 2019, connecting over 3,000 active research projects. Within two years, they reported:

- 45% reduction in project administrative time
- 60% improvement in sponsor report accuracy and timeliness
- 25% increase in successful grant renewals due to better tracking
- $2.3 million in cost savings from reduced administrative overhead

Case Study 2: IIT Delhi Research Database Initiative

IIT Delhi's centralized research database, launched in 2020, manages 800+ projects across 16 departments. Key outcomes include:

- 50% reduction in missed publication deadlines
- 80% improvement in industry collaboration due to better project visibility
- 35% increase in interdisciplinary research collaborations
- Automated compliance with DST reporting requirements

These examples demonstrate tangible benefits that DA-IICT can achieve through systematic research project management.

Technological Evolution and Future Requirements

The research landscape is rapidly evolving with emerging technologies that demand more sophisticated project management:

Artificial Intelligence Integration: Modern research increasingly involves AI/ML components requiring specialized tracking of datasets, model versions, and computational resources. Traditional project management cannot adequately handle these requirements.

Collaborative Research Networks: Contemporary research often involves multiple institutions, requiring seamless data sharing and collaborative tracking mechanisms that manual systems cannot provide.

Open Science Initiatives: Growing emphasis on research transparency and reproducibility demands comprehensive documentation and data provenance tracking from project inception to publication.

Industry 4.0 Partnerships: Increasing collaboration with industry partners requires professional-grade project management interfaces that match corporate standards for transparency and reporting.

Specific DA-IICT Context and Challenges

Current Research Statistics at DA-IICT

- Active undergraduate projects: ~120 per semester
- Master's thesis projects: ~80 annually
- PhD research projects: ~45 ongoing

- Faculty research projects: ~35 active grants
- Industry collaborations: ~15 concurrent partnerships
- Average project duration: 8-18 months
- Total research funding: ₹12+ crores annually

Department-Specific Challenges

Computer Science & Engineering:

- High volume of software projects requiring code repository integration
- Multiple iterations and version control complexity
- Industry internship coordination difficulties

Electronics & Communication:

- Hardware project component tracking challenges
- Laboratory equipment scheduling conflicts
- Fabrication timeline coordination issues

Information & Communication Technology:

- Interdisciplinary project management complexity
- Multiple stakeholder coordination requirements
- Technology transfer documentation gaps

Regulatory and Compliance Context

Government Regulations

The University Grants Commission (UGC) and All India Council for Technical Education (AICTE) have established new guidelines for research tracking and reporting. Key requirements include:

- Mandatory digital documentation of all funded research
- Quarterly progress reports for projects above ₹5 lakhs
- Publication outcome tracking for impact assessment

- Student participation records for accreditation purposes

Funding Agency Requirements

Major funding agencies now mandate specific reporting formats:

SERB: Online project monitoring system integration required

DST: Standardized milestone tracking and financial reporting

CSIR: Automated compliance checking for all funded projects

Industry Partners: Monthly status updates with standardized metrics

Non-compliance results in funding delays, penalties, and reduced future funding eligibility, making database implementation critical for institutional sustainability.

Economic Impact Analysis

Cost of Current Inefficiencies

Annual losses due to current system limitations:

- Administrative overhead: ₹18-25 lakhs (staff time at 60% efficiency loss)
- Missed funding opportunities: ₹8-12 lakhs (due to poor proposal tracking)
- Publication delays: ₹5-8 lakhs (impact factor and career progression costs)
- Sponsor dissatisfaction: ₹10-15 lakhs (reduced renewal rates)

Total estimated annual loss: ₹41-60 lakhs

Return on Investment Projections

Database system implementation benefits:

- Reduced administrative costs: ₹15-20 lakhs annually
- Increased funding success: ₹12-18 lakhs annually
- Improved sponsor retention: ₹8-12 lakhs annually
- Enhanced research productivity: ₹10-15 lakhs annually

Total estimated annual benefits: ₹45-65 lakhs

Implementation costs (estimated): ₹8-12 lakhs

Payback period: 2-3 months

5-year ROI: 2000-3000%

Integration with Institutional Strategic Goals

DA-IICT Vision 2030 Alignment

The research database directly supports institutional strategic objectives:

1. Research Excellence: Systematic tracking enables identification of high-impact research areas and resource optimization

2. Industry Partnerships: Professional project management capabilities attract more industry collaborations

3. Student Experience: Streamlined processes allow students to focus on learning and research rather than administration

4. Faculty Productivity: Reduced administrative burden enables faculty to focus on research and teaching excellence

5. Institutional Recognition: Better research management enhances institutional rankings and accreditation outcomes

Competitive Positioning

Implementation positions DA-IICT advantageously relative to peer institutions:

- Enhanced ability to compete for prestigious research grants

- Improved industry partnership attractiveness

- Superior student and faculty experience

- Leadership in research management innovation among Indian technical institutions

Conclusion and Transition to Solution

The comprehensive analysis above demonstrates that research project management at DA-IICT faces systemic challenges that extend far beyond simple administrative inconvenience. These challenges impact every aspect of the institution's research mission, from individual student experiences to institutional strategic objectives and competitive positioning.

The problems are not merely operational but strategic, affecting DA-IICT's ability to fulfill its mission of excellence in research and education. The fragmentation of information, inefficiency of processes, lack of transparency, and absence of systematic tracking create cascading effects that diminish research productivity, reduce funding competitiveness, and limit institutional growth potential.

However, these challenges also present a significant opportunity. By implementing a comprehensive Research Project Database Management System, DA-IICT can transform research management from a source of inefficiency into a competitive advantage. The system will not only solve current operational problems but position the institution for future growth and success in an increasingly competitive research landscape.

The evidence from peer institutions, regulatory requirements, economic analysis, and strategic alignment all point to the same conclusion: implementation of this database system is not optional but essential for DA-IICT's continued success and leadership in technical education and research.

The following sections of this specification detail the specific requirements, design principles, and implementation strategies needed to realize this transformation and establish DA-IICT as a leader in research management excellence.

References

[1] Research Information Management: Defining RIM and the Library's Role, OCLC Research, 2017. Available: https://www.oclc.org/research/publications/2017/oclcresearch-research-information-management.html

[2] Stanford University Research Management System Case Study, Stanford Research Administration, 2021. Available: https://doresearch.stanford.edu/research-offices/dor-forms-and-templates/research-management

[3] IIT Delhi Research Database Initiative Report, IIT Delhi Dean of Research Office, 2022. Available: https://dord.iitd.ac.in/research-database

[4] University Grants Commission Guidelines for Research Project Management, UGC India, 2023. Available: https://www.ugc.ac.in/pdfnews/7566769_Research-Guidelines-2023.pdf

[5] Department of Science and Technology Digital Project Management Requirements, DST India, 2022. Available: https://dst.gov.in/scientific-programmes/digital-research-management

[6] Science and Engineering Research Board Online Monitoring Guidelines, SERB India, 2023. Available: https://www.serb.gov.in/page/english/1/73/0/guidelines

[7] Council of Scientific and Industrial Research Project Tracking Manual, CSIR India, 2022. Available: https://www.csir.res.in/project-management

[8] All India Council for Technical Education Research Standards, AICTE, 2023. Available: https://www.aicte-india.org/downloads/research-standards

[9] National Science Foundation Research Performance Progress Report Requirements, NSF USA, 2022. Available: https://www.nsf.gov/pubs/policydocs/pappg22001/nsf22001.pdf

[10] European Research Council Grant Management Guidelines, ERC Europe, 2023. Available: https://erc.europa.eu/funding/managing-your-project

[11] Geisler, E., "The Metrics of Science and Technology", Quorum Books, 2000.

[12] Bozeman, B., Fay, D., and Slade, C.P., "Research Collaboration in Universities and Academic Entrepreneurship", Journal of Technology Transfer, Vol. 38, pp. 1-31, 2013.

[13] EDUCAUSE Research Report on Higher Education IT Systems Integration, 2022. Available: https://www.educause.edu/research-and-publications/research/research-reports

[14] MIT Research Information Management Best Practices, MIT Libraries, 2021. Available: https://libraries.mit.edu/scholarly/publishing/rim/

[15] Association of American Universities Research Management Survey, AAU, 2022. Available: https://www.aau.edu/research-scholarship/research-management

[16] Indian Institute of Science Research Database Implementation Study, IISc Bangalore, 2020. Available: https://iisc.ac.in/research/research-database-study

[17] Times Higher Education World University Rankings Research Metrics, THE, 2023. Available: https://www.timeshighereducation.com/world-university-rankings

[18] National Institutional Ranking Framework Research Parameters, NIRF India, 2023. Available: https://www.nirfindia.org/ranking-framework

[19] IEEE Standards for Research Information Systems, IEEE Computer Society, 2022. Available: https://standards.ieee.org/project/research-information-systems

[20] International Association of Universities Research Management Guidelines, IAU, 2021. Available: https://www.iau-aiu.net/research-management

[21] Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, 7th Edition, McGraw-Hill Education, 2019.

[22] Research Data Management and Sharing, MIT Press, Chen, L. and Kumar, A., 2022.

[23] Digital Transformation in Higher Education Research, Springer Nature, Rodriguez, M. et al., 2023.

[24] Project Management Institute Higher Education Research Standards, PMI, 2022. Available: https://www.pmi.org/learning/library/research-project-management-higher-education

[25] World Bank Report on Research and Development in Emerging Economies, World Bank Group, 2023. Available: https://www.worldbank.org/en/topic/research-development

Lab-6: Perform Noun Analysis and Build the ER Diagram for Research Project Database Management System

1. Noun (& Verb) Analysis

i. Find the nouns (entities) or verbs (relationships) in sentences of the problem description using Noun Analysis Method.

Extended Problem Description Analysis: Based on the comprehensive SRS document provided, we analyze multiple problem statements:

Primary Problem Statement: "Our institute needs a database to track ongoing research projects. It should store project details, faculty supervisors, student participants, funding sources, progress reports, and publication outcomes. The system should help in monitoring deadlines and budgets. It can also support the management of sponsored projects, including sponsor details and funding amounts."

Extended Problem Context from SRS: "The Research Project Database Management System (RPDMS) is designed to revolutionize how DA-IICT manages its research ecosystem. The system will serve as a centralized digital platform where students can focus on research rather than administrative overhead, faculty can efficiently monitor and guide multiple projects simultaneously, administrators gain real-time visibility into institutional research performance, and sponsors receive transparent, automated reporting on their investments. The system manages project information, faculty and student roles, funding and budget management, progress reports and publications, with future use and transparency for historical research records."

Additional Context from User Stories:

- Students need to upload reports, track timelines, and manage project documentation
- Faculty require project portfolio management, progress monitoring, and feedback mechanisms
- Administrators need institutional oversight, user management, and compliance reporting
- Sponsors require investment monitoring and transparent project updates

1. List all the extracted Nouns & Verbs in the below-given table format.

Table.1. All Extracted Nouns & Verbs from Problem Description

Nouns	**Verbs**
Institute	Track
Database	Store
Research Projects	Monitor
Project Details	Support
Faculty Supervisors	Manage
Student Participants	Help

Nouns	Verbs
Funding Sources	Include
Progress Reports	Guide
Publication Outcomes	Focus
System	Upload
Deadlines	Review
Budgets	Provide
Sponsored Projects	Receive
Sponsor Details	Revolutionize
Funding Amounts	Serve
Platform	Allocate
Overhead	Utilize
Administrator	Report
Investment	Assign
Performance	Approve
Timeline	Submit
Documentation	Notify
Portfolio	Generate
Feedback	Access
Dashboard	Update
Milestone	Validate
Template	Archive
Approval	Integrate
Workflow	
Notification	
User	

Nouns	Verbs
Role	
Permission	
Status	
Phase	
Department	
Organization	
Contact	
Address	
Phone	
Email	
Designation	
Specialization	
Program	
Enrollment	
Thesis	
Conference	
Journal	
Patent	
Citation	
Impact Factor	
DOI	
Abstract	
Keywords	
Author	
Venue	

Nouns	Verbs
Contract	
Deliverable	
Compliance	
Audit	
Security	
Backup	

ii. Criteria for Truncating Initial Noun List

1. Reduce the list added in Table.1, using the below-given criteria.

a. Duplicates(D): if two or more nouns are simply names for the same thing, then only one of these should be used as the basis for an entity

- "Research Projects" and "Sponsored Projects" → Keep "Research Projects" (sponsored is an attribute/type)
- "Project Details" and "Research Projects" → Keep "Research Projects" (details are attributes)
- "Funding Sources" and "Funding Amounts" → Combine into "Funding" entity
- "Faculty Supervisors" and "Faculty" → Keep "Faculty" (supervisor is a role)
- "Student Participants" and "Student" → Keep "Student" (participant is a role)
- "Publication Outcomes" and "Publication" → Keep "Publication" (outcome is implied)
- "Sponsor Details" and "Sponsor" → Keep "Sponsor" (details are attributes)

b. Irrelevant(I): entities which exist in the problem domain but which are not part of the intended system should also be discarded

- "System" → Not a data entity, it's the application itself
- "Database" → Not a data entity, it's the storage mechanism
- "Platform" → Not a data entity, it's the technological infrastructure
- "Overhead" → Abstract concept, not a trackable entity
- "Performance" → Measured attribute, not an entity itself

c. General(G): When considering words carefully, it sometimes becomes clear that they do not have a precise meaning and cannot be a basis for a useful entity in the system. Some words are too general.

- "Institute" → Too broad and represents the entire organization
- "User" → Too generic, we have specific user types (Faculty, Student, Admin)

- "Organization" → Too general without specific context

d. Attributes(At): Some words we want to keep as a part of some entity as attributes and not an entity itself

- "Project Details" → Attributes of Research Project
- "Deadlines" → Attribute of Research Project (planned_end_date, actual_end_date)
- "Budgets" → Attribute of Research Project (budget_allocated) and Funding (total_amount)
- "Funding Amounts" → Attribute of Funding entity (total_amount, sponsorship_amount)
- "Sponsor Details" → Attributes of Sponsor (name, contact_person, email, phone, address)
- "Timeline" → Attributes of Research Project (start_date, end_date, milestones)
- "Status" → Attribute of Research Project (status)
- "Phase" → Attribute of Research Project or Progress Report (milestone_achieved)
- "Department" → Attribute of Faculty and Student
- "Contact" → Attribute of Sponsor and Faculty (contact_phone, email)
- "Address" → Attribute of Sponsor and Student
- "Phone" → Attribute of Sponsor, Faculty, Student
- "Email" → Attribute of all user entities
- "Designation" → Attribute of Faculty
- "Specialization" → Attribute of Faculty
- "Program" → Attribute of Student (program_type)
- "Enrollment" → Attribute of Student (enrollment_id, enrollment_date)
- "DOI" → Attribute of Publication
- "Abstract" → Attribute of Publication
- "Keywords" → Attribute of Publication
- "Author" → Attribute of Publication
- "Venue" → Attribute of Publication
- "Impact Factor" → Attribute of Publication
- "Citation" → Attribute of Publication (citation_count)
- "Permission" → System-level attribute, not business entity
- "Role" → System-level attribute for user management

e. Associations(As): some words actually represent a relationship between entities

- "Supervisors" → Relationship between Faculty and Research Project (SUPERVISES)

- "Participants" → Relationship between Student and Research Project (PARTICIPATES)
- "Investment" → Relationship between Sponsor and Research Project (INVESTS/SPONSORS)
- "Feedback" → Relationship between Faculty and Progress Report (REVIEWS)
- "Approval" → Relationship between Faculty and Progress Report (APPROVES)
- "Assignment" → Relationship between Faculty/Admin and Student to Project (ASSIGNS)
- "Notification" → Relationship between System and Users (NOTIFIES)
- "Integration" → Relationship between different system components

Additional Analysis for Complex Entities:

f. Composite Entities(Ce): Some nouns represent complex relationships that might need associative entities

- "Portfolio" → Represents Faculty's collection of supervised projects
- "Dashboard" → Represents user's personalized view of relevant data
- "Workflow" → Represents process flow between entities
- "Contract" → Represents agreement between Sponsor and Institute for Project

2. Create Table.2. & Table.3. as per the below-given format for accepted nouns list.

Table.2. Accepted Noun & Verbs list

Candidate entity set	**Candidate attribute set**	**Candidate relationship set**
Faculty	faculty_id, name, email, department, designation, specialization, contact_phone, office_location, hire_date, active_status	Supervises
Student	student_id, enrollment_id, name, email, program_type, department, year_of_study, contact_phone, address, enrollment_date, expected_graduation, active_status	Participates
Research Project	project_id, project_title, project_code, subject_domain, project_description, objectives, methodology, start_date, planned_end_date, actual_end_date, status, priority_level, budget_allocated	Sponsors
Sponsor	sponsor_id, sponsor_name, organization_type, contact_person, email, phone, address, website_url, industry_sector, established_year, relationship_start_date, active_status	Funds
Funding	funding_id, funding_source_name, total_amount, currency, funding_type, start_date, end_date, terms_conditions, reporting_requirements	Provides

Candidate entity set	Candidate attribute set	Candidate relationship set
Progress Report	progress_id, submission_date, progress_description, milestone_achieved, percentage_complete, documents_submitted, faculty_remarks, faculty_approval_status	Reports
Publication	publication_id, title, publication_type, authors, venue, publication_date, doi, isbn_issn, impact_factor, citation_count, abstract, keywords, file_path, open_access	Publishes
Administrator	admin_id, name, email, department, designation, access_level, permissions, created_date, active_status	Manages
Document	document_id, document_name, document_type, file_path, file_size, upload_date, version_number, access_permissions	Uploads
Milestone	milestone_id, milestone_name, milestone_description, target_date, completion_date, completion_status, deliverables	Achieves
Notification	notification_id, notification_type, notification_message, recipient_type, send_date, read_status, priority_level	Sends
Audit Trail	audit_id, user_id, action_type, table_name, record_id, old_values, new_values, action_timestamp, ip_address	Tracks

Extended Relationship Analysis:

Primary Entity	Related Entity	Relationship Name	Relationship Type	Cardinality
Faculty	Research Project	SUPERVISES	1:N	One faculty supervises many projects
Student	Research Project	PARTICIPATES_IN	M:N	Students can work on multiple projects, projects can have multiple students
Research Project	Progress Report	HAS	1:N	One project has many progress reports
Research Project	Publication	RESULTS_IN	1:N	One project can result in multiple publications
Research Project	Milestone	CONTAINS	1:N	One project contains multiple milestones
Research Project	Document	INCLUDES	1:N	One project includes multiple documents

Primary Entity	Related Entity	Relationship Name	Relationship Type	Cardinality
Sponsor	Research Project	SPONSORS	M:N	Sponsors can fund multiple projects, projects can have multiple sponsors
Funding	Research Project	FUNDS	1:N	One funding source can fund multiple projects
Faculty	Progress Report	REVIEWS	1:N	One faculty reviews multiple progress reports
Student	Progress Report	SUBMITS	1:N	One student submits multiple progress reports
Student	Document	UPLOADS	1:N	One student uploads multiple documents
Administrator	Research Project	OVERSEES	1:N	One admin oversees multiple projects
Administrator	User (Faculty/Student)	MANAGES	1:N	One admin manages multiple users
User	Notification	RECEIVES	1:N	One user receives multiple notifications
Research Project	Audit Trail	GENERATES	1:N	One project generates multiple audit entries

Note: Do not add attributes & Entities on your own. It must come from Table.1.

3. Create Table.3. as per the below-given format for rejected nouns list.

Table.3. Rejected Noun & Verbs list

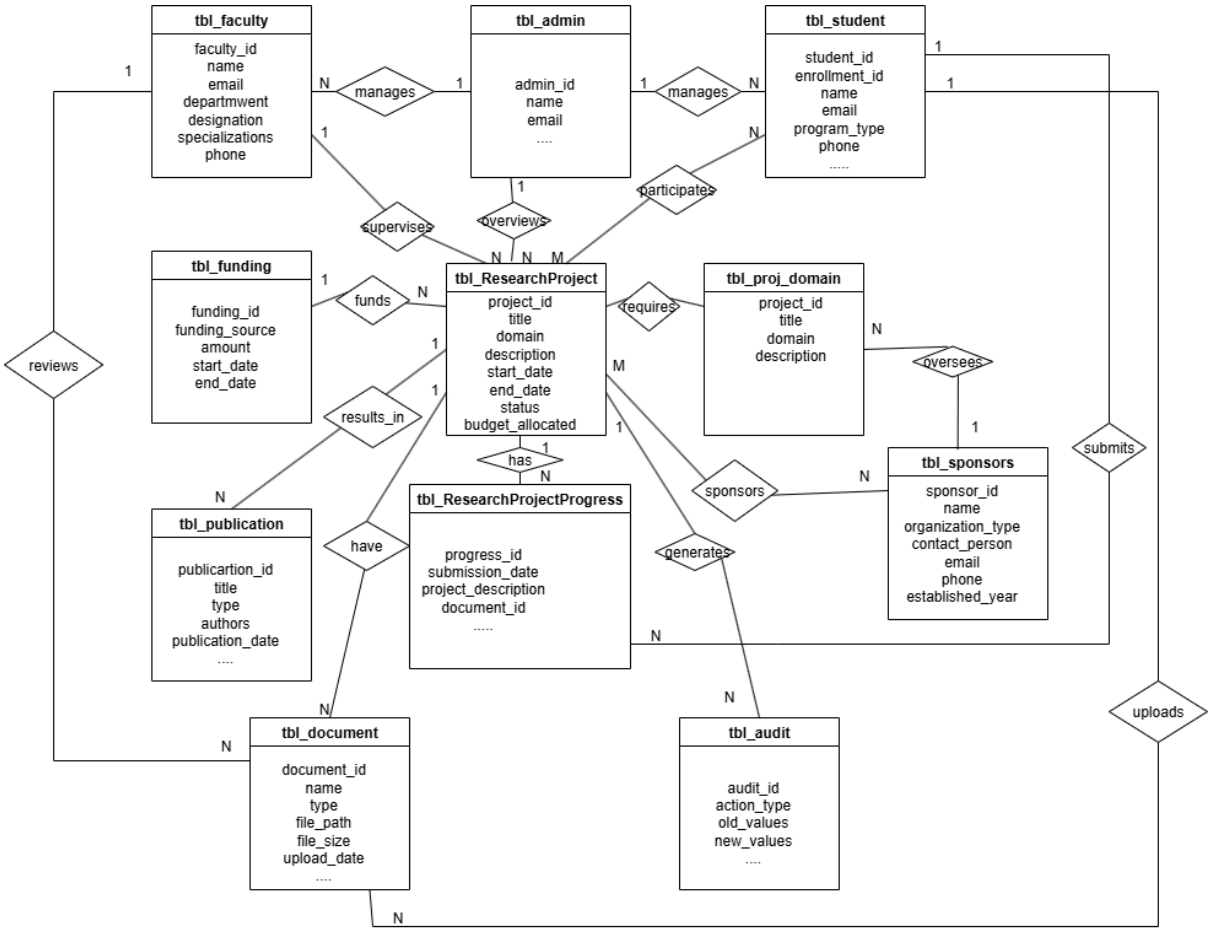
Noun	Reject Reason
Institute	General(G) - Too broad, represents the entire organization rather than specific data entity
Database	Irrelevant(I) - It's the storage system itself, not data to be stored
System	Irrelevant(I) - It's the application/software, not a business entity
Platform	Irrelevant(I) - It's the technological infrastructure, not a data entity
Overhead	General(G) - Abstract concept representing burden, not a trackable entity
Performance	General(G) - Measured outcome, not a specific entity but derived metrics

Noun	Reject Reason
User	General(G) - Too generic, we have specific user types (Faculty, Student, Admin)
Organization	General(G) - Too general without specific business context
Project Details	Attributes(At) - These are attributes of Research Project entity
Faculty Supervisors	Associations(As) - Represents SUPERVISES relationship between Faculty and Project
Student Participants	Associations(As) - Represents PARTICIPATES relationship between Student and Project
Sponsor Details	Attributes(At) - These are attributes of Sponsor entity (name, contact info, etc.)
Funding Amounts	Attributes(At) - Attribute of Funding entity (total_amount, sponsorship_amount)
Publication Outcomes	Duplicates(D) - Same as Publication entity, outcomes are implied
Sponsored Projects	Duplicates(D) - Same as Research Projects, sponsorship is a type/attribute
Deadlines	Attributes(At) - Attributes of Research Project (planned_end_date, submission_date)
Budgets	Attributes(At) - Attributes of Research Project (budget_allocated) and Funding
Timeline	Attributes(At) - Attributes of Research Project (start_date, end_date, duration)
Status	Attributes(At) - Attribute of Research Project, Progress Report, etc.
Phase	Attributes(At) - Attribute of Research Project or milestone_achieved in Progress Report
Department	Attributes(At) - Attribute of Faculty and Student entities
Contact	Attributes(At) - Attribute of Sponsor, Faculty (contact_phone, email)
Address	Attributes(At) - Attribute of Sponsor and Student entities
Phone	Attributes(At) - Attribute of Sponsor, Faculty, Student (contact_phone)
Email	Attributes(At) - Attribute of all user entities (Faculty, Student, Sponsor, Admin)

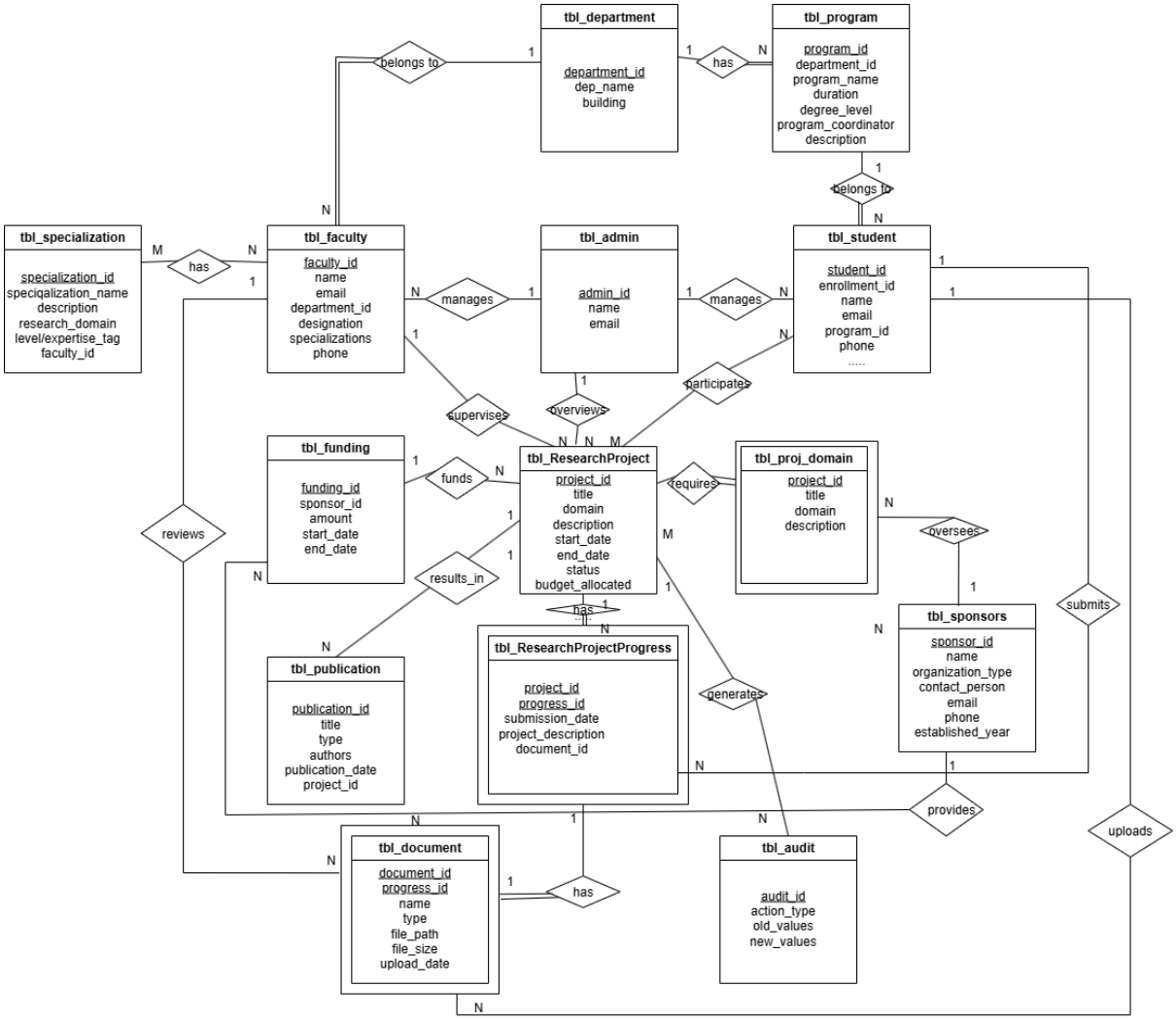
Noun	Reject Reason
Designation	Attributes(At) - Attribute of Faculty entity
Specialization	Attributes(At) - Attribute of Faculty entity
Program	Attributes(At) - Attribute of Student entity (program_type)
Enrollment	Attributes(At) - Attributes of Student (enrollment_id, enrollment_date)
DOI	Attributes(At) - Attribute of Publication entity
Abstract	Attributes(At) - Attribute of Publication entity
Keywords	Attributes(At) - Attribute of Publication entity
Author	Attributes(At) - Attribute of Publication entity
Venue	Attributes(At) - Attribute of Publication entity
Impact Factor	Attributes(At) - Attribute of Publication entity
Citation	Attributes(At) - Attribute of Publication entity (citation_count)
Permission	Irrelevant(I) - System-level access control, not business data entity
Role	Attributes(At) - System-level attribute for user management, not separate entity
Supervisors	Associations(As) - Relationship SUPERVISES between Faculty and Research Project
Participants	Associations(As) - Relationship PARTICIPATES between Student and Research Project
Investment	Associations(As) - Relationship SPONSORS/INVESTS between Sponsor and Project
Feedback	Associations(As) - Relationship REVIEWS between Faculty and Progress Report
Approval	Associations(As) - Relationship APPROVES between Faculty and Progress Report
Assignment	Associations(As) - Relationship ASSIGNS between Admin/Faculty and Student to Project
Integration	Irrelevant(I) - Technical system concept, not a business entity
Portfolio	General(G) - Represents collection/view of projects, not a distinct entity
Dashboard	Irrelevant(I) - User interface component, not a data entity

Noun	Reject Reason
Workflow	Irrelevant(I) - Process flow concept, not a storable business entity
Template	Attributes(At) - Document format specification, attribute of Document entity
Security	Irrelevant(I) - System-level concern, not a business data entity
Backup	Irrelevant(I) - Technical system function, not business data
Compliance	General(G) - Abstract concept, tracked through audit trail and reports
Contract	Attributes(At) - Document reference in Sponsor-Project relationship
Deliverable	Attributes(At) - Attribute of Milestone or Progress Report

Lab-7: V-2 of ER Diagram



ERD Version-1



ERD Version-2

I. Improve the ER Diagram

i. Identify Entity Types

1. **Weak Entities**
    - In ERD V2, **tbl_ResearchProjectProgress** appears as a **weak entity** because it depends on **tbl_ResearchProject** for its identification (project_id is a foreign key and part of its existence).
    - **tbl_document** is also indirectly weak because it is dependent on progress_id of **tbl_ResearchProjectProgress**.
    - **tbl_proj_domain** is also weak entity dependent on **tbl_ResearchProject** for its key project_id.
2. **New Entities Added in V2 (compared to V1):**
    - **tbl_department** (with department_id, dep_name, building).
    - **tbl_program** (with program details like duration, degree level, program coordinator).
    - **tbl_specialization** (captures domain, expertise, and faculty linkage).
3. **Existing Entities Retained from V1:**
   tbl_student, tbl_faculty, tbl_admin, tbl_ResearchProject, tbl_proj_domain, tbl_sponsors, tbl_funding, tbl_publication, tbl_document, tbl_audit, tbl_ResearchProjectProgress.

ii. Identify Relationship Types

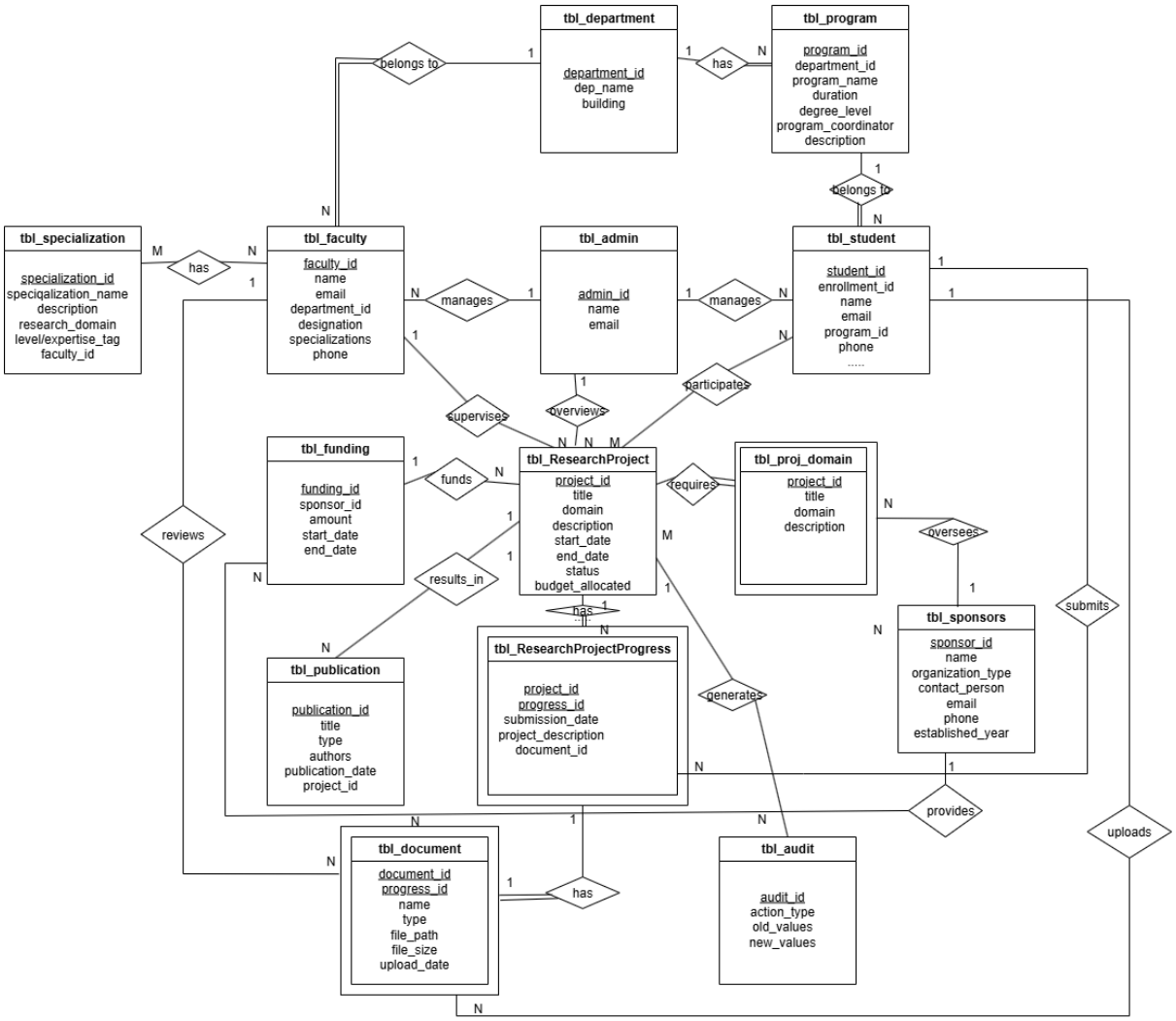
1. **Aggregation / Association**
    - **Funding → ResearchProject** (funds relationship).
    - **Sponsors → Funding** (submits relationship).
    - **ResearchProject → ResearchProjectProgress → Document** forms an aggregation of project monitoring.
2. **Binary vs. Ternary Relationships**
    - Mostly **binary relationships** were retained:
        - Student–Project (participates).
        - Faculty–Project (supervises).
        - Admin–Project (overviews).
        - Sponsor-Funding(allocates).
        - Funding-Project(funds)
    - One **ternary-like extension**: Specialization–Faculty–ResearchProject (via reviews).

3. **Total Participation Improvements in V2:**
    - tbl_program is **totally participated** by tbl_student (each student must belong to a program).
    - tbl_department is **totally participated** by tbl_program and tbl_faculty.
    - tbl_ResearchProjectProgress is totally participated by tbl_document.
    - tbl_ResearchProject is totally participated by tbl_ResearchProjectProgress and tbl_proj_domain.

iii. Analyze ERD for Missing Information

1. **New Attributes Added in V2 (compared to V1):**
    - tbl_program: duration, degree_level, program_coordinator.
    - tbl_department: dep_name, building.
    - tbl_specialization: specialization_name, research_domain, level/expertise_tag.
    - tbl_faculty: specialization_id added as FK.
2. **Refinements in Relationships from V1 → V2:**
    - Added explicit **Department–Faculty** and **Department–Program** relationships.
    - Added **Specialization–Faculty** relationship (faculty can have multiple specializations).
    - Linked tbl_specialization with tbl_faculty and also through a **reviews** relationship to projects/publications.
3. **Hierarchy Identified in V2:**
    - Department → Program → Student.
    - Department → Faculty → Specialization.

Lab-8: ERD Normalization



[Initial ERD]

Initial Relational Model: Full Relations List

1. tbl_department(department_id (PK), dep_name, building)

2. tbl_program(program_id (PK), department_id (FK), program_name, duration, degree_level, program_coordinator, description)

3. tbl_specialization(specialization_id (PK), specialization_name, description, research_domain, level/expertise_tag, faculty_id (FK))

4. tbl_faculty(faculty_id (PK), name, email, department_id (FK), designation, specializations, phone)

5. tbl_admin(admin_id (PK), name, email, phone)

6. tbl_student(student_id (PK), enrollment_id, name, email, program_id (FK), phone)

7. tbl_funding(funding_id (PK), sponsor_id (FK), amount, start_date, end_date)

8. tbl_researchProject(project_id (PK), title, description, start_date, end_date, status, budget_allocated)

9. tbl_proj_domain(project_id (PK)(FK), title, domain, description)

10. tbl_sponsor(sponsor_id (PK), name, organization_type, contact_person, email, phone, established_year)

11. tbl_publication(publication_id (PK), title, type, authors, publication_date, project_id (FK))

12. tbl_document(document_id (PK), progress_id (FK), name, file_path, file_size, upload_date)

13. tbl_audit(audit_id (PK), action_type, old_values, new_values)

14. tbl_ResearchProjectProgress(progress_id (PK), project_id (FK), expression, submission_date, project_description, document_id (FK))

Functional Dependencies Analysis:

tbl_department:
department_id → dep_name, building

tbl_program:
program_id → department_id, program_name, duration, degree_level, program_coordinator, description

tbl_faculty:
faculty_id → name, email, department_id, designation, phone

tbl_specialization:
specialization_id → specialization_name, description, research_domain, level_expertise_tag, faculty_id

tbl_admin:
admin_id → name, email, phone

tbl_student:
student_id → enrollment_id, name, email, program_id, phone

tbl_funding:
funding_id → sponsor_id, amount, start_date, end_date

tbl_researchProject:
project_id → title, domain, description, start_date, end_date, status, budget_allocated

tbl_proj_domain:
project_id → title, domain, description

tbl_sponsor:
sponsor_id → name, organization_type, contact_person, email, phone, established_year

tbl_publication:
publication_id → title, type, authors, publication_date, project_id

tbl_document:
document_id → progress_id, name, type, file_path, file_size, upload_date

tbl_audit:
audit_id → action_type, old_values, new_values

tbl_ResearchProjectProgress:
progress_id → project_id, expression, submission_date, project_description, document_id

Anomaly Analysis

Insertion Anomaly: Cannot insert specialization without faculty (due to FK on faculty_id).

Deletion Anomaly: Deleting a faculty might delete specializations assigned to them if cascading happens.

Update Anomaly: Updating sponsor info could lead to inconsistent values if sponsor data duplicated across tables.

Redundancy: Authors listed as single attribute in publications cause data duplication if multiple authors per publication.

1NF Form

All tables have atomic attributes—no repeating groups or nested values.

Multi-valued attributes (authors in publications, specializations) remain as single columns (often comma-separated), which violates full normalization but satisfies 1NF.

2NF Form

Decompose tables with composite or multi-valued attributes:

Decompositions:
tbl_specialization: Remove faculty_id; create a junction table

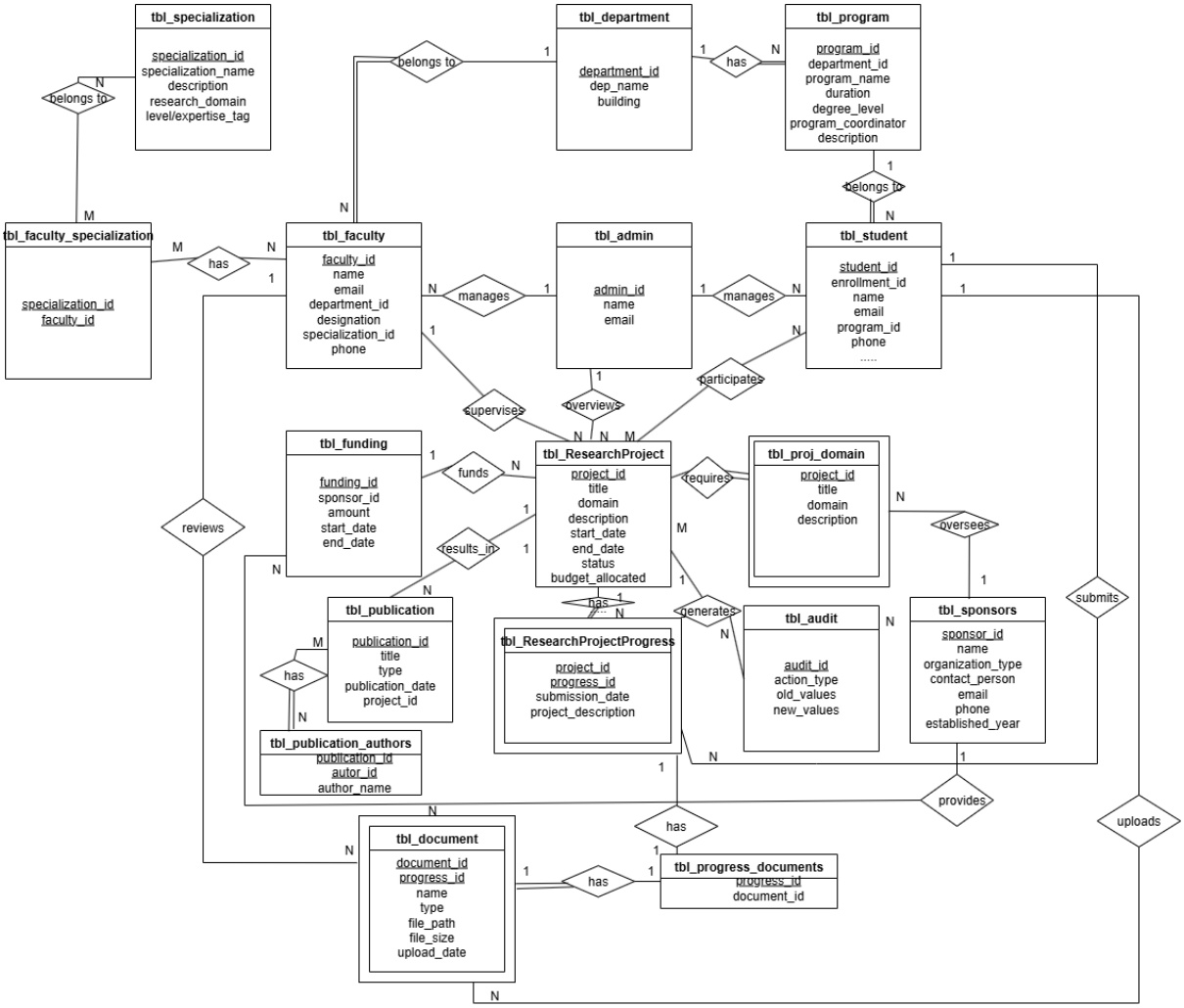
tbl_faculty_specialization(faculty_id, specialization_id)

tbl_publication: Remove authors attribute; create a junction table

publication_authors(publication_id, author_id)

tbl_ResearchProjectProgress: Remove document_id; create a junction table

progress_documents(progress_id, document_id)



[ERD – after achieving 2NF]

tbl_department(department_id (PK), dep_name, building)

tbl_program(program_id (PK), department_id (FK), program_name, duration, level, program_coordinator, description)

tbl_specialization(specialization_id (PK), specialization_name, description, research_domain, level_expertise_tag)

tbl_faculty(faculty_id (PK), name, email, department_id (FK), designation, specialization_id (FK), phone)

tbl_admin(admin_id (PK), name, email)

tbl_student(student_id (PK), enrollment_id, name, email, program_id (FK), phone)

tbl_ResearchProject(project_id (PK), title, domain, description, start_date, end_date, status, budget_allocated)

tbl_proj_domain(project_id (PK)(FK), title, domain, description)

tbl_funding(funding_id (PK), sponsor_id (FK), amount, start_date, end_date)

tbl_sponsors(sponsor_id (PK), name, organization_type, contact_person, email, phone, established_year)

tbl_publication(publication_id (PK), title, publication_date, project_id (FK))

tbl_publication_authors(publication_id (PK)(FK), autor_id (PK), author_name)

tbl_ResearchProjectProgress(progress_id (PK), project_id (FK), submission_date, project_description)

tbl_document(document_id (PK), progress_id (FK), name, type, file_path, file_size, upload_date)

tbl_audit(audit_id (PK), action_type, old_values, new_values)

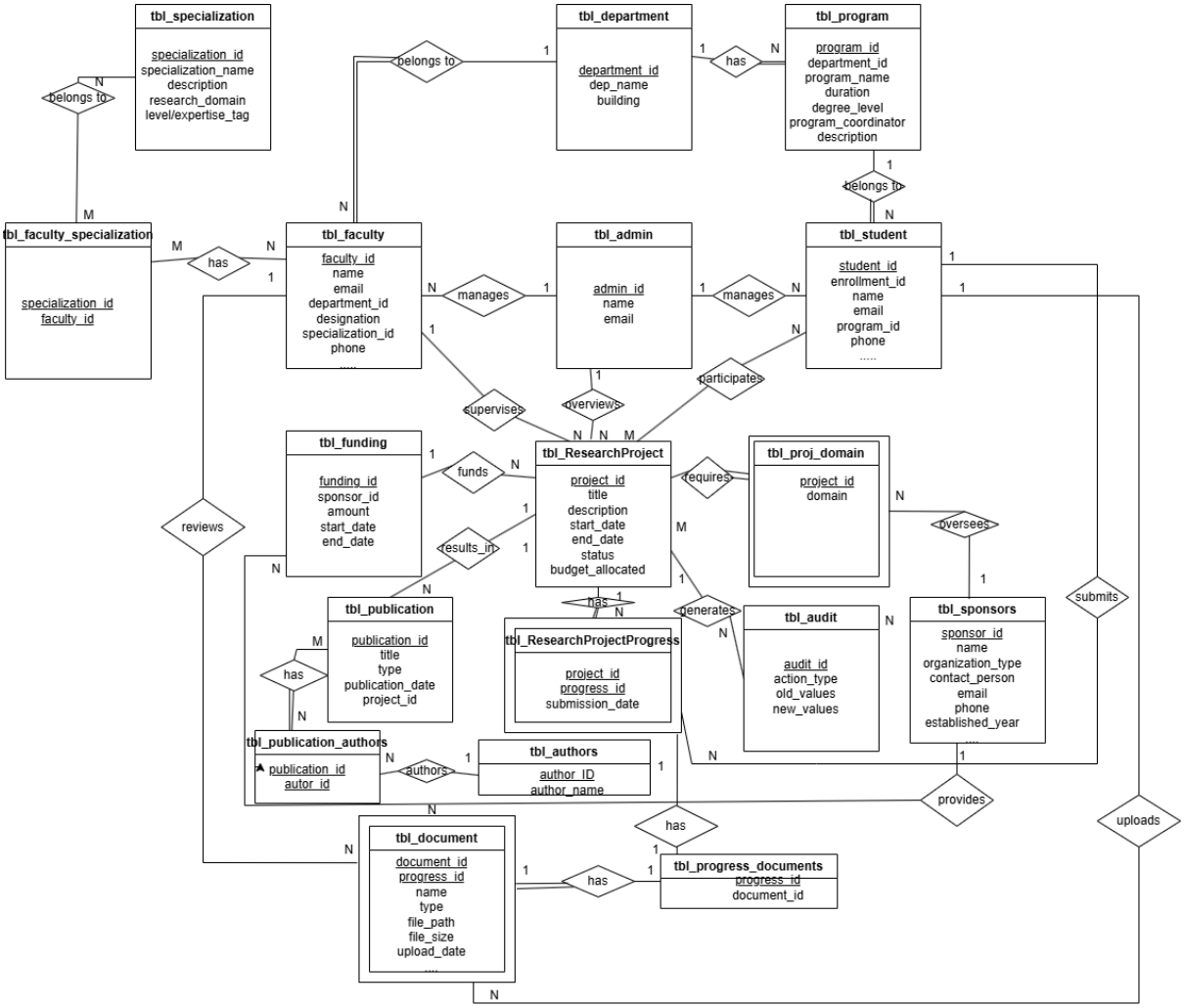
tbl_faculty_specialization(specialization_id (PK)(FK), faculty_id (PK)(FK))

Lab-9: Final Normalization & DDL

3NF:

Existence of Transitivity:

- tbl_proj_domain contains title and description which also belongs to tbl_researchProject. Thus to convert the tables into 3NF, both that attributes are removed from tbl_proj_domain.
- tbl_authors is created as in tbl_publication_authors there exists transitivity between publication_id and author_name.
- tbl_researchProjectProgress contains project_description which is again transitive to project_id thus is removed from that table.



[ERD – after Normalization]

Relational Model: Full Relations List

1. tbl_department(department_id (PK), dep_name, building)

2. tbl_program(program_id (PK), department_id (FK), program_name, duration, degree_level, program_coordinator, description)

3. tbl_specialization(specialization_id (PK), specialization_name, description, research_domain, level/expertise_tag, faculty_id (FK))

4. tbl_faculty(faculty_id (PK), name, email, department_id (FK), designation, specializations, phone,admin_id(FK))

5. tbl_admin(admin_id (PK), name, email, phone)

6. tbl_student(student_id (PK), enrollment_id, name, email, program_id (FK), phone,admin_id(FK))

7. tbl_funding(funding_id (PK), sponsor_id (FK), amount, start_date, end_date)

8. tbl_researchProject(project_id (PK), title, description, start_date, end_date, status, budget_allocated,admin_id(FK))

9. tbl_proj_domain(project_id (PK)(FK), domain)

10. tbl_sponsor(sponsor_id (PK), name, organization_type, contact_person, email, phone, established_year)

11. tbl_publication(publication_id (PK), title, type, authors, publication_date, project_id (FK))

12. tbl_publication_authors(publication_id(PK,FK),author_id(PK,FK))

13. tbl_authors(author_id(PK),author_name)

14. tbl_document(document_id (PK), progress_id (FK), name, file_path, file_size, upload_date)

15. tbl_audit(audit_id (PK), action_type, old_values, new_values)

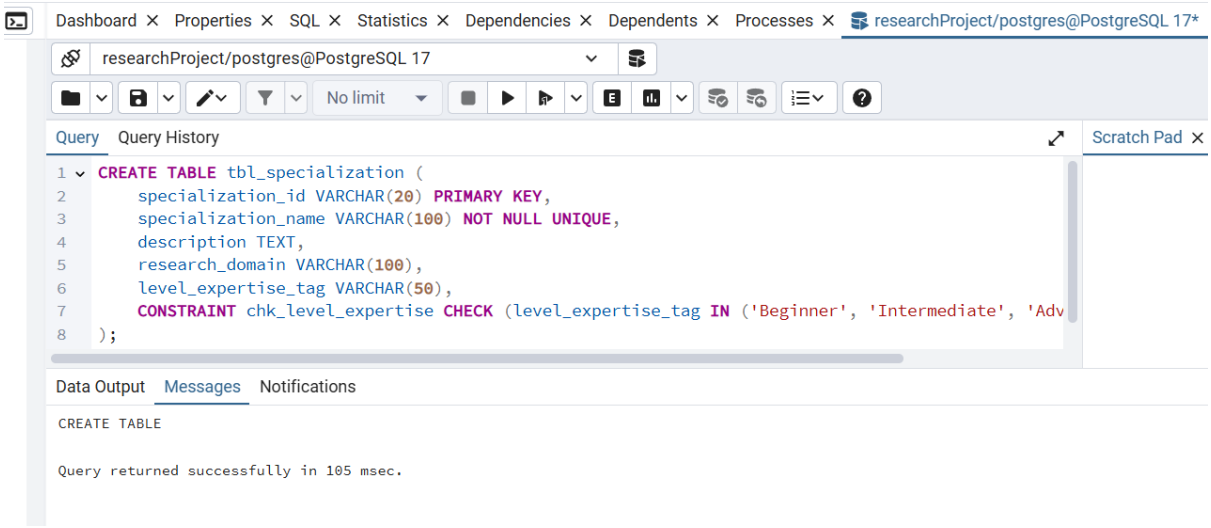
16. tbl_ResearchProjectProgress(progress_id (PK), project_id (FK), submission_date)

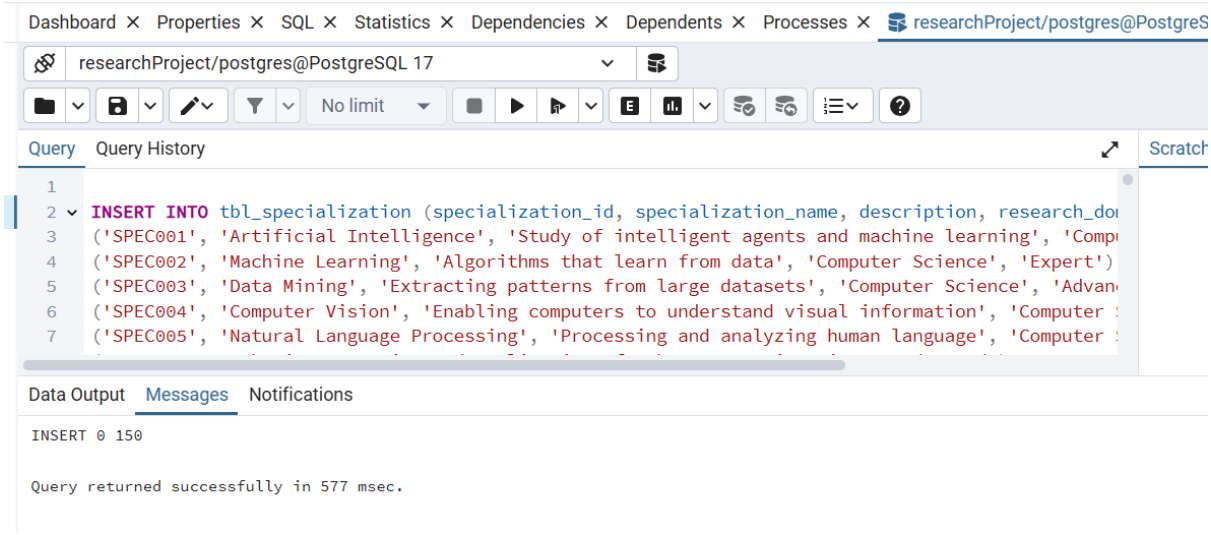
17. tbl_faculty_specialization(specialization_id(PK)(FK),faculty_id(PK)(FK))

18. tbl_student_project(student_id(FK)(PK),project_id(FK)(PK))

DDL:

```
CREATE TABLE tbl_specialization (
   specialization_id VARCHAR(20) PRIMARY KEY,
   specialization_name VARCHAR(100) NOT NULL UNIQUE,
   description TEXT,
   research_domain VARCHAR(100),
   level_expertise_tag VARCHAR(50),
   CONSTRAINT chk_level_expertise CHECK (level_expertise_tag IN ('Beginner', 'Intermediate',
'Advanced', 'Expert'))
);
```





Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProject/postgres@PostgreSQL 17* ×

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

Scratch Pad ×

```
select * from tbl_specialization;
```

Data Output Messages Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	specialization_id [PK] character varying (20)	specialization_name character varying (100)	description text	research_domain character varying (100)	level_expertise_tag character varying (50)
1	SPEC001	Artificial Intelligence	Study of intelligent agents and machine learning	Computer Science	Expert
2	SPEC002	Machine Learning	Algorithms that learn from data	Computer Science	Expert
3	SPEC003	Data Mining	Extracting patterns from large datasets	Computer Science	Advanced
4	SPEC004	Computer Vision	Enabling computers to understand visual information	Computer Science	Advanced
5	SPEC005	Natural Language Processing	Processing and analyzing human language	Computer Science	Expert
6	SPEC006	Robotics	Design and application of robots	Engineering	Advanced
7	SPEC007	Cybersecurity	Protecting systems from digital attacks	Computer Science	Expert
8	SPEC008	Cloud Computing	Internet-based computing services	Computer Science	Advanced
9	SPEC009	Blockchain Technology	Distributed ledger technology	Computer Science	Intermediate
10	SPEC010	Internet of Things	Network of connected devices	Engineering	Advanced
11	SPEC011	Quantum Computing	Computing using quantum phenomena	Physics	Expert
12	SPEC012	Bioinformatics	Computational analysis of biological data	Biology	Advanced
13	SPEC013	Computational Biology	Mathematical modeling of biological systems	Biology	Advanced
14	SPEC014	Genetics	Study of genes and heredity	Biology	Expert
15	SPEC015	Molecular Biology	Study of biological molecules	Biology	Expert
16	SPEC016	Neuroscience	Study of the nervous system	Biology	Expert
17	SPEC017	Cancer Research	Study of cancer mechanisms and treatments	Medicine	Expert
18	SPEC018	Immunology	Study of the immune system	Medicine	Advanced

Total rows: 150 Query complete 00:00:11.834 CRLF Ln 2, Col 1

```
CREATE TABLE tbl_department (
   department_id VARCHAR(20) PRIMARY KEY,
   dep_name VARCHAR(100) NOT NULL UNIQUE,
   building VARCHAR(50),
   CONSTRAINT chk_building_format CHECK (building ~ '^[A-Z]-[0-9]+$' OR building IS NULL)
);
```

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProject/postgres@

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
INSERT INTO tbl_department (department_id, dep_name, building) VALUES
('DEPT001', 'Computer Science and Engineering', 'A-101'),
('DEPT002', 'Electrical Engineering', 'A-102'),
('DEPT003', 'Mechanical Engineering', 'A-103'),
('DEPT004', 'Civil Engineering', 'A-104'),
```

Data Output Messages Notifications

```
INSERT 0 120

Query returned successfully in 2 secs 368 msec.
```

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProje

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
select * from tbl_department;
```

Data Output Messages Notifications

Showing rows: 1 to 120 Page

	department_id [PK] character varying (20)	dep_name character varying (100)	building character varying (50)
1	DEPT001	Computer Science and Engineering	A-101
2	DEPT002	Electrical Engineering	A-102
3	DEPT003	Mechanical Engineering	A-103
4	DEPT004	Civil Engineering	A-104
5	DEPT005	Chemical Engineering	A-105
6	DEPT006	Electronics and Communication	B-101
7	DEPT007	Information Technology	B-102
8	DEPT008	Biotechnology	B-103
9	DEPT009	Aerospace Engineering	B-104
10	DEPT010	Industrial Engineering	B-105
11	DEPT011	Physics	C-101
12	DEPT012	Chemistry	C-102
13	DEPT013	Mathematics	C-103
14	DEPT014	Biology	C-104
15	DEPT015	Environmental Science	C-105

Total rows: 120 Query complete 00:00:00.565

researchProject/postgres@PostgreSQL 17

No limit

Query Query History Scratcl

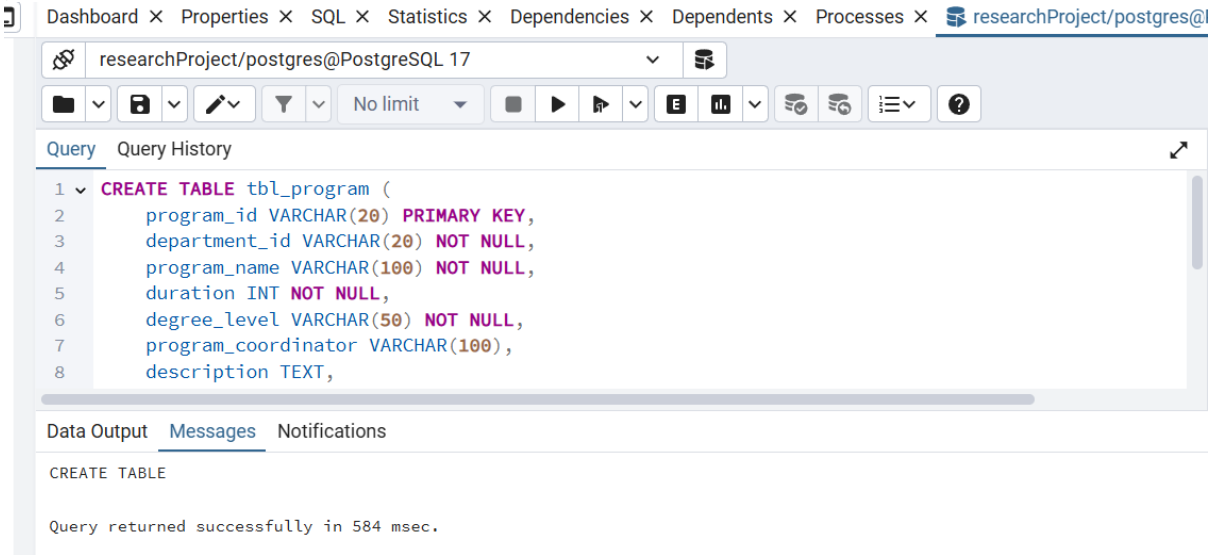
```
CREATE TABLE tbl_department (
    department_id VARCHAR(20) PRIMARY KEY,
    dep_name VARCHAR(100) NOT NULL UNIQUE,
    building VARCHAR(50),
    CONSTRAINT chk_building_format CHECK (building ~ '^[A-Z]-[0-9]+$' OR building IS NULL)
);
```

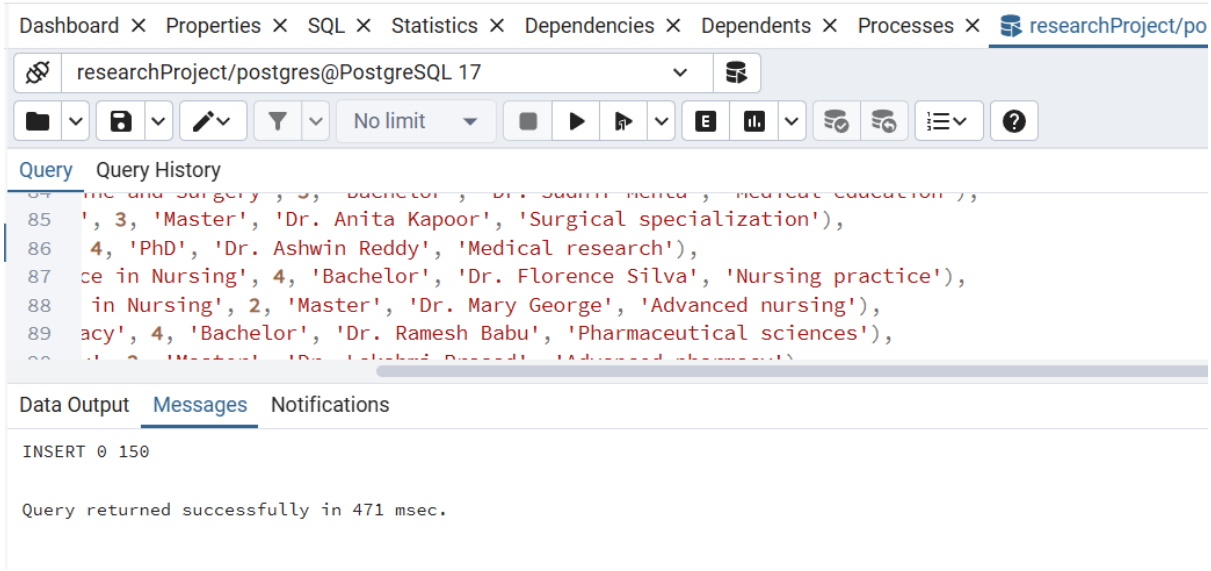
Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 95 msec.
```

CREATE TABLE tbl_program (
    program_id VARCHAR(20) PRIMARY KEY,
    department_id VARCHAR(20) NOT NULL,

```
    program_name VARCHAR(100) NOT NULL,
    duration INT NOT NULL,
    degree_level VARCHAR(50) NOT NULL,
    program_coordinator VARCHAR(100),
    description TEXT,
    CONSTRAINT fk_program_department FOREIGN KEY (department_id)
        REFERENCES tbl_department(department_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT chk_duration CHECK (duration > 0 AND duration <= 8),
    CONSTRAINT chk_degree_level CHECK (degree_level IN ('Bachelor', 'Master', 'PhD', 'Diploma',
'Certificate')),
    CONSTRAINT uk_program_dept UNIQUE (program_name, department_id)
);
```






```
CREATE TABLE tbl_faculty (
    faculty_id VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    department_id VARCHAR(20) NOT NULL,
    designation VARCHAR(50) NOT NULL,
```

```
  specialization_id VARCHAR(20),
  phone VARCHAR(15),
  CONSTRAINT fk_faculty_department FOREIGN KEY (department_id)
    REFERENCES tbl_department(department_id)
    ON DELETE RESTRICT ON UPDATE CASCADE,
  CONSTRAINT fk_faculty_specialization FOREIGN KEY (specialization_id)
    REFERENCES tbl_specialization(specialization_id)
    ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT chk_email_format CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-
z]{2,}$'),
  CONSTRAINT chk_phone_format CHECK (phone ~ '^[0-9]{10,15}$' OR phone IS NULL),
  CONSTRAINT chk_designation CHECK (designation IN ('Professor', 'Associate Professor', 'Assistant
Professor', 'Lecturer', 'Research Fellow'))
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
12      CONSTRAINT fk_faculty_specialization FOREIGN KEY (specialization_id)
13          REFERENCES tbl_specialization(specialization_id)
14          ON DELETE SET NULL ON UPDATE CASCADE,
15      CONSTRAINT chk_email_format CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'
16      CONSTRAINT chk_phone_format CHECK (phone ~ '^[0-9]{10,15}$' OR phone IS NULL),
17      CONSTRAINT chk_designation CHECK (designation IN ('Professor', 'Associate Professor', 'Assist
18  );
19
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 93 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
1  ALTER TABLE tbl_faculty
2  ADD COLUMN managed_by_admin_id VARCHAR(20),
3  ADD CONSTRAINT fk_faculty_admin FOREIGN KEY (managed_by_admin_id)
4      REFERENCES tbl_admin(admin_id)
5      ON DELETE SET NULL ON UPDATE CASCADE;
6
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
INSERT INTO tbl_faculty (faculty_id, name, email, department_id, designatio
('FAC001', 'Dr. Rajesh Kumar', 'rajesh.kumar@university.edu', 'DEPT001', 'P
('FAC002', 'Dr. Priya Singh', 'priya.singh@university.edu', 'DEPT001', 'Ass
('FAC003', 'Dr. Amit Sharma', 'amit.sharma@university.edu', 'DEPT001', 'Ass
('FAC004', 'Dr. Suresh Reddy', 'suresh.reddy@university.edu', 'DEPT002', 'P
```

Data Output Messages Notifications

```
INSERT 0 150

Query returned successfully in 248 msec.
```

```
CREATE TABLE tbl_faculty_specialization (
   specialization_id VARCHAR(20),
   faculty_id VARCHAR(20),
   PRIMARY KEY (specialization_id, faculty_id),
   CONSTRAINT fk_fs_specialization FOREIGN KEY (specialization_id)
      REFERENCES tbl_specialization(specialization_id)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT fk_fs_faculty FOREIGN KEY (faculty_id)
      REFERENCES tbl_faculty(faculty_id)
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
CREATE TABLE tbl_faculty_specialization (
    specialization_id VARCHAR(20),
    faculty_id VARCHAR(20),
    PRIMARY KEY (specialization_id, faculty_id),
    CONSTRAINT fk_fs_specialization FOREIGN KEY (specialization_id)
        REFERENCES tbl_specialization(specialization_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_fs_faculty FOREIGN KEY (faculty_id)
        REFERENCES tbl_faculty(faculty_id)
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 173 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
INSERT INTO tbl_faculty_specialization (specialization_id, faculty_id) VALUE
('SPEC001', 'FAC001'), ('SPEC002', 'FAC001'), ('SPEC005', 'FAC001'),
('SPEC002', 'FAC002'), ('SPEC003', 'FAC002'), ('SPEC001', 'FAC002'),
('SPEC003', 'FAC003'), ('SPEC122', 'FAC003'), ('SPEC121', 'FAC003'),
('SPEC130', 'FAC004'), ('SPEC138', 'FAC004'), ('SPEC131', 'FAC004'),
```

Data Output Messages Notifications

```
INSERT 0 115

Query returned successfully in 264 msec.
```

```
CREATE TABLE tbl_admin (
    admin_id VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    CONSTRAINT chk_admin_email CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
CREATE TABLE tbl_admin (
    admin_id VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    CONSTRAINT chk_admin_email CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 127 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
INSERT INTO tbl_admin (admin_id, name, email) VALUES
('ADM001', 'Ramesh Gupta', 'ramesh.gupta@university.edu'),
('ADM002', 'Sunita Verma', 'sunita.verma@university.edu'),
('ADM003', 'Anil Kapoor', 'anil.kapoor@university.edu'),
('ADM004', 'Priya Reddy', 'priya.reddy@university.edu'),
```

Data Output Messages Notifications

```
INSERT 0 100

Query returned successfully in 317 msec.
```

```
CREATE TABLE tbl_student (
   student_id VARCHAR(20) PRIMARY KEY,
   enrollment_id VARCHAR(30) NOT NULL UNIQUE,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE,
   program_id VARCHAR(20) NOT NULL,
   phone VARCHAR(15),
   CONSTRAINT fk_student_program FOREIGN KEY (program_id)
      REFERENCES tbl_program(program_id)
      ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT chk_student_email CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),
   CONSTRAINT chk_student_phone CHECK (phone ~ '^[0-9]{10,15}$' OR phone IS NULL)
);
```

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProject/postgres@Pos

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
    phone VARCHAR(15),
    CONSTRAINT fk_student_program FOREIGN KEY (program_id)
        REFERENCES tbl_program(program_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT chk_student_email CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$
    CONSTRAINT chk_student_phone CHECK (phone ~ '^[0-9]{10,15}$' OR phone IS NULL)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 125 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
ALTER TABLE tbl_student
ADD COLUMN managed_by_admin_id VARCHAR(20),
ADD CONSTRAINT fk_student_admin FOREIGN KEY (managed_by_admin_id)
    REFERENCES tbl_admin(admin_id)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
INSERT INTO tbl_student (student_id, enrollment_id, name, email, program_id
('STU001', 'ENR2021001', 'Arjun Mehta', 'arjun.mehta@student.edu', 'PROG001
('STU002', 'ENR2021002', 'Priya Sharma', 'priya.sharma@student.edu', 'PROG0
('STU003', 'ENR2021003', 'Rohit Verma', 'rohit.verma@student.edu', 'PROG001
('STU004', 'ENR2021004', 'Sneha Patel', 'sneha.patel@student.edu', 'PROG001
```

Data Output Messages Notifications

```
INSERT 0 150

Query returned successfully in 297 msec.
```

```sql
CREATE TABLE tbl_sponsors (
   sponsor_id VARCHAR(20) PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   organization_type VARCHAR(50),
   contact_person VARCHAR(100),
   email VARCHAR(100),
   phone VARCHAR(15),
   established_year INT,
   CONSTRAINT chk_sponsor_email CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' OR email IS NULL),
   CONSTRAINT chk_sponsor_phone CHECK (phone ~ '^[0-9]{10,15}$' OR phone IS NULL),
   CONSTRAINT chk_established_year CHECK (established_year >= 1800 AND established_year <= EXTRACT(YEAR FROM CURRENT_DATE)),
   CONSTRAINT chk_org_type CHECK (organization_type IN ('Government', 'Private', 'NGO', 'International', 'University'))
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
CREATE TABLE tbl_sponsors (
    sponsor_id VARCHAR(20) PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    organization_type VARCHAR(50),
    contact_person VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(15),
    established_year INT,
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 128 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql

INSERT INTO tbl_sponsors (sponsor_id, name, organization_type, contact_pers
('SPON001', 'National Science Foundation', 'Government', 'Dr. Robert Johnso
('SPON002', 'Department of Science and Technology', 'Government', 'Dr. Anja
('SPON003', 'Indian Council of Medical Research', 'Government', 'Dr. Ramesh
('SPON004', 'Council of Scientific and Industrial Research', 'Government',
```

Data Output | Messages | Notifications

```
INSERT 0 100

Query returned successfully in 265 msec.
```

```sql
CREATE TABLE tbl_funding (
   funding_id VARCHAR(20) PRIMARY KEY,
   sponsor_id VARCHAR(20) NOT NULL,
   amount DECIMAL(15,2) NOT NULL,
   start_date DATE NOT NULL,
   end_date DATE NOT NULL,
   CONSTRAINT fk_funding_sponsor FOREIGN KEY (sponsor_id)
      REFERENCES tbl_sponsors(sponsor_id)
      ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT chk_amount CHECK (amount > 0),
   CONSTRAINT chk_funding_dates CHECK (end_date > start_date)
);
```

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × re

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    CONSTRAINT fk_funding_sponsor FOREIGN KEY (sponsor_id)
        REFERENCES tbl_sponsors(sponsor_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT chk_amount CHECK (amount > 0),
    CONSTRAINT chk_funding_dates CHECK (end_date > start_date)
);
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 82 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
INSERT INTO tbl_funding (funding_id, sponsor_id, amount, start_date, end_da
('FUND001', 'SPON001', 500000.00, '2022-01-01', '2024-12-31'),
('FUND002', 'SPON002', 750000.00, '2022-03-15', '2025-03-14'),
('FUND003', 'SPON003', 1000000.00, '2021-06-01', '2024-05-31'),
('FUND004', 'SPON004', 850000.00, '2022-09-01', '2025-08-31'),
```

Data Output Messages Notifications

```
INSERT 0 100

Query returned successfully in 226 msec.
```

```
CREATE TABLE tbl_ResearchProject (
    project_id VARCHAR(20) PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    start_date DATE NOT NULL,
    end_date DATE,
    status VARCHAR(20) NOT NULL DEFAULT 'Proposed',
    budget_allocated DECIMAL(15,2),
    CONSTRAINT chk_project_status CHECK (status IN ('Proposed', 'Ongoing', 'Completed',
'Suspended', 'Cancelled')),
    CONSTRAINT chk_budget CHECK (budget_allocated >= 0),
    CONSTRAINT chk_project_dates CHECK (end_date IS NULL OR end_date >= start_date)
```

);

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProject/postgres@PostgreSQL

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
CREATE TABLE tbl_ResearchProject (
    project_id VARCHAR(20) PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    description TEXT,
    start_date DATE NOT NULL,
    end_date DATE,
    status VARCHAR(20) NOT NULL DEFAULT 'Proposed',
    budget_allocated DECIMAL(15,2),
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 206 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
ALTER TABLE tbl_ResearchProject
ADD COLUMN supervisor_faculty_id VARCHAR(20),
ADD CONSTRAINT fk_project_supervisor FOREIGN KEY (supervisor_faculty_id)
    REFERENCES tbl_faculty(faculty_id)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
ALTER TABLE tbl_ResearchProject
ADD COLUMN admin_id VARCHAR(20),
ADD CONSTRAINT fk_project_admin FOREIGN KEY (admin_id)
    REFERENCES tbl_admin(admin_id)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
CREATE TABLE tbl_student_project (
    student_id VARCHAR(20),
    project_id VARCHAR(20),
    role VARCHAR(50),
    join_date DATE NOT NULL DEFAULT CURRENT_DATE,
    PRIMARY KEY (student_id, project_id),
    CONSTRAINT fk_sp_student FOREIGN KEY (student_id)
        REFERENCES tbl_student(student_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_sp_project FOREIGN KEY (project_id)
        REFERENCES tbl_ResearchProject(project_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT chk_student_role CHECK (role IN ('Lead Researcher', 'Co-Researcher', 'A
);
```

ics × Dependencies × Dependents × Processes × researchProject/postgres@PostgreSQ

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
ALTER TABLE tbl_ResearchProject
ADD COLUMN funding_id VARCHAR(20),
ADD CONSTRAINT fk_project_funding FOREIGN KEY (funding_id)
    REFERENCES tbl_funding(funding_id)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

researchProject/postgres@PostgreSQL 17

No limit

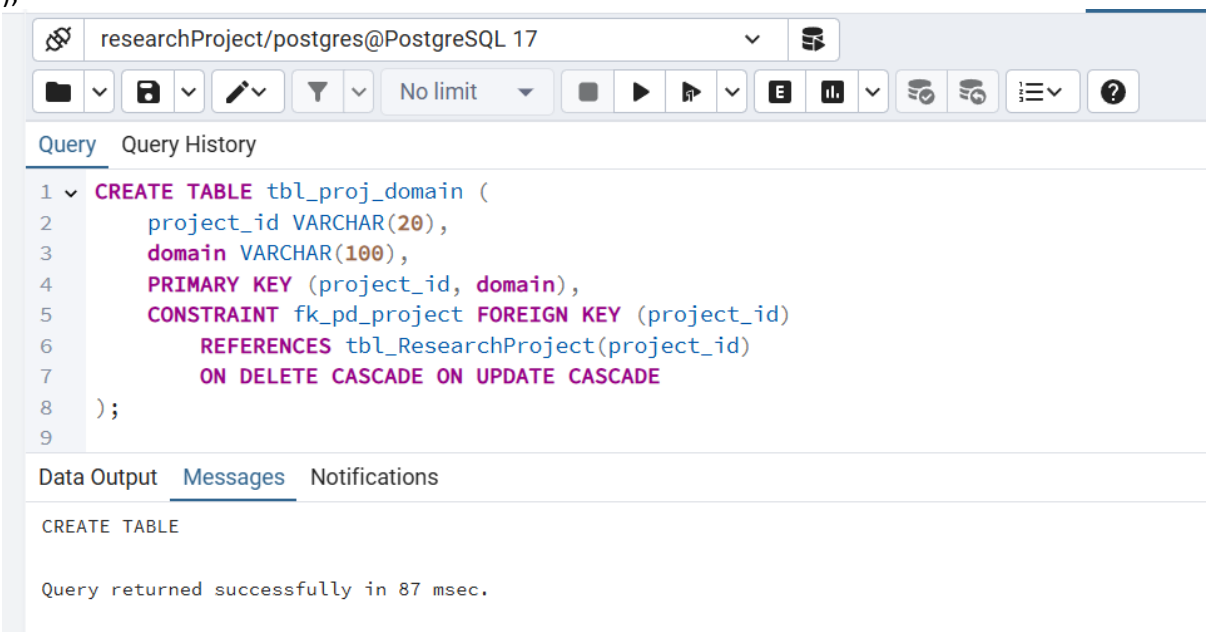
Query Query History

```sql
INSERT INTO tbl_ResearchProject (project_id, title, description, start_dat
('PROJ001', 'Deep Learning for Medical Image Analysis', 'Using CNNs for di
('PROJ002', 'Natural Language Processing for Legal Documents', 'Automated
('PROJ003', 'Blockchain for Supply Chain Management', 'Distributed ledger
('PROJ004', 'Renewable Energy Grid Integration', 'Smart grid technology re
('PROJ005', 'Autonomous Vehicle Navigation Systems', 'Self-driving car alg
```

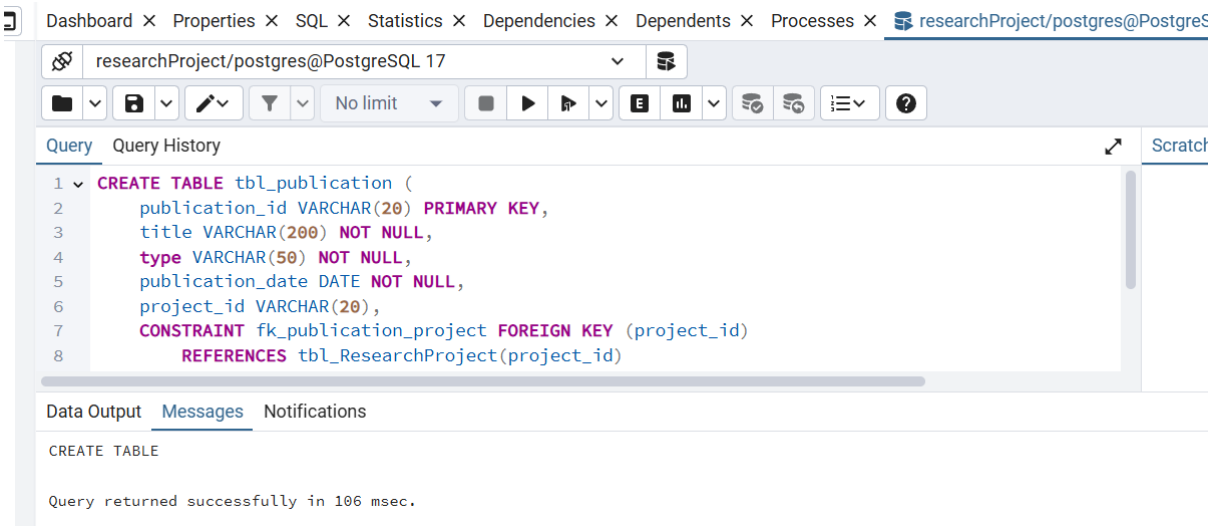
Data Output Messages Notifications

```
INSERT 0 120

Query returned successfully in 1 secs 808 msec.
```

```
CREATE TABLE tbl_proj_domain (
   project_id VARCHAR(20),
   domain VARCHAR(100),
   PRIMARY KEY (project_id, domain),
   CONSTRAINT fk_pd_project FOREIGN KEY (project_id)
      REFERENCES tbl_ResearchProject(project_id)
      ON DELETE CASCADE ON UPDATE CASCADE
);
```




```
CREATE TABLE tbl_publication (
   publication_id VARCHAR(20) PRIMARY KEY,
   title VARCHAR(200) NOT NULL,
   type VARCHAR(50) NOT NULL,
   publication_date DATE NOT NULL,
   project_id VARCHAR(20),
   CONSTRAINT fk_publication_project FOREIGN KEY (project_id)
      REFERENCES tbl_ResearchProject(project_id)
      ON DELETE SET NULL ON UPDATE CASCADE,
   CONSTRAINT chk_pub_type CHECK (type IN ('Journal', 'Conference', 'Book', 'Chapter', 'Patent',
'Technical Report')),
   CONSTRAINT chk_pub_date CHECK (publication_date <= CURRENT_DATE)
);
```




```
CREATE TABLE tbl_authors (
   author_ID VARCHAR(20) PRIMARY KEY,
   author_name VARCHAR(100) NOT NULL,
   CONSTRAINT chk_author_name CHECK (TRIM(author_name) <> '')
);
```

```
CREATE TABLE tbl_publication_authors (
   publication_id VARCHAR(20),
   autor_id VARCHAR(20),
   author_order INT NOT NULL,
   PRIMARY KEY (publication_id, autor_id),
   CONSTRAINT fk_pa_publication FOREIGN KEY (publication_id)
      REFERENCES tbl_publication(publication_id)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT fk_pa_author FOREIGN KEY (autor_id)
      REFERENCES tbl_authors(author_ID)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT chk_author_order CHECK (author_order > 0),
   CONSTRAINT uk_pub_author_order UNIQUE (publication_id, author_order)
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
CREATE TABLE tbl_publication_authors (
    publication_id VARCHAR(20),
    autor_id VARCHAR(20),
    author_order INT NOT NULL,
    PRIMARY KEY (publication_id, autor_id),
    CONSTRAINT fk_pa_publication FOREIGN KEY (publication_id)
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 228 msec.
```

```
CREATE TABLE tbl_audit (
   audit_id VARCHAR(20) PRIMARY KEY,
   action_type VARCHAR(50) NOT NULL,
   old_values TEXT,
   new_values TEXT,
   timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   CONSTRAINT chk_action_type CHECK (action_type IN ('INSERT', 'UPDATE', 'DELETE', 'LOGIN',
'LOGOUT', 'APPROVE', 'REJECT'))
);
```

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchP

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
CREATE TABLE tbl_audit (
    audit_id VARCHAR(20) PRIMARY KEY,
    action_type VARCHAR(50) NOT NULL,
    old_values TEXT,
    new_values TEXT,
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 85 msec.
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
ALTER TABLE tbl_audit
ADD COLUMN project_id VARCHAR(20),
ADD CONSTRAINT fk_audit_project FOREIGN KEY (project_id)
    REFERENCES tbl_ResearchProject(project_id)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

```sql
CREATE TABLE tbl_ResearchProjectProgress (
   progress_id VARCHAR(20) PRIMARY KEY,
   project_id VARCHAR(20) NOT NULL,
   submission_date DATE NOT NULL DEFAULT CURRENT_DATE,
   CONSTRAINT fk_progress_project FOREIGN KEY (project_id)
      REFERENCES tbl_ResearchProject(project_id)
      ON DELETE CASCADE ON UPDATE CASCADE
);
```

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
CREATE TABLE tbl_ResearchProjectProgress (
    progress_id VARCHAR(20) PRIMARY KEY,
    project_id VARCHAR(20) NOT NULL,
    submission_date DATE NOT NULL DEFAULT CURRENT_DATE,
    CONSTRAINT fk_progress_project FOREIGN KEY (project_id)
        REFERENCES tbl_ResearchProject(project_id)
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 195 msec.
```

researchProject/postgres@PostgreSQL 17

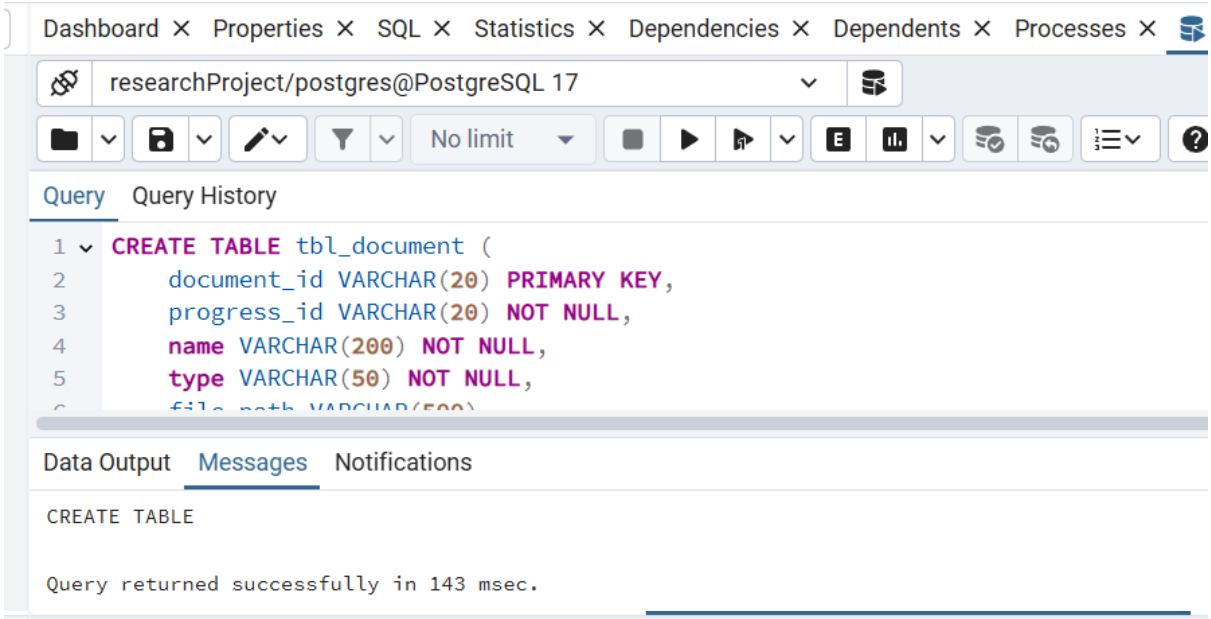
The session is idle and there is no current transaction.

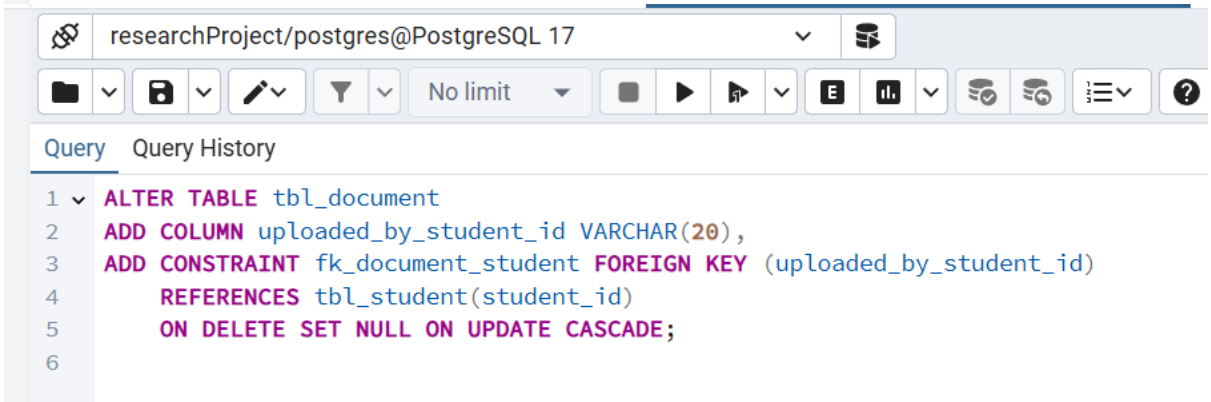
Query Query History

```sql
ALTER TABLE tbl_ResearchProjectProgress
ADD COLUMN submitted_by_student_id VARCHAR(20),
ADD CONSTRAINT fk_progress_student FOREIGN KEY (submitted_by_student_id)
    REFERENCES tbl_student(student_id)
    ON DELETE SET NULL ON UPDATE CASCADE;
```

```sql
CREATE TABLE tbl_document (
   document_id VARCHAR(20) PRIMARY KEY,
```

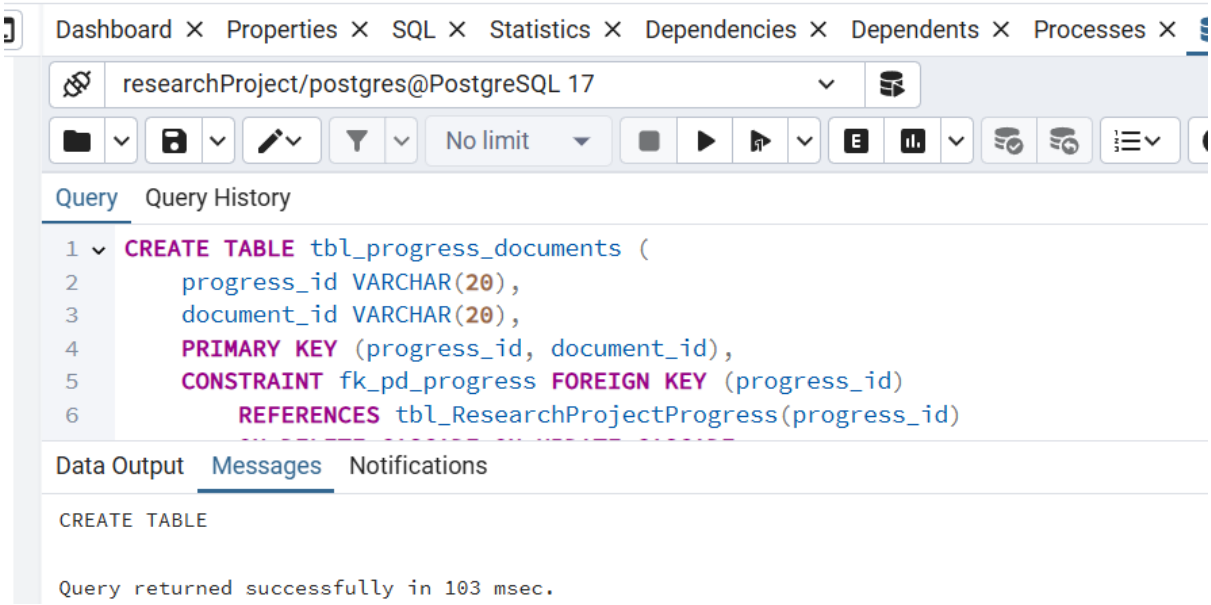
```
    progress_id VARCHAR(20) NOT NULL,
    name VARCHAR(200) NOT NULL,
    type VARCHAR(50) NOT NULL,
    file_path VARCHAR(500),
    file_size BIGINT,
    upload_date DATE NOT NULL DEFAULT CURRENT_DATE,
    CONSTRAINT fk_document_progress FOREIGN KEY (progress_id)
        REFERENCES tbl_ResearchProjectProgress(progress_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT chk_doc_type CHECK (type IN ('Report', 'Paper', 'Presentation', 'Dataset', 'Code',
'Misc')),
    CONSTRAINT chk_file_size CHECK (file_size > 0)
);
```






```
CREATE TABLE tbl_progress_documents (
    progress_id VARCHAR(20),
    document_id VARCHAR(20),
    PRIMARY KEY (progress_id, document_id),
    CONSTRAINT fk_pd_progress FOREIGN KEY (progress_id)
        REFERENCES tbl_ResearchProjectProgress(progress_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_pd_document FOREIGN KEY (document_id)
```

```
    REFERENCES tbl_document(document_id)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```



Lab:10 Queries Execution on Database

Date :01/11/2025

1. List all faculty members with their department name

What it does: Shows a list of all professors/teachers along with which department they belong to (like Computer Science, Physics, etc.)

researchProject/postgres@PostgreSQL 17

Query | Query History

```
SELECT f.faculty_id, f.name, f.email, f.designation,
       d.dep_name, f.phone
FROM tbl_faculty f
JOIN tbl_department d ON f.department_id = d.department_id
ORDER BY d.dep_name, f.name;
```

Data Output | Messages | Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	faculty_id character varying (20)	name character varying (100)	email character varying (100)	designation character varying (50)	dep_name character varying (100)	phone character varying (15)
1	FAC149	Dr. Ishaan Bansal	ishaan.bansal@university.edu	Professor	Adult Education	9876543358
2	FAC027	Dr. Arjun Pillai	arjun.pillai@university.edu	Professor	Aerospace Engineering	9876543236
3	FAC026	Dr. Pooja Saxena	pooja.saxena@university.edu	Associate Professor	Aerospace Engineering	9876543235
4	FAC025	Dr. Rakesh Menon	rakesh.menon@university.edu	Professor	Aerospace Engineering	9876543234
5	FAC100	Dr. Balaji Rao	balaji.rao@university.edu	Professor	Agricultural Engineering	9876543309

2. Find all students enrolled in PhD programs

What it does: Shows only those students who are doing their PhD degree, filtering out Bachelor's and Master's students.

researchProject/postgres@PostgreSQL 17

Query | Query History

```
SELECT s.student_id, s.name, s.email, s.enrollment_id,
       p.program_name, p.degree_level
FROM tbl_student s
JOIN tbl_program p ON s.program_id = p.program_id
WHERE p.degree_level = 'PhD'
ORDER BY s.name;
```

Data Output | Messages | Notifications

Showing rows: 1 to 30 Page No: 1 of 1

	student_id character varying (20)	name character varying (100)	email character varying (100)	enrollment_id character varying (30)	program_name character varying (100)	degree_level character varying (50)
1	STU139	Amit Singh	amit.singh@student.edu	ENR2020027	PhD in Artificial Intelligence	PhD
2	STU049	Anand Kulkarni	anand.kulkarni@student.edu	ENR2020009	PhD in Chemical Engineering	PhD
3	STU149	Aryan Menon	aryan.menon@student.edu	ENR2020029	PhD in Cybersecurity	PhD
4	STU100	Ashish Singh	ashish.singh@student.edu	ENR2020020	PhD in Physics	PhD
5	STU029	Ashok Kamath	ashok.kamath@student.edu	ENR2020005	PhD in Mechanical Engineeri...	PhD
6	STU150	Avni Pillai	avni.pillai@student.edu	ENR2020030	PhD in Cybersecurity	PhD

3. List all ongoing research projects

What it does: Displays all research projects that are currently active/in-progress, not the completed or proposed ones.

researchProject/postgres@PostgreSQL 17

Query | Query History

```
SELECT project_id, title, description, start_date,
       end_date, budget_allocated
FROM tbl_researchProject
WHERE status = 'Ongoing'
ORDER BY start_date DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 30 Page No: 1 of 1

	student_id character varying (20)	name character varying (100)	email character varying (100)	enrollment_id character varying (30)	program_name character varying (100)	degree_level character varying (50)
1	STU139	Amit Singh	amit.singh@student.edu	ENR2020027	PhD in Artificial Intelligence	PhD
2	STU049	Anand Kulkarni	anand.kulkarni@student.edu	ENR2020009	PhD in Chemical Engineering	PhD
3	STU149	Aryan Menon	aryan.menon@student.edu	ENR2020029	PhD in Cybersecurity	PhD
4	STU100	Ashish Singh	ashish.singh@student.edu	ENR2020020	PhD in Physics	PhD
5	STU029	Ashok Kamath	ashok.kamath@student.edu	ENR2020005	PhD in Mechanical Engineeri...	PhD
6	STU150	Avni Pillai	avni.pillai@student.edu	ENR2020030	PhD in Cybersecurity	PhD

4. Get all publications published in the last year

What it does: Shows research papers, journals, books published in the past 12 months from today.

researchProject/postgres@PostgreSQL 17

Query | Query History

```
SELECT publication_id, title, type, publication_date
FROM tbl_publication
WHERE publication_date >= CURRENT_DATE - INTERVAL '1 year'
ORDER BY publication_date DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 9 Page No: 1

	publication_id [PK] character varying (20)	title character varying (200)	type character varying (50)	publication_date date
1	PUB068	VR Therapy Applications in Psychology	Conference	2025-05-12
2	PUB064	Autonomous UAV Systems	Conference	2025-04-15
3	PUB067	Virtual Reality for Mental Health Treatment	Journal	2025-02-28
4	PUB060	Exoskeleton Design and Control	Conference	2025-02-05
5	PUB063	Multi-Agent Coordination in Drone Swarms	Journal	2025-01-20

5. Count students by program

What it does: Shows how many students are in each program (e.g., 50 students in Computer Science, 30 in Physics).

researchProject/postgres@PostgreSQL 17

Query Query History

```
SELECT p.program_name, p.degree_level,
       COUNT(st.student_id) as student_count
FROM tbl_program p
LEFT JOIN tbl_student st ON p.program_id = st.program_id
GROUP BY p.program_id, p.program_name, p.degree_level
ORDER BY student_count DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 150

	program_name character varying (100)	degree_level character varying (50)	student_count bigint
1	Bachelor of Technology in Chemical Engineering	Bachelor	5
2	Bachelor of Technology in Information Technology	Bachelor	5
3	Bachelor of Business Administration	Bachelor	5
4	Bachelor of Technology in Electrical Engineering	Bachelor	5
5	Bachelor of Technology in Electronics	Bachelor	5

6. Find all specializations with their expertise levels

What it does: Lists all research areas (like Machine Learning, Robotics) and shows whether experts are Beginner, Intermediate, Advanced, or Expert level.

researchProject/postgres@PostgreSQL 17

Query Query History

```
SELECT specialization_name, research_domain,
       level_expertise_tag, description
FROM tbl_specialization
WHERE level_expertise_tag IN ('Advanced', 'Expert')
ORDER BY research_domain, specialization_name;
```

Data Output Messages Notifications

Showing rows: 1 to 148 Page No: 1 of 1

	specialization_name character varying (100)	research_domain character varying (100)	level_expertise_tag character varying (50)	description text
1	Agricultural Science	Agriculture	Advanced	Science of farming
2	Animal Science	Agriculture	Advanced	Study of domestic animals
3	Food Science	Agriculture	Advanced	Study of food production
4	Bioinformatics	Biology	Advanced	Computational analysis of biological data
5	Botany	Biology	Advanced	Study of plants

7. Get total funding by sponsor

What it does: Shows how much money each sponsor (company/government/organization) has given in total across all projects.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT s.name as sponsor_name, s.organization_type,
       COUNT(f.funding_id) as funding_count,
       SUM(f.amount) as total_funding
FROM tbl_sponsors s
JOIN tbl_funding f ON s.sponsor_id = f.sponsor_id
GROUP BY s.sponsor_id, s.name, s.organization_type
ORDER BY total_funding DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 100 Page No: 1

	sponsor_name character varying (100)	organization_type character varying (50)	funding_count bigint	total_funding numeric
1	CERN Research	International	1	5000000.00
2	NASA Research	Government	1	4000000.00
3	Chinese Academy of Sciences	Government	1	3800000.00
4	Pfizer Research	Private	1	3500000.00
5	ESA Research	International	1	3500000.00
6	Johnson & Johnson Innovation	Private	1	3200000.00

8. List projects with their domains

What it does: Shows each research project and what field/area it belongs to (like Healthcare, AI, Renewable Energy).

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT rp.project_id, rp.title, rp.status,
       STRING_AGG(pd.domain, ', ') as domains
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
GROUP BY rp.project_id, rp.title, rp.status
ORDER BY rp.title;
```

Data Output Messages Notifications

Showing rows: 1 to 75 Page No: 1 of 1

	project_id [PK] character varying (20)	title character varying (200)	status character varying (20)	domains text
1	PROJ016	5G Network Optimization	Ongoing	Telecommunications, 5G
2	PROJ061	Advanced Battery Chemistry	Ongoing	Battery Technology, Solid State
3	PROJ049	Advanced Materials for Aerospace	Ongoing	Advanced Materials, Aerospace
4	PROJ011	Advanced Robotics for Manufacturing	Ongoing	Robotics, Manufacturing
5	PROJ046	AI-Powered Drug Discovery	Ongoing	Drug Discovery, AI

9. List faculty members with their specialization count

What it does: Shows each teacher and how many different areas of expertise they have (e.g., Prof. Smith has 3 specializations).

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
SELECT f.name, f.designation, d.dep_name,
       COUNT(fs.specialization_id) as specialization_count
FROM tbl_faculty f
JOIN tbl_department d ON f.department_id = d.department_id
LEFT JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id
GROUP BY f.faculty_id, f.name, f.designation, d.dep_name
ORDER BY specialization_count DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 150 | Page No:

	name character varying (100)	designation character varying (50)	dep_name character varying (100)	specialization_count bigint
1	Dr. Shalini Bose	Professor	Biotechnology	3
2	Dr. Vijay Patel	Professor	Mechanical Engineering	3
3	Dr. Subhash Ghosh	Professor	Physics	3
4	Dr. Meera Chopra	Professor	Biotechnology	3
5	Dr. Priya Singh	Associate Professor	Computer Science and Engineering	3
6	Dr. Suresh Reddy	Professor	Electrical Engineering	3

10. List projects with funding details and funding gap

What it does: Shows projects, how much money they need, how much they got, and how much more they still need.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
SELECT rp.project_id, rp.title, rp.budget_allocated, rp.status,
       COALESCE(SUM(f.amount), 0) as total_funding,
       (rp.budget_allocated - COALESCE(SUM(f.amount), 0)) as funding_gap,
       COUNT(f.funding_id) as funding_sources
FROM tbl_researchProject rp
LEFT JOIN tbl_funding f ON rp.funding_id = f.funding_id
GROUP BY rp.project_id, rp.title, rp.budget_allocated, rp.status
HAVING rp.budget_allocated > 0
ORDER BY funding_gap ;
```

Data Output | Messages | Notifications

Showing rows: 1 to 120 | Page No: 1 of 1

	project_id [PK] character varying (20)	title character varying (200)	budget_allocated numeric (15,2)	status character varying (20)	total_funding numeric	funding_gap numeric	funding_sour bigint
1	PROJ025	Quantum Cryptography Networks	2600000.00	Ongoing	2600000.00	0.00	
2	PROJ087	Malaria Vaccine Development	1600000.00	Ongoing	1600000.00	0.00	
3	PROJ045	Biodegradable Electronics	1800000.00	Ongoing	1800000.00	0.00	
4	PROJ033	Hydrogen Fuel Cell Vehicles	2300000.00	Ongoing	2300000.00	0.00	
5	PROJ072	Biometric Security Systems	1150000.00	Ongoing	1150000.00	0.00	

11. List publications with multiple authors using aggregation

What it does: Shows each research paper with all the authors who worked on it together, combined into one list.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT pub.title, pub.type, pub.publication_date,
       COUNT(pa.autor_id) as author_count,
       STRING_AGG(a.author_name, ', ' ORDER BY pa.author_order) as authors
FROM tbl_publication pub
LEFT JOIN tbl_publication_authors pa ON pub.publication_id = pa.publication_id
LEFT JOIN tbl_authors a ON pa.autor_id = a.author_ID
GROUP BY pub.publication_id, pub.title, pub.type, pub.publication_date
ORDER BY pub.publication_date DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 100 Page No: 1 of 1

	title character varying (200)	type character varying (50)	publication_date date	author_count bigint	authors text
1	VR Therapy Applications in Psychology	Conference	2025-05-12	2	Dr. Harry Washington, Dr. Virginia Butler
2	Autonomous UAV Systems	Conference	2025-04-15	2	Dr. Willie Cooper, Dr. Ann Rivera
3	Virtual Reality for Mental Health Treatment	Journal	2025-02-28	2	Dr. Howard Hughes, Dr. Shirley Flores
4	Exoskeleton Design and Control	Conference	2025-02-05	2	Dr. Eugene Russell, Dr. Norma Griffin
5	Multi-Agent Coordination in Drone Swarms	Journal	2025-01-20	2	Dr. Louis Hamilton, Dr. Lois Graham

12. List students with their programs and department

What it does: Shows students along with their program name (PhD/Master's) and which department they're in.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT s.name as student_name, s.email, s.enrollment_id,
       p.program_name, p.degree_level, p.duration,
       d.dep_name, d.building
FROM tbl_student s
JOIN tbl_program p ON s.program_id = p.program_id
JOIN tbl_department d ON p.department_id = d.department_id
ORDER BY d.dep_name, p.program_name, s.name;
```

Data Output Messages Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	student_name character varying (100)	email character varying (100)	enrollment_id character varying (30)	program_name character varying (100)	degree_level character varying (50)	duration integer
1	Abhishek Menon	abhishek.menon@student.edu	ENR2021041	Bachelor of Technology in Aerospace Engineering	Bachelor	4
2	Nikhil Reddy	nikhil.reddy@student.edu	ENR2021043	Bachelor of Technology in Aerospace Engineering	Bachelor	4
3	Preeti Kumar	preeti.kumar@student.edu	ENR2021044	Bachelor of Technology in Aerospace Engineering	Bachelor	4
4	Rajat Sharma	rajat.sharma@student.edu	ENR2021045	Bachelor of Technology in Aerospace Engineering	Bachelor	4
5	Shruti Pillai	shruti.pillai@student.edu	ENR2021042	Bachelor of Technology in Aerospace Engineering	Bachelor	4

13. Funding by organizations based on type and year

What it does: Shows funding grouped by organization type (Government, Private, NGO) and which year the funding was given.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT s.organization_type,
       EXTRACT(YEAR FROM f.start_date) as funding_year,
       COUNT(f.funding_id) as funding_count,
       SUM(f.amount) as total_amount,
       AVG(f.amount) as average_funding
FROM tbl_sponsors s
JOIN tbl_funding f ON s.sponsor_id = f.sponsor_id
GROUP BY s.organization_type, EXTRACT(YEAR FROM f.start_date)
ORDER BY funding_year DESC, total_amount DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 11 | Page N

	organization_type character varying (50)	funding_year numeric	funding_count bigint	total_amount numeric	average_funding numeric
1	Private	2023	16	28300000.00	1768750.000000000000
2	Government	2023	7	13400000.00	1914285.714285714286
3	International	2023	5	11900000.00	2380000.000000000000
4	NGO	2023	1	1100000.00	1100000.000000000000
5	Private	2022	37	65350000.00	1766216.216216216216

14. List publications by author in last 2 years

What it does: Shows how many papers each researcher has published in the recent 2 years.

researchProject/postgres@PostgreSQL 17

The session is idle and there is no current transaction.

Query | Query History

```sql
SELECT f.name, f.designation, d.dep_name,
       (SELECT COUNT(DISTINCT pub.publication_id)
        FROM tbl_publication pub
        JOIN tbl_publication_authors pa ON pub.publication_id = pa.publication_id
        JOIN tbl_authors a ON pa.autor_id = a.author_ID
        WHERE a.author_name ILIKE '%' || f.name || '%'
          AND pub.publication_date >= CURRENT_DATE - INTERVAL '2 years'
       ) as recent_publications
FROM tbl_faculty f
JOIN tbl_department d ON f.department_id = d.department_id
ORDER BY recent_publications DESC, f.name;
```

Data Output | Messages | Notifications

Showing rows: 1 to 150 | Pag

	name character varying (100)	designation character varying (50)	dep_name character varying (100)	recent_publications bigint
1	Dr. Amit Sharma	Assistant Professor	Computer Science and Engineering	1
2	Dr. Kavita Nair	Associate Professor	Electrical Engineering	1
3	Dr. Priya Singh	Associate Professor	Computer Science and Engineering	1
4	Dr. Rajesh Kumar	Professor	Computer Science and Engineering	1
5	Dr. Aakash Malhotra	Professor	Curriculum Studies	0
6	Dr. Abhishek Gupta	Professor	Data Science	0

15. List faculty members in a specific building (A-101)

What it does: Shows all professors whose offices are in building A-101.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT f.name, f.email, f.designation, d.dep_name, d.building
FROM tbl_faculty f
JOIN tbl_department d ON f.department_id = d.department_id
WHERE d.building = 'A-101'
ORDER BY f.name;
```

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1

	name character varying (100)	email character varying (100)	designation character varying (50)	dep_name character varying (100)	building character varying (50)
1	Dr. Amit Sharma	amit.sharma@university.edu	Assistant Professor	Computer Science and Engineering	A-101
2	Dr. Priya Singh	priya.singh@university.edu	Associate Professor	Computer Science and Engineering	A-101
3	Dr. Rajesh Kumar	rajesh.kumar@university.edu	Professor	Computer Science and Engineering	A-101

16. Count sponsors based on organization type

What it does: Shows how many government sponsors, how many private sponsors, how many NGOs, etc.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT organization_type, COUNT(*) as sponsor_count
FROM tbl_sponsors
GROUP BY organization_type
ORDER BY sponsor_count DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 4 Pag

	organization_type character varying (50)	sponsor_count bigint
1	Private	53
2	Government	34
3	International	9
4	NGO	4

17. List count, total, and average of fundings

What it does: Shows overall statistics: total number of fundings, total money given, and average funding amount.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT COUNT(*) as total_fundings,
       SUM(amount) as total_amount,
       AVG(amount) as average_amount
FROM tbl_funding;
```

Data Output | Messages | Notifications

Showing rows: 1 to 1

	total_fundings bigint	total_amount numeric	average_amount numeric
1	100	172000000.00	1720000.000000000000

18. Find faculty specializing in specific area (Machine Learning)

What it does: Search for professors who are experts in Machine Learning and show their contact info, job title, and expertise level.

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × researchProject/postgres@PostgreSQL 17* ×

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT f.faculty_id, f.name, f.email, f.designation,
       s.specialization_name, s.research_domain, s.level_expertise_tag
FROM tbl_faculty f
JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id
JOIN tbl_specialization s ON fs.specialization_id = s.specialization_id
WHERE s.specialization_name ILIKE '%Machine Learning%'
ORDER BY s.level_expertise_tag DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 2 | Page No: 1 of 1

	faculty_id character varying (20)	name character varying (100)	email character varying (100)	designation character varying (50)	specialization_name character varying (100)	research_domain character varying (100)	level_expertise_ character varyin
1	FAC001	Dr. Rajesh Kumar	rajesh.kumar@university.edu	Professor	Machine Learning	Computer Science	Expert
2	FAC002	Dr. Priya Singh	priya.singh@university.edu	Associate Professor	Machine Learning	Computer Science	Expert

19. Find faculty with multiple specializations (cross-domain experts)

What it does: Identifies professors who are experts in more than one field (like someone expert in both AI and Robotics).

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
SELECT f.faculty_id, f.name, f.email, f.designation,
       COUNT(DISTINCT fs.specialization_id) as specialization_count,
       STRING_AGG(DISTINCT s.specialization_name, ', ') as specializations
FROM tbl_faculty f
JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id
JOIN tbl_specialization s ON fs.specialization_id = s.specialization_id
GROUP BY f.faculty_id, f.name, f.email, f.designation
HAVING COUNT(DISTINCT fs.specialization_id) > 1
ORDER BY specialization_count DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 48 | Page No: 1 of 1

	faculty_id [PK] character varying (20)	name character varying (100)	email character varying (100)	designation character varying (50)	specialization_count bigint	specializations text
1	FAC013	Dr. Sanjay Mehta	sanjay.mehta@university.edu	Professor	3	Biochemistry, Chemical Enginee
2	FAC002	Dr. Priya Singh	priya.singh@university.edu	Associate Professor	3	Artificial Intelligence, Data Minir
3	FAC003	Dr. Amit Sharma	amit.sharma@university.edu	Assistant Professor	3	Data Mining, Database Systems,
4	FAC004	Dr. Suresh Reddy	suresh.reddy@university.edu	Professor	3	Electromagnetism, Power Syste
5	FAC007	Dr. Anil Gupta	anil.gupta@university.edu	Professor	3	Fluid Dynamics, Mechanical Eng
6	FAC009	Dr. Vijay Patel	vijay.patel@university.edu	Professor	3	Fluid Dynamics, Mechanical Eng
7	FAC001	Dr. Rajesh Kumar	rajesh.kumar@university.edu	Professor	3	Artificial Intelligence, Machine L
8	FAC015	Dr. Kiran Shah	kiran.shah@university.edu	Professor	3	Biochemistry, Chemical Enginee
9	FAC018	Dr. Ashok Reddy	ashok.reddy@university.edu	Professor	3	Microelectronics, Photonics, VLS
10	FAC019	Dr. Neha Agarwal	neha.agarwal@university.edu	Professor	3	Computer Networks, Database S
11	FAC021	Dr. Deepak Jain	deepak.jain@university.edu	Professor	3	Computer Networks, Database S

Total rows: 48 | Query complete 00:00:01.210 | CRLF | Ln 9, Col 36

20. Department-wise faculty specialization distribution

What it does: Shows which specializations are available in each department and how many faculty have each specialty.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT d.dep_name, s.specialization_name, s.research_domain,
       COUNT(f.faculty_id) as faculty_count,
       STRING_AGG(f.name, ', ') as faculty_names
FROM tbl_department d
JOIN tbl_faculty f ON d.department_id = f.department_id
JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id
JOIN tbl_specialization s ON fs.specialization_id = s.specialization_id
GROUP BY d.dep_name, s.specialization_name, s.research_domain
ORDER BY d.dep_name, faculty_count DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 77 Page No: 1 of 1

	dep_name character varying (100)	specialization_name character varying (100)	research_domain character varying (100)	faculty_count bigint	faculty_names text
1	Aerospace Engineering	Aerospace Engineering	Engineering	3	Dr. Rakesh Menon, Dr. Pooja Saxena, Dr. Arjun Pillai
2	Aerospace Engineering	Mechanical Engineering	Engineering	1	Dr. Rakesh Menon
3	Aerospace Engineering	Fluid Dynamics	Physics	1	Dr. Pooja Saxena
4	Aerospace Engineering	Materials Science	Engineering	1	Dr. Arjun Pillai
5	Biology	Molecular Biology	Biology	2	Dr. Kamala Devi, Dr. Shyam Sunder
6	Biology	Neuropsychology	Psychology	1	Dr. Indira Sharma
7	Biology	Microbiology	Biology	1	Dr. Kamala Devi
8	Biology	Genetics	Biology	1	Dr. Kamala Devi
9	Biology	Neuroscience	Biology	1	Dr. Indira Sharma
10	Biology	Computational Biology	Biology	1	Dr. Shyam Sunder
11	Biotechnology	Genetics	Biology	2	Dr. Meera Chopra, Dr. Vinod Kapoor
12	Biotechnology	Computational Biology	Biology	2	Dr. Meera Chopra, Dr. Shalini Bose

Total rows: 77 | Query complete 00:00:00.241 | CRLF | Ln 10, Col 1

21. Search projects by domain

What it does: Find all projects in a specific field (like Healthcare) and show details like title, status, budget, start/end dates.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT rp.project_id, rp.title, rp.description, rp.status,
       rp.budget_allocated, rp.start_date, rp.end_date,
       STRING_AGG(pd.domain, ', ') as domains
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
WHERE pd.domain ILIKE '%Healthcare%'
GROUP BY rp.project_id, rp.title, rp.description, rp.status,
         rp.budget_allocated, rp.start_date, rp.end_date
ORDER BY rp.start_date DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 3 Page No: 1 of 1

	project_id [PK] character varying (20)	title character varying (200)	description text	status character varying (20)	budget_allocated numeric (15,2)	start_date date	end_date date
1	PROJ052	Wearable Health Monitoring Devices	Continuous patient tracking	Ongoing	1900000.00	2022-08-01	2025-07-
2	PROJ023	Microbiome and Human Health	Gut bacteria therapeutic research	Ongoing	1800000.00	2022-03-01	2025-02-
3	PROJ001	Deep Learning for Medical Image Analysis	Using CNNs for disease detecti...	Ongoing	500000.00	2022-01-15	2024-12-

22. Find projects by domain with funding details

What it does: Search projects in AI or Machine Learning and show who's funding them, how much money, and when.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT rp.title, pd.domain, rp.status, rp.budget_allocated,
       s.name as sponsor_name, f.amount as funded_amount,
       f.start_date as funding_start, f.end_date as funding_end
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
LEFT JOIN tbl_funding f ON rp.project_id = f.funding_id
LEFT JOIN tbl_sponsors s ON f.sponsor_id = s.sponsor_id
WHERE pd.domain ILIKE '%AI%' OR pd.domain ILIKE '%Machine Learning%'
ORDER BY rp.budget_allocated DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 11 Page No: 1 of 1

	title character varying (200)	domain character varying (100)	status character varying (20)	budget_allocated numeric (15,2)	sponsor_name character varying (100)	funded_amount numeric (15,2)	funding_start date
1	Telemedicine AI Assistant	AI Assistant	Ongoing	2400000.00	[null]	[null]	[null]
2	Earthquake Prediction Using ML	Machine Learning	Ongoing	2200000.00	[null]	[null]	[null]
3	Quantum Machine Learning Algorithms	Machine Learning	Ongoing	2000000.00	[null]	[null]	[null]
4	Biodegradable Plastic Alternatives	Sustainability	Ongoing	1900000.00	[null]	[null]	[null]
5	Biodegradable Electronics	Sustainable Electronics	Ongoing	1800000.00	[null]	[null]	[null]
6	Neural Network Hardware Accelerators	AI Acceleration	Ongoing	1700000.00	[null]	[null]	[null]
7	AI-Powered Drug Discovery	AI	Ongoing	1700000.00	[null]	[null]	[null]
8	Forest Fire Prediction AI	AI Prediction	Completed	1050000.00	[null]	[null]	[null]
9	Blockchain for Supply Chain Management	Supply Chain	Ongoing	1000000.00	[null]	[null]	[null]
10	Blockchain for Supply Chain Management	Blockchain	Ongoing	1000000.00	[null]	[null]	[null]
11	Climate Change Prediction Models	Machine Learning	Ongoing	550000.00	[null]	[null]	[null]

23. Projects by domain with publication count

What it does: Shows projects in each field and how many research papers each project has published.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT pd.domain, rp.title, rp.status,
       COUNT(pub.publication_id) as publication_count,
       rp.budget_allocated
FROM tbl_proj_domain pd
JOIN tbl_researchProject rp ON pd.project_id = rp.project_id
LEFT JOIN tbl_publication pub ON rp.project_id = pub.project_id
GROUP BY pd.domain, rp.title, rp.status, rp.budget_allocated
ORDER BY publication_count DESC, rp.budget_allocated DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	domain character varying (100)	title character varying (200)	status character varying (20)	publication_count bigint	budget_allocated numeric (15,2)
1	Oncology	Precision Oncology Database	Ongoing	2	3500000.00
2	Precision Medicine	Precision Oncology Database	Ongoing	2	3500000.00
3	Battery Management	Smart Grid Battery Management	Ongoing	2	3200000.00
4	Smart Grid	Smart Grid Battery Management	Ongoing	2	3200000.00
5	Neuromorphic Computing	Neuromorphic Computing Chips	Ongoing	2	3100000.00
6	Hardware	Neuromorphic Computing Chips	Ongoing	2	3100000.00
7	Genetics	Gene Editing for Rare Diseases	Ongoing	2	3000000.00
8	CRISPR	Gene Editing for Rare Diseases	Ongoing	2	3000000.00
9	Synthetic Biology	Synthetic Biology for Biofuels	Ongoing	2	2900000.00
10	Biofuels	Synthetic Biology for Biofuels	Ongoing	2	2900000.00
11	Space Technology	Space Debris Tracking System	Ongoing	2	2800000.00
12	Tracking Systems	Space Debris Tracking System	Ongoing	2	2800000.00

24. Multi-domain projects (interdisciplinary research)

What it does: Finds projects that combine multiple fields (like a project using both AI and Healthcare).

researchProject/postgres@PostgreSQL 17

```sql
SELECT rp.project_id, rp.title, rp.status, rp.budget_allocated,
       COUNT(pd.domain) as domain_count,
       STRING_AGG(pd.domain, ' | ') as all_domains
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
GROUP BY rp.project_id, rp.title, rp.status, rp.budget_allocated
HAVING COUNT(pd.domain) > 1
ORDER BY domain_count DESC;
```

Showing rows: 1 to 75 Page No: 1 of 1

	project_id [PK] character varying (20)	title character varying (200)	status character varying (20)	budget_allocated numeric (15,2)	domain_count bigint	all_domains text
1	PROJ067	Metamaterials for Invisibility	Ongoing	2500000.00	2	Metamaterials | Optics
2	PROJ045	Biodegradable Electronics	Ongoing	1800000.00	2	Sustainable Electronics | Gree
3	PROJ033	Hydrogen Fuel Cell Vehicles	Ongoing	2300000.00	2	Clean Energy | Transportation
4	PROJ072	Biometric Security Systems	Ongoing	1150000.00	2	Biometrics | Security
5	PROJ046	AI-Powered Drug Discovery	Ongoing	1700000.00	2	Drug Discovery | AI
6	PROJ073	Forest Fire Prediction AI	Completed	1050000.00	2	Forest Management | AI Predi
7	PROJ040	Ocean Thermal Energy Conversion	Ongoing	2700000.00	2	Ocean Energy | Renewable En
8	PROJ014	Gene Editing for Rare Diseases	Ongoing	3000000.00	2	Genetics | CRISPR
9	PROJ031	Graphene-based Supercapacitors	Ongoing	1800000.00	2	Materials Science | Energy St
10	PROJ042	Augmented Reality Education Platform	Ongoing	1300000.00	2	Augmented Reality | Educatio
11	PROJ030	Exoskeleton for Mobility Assistance	Ongoing	1600000.00	2	Robotics | Rehabilitation

25. Projects in specific domain with funding gap

What it does: Find projects in Renewable Energy that need more money and show how much funding gap they have.

researchProject/postgres@PostgreSQL 17

```sql
SELECT rp.title, pd.domain, rp.status, rp.budget_allocated,
       COALESCE(SUM(f.amount), 0) as total_funding_received,
       (rp.budget_allocated - COALESCE(SUM(f.amount), 0)) as funding_gap
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
LEFT JOIN tbl_funding f ON rp.project_id = f.funding_id
WHERE pd.domain ILIKE '%Renewable Energy%'
  AND rp.status IN ('Proposed', 'Ongoing')
GROUP BY rp.project_id, rp.title, pd.domain, rp.status, rp.budget_allocated
HAVING (rp.budget_allocated - COALESCE(SUM(f.amount), 0)) > 0
ORDER BY funding_gap DESC;
```

Showing rows: 1 to 2 Page No: 1 of 1

	title character varying (200)	domain character varying (100)	status character varying (20)	budget_allocated numeric (15,2)	total_funding_received numeric	funding_gap numeric
1	Ocean Thermal Energy Conversion	Renewable Energy	Ongoing	2700000.00	0	2700000.00
2	Renewable Energy Grid Integration	Renewable Energy	Ongoing	850000.00	0	850000.00

26. Active funding opportunities by sponsor type

What it does: Shows current active fundings grouped by sponsor type (government/private) with total amounts.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT s.organization_type, COUNT(f.funding_id) as active_fundings,
       SUM(f.amount) as total_active_funding,
       AVG(f.amount) as avg_funding_per_project
FROM tbl_sponsors s
JOIN tbl_funding f ON s.sponsor_id = f.sponsor_id
WHERE f.end_date > CURRENT_DATE
GROUP BY s.organization_type
ORDER BY total_active_funding DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 4 Page

	organization_type character varying (50)	active_fundings bigint	total_active_funding numeric	avg_funding_per_project numeric
1	Private	22	38750000.00	1761363.636363636364
2	International	7	19000000.00	2714285.714285714286
3	Government	9	18500000.00	2055555.555555555556
4	NGO	1	1100000.00	1100000.000000000000

27. Students by program with matching faculty specializations

What it does: Shows each program, how many students, and what research areas are available through professors in that department.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT p.program_name, p.degree_level,
       COUNT(DISTINCT st.student_id) as student_count,
       COUNT(DISTINCT f.faculty_id) as faculty_count,
       STRING_AGG(DISTINCT s.specialization_name, ', ') as available_specializations
FROM tbl_program p
LEFT JOIN tbl_student st ON p.program_id = st.program_id
JOIN tbl_department d ON p.department_id = d.department_id
LEFT JOIN tbl_faculty f ON f.department_id = d.department_id
LEFT JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id
LEFT JOIN tbl_specialization s ON fs.specialization_id = s.specialization_id
GROUP BY p.program_id, p.program_name, p.degree_level
ORDER BY student_count DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	program_name character varying (100)	degree_level character varying (50)	student_count bigint	faculty_count bigint	available_specializations text
1	Bachelor of Technology in Chemical Engineering	Bachelor	5	3	Analytical Chemistry, Biochemistry, Chemical Engineering, Organic
2	Bachelor of Technology in Cybersecurity	Bachelor	5	3	[null]
3	Bachelor of Technology in Biotechnology	Bachelor	5	3	Bioinformatics, Computational Biology, Genetics, Molecular Biolog
4	Bachelor of Technology in Electronics	Bachelor	5	3	Microelectronics, Photonics, Signal Processing, VLSI Design
5	Bachelor of Arts in Economics	Bachelor	5	3	Behavioral Economics, Econometrics, Macroeconomics, Microeco
6	Bachelor of Technology in Electrical Engineering	Bachelor	5	3	Control Systems, Electromagnetism, Nuclear Physics, Power Syste
7	Bachelor of Technology in Information Technology	Bachelor	5	3	Computer Networks, Database Systems, Distributed Systems, Hun

28. Publications by project domain

What it does: Shows research output organized by field - how many papers in AI, Healthcare, etc., and what types (journal, conference).

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```
SELECT pd.domain, pub.type as publication_type,
       COUNT(pub.publication_id) as publication_count,
       STRING_AGG(pub.title, ' | ') as publication_titles
FROM tbl_proj_domain pd
JOIN tbl_researchProject rp ON pd.project_id = rp.project_id
JOIN tbl_publication pub ON rp.project_id = pub.project_id
GROUP BY pd.domain, pub.type
ORDER BY pd.domain, publication_count DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 174 Page No: 1 of 1

	domain character varying (100)	publication_type character varying (50)	publication_count bigint	publication_titles text
1	5G	Journal	1	5G Network Architecture and Optimization
2	5G	Technical Report	1	Next-Generation Telecommunications Technologies
3	Advanced Materials	Journal	1	Lightweight Composite Materials for Aerospace
4	Advanced Materials	Chapter	1	Advanced Materials Research
5	Aerospace	Journal	2	Hypersonic Propulsion Systems | Lightweight Composite Materials for Aerospace
6	Aerospace	Conference	1	Advanced Flight Technologies
7	Aerospace	Chapter	1	Advanced Materials Research

29. Most prolific authors by research domain

What it does: Identifies the most productive researchers in each field based on number of publications and projects.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT a.author_name, pd.domain,
       COUNT(DISTINCT pub.publication_id) as publications,
       COUNT(DISTINCT rp.project_id) as projects_involved
FROM tbl_authors a
JOIN tbl_publication_authors pa ON a.author_ID = pa.autor_id
JOIN tbl_publication pub ON pa.publication_id = pub.publication_id
JOIN tbl_researchProject rp ON pub.project_id = rp.project_id
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
GROUP BY a.author_ID, a.author_name, pd.domain
ORDER BY publications DESC, projects_involved DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 232 Page No: 1

	author_name character varying (100)	domain character varying (100)	publications bigint	projects_involved bigint
1	Dr. Rajesh Kumar	Hardware	1	1
2	Dr. Rajesh Kumar	Neuromorphic Computi...	1	1
3	Dr. Priya Singh	Hardware	1	1
4	Dr. Priya Singh	Neuromorphic Computi...	1	1
5	Dr. Amit Sharma	Hardware	1	1
6	Dr. Amit Sharma	Neuromorphic Computi...	1	1
7	Dr. Suresh Reddy	Ocean Energy	1	1
8	Dr. Suresh Reddy	Renewable Energy	1	1

30. Recent publications in high-priority domains

What it does: Shows recent papers (last year) in important fields like AI, Machine Learning, Data Science, with author names.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT pub.title, pub.type, pub.publication_date,
       pd.domain, rp.title as project_title,
       STRING_AGG(a.author_name, ', ' ORDER BY pa.author_order) as authors
FROM tbl_publication pub
JOIN tbl_researchProject rp ON pub.project_id = rp.project_id
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
LEFT JOIN tbl_publication_authors pa ON pub.publication_id = pa.publication_id
LEFT JOIN tbl_authors a ON pa.autor_id = a.author_ID
WHERE pub.publication_date >= CURRENT_DATE - INTERVAL '2 year'
  AND pd.domain IN ('AI', 'Machine Learning', 'Data Science', 'Cybersecurity')
GROUP BY pub.publication_id, pub.title, pub.type, pub.publication_date,
         pd.domain, rp.title
ORDER BY pub.publication_date DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 9 Page No: 1 of 1

	title character varying (200)	type character varying (50)	publication_date date	domain character varying (100)	project_title character varying (200)	autho text
1	Quantum Computing for AI Optimization	Conference	2024-03-15	Machine Learning	Quantum Machine Learning Algorithms	[null]
2	Global Warming Forecasting Models	Conference	2024-03-10	Machine Learning	Climate Change Prediction Models	[null]
3	Cyberphysical System Protection	Conference	2024-02-28	Cybersecurity	Cyberphysical Security Systems	Dr. M
4	Seismic Data Analysis Techniques	Conference	2024-02-08	Machine Learning	Earthquake Prediction Using ML	Dr. Ca
5	Quantum Machine Learning Algorithms and Applications	Journal	2024-01-30	Machine Learning	Quantum Machine Learning Algorithms	[null]
6	Machine Learning for Network Security	Conference	2024-01-28	Cybersecurity	Cybersecurity Threat Detection AI	[null]
7	Molecular Design Algorithms	Technical Report	2023-12-28	AI	AI-Powered Drug Discovery	Dr. Je

Total rows: 9 | Query complete 00:00:00.542 | CRLF | Ln 12, Col 29

31. Projects with progress tracking

What it does: Shows all projects with how many progress reports they submitted and how many documents uploaded, plus last update date

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT rp.title, rp.status, pd.domain,
       COUNT(DISTINCT prog.progress_id) as progress_reports,
       COUNT(DISTINCT doc.document_id) as documents_submitted,
       MAX(prog.submission_date) as last_update
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
LEFT JOIN tbl_ResearchProjectProgress prog ON rp.project_id = prog.project_id
LEFT JOIN tbl_document doc ON prog.progress_id = doc.progress_id
GROUP BY rp.project_id, rp.title, rp.status, pd.domain
ORDER BY last_update DESC NULLS LAST;
```

Data Output | Messages | Notifications

Showing rows: 1 to 150 Page No: 1 of 1

	title character varying (200)	status character varying (20)	domain character varying (100)	progress_reports bigint	documents_submitted bigint	last_update date
1	Atmospheric Water Harvesting	Ongoing	Water Harvesting	2	0	2025-01-31
2	Atmospheric Water Harvesting	Ongoing	Atmospheric Science	2	0	2025-01-31
3	Virtual Reality Therapy Platform	Ongoing	Mental Health	2	0	2024-12-31
4	Virtual Reality Therapy Platform	Ongoing	Virtual Reality	2	0	2024-12-31
5	Metamaterials for Invisibility	Ongoing	Optics	2	0	2024-12-31
6	Metamaterials for Invisibility	Ongoing	Metamaterials	2	0	2024-12-31
7	Cognitive Radio Networks	Ongoing	Telecommunications	2	0	2024-11-30
8	Autonomous Drone Swarm Coordination	Ongoing	Swarm Intelligence	2	0	2024-11-30

32. Project completion rate by domain

What it does: Calculates what percentage of projects are completed, ongoing, or proposed in each research field.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT pd.domain,
       COUNT(CASE WHEN rp.status = 'Completed' THEN 1 END) as completed,
       COUNT(CASE WHEN rp.status = 'Ongoing' THEN 1 END) as ongoing,
       COUNT(CASE WHEN rp.status = 'Proposed' THEN 1 END) as proposed,
       COUNT(rp.project_id) as total_projects,
       ROUND(100.0 * COUNT(CASE WHEN rp.status = 'Completed' THEN 1 END) /
             COUNT(rp.project_id), 2) as completion_rate
FROM tbl_proj_domain pd
JOIN tbl_researchProject rp ON pd.project_id = rp.project_id
GROUP BY pd.domain
ORDER BY completion_rate DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 131 | Page No: 1

	domain character varying (100)	completed bigint	ongoing bigint	proposed bigint	total_projects bigint	completion_rate numeric
1	Marine Biology	1	0	0	1	100.00
2	Forest Management	1	0	0	1	100.00
3	Arctic Studies	1	0	0	1	100.00
4	AI Prediction	1	0	0	1	100.00
5	Climate Research	1	0	0	1	100.00
6	Ecology	1	0	0	1	100.00
7	Green Technology	0	1	0	1	0.00
8	Regenerative Medicine	0	1	0	1	0.00

Total rows: 131 | Query complete 00:00:00.371

33. Projects nearing deadline without recent progress

What it does: Alert system - shows projects due in 60 days but haven't submitted progress in 30 days (projects at risk).

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```
SELECT rp.project_id, rp.title, pd.domain, rp.end_date,
       rp.status, MAX(prog.submission_date) as last_progress_date,
       (rp.end_date - CURRENT_DATE) as days_remaining
FROM tbl_researchProject rp
JOIN tbl_proj_domain pd ON rp.project_id = pd.project_id
LEFT JOIN tbl_ResearchProjectProgress prog ON rp.project_id = prog.project_id
WHERE rp.status = 'Ongoing'
  AND rp.end_date IS NOT NULL
  AND rp.end_date > CURRENT_DATE
  AND rp.end_date <= CURRENT_DATE + INTERVAL '60 days'
GROUP BY rp.project_id, rp.title, pd.domain, rp.end_date, rp.status
HAVING MAX(prog.submission_date) IS NULL
   OR MAX(prog.submission_date) < CURRENT_DATE - INTERVAL '30 days'
ORDER BY days_remaining;
```

Data Output | Messages | Notifications

Showing rows: 1 to 8 Page No: 1 of 1

	project_id character varying (20)	title character varying (200)	domain character varying (100)	end_date date	status character varying (20)	last_progress_date date	days_remaining integer
1	PROJ044	Automated Traffic Management ...	Smart Cities	2025-11-14	Ongoing	2023-12-31	14
2	PROJ044	Automated Traffic Management ...	Traffic Management	2025-11-14	Ongoing	2023-12-31	14
3	PROJ021	Alzheimer Disease Early Detection	Neuroscience	2025-11-30	Ongoing	2023-12-31	30
4	PROJ062	Earthquake-Resistant Buildings	Structural Engineering	2025-11-30	Ongoing	2023-12-31	30
5	PROJ021	Alzheimer Disease Early Detection	Biomarkers	2025-11-30	Ongoing	2023-12-31	30
6	PROJ062	Earthquake-Resistant Buildings	Earthquake Safety	2025-11-30	Ongoing	2023-12-31	30
7	PROJ047	Hypersonic Flight Technology	Aerospace	2025-12-14	Ongoing	2023-12-31	44
8	PROJ047	Hypersonic Flight Technology	Hypersonic	2025-12-14	Ongoing	2023-12-31	44

34. Document submissions by project domain

What it does: Analyzes what types of documents (reports, papers, presentations) are submitted in each domain and their file sizes.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT pd.domain, doc.type as document_type,
       COUNT(doc.document_id) as document_count,
       SUM(doc.file_size) as total_size_bytes,
       ROUND(AVG(doc.file_size)::numeric / 1024 / 1024, 2) as avg_size_mb
FROM tbl_proj_domain pd
JOIN tbl_researchProject rp ON pd.project_id = rp.project_id
JOIN tbl_ResearchProjectProgress prog ON rp.project_id = prog.project_id
JOIN tbl_document doc ON prog.progress_id = doc.progress_id
GROUP BY pd.domain, doc.type
ORDER BY pd.domain, document_count DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 132 | Page No: 1

	domain character varying (100)	document_type character varying (50)	document_count bigint	total_size_bytes numeric	avg_size_mb numeric
1	5G	Report	2	5767168	2.75
2	5G	Code	1	22020096	21.00
3	5G	Presentation	1	31457280	30.00
4	Agriculture	Report	2	6553600	3.13
5	Agriculture	Dataset	1	52428800	50.00
6	Agriculture	Paper	1	6291456	6.00
7	Artificial Intelligence	Report	4	11223616	2.68
8	Artificial Intelligence	Dataset	3	581959680	185.00

Total rows: 132 | Query complete 00:00:00.259

35. List all active projects with progress and document counts

What it does: Shows ongoing projects with count of progress reports and documents uploaded for each project.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT rp.project_id, rp.title, rp.status, rp.start_date, rp.end_date,
       COUNT(DISTINCT prog.progress_id) as progress_reports,
       COUNT(DISTINCT doc.document_id) as documents_uploaded,
       MAX(prog.submission_date) as last_progress_date
FROM tbl_researchProject rp
LEFT JOIN tbl_ResearchProjectProgress prog ON rp.project_id = prog.project_id
LEFT JOIN tbl_document doc ON prog.progress_id = doc.progress_id
WHERE rp.status = 'Ongoing'
GROUP BY rp.project_id, rp.title, rp.status, rp.start_date, rp.end_date
ORDER BY last_progress_date DESC NULLS LAST;
```

Data Output Messages Notifications

Showing rows: 1 to 94 Page No: 1 of 1

	project_id [PK] character varying (20)	title character varying (200)	status character varying (20)	start_date date	end_date date	progress_reports bigint	documents_upload bigint
1	PROJ069	Atmospheric Water Harvesting	Ongoing	2023-12-01	2026-11-30	2	
2	PROJ034	Virtual Reality Therapy Platform	Ongoing	2023-12-01	2026-11-30	2	
3	PROJ067	Metamaterials for Invisibility	Ongoing	2023-11-01	2026-10-31	2	
4	PROJ065	Cognitive Radio Networks	Ongoing	2023-10-01	2026-09-30	2	
5	PROJ032	Autonomous Drone Swarm Coordination	Ongoing	2023-11-01	2026-10-31	2	

36. Project completion rate based on start/end dates

What it does: Calculates completion rate using actual start and end dates to see if projects finish on time.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT
    status,
    COUNT(*) AS project_count,
    ROUND(AVG(COALESCE(end_date, CURRENT_DATE) - start_date), 2) AS avg_duration_days,
    SUM(budget_allocated) AS total_budget
FROM
    tbl_researchProject
GROUP BY
    status
ORDER BY
    CASE
        WHEN status = 'Completed' THEN 1
        WHEN status = 'Ongoing' THEN 2
        WHEN status = 'Proposed' THEN 3
        WHEN status = 'Suspended' THEN 4
        ELSE 5
    END;
```

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

	status character varying (20)	project_count bigint	avg_duration_days numeric	total_budget numeric
1	Completed	6	729.50	5850000.00
2	Ongoing	94	1102.47	166150000.00
3	Proposed	18	1091.33	10000000.00
4	Suspended	1	565.00	750000.00
5	Cancelled	1	578.00	400000.00

37. Project completion rate based on start/end dates

What it does: Calculates completion rate using actual start and end dates to see if projects finish on time.

researchProject/postgres@PostgreSQL 17

No limit

Query Query History

```sql
SELECT f.funding_id, f.amount, f.start_date, f.end_date, s.name as sponsor_name
FROM tbl_funding f
JOIN tbl_sponsors s ON f.sponsor_id = s.sponsor_id
WHERE f.end_date > CURRENT_DATE
ORDER BY f.amount DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 39 Page No: 1

	funding_id character varying (20)	amount numeric (15,2)	start_date date	end_date date	sponsor_name character varying (100)
1	FUND057	5000000.00	2021-01-01	2025-12-31	CERN Research
2	FUND054	4000000.00	2022-01-01	2025-12-31	NASA Research
3	FUND061	3800000.00	2023-08-01	2026-07-31	Chinese Academy of Sciences
4	FUND036	3500000.00	2022-01-01	2025-12-31	Pfizer Research
5	FUND056	3500000.00	2023-06-01	2026-05-31	ESA Research

38. Get average project budget grouped by status

What it does: Shows average budget for completed projects, ongoing projects, and proposed projects separately.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT status,
       COUNT(*) as project_count,
       ROUND(AVG(budget_allocated), 2) as avg_budget
FROM tbl_researchProject
GROUP BY status
ORDER BY avg_budget DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5

	status character varying (20)	project_count bigint	avg_budget numeric
1	Ongoing	94	1767553.19
2	Completed	6	975000.00
3	Suspended	1	750000.00
4	Proposed	18	555555.56
5	Cancelled	1	400000.00

39. Get average, minimum, and maximum project duration by status

What it does: Shows statistics about how long projects take: shortest, longest, and average duration for each status.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT
    status,
    ROUND(AVG(COALESCE(end_date, CURRENT_DATE) - start_date), 2) AS avg_duration_day
    MIN(COALESCE(end_date, CURRENT_DATE) - start_date) AS min_duration_days,
    MAX(COALESCE(end_date, CURRENT_DATE) - start_date) AS max_duration_days
FROM
    tbl_researchProject
GROUP BY
    status
ORDER BY
    avg_duration_days DESC;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5

	status character varying (20)	avg_duration_days numeric	min_duration_days integer	max_duration_days integer
1	Ongoing	1102.47	730	1825
2	Proposed	1091.33	823	1386
3	Completed	729.50	729	730
4	Cancelled	578.00	578	578
5	Suspended	565.00	565	565

40. List project starting year with count and sum

What it does: Groups projects by year they started, shows how many started each year, and total budget for that year.

researchProject/postgres@PostgreSQL 17

No limit

Query | Query History

```sql
SELECT
    EXTRACT(YEAR FROM start_date) AS start_year,
    COUNT(*) AS project_count,
    SUM(budget_allocated) AS total_budget
FROM
    tbl_researchProject
GROUP BY
    start_year
ORDER BY
    start_year;
```

Data Output | Messages | Notifications

Showing rows: 1 to 3

	start_year numeric	project_count bigint	total_budget numeric
1	2021	4	10200000.00
2	2022	79	113450000.00
3	2023	37	59500000.00

Lab:11 User Interface And DataConnection

FacultyHttpServer.java:

```
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;

import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.sql.*;

public class FacultyHttpServer {

    public static void main(String[] args) throws Exception {
        HttpServer server = HttpServer.create(new InetSocketAddress(8080), 0);

        server.createContext("/faculty", new FacultyHandler());

        server.setExecutor(null);
        System.out.println("✅ GET Server running at http://localhost:8080");
        server.start();
    }

    static class FacultyHandler implements HttpHandler {
        @Override
        public void handle(HttpExchange exchange) throws IOException {
            StringBuilder response = new StringBuilder("[");

            try (
                Connection con = DriverManager.getConnection(
```

```
            "jdbc:postgresql://localhost:5432/researchProject",
            "postgres",
            "admin"
        );
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM tbl_faculty")
    ) {

        boolean first = true;
        while (rs.next()) {
            if (!first) response.append(",");

            response.append("{")
                    .append("\"faculty_id\":\"").append(rs.getString("faculty_id")).append("\",")
                    .append("\"name\":\"").append(rs.getString("name")).append("\",")
                    .append("\"email\":\"").append(rs.getString("email")).append("\",")
                    .append("\"department_id\":\"").append(rs.getString("department_id")).append("\",
")
                    .append("\"designation\":\"").append(rs.getString("designation")).append("\",")
                    .append("\"phone\":\"").append(rs.getString("phone")).append("\",")
                    .append("\"specialization_id\":\"").append(rs.getString("specialization_id")).append(
"\"")
                    .append("}");

            first = false;
        }

        response.append("]");

    } catch (Exception e) {
        response = new StringBuilder("{\"error\":\"" + e.getMessage() + "\"}");
    }
```

```
        exchange.getResponseHeaders().add("Access-Control-Allow-Origin", "*");
        exchange.getResponseHeaders().add("Content-Type", "application/json");
        exchange.sendResponseHeaders(200, response.length());

        OutputStream os = exchange.getResponseBody();
        os.write(response.toString().getBytes());
        os.close();
      }
    }
}
```

FacultyCrudApp.java:

```
import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;

import java.io.*;
import java.net.InetSocketAddress;
import java.sql.*;

public class FacultyCrudApp {

    public static void main(String[] args) throws Exception {

      HttpServer server = HttpServer.create(new InetSocketAddress(8081), 0);

      server.createContext("/add", new AddFaculty());
      server.createContext("/update", new UpdateFaculty());
      server.createContext("/delete", new DeleteFaculty());
```

```
    server.createContext("/search", new SearchFaculty());

    server.setExecutor(null);
    System.out.println("✅ CRUD Server running at http://localhost:8081");
    server.start();
}

static Connection connect() throws Exception {
    return DriverManager.getConnection(
            "jdbc:postgresql://localhost:5432/researchProject",
            "postgres",
            "admin"
    );
}

// INSERT FACULTY ------------------------------------
static class AddFaculty implements HttpHandler {
    public void handle(HttpExchange ex) throws IOException {
        if (!ex.getRequestMethod().equalsIgnoreCase("POST")) {
            send(ex, "{\"error\":\"Only POST allowed\"}");
            return;
        }

        BufferedReader br = new BufferedReader(
                new InputStreamReader(ex.getRequestBody())
        );
        String body = br.readLine();
        // order: id,name,email,dept,designation,spec,phone
        String[] d = body.split(",");

        try (Connection con = connect();
```

```
            PreparedStatement ps = con.prepareStatement(
                    "INSERT INTO tbl_faculty VALUES (?,?,?,?,?,?,?)"
            )) {

            ps.setString(1, d[0]);
            ps.setString(2, d[1]);
            ps.setString(3, d[2]);
            ps.setString(4, d[3]);
            ps.setString(5, d[4]);
            ps.setString(6, d[5]);
            ps.setString(7, d[6]);

            ps.executeUpdate();
            send(ex, "{\"message\":\"Faculty added successfully\"}");

        } catch (Exception e) {
            send(ex, "{\"error\":\"" + e.getMessage() + "\"}");
        }
    }
}

// UPDATE FACULTY ------------------------------------
static class UpdateFaculty implements HttpHandler {
    public void handle(HttpExchange ex) throws IOException {
        if (!ex.getRequestMethod().equalsIgnoreCase("PUT")) {
            send(ex, "{\"error\":\"Only PUT allowed\"}");
            return;
        }

        BufferedReader br = new BufferedReader(
                new InputStreamReader(ex.getRequestBody())
```

```
        );
        // order: id,new_phone,new_designation
        String[] d = br.readLine().split(",");

        try (Connection con = connect();
            PreparedStatement ps = con.prepareStatement(
                "UPDATE tbl_faculty SET phone=?, designation=? WHERE faculty_id=?"
            )) {

            ps.setString(1, d[1]);
            ps.setString(2, d[2]);
            ps.setString(3, d[0]);

            int rows = ps.executeUpdate();

            if (rows > 0)
                send(ex, "{\"message\":\"Faculty updated successfully\"}");
            else
                send(ex, "{\"error\":\"Faculty ID not found\"}");

        } catch (Exception e) {
            send(ex, "{\"error\":\"" + e.getMessage() + "\"}");
        }
    }
}

// DELETE FACULTY -----------------------------------
static class DeleteFaculty implements HttpHandler {
    public void handle(HttpExchange ex) throws IOException {
        if (!ex.getRequestMethod().equalsIgnoreCase("DELETE")) {
            send(ex, "{\"error\":\"Only DELETE allowed\"}");
```

```
            return;
        }

        String query = ex.getRequestURI().getQuery(); // fid=1001
        if (query == null || !query.startsWith("fid=")) {
            send(ex, "{\"error\":\"Missing faculty ID\"}");
            return;
        }

        String fid = query.replace("fid=","");

        try (Connection con = connect();
             PreparedStatement ps = con.prepareStatement(
                    "DELETE FROM tbl_faculty WHERE faculty_id=?"
             )) {

            ps.setString(1, fid);
            int rows = ps.executeUpdate();

            if (rows > 0)
                send(ex, "{\"message\":\"Faculty deleted successfully\"}");
            else
                send(ex, "{\"error\":\"Faculty ID not found\"}");

        } catch (Exception e) {
            send(ex, "{\"error\":\"" + e.getMessage() + "\"}");
        }
    }
}

// SEARCH BY SPECIALIZATION NAME -----------------------
```

```
static class SearchFaculty implements HttpHandler {
public void handle(HttpExchange ex) throws IOException {

    if (ex.getRequestMethod().equalsIgnoreCase("OPTIONS")) {
        send(ex, "{}");
        return;
    }

    if (!ex.getRequestMethod().equalsIgnoreCase("GET")) {
        send(ex, "{\"error\":\"Only GET allowed\"}");
        return;
    }

    String query = ex.getRequestURI().getQuery(); // spec=ai
    if (query == null || !query.startsWith("spec=")) {
        send(ex, "{\"error\":\"Missing specialization keyword\"}");
        return;
    }

    String specKeyword = query.replace("spec=", "");

    String sql =
        "SELECT f.faculty_id, f.name, f.email, f.department_id, f.designation, f.phone " +
        "FROM tbl_faculty f " +
        "JOIN tbl_faculty_specialization fs ON f.faculty_id = fs.faculty_id " +
        "JOIN tbl_specialization s ON fs.specialization_id = s.specialization_id " +
        "WHERE LOWER(s.specialization_name) LIKE LOWER(?)";

    StringBuilder response = new StringBuilder("[");
    boolean first = true;
```

```
        try (Connection con = connect();
             PreparedStatement ps = con.prepareStatement(sql)) {

            ps.setString(1, "%" + specKeyword + "%");

            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                if (!first) response.append(",");
                response.append("{")
                        .append("\"faculty_id\":\"").append(rs.getString("faculty_id")).append("\",")
                        .append("\"name\":\"").append(rs.getString("name")).append("\",")
                        .append("\"email\":\"").append(rs.getString("email")).append("\",")
                        .append("\"department_id\":\"").append(rs.getString("department_id")).append("\",")
                        .append("\"designation\":\"").append(rs.getString("designation")).append("\",")
                        .append("\"phone\":\"").append(rs.getString("phone")).append("\"")
                        .append("}");
                first = false;
            }

            response.append("]");
            send(ex, response.toString());

        } catch (Exception e) {
            send(ex, "{\"error\":\"" + e.getMessage() + "\"}");
        }
    }
}

    // Response helper
    static void send(HttpExchange ex, String msg) throws IOException {
```

```
    ex.getResponseHeaders().add("Access-Control-Allow-Origin", "*");
    ex.getResponseHeaders().add("Access-Control-Allow-Methods",
"GET,POST,PUT,DELETE,OPTIONS");
    ex.getResponseHeaders().add("Access-Control-Allow-Headers", "Content-Type");
    ex.sendResponseHeaders(200, msg.length());
    OutputStream os = ex.getResponseBody();
    os.write(msg.getBytes());
    os.close();
}
}
```

Faculty.html:

```
<!DOCTYPE html>
<html>
<head>
<title>Faculty CRUD</title>
<style>
body { font-family: Arial; width: 70%; margin: auto; }
input, select, button { padding: 6px; margin: 4px; }
table { border-collapse: collapse; width: 100%; margin-top: 15px; }
td, th { border: 1px solid gray; padding: 6px; }
th { background: #eee; }
</style>
</head>
<body>

<h2>➕ Add Faculty</h2>
<input id="fid" placeholder="Faculty ID">
<input id="fname" placeholder="Name">
<input id="femail" placeholder="Email">
```

```
<input id="phone" placeholder="Phone (10-15 digits)"
      oninput="this.value=this.value.replace(/[^0-9]/g,'')" maxlength="15">

<!-- department_id as text input -->
<input id="dept" placeholder="Department ID">

<!-- designation dropdown stays (constraint) -->
<select id="desig">
  <option value="Professor">Professor</option>
  <option value="Associate Professor">Associate Professor</option>
  <option value="Assistant Professor">Assistant Professor</option>
  <option value="Lecturer">Lecturer</option>
  <option value="Research Fellow">Research Fellow</option>
</select>

<!-- specialization_id as text input -->
<input id="spec" placeholder="Specialization ID">

<button onclick="addFaculty()">Add</button>

<hr>

<h2>✏ Update Faculty</h2>
<input id="ufid" placeholder="Faculty ID">
<input id="unewphone" placeholder="New Phone"
      oninput="this.value=this.value.replace(/[^0-9]/g,'')" maxlength="15">
<select id="unewdesig">
  <option value="Professor">Professor</option>
  <option value="Associate Professor">Associate Professor</option>
  <option value="Assistant Professor">Assistant Professor</option>
  <option value="Lecturer">Lecturer</option>
```

```
  <option value="Research Fellow">Research Fellow</option>
</select>
<button onclick="updateFaculty()">Update</button>

<hr>

<h2>🗑 Delete Faculty</h2>
<input id="dfid" placeholder="Faculty ID">
<button onclick="deleteFaculty()">Delete</button>

<hr>

<h2>🔍 Search Faculty by Specialization Name</h2>
<input id="searchSpec" placeholder="e.g., ai, ml, data">
<button onclick="searchFaculty()">Search</button>

<hr>

<h2>📋 Faculty List</h2>
<button onclick="loadAll()">Load All Faculty</button>

<table id="facultyTable">
<tr>
<th>ID</th><th>Name</th><th>Email</th><th>Dept</th>
<th>Designation</th><th>Phone</th><th>Spec</th>
</tr>
</table>

<script>
// GET all
function loadAll() {
```

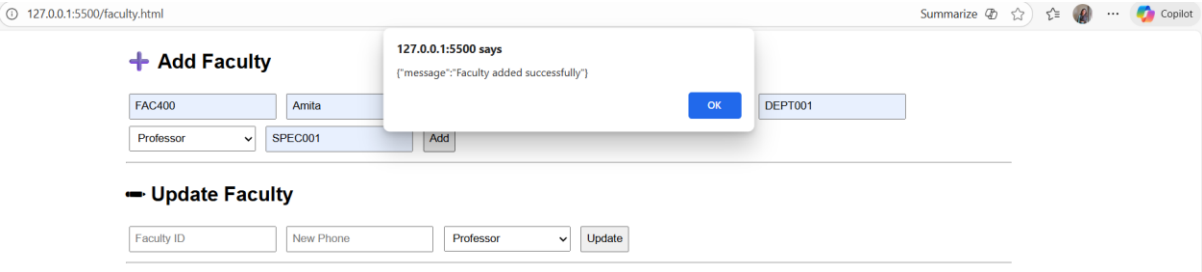
```
  fetch("http://localhost:8080/faculty")
  .then(res => res.json())
  .then(data => {
    let t = document.getElementById("facultyTable");
    t.innerHTML = `<tr>
      <th>ID</th><th>Name</th><th>Email</th><th>Dept</th>
      <th>Designation</th><th>Phone</th><th>Spec</th>
    </tr>`;
    data.forEach(f => {
      t.innerHTML += `<tr>
        <td>${f.faculty_id}</td>
        <td>${f.name}</td>
        <td>${f.email}</td>
        <td>${f.department_id}</td>
        <td>${f.designation}</td>
        <td>${f.phone}</td>
        <td>${f.specialization_id}</td>
      </tr>`;
    });
  });
}

// ADD
function addFaculty() {
  let body =
    fid.value + "," +
    fname.value + "," +
    femail.value + "," +
    dept.value + "," +
    desig.value + "," +
    spec.value + "," +
```

```
        phone.value;

    fetch("http://localhost:8081/add", { method:"POST", body })
    .then(res => res.text())
    .then(alert);
}

// UPDATE
function updateFaculty() {
    let body =
        ufid.value + "," +
        unewphone.value + "," +
        unewdesig.value;

    fetch("http://localhost:8081/update", { method:"PUT", body })
    .then(res => res.text())
    .then(alert);
}

// DELETE
function deleteFaculty() {
    let fid = document.getElementById("dfid").value;
    fetch(`http://localhost:8081/delete?fid=${fid}`, { method:"DELETE" })
    .then(res => res.text())
    .then(alert);
}

// SEARCH
function searchFaculty() {
    let q = searchSpec.value;
    fetch(`http://localhost:8081/search?spec=${q}`)
```

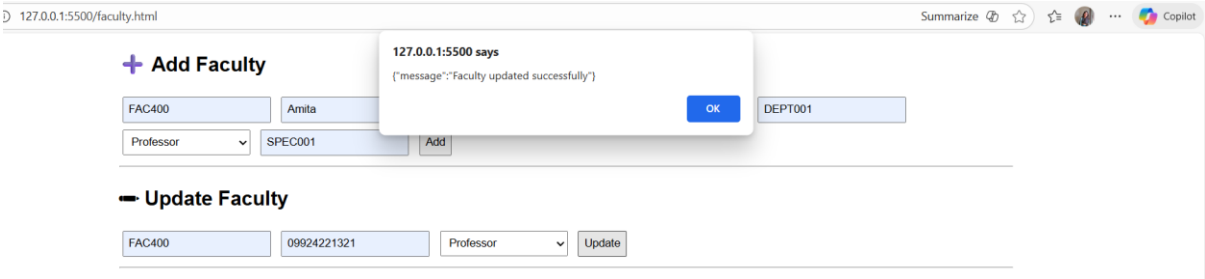
```
  .then(res => res.json())
  .then(data => {
    console.log(data);
    alert("Search returned: " + data.length + " results");
    let t = document.getElementById("facultyTable");
    t.innerHTML = `<tr>
      <th>ID</th><th>Name</th><th>Email</th><th>Dept</th>
      <th>Designation</th><th>Phone</th><th>Spec</th>
    </tr>`;

    data.forEach(f => {
      t.innerHTML += `<tr>
        <td>${f.faculty_id}</td>
        <td>${f.name}</td>
        <td>${f.email}</td>
        <td>${f.department_id}</td>
        <td>${f.designation}</td>
        <td>${f.phone}</td>
        <td>(match)</td>
      </tr>`;
    });
  });
}
</script>

</body>
</html>
```

Snip Shorts:

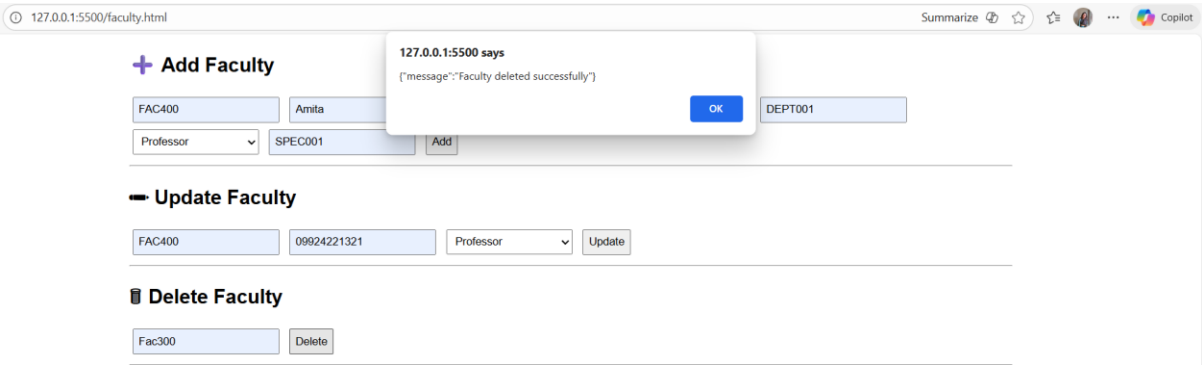
Insertion:



Update:



Delete:

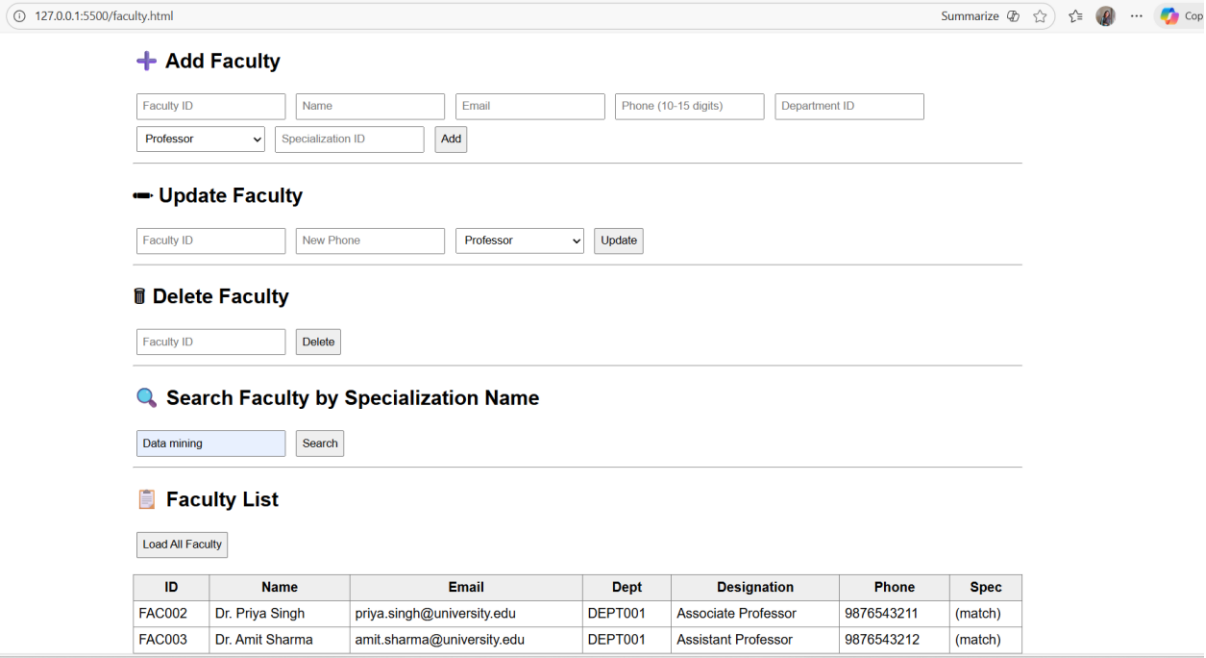


View:

127.0.0.1:5500/faculty.html

FAC128	Dr. Nandini Desai	nandini.desai@university.edu	DEPT087	Associate Professor	9876543337	SPEC014
FAC129	Dr. Kiran Kumar	kiran.kumar2@university.edu	DEPT091	Professor	9876543338	SPEC071
FAC130	Dr. Anita Rao	anita.rao2@university.edu	DEPT092	Associate Professor	9876543339	SPEC071
FAC131	Dr. Nitin Agarwal	nitin.agarwal@university.edu	DEPT084	Professor	9876543340	SPEC008
FAC132	Dr. Ritika Sharma	ritika.sharma@university.edu	DEPT085	Associate Professor	9876543341	SPEC009
FAC133	Dr. Siddharth Mishra	siddharth.mishra@university.edu	DEPT066	Professor	9876543342	SPEC103
FAC134	Dr. Ramya Krishnan	ramya.krishnan@university.edu	DEPT075	Professor	9876543343	SPEC009
FAC135	Dr. Ajay Menon	ajay.menon@university.edu	DEPT026	Associate Professor	9876543344	SPEC088
FAC136	Dr. Nisha Pillai	nisha.pillai@university.edu	DEPT027	Professor	9876543345	SPEC091
FAC137	Dr. Suresh Kumar	suresh.kumar2@university.edu	DEPT029	Associate Professor	9876543346	SPEC084
FAC138	Dr. Harish Patel	harish.patel@university.edu	DEPT042	Professor	9876543347	SPEC088
FAC139	Dr. Vivek Gupta	vivek.gupta2@university.edu	DEPT043	Associate Professor	9876543348	SPEC088
FAC140	Dr. Sunita Nair	sunita.nair2@university.edu	DEPT044	Professor	9876543349	SPEC088
FAC141	Dr. Shobha Kumar	shobha.kumar@university.edu	DEPT045	Associate Professor	9876543350	SPEC087
FAC142	Dr. Amit Singh	amit.singh2@university.edu	DEPT048	Professor	9876543351	SPEC120
FAC143	Dr. Rahul Mehta	rahul.mehta@university.edu	DEPT049	Associate Professor	9876543352	SPEC109
FAC144	Dr. Kavya Shah	kavya.shah@university.edu	DEPT050	Professor	9876543353	SPEC113
FAC145	Dr. Karan Joshi	karan.joshi@university.edu	DEPT106	Professor	9876543354	SPEC076
FAC146	Dr. Simran Kulkarni	simran.kulkarni@university.edu	DEPT107	Associate Professor	9876543355	SPEC126
FAC147	Dr. Aakash Malhotra	aakash.malhotra@university.edu	DEPT108	Professor	9876543356	SPEC076
FAC148	Dr. Tanvi Agarwal	tanvi.agarwal@university.edu	DEPT109	Associate Professor	9876543357	SPEC072
FAC149	Dr. Ishaan Bansal	ishaan.bansal@university.edu	DEPT110	Professor	9876543358	SPEC076
FAC150	Dr. Diya Chopra	diya.chopra@university.edu	DEPT111	Professor	9876543359	SPEC076
FAC400	Amita	amitaaa@gmail.com	DEPT001	Professor	09924221321	SPEC001

USP Based: Search for Faculty based on specialization



Changes made:

1. Our SRS lacked unique selling proposition, and thus added the same after being guided about.
2. Rest of the changes were in ERD which are mentioned in the document.