***DBMS OF Pakistan Science and Technology***

**Database Design Document**

**V 3.0**

**By**

|  |  |  |
| --- | --- | --- |
| *1* | *Anam Fatima* | *NUM-BSCS-2022-01* |
| *2* | *Sultan Haider* | *NUM-BSCS-2022-48* |
| *3* | *Abdul Qadeer* | *NUM-BSCS-2022-56* |



**Department of Computer Sciences**

**Namal University**

**Mianwali, Pakistan**

**Submission Date: 30th June, 2024**

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Approved by** |
| 10/06/24 | V 2.1 | Removed Associative Entity SoftwareHouseTelecomProvider | Mam ASIYA |
| 10/06/24 | V2.2 | Therelationship between Software House and Telecommunication is changed to One-to Many | Mam ASIYA |
| 10/06/24 | V2.3 | The relationship between software House and software development companies is changed to many to many and thus added associative entity Software Projects | Mam ASIYA |
| 10/06/24 | V2.4 | Improved the structure of ERD | Mam ASIYA |
| 22/04/24 | V 1.1 | - Corrected indentation on title page | Mam ASIYA |
| 22/04/24 | V 1.2 | - Removed one logo from the title page | Mam ASIYA |
| 22/04/24 | V 1.3 | - Specified the type of data the project will store in the introduction | Mam ASIYA |
| 22/04/24 | V 1.4 | -removed objective point 3 in which we were providing the recommendations.  -removed the further contribution line from objective point 5 in the field of science and tech | Mam ASIYA |
| 22/04/24 | V 1.5 | - Removed Recommendation Generation from project scope | Mam ASIYA |
| 22/04/24 | V 1.6 | - Removed Visualization Tool from project scope | Mam ASIYA |
| 22/04/24 | V 1.7 | - Improved System Functionality | Mam ASIYA |

*.*

**TABLE OF CONTENTS**

[CHAPTER 1: PROJECT OVERVIEW 4](#_Toc170133991)

[1.1. INTRODUCTION: 4](#_Toc170133992)

[1.2. PROBLEM STATEMENT: 4](#_Toc170133993)

[1.3. PROJECT OBJECTIVES: 4](#_Toc170133994)

[1.4. DOCUMENT OBJECTIVES: 4](#_Toc170133995)

[CHAPTER 2: DETAILED DATABASE DESIGN 5](#_Toc170133996)

[2.1. ENTITY: 5](#_Toc170133998)

[2.2. DATA DICTIONARY: 5](#_Toc170133999)

[2.3. RELATIONSHIPS: 5](#_Toc170134000)

[2.4. ENTITY RELATIONSHIP DIAGRAM: 6](#_Toc170134001)

[CHAPTER 3 : Logical DATABASE DESIGN 7](#_Toc170134002)

[3.1. RELATIONAL SCHEMA: 7](#_Toc170134003)

[3.2. FUNCTIONAL DEPENDENCIES: 7](#_Toc170134004)

[3.3. NORMALIZATION: 7](#_Toc170134005)

[CHAPTER 4 : Physical DATABASE DESIGN 8](#_Toc170134006)

[4.1. STRUCTURE OF THE TABLES: 8](#_Toc170134007)

[4.2. DATA SAMPLES INSIDE TABLES: 8](#_Toc170134008)

[4.3. QUERIES RESULTS: 8](#_Toc170134009)

[CHAPTER 5 : Interface Design 9](#_Toc170134010)

[5.1. LANGUAGE/FRAMEWORK: 9](#_Toc170134011)

[5.2. DATABASE CONNECTIVITY: 9](#_Toc170134012)

[5.3. STORED PROCEDURES AND FUNCTIONS: 9](#_Toc170134013)

[5.4. INTERFACES: 9](#_Toc170134014)

[CHAPTER 6 : CONCLUSION 10](#_Toc170134015)

[6.1. LESSONS LEARNED: 10](#_Toc170134016)

[6.2. CHALLENGES AND SOLUTIONS: 10](#_Toc170134017)

[6.3. FUTURE WORK AND IMPROVEMENTS: 10](#_Toc170134018)

[6.4. FINAL THOUGHTS: 10](#_Toc170134019)

[REFERENCES 11](#_Toc170134020)

# PROJECT OVERVIEW

## INTRODUCTION:

## Our project aims to develop a specialized Database Management System (DBMS) tailored for Pakistan's science and technology sector, leveraging MySQL and Python for efficient data management and predictive analytics. Key datasets include Tech Indicators, Software Houses, Software Development Companies, Freelancers, Telecommunication Statistics, Telecommunication, Scientists, Software House Telecom Provider, and Freelancer Company Assignment. [1] [2] By analyzing these datasets, our DBMS will provide valuable insights, fostering progress in Pakistan's science and technology landscape.

## PROBLEM STATEMENT:

There is a lack of specialized DBMS solutions tailored to the needs of science and technology domains, hindering efficient data management and analysis. Additionally, the absence of predictive analytics regarding Pakistan's future in science and technology limits strategic planning in this crucial sector.

## PROJECT OBJECTIVES:

1. Develop a user-friendly DBMS for science and technology data management.
2. Predict Pakistan's future standing in science and technology based on data analysis.
3. Develop a comprehensive database system to catalog and organize information on scientific research, technological advancements, and innovations across various domains.
4. This database should facilitate efficient storage, retrieval, and analysis of data, enabling technologists to access relevant information.

## DOCUMENT OBJECTIVES:

1. **Clarity:** Ensure that the documentation clearly articulates the goals, scope, and significance of the database project to stakeholders and team members.
2. **Conciseness:** Present the information in a concise manner, avoiding unnecessary details while covering all essential aspects of the project.
3. **Coherence:** Organize the documentation in a logical and coherent manner, facilitating easy navigation and understanding for readers.
4. **Completeness:** Ensure that all relevant aspects of the project, including data description, system architecture, functional and non-functional requirements, implementation plan, and testing approach, are adequately addressed.
5. **Accessibility:** Make the documentation accessible to a diverse audience, including project developers and other interested parties(instructor, groupmates), by using clear language and avoiding technical jargon whenever possible.

# DETAILED DATABASE DESIGN



## ENTITY:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Entity Name** | **Description** |
| 01 | Tech Indicators | Tech Indicators is a collection of data representing scientific and technological indicators along with their values across Pakistan in different years. |
| 02 | Software Houses | Software house is a registered software development center operating within Pakistan showing the description of these centers. |
| 03 | Software Development Companies | Software Development Companies are an established software development firms with ratings, locations, and notable client lists, documented within the system. |
| 04 | Freelancers | Freelancers are individuals offering their services independently within the Pakistani market. |
| 05 | Telecommunication Statistics | Telecommunication statistics is the collection of telecommunications metrics over time in Pakistan. |
| 06 | Telecommunication | Telecommunication is a monthly usage statistics associated with major telecom providers operating in Pakistan. |
| 07 | Scientist | Scientists are the individuals recognized within the system for their contributions to science, affiliated with specific disciplines and institutions within Pakistan. |
| 08 | Software Project (Associative) | Software Project act as a linker between the software houses and the software development companies which assigns projects to various Software Houses. |
| 09 | Freelancer Company Assignment (Associative) | Freelancer Company Assignment acts as a linker that establishes connections between individual freelancers and the companies they work for or collaborate with. |

## DATA DICTIONARY:

* + 1. **Tech Indicators:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Technology ID (Primary Key) | Number | NOT NULL | Technology ID of the indicator and year |
| 02 | Indicator | Text | None | The name of scientific and technological indicator |
| 03 | Indicator Code | Text | None | A unique Code identifying the indicator |
| 04 | Value | Number | None | The value of indicator in specific year |
| 05 | Year | Number | BETWEEN 1990 to 2022 | The year in which data is measured |

* + 1. **Software Houses:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Software House ID (Primary Key) | Number | NOT NULL | Software House ID of each software house |
| 02 | Name | Text | Between 1 and 70 | The name of the sotware house |
| 03 | Description | Text | None | A one-liner description of each software house |

* + 1. **Software Development Companies:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Company ID (Primary Key) | Number | NOT NULL | The ID of the software development companies |
| 02 | Company Name | Text | Between 1 and 70 | The name of the software development company associated with the ID |
| 03 | Rating | Number | Between 0 and 5 | The rating or reputation of the company |
| 04 | Location | Text | Any constraint | The location of the company (city) |
| 05 | Company Clients | Text | Multi valued attribute | The clients with which the company is associated |

* + 1. **Freelancers:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Freelancer ID (Primary Key) | Number | NOT NULL | Freelancer ID of each individual |
| 02 | Name | Text | Between 1 and 70 | The name of the person |
| 03 | Profession | Text | Multi valued attribute | The profession or expertise o the freelancer |
| 04 | Reviews | Number | None | The total numbers of reviews or feedback received from the clients |
| 05 | Hourly Rate | Currency | Must be In Dollars | The hourly rate charged by the freelancer |

* + 1. **Telecommunication Statistics:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | TS ID (Primary Key) | Number | NOT NULL | The ID of each statistic |
| 02 | Year | Number | Between 2010 to 2020 | The year in which the statistics are measured |
| 03 | Cellular Mobile | Number | NONE | Statistics related to cellular mobile services |
| 04 | Wireless Local | Number | NONE | Statistics related to wireless local telecommunication services |

* + 1. **Telecommunication:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Telecommunication ID (Primary Key) | Number | NOT NULL | Telecommunication ID of the given month and service provider |
| 02 | Month | Text | January-December | The month for which statistics are recorded |
| 03 | Jazz | Number | NONE | Usage statistics associated with Jazz telecom provider. |
| 04 | Zong | Number | NONE | Usage statistics associated with each Ufone provider. |
| 05 | Ufone | Number | NONE | Usage statistics associated with Ufone telecom provider. |
| 06 | Telenor | Number | NONE | Usage statistics associated with each Telenor provider. |
| 07 | Warid | Number | NONE | Usage statistics associated with Warid telecom provider. |

* + 1. **Scientists:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Scientist ID (Primary Key) | Number | NOT NULL | Unique Id of each scientist |
| 02 | Name | Text | NONE | The name of scientist |
| 03 | Discipline | Text | NONE | Scientific discipline or field of expertise |
| 04 | Affiliation | Text | NONE | Institution with which scientist is affiliated |
| 05 | Qualification | Text | NONE | Academic Qualification or degree of each scientist |

* + 1. **Software Project (Associative)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Software Project ID (Primary Key) | Number | NOT NULL | Unique ID |
| 02 | Project Type | text | NONE | Type of project assigned by the software company to software house |

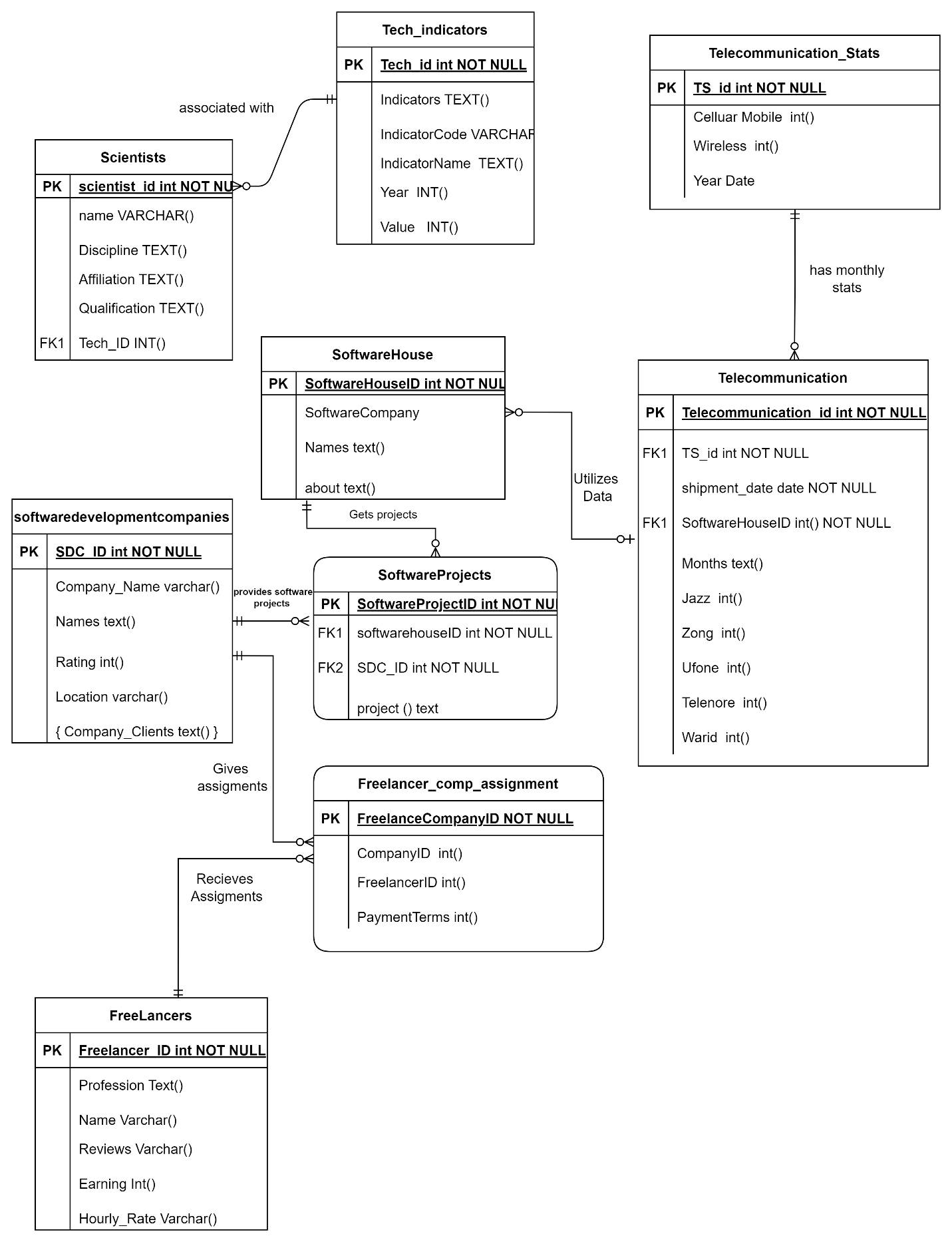
* + 1. **Freelancer Company Assignment (Associative)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No** | **Name** | **Data Type** | **Constraint** | **Description** |
| 01 | Freelance\_Company ID (Primary Key) | Number | NOT NULL | Unique ID |
| 02 | Payment Terms | Currency | Must be In Dollars | The total earnings of the freelancer |

## RELATIONSHIPS:

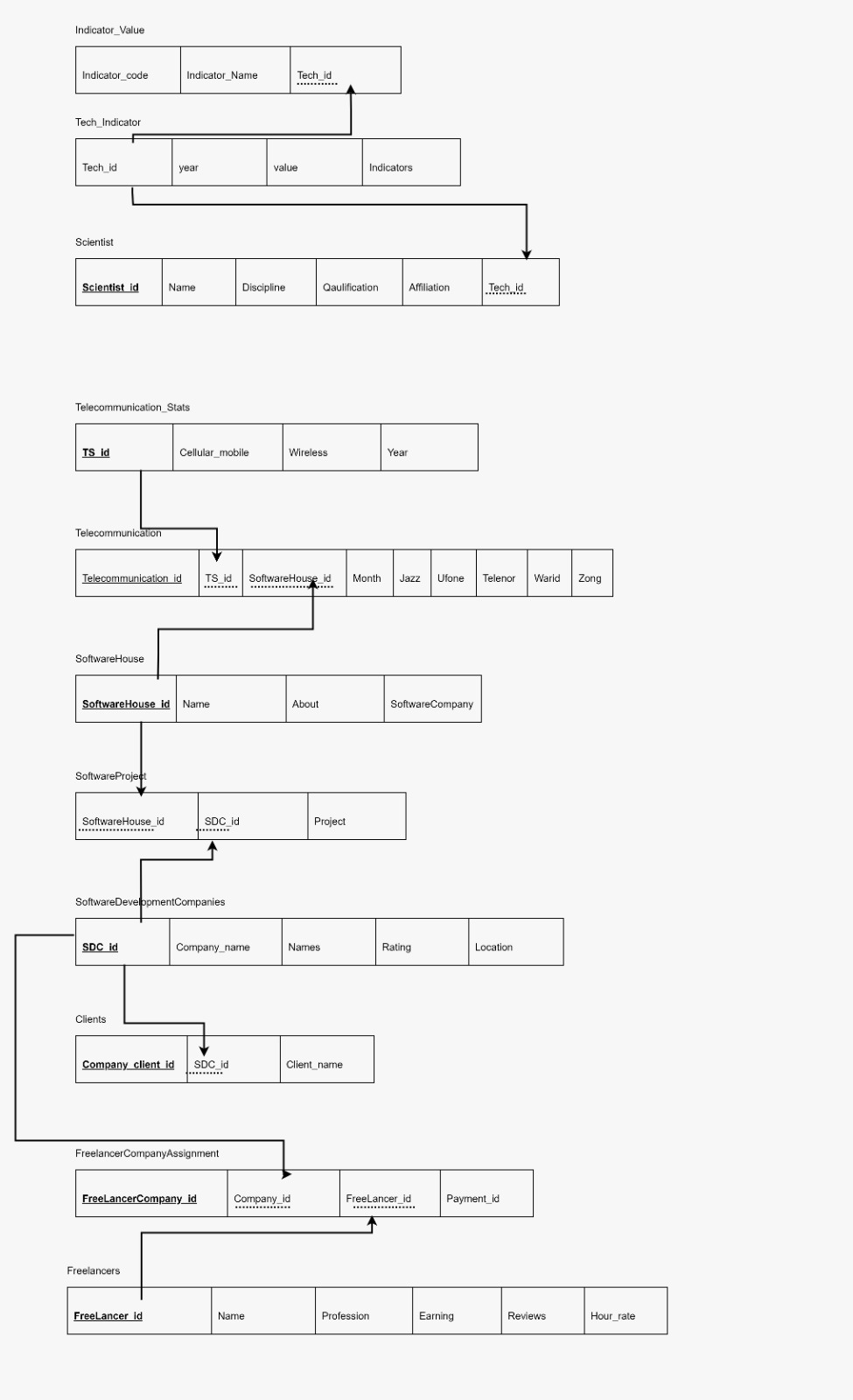
|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Participating Entities** | **Relation** | **Business Rule** |
| 01 | Software Houses, Software Development Companies | Software Development Companies provides projects to Software Houses | One Software Development company provides project to many software houses. One software house can get project from multiple Software Development company. |
| 02 | Telecommunication, Telecommunication statistics | Telecommunication relates monthly Telecommunication statistics | One statistic is of one Telecommunication. One telecommunication can have more than one statistics. |
| 03 | Tech Indicators, Scientist | Tech Indicators associates Scientist data | Multiple scientists can be associated with the same indicator. Each individual scientist record is linked to only one indicator |
| 04 | Freelancer, Software Development Companies | Freelancer interacts with Software Development Companies | One Freelancer (individual) can have multiple assignments by multiple Software Development Companies. Many software development companies give assignments to various freelancers. |
| 05 | Software House , Telecommunication | Telecommunication represents Internet usage by the software houses | One software house utilizes communication and internet from many telecommunication companies and have records of many months. Many telecommunication record represents one software house.. |

## ENTITY RELATIONSHIP DIAGRAM:



# : Logical DATABASE DESIGN

* 1. **RELATIONAL SCHEMA:**



* 1. **FUNCTIONAL DEPENDENCIES:**

The functional dependencies for each table.

1. **Tech\_Indicator**:
   * TechID → year, Value, Indicator\_Code
2. **Indicator\_Value**:
   * Indicator\_Code → Indicator\_Name
3. **Scientist**:
   * ScientistID → Name, Discipline, Qualification, Affiliation, TechID
   * TechID → Tech\_Indicator (Foreign Key Dependency)
4. **Telecommunication\_Stats**:
   * TSID → Cellular\_Mobile, Wireless, Year
5. **SoftwareHouse**:
   * SoftwareHouse\_ID → Name, About, SoftwareCompany
6. **Telecommunication**:
   * Telecommunication\_id → TSID, SoftwareHouse\_ID, Month, Jazz, Ufone, Telenor, Warid, Zong
   * TSID → Telecommunication\_Stats (Foreign Key Dependency)
   * SoftwareHouse\_ID → SoftwareHouse (Foreign Key Dependency)
7. **SoftwareDevelopmentCompanies**:
   * SDC\_ID → Company\_Name, Rating, Location
8. **SoftwareProject**:
   * SoftwareHouse\_ID, SDC\_ID → Project
   * SoftwareHouse\_ID → SoftwareHouse (Foreign Key Dependency)
   * SDC\_ID → SoftwareDevelopmentCompanies (Foreign Key Dependency)
9. **Clients**:
   * CompanyClient\_ID → SDC\_ID, Client\_Name
   * SDC\_ID → SoftwareDevelopmentCompanies (Foreign Key Dependency)
10. **Freelancers**:
    * Freelancer\_ID → Name, Profession, Earning, Reviews, Hour\_Rate
11. **FreelancerCompanyAssignment**:
    * FreelancerCompany\_ID → Company\_ID, Freelancer\_ID, Payment
    * Freelancer\_ID → Freelancers (Foreign Key Dependency)
    * Company\_ID → SoftwareDevelopmentCompanies (Foreign Key Dependency)

*.*

* 1. **NORMALIZATION:**

Let's check each table for these normal forms:

1. **Tech\_Indicator**: Already in 3NF.
   * Primary Key: TechID
   * Transitive dependeny of IndicatorValue on IndicatorCode
2. **Scientist**: Already in 3NF.
   * Primary Key: ScientistID
   * No partial or transitive dependencies.
3. **Telecommunication\_Stats**: Already in 3NF.
   * Primary Key: TSID
   * No partial or transitive dependencies.
4. **SoftwareHouse**: Already in 3NF.
   * Primary Key: SoftwareHouse\_ID
   * No partial or transitive dependencies.
5. **Telecommunication**: Already in 3NF.
   * Primary Key: Telecommunication\_id
   * No partial or transitive dependencies.
6. **SoftwareDevelopmentCompanies**: Already in 3NF.
   * Primary Key: SDC\_ID
   * Repetition due to Company Clients
7. **SoftwareProject**: Already in 3NF.
   * Primary Key: (SoftwareHouse\_ID, SDC\_ID)
   * No partial or transitive dependencies.
8. **Freelancers**: Already in 3NF.
   * Primary Key: Freelancer\_ID
   * No partial or transitive dependencies.
9. **FreelancerCompanyAssignment**: Already in 3NF.
   * Primary Key: FreelancerCompany\_ID
   * No partial or transitive dependencies.

To normalize the tables, we need to ensure they are in 1NF, 2NF, and 3NF.

**First Normal Form (1NF):**

* Ensure that all attributes have atomic values.

There is repetition in the table SoftwareDevelopmentCompanies due to the Multi-Valued attribute **Company clients.** Hence, to remove the redundancy we will split the table in 2 tables as:

**SoftwareDevelopmentCompany**

|  |  |  |  |
| --- | --- | --- | --- |
| **SDCID** | CompanyName | Rating | Location |

**Clients**

|  |  |  |
| --- | --- | --- |
| **CompanyClientID** | **\_ \_SDCID\_ \_** | ClientName |

Now all the tables are in First natural Form.

**Second Normal Form (2NF):**

* Ensure that all non-key attributes are fully functionally dependent on the primary key.
* Removes partial dependencies.
* There is no partial Depedency so Every relation is in Second Normal Form.

**Third Normal Form (3NF):**

There is Transitive Depedency of Indicator value on Indicator in the Tech\_Indicator Tabe. Hence, to remove the transitive dependency we will split in 2 tables as follow:

**Tech\_Indicaor**

|  |  |  |  |
| --- | --- | --- | --- |
| **TechID** | Year | Value | Indicator\_Code |

**Indicator\_Value**

|  |  |
| --- | --- |
| **Indicator\_Code** | IndicatorName |

Now all the tables are in Third natural Form.

**NORMALIZED TABLE:**

Now we have the normalized forms of each table as follow:

1. **Tech\_Indicator**: Already in 3NF.
   * Primary Key: TechID
   * No partial or transitive dependencies.
2. **Indicator\_Value**: Already in 3NF.
   * Primary Key: Indicator\_Code
   * No partial or transitive dependencies.
3. **Scientist**: Already in 3NF.
   * Primary Key: ScientistID
   * No partial or transitive dependencies.
4. **Telecommunication\_Stats**: Already in 3NF.
   * Primary Key: TSID
   * No partial or transitive dependencies.
5. **SoftwareHouse**: Already in 3NF.
   * Primary Key: SoftwareHouse\_ID
   * No partial or transitive dependencies.
6. **Telecommunication**: Already in 3NF.
   * Primary Key: Telecommunication\_id
   * No partial or transitive dependencies.
7. **SoftwareDevelopmentCompanies**: Already in 3NF.
   * Primary Key: SDC\_ID
   * No partial or transitive dependencies.
8. **SoftwareProject**: Already in 3NF.
   * Primary Key: (SoftwareHouse\_ID, SDC\_ID)
   * No partial or transitive dependencies.
9. **Clients**: Already in 3NF.
   * Primary Key: CompanyClient\_ID
   * No partial or transitive dependencies.
10. **Freelancers**: Already in 3NF.
    * Primary Key: Freelancer\_ID
    * No partial or transitive dependencies.
11. **FreelancerCompanyAssignment**: Already in 3NF.
    * Primary Key: FreelancerCompany\_ID
    * No partial or transitive dependencies.

### **3NF Relations**

After ensuring that all tables are in 3NF, we have the following relations:

1. **Tech\_Indicator (TechID, year, Value, Indicator\_Code)**
2. **Indicator\_Value (Indicator\_Code, Indicator\_Name)**
3. **Scientist (ScientistID, Name, Discipline, Qualification, Affiliation, TechID)**
4. **Telecommunication\_Stats (TSID, Cellular\_Mobile, Wireless, Year)**
5. **SoftwareHouse (SoftwareHouse\_ID, Name, About, SoftwareCompany)**
6. **Telecommunication (Telecommunication\_id, TSID, SoftwareHouse\_ID, Month, Jazz, Ufone, Telenor, Warid, Zong)**
7. **SoftwareDevelopmentCompanies (SDC\_ID, Company\_Name, Rating, Location)**
8. **SoftwareProject (SoftwareHouse\_ID, SDC\_ID, Project)**
9. **Clients (CompanyClient\_ID, SDC\_ID, Client\_Name)**
10. **Freelancers (Freelancer\_ID, Name, Profession, Earning, Reviews, Hour\_Rate)**
11. **FreelancerCompanyAssignment (FreelancerCompany\_ID, Company\_ID, Freelancer\_ID, Payment)**

These tables are now normalized to 3NF, ensuring there are no partial or transitive dependencies, and all functional dependencies are properly addressed.

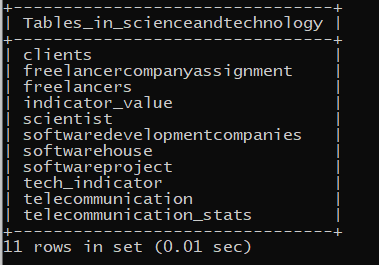
# : Physical DATABASE DESIGN

* 1. **STRUCTURE OF THE TABLES:**

### Querry:

SHOW TABLES;

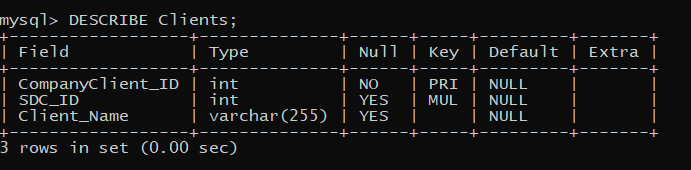
### Output:



### Querry:

DESCRIBE Clients;

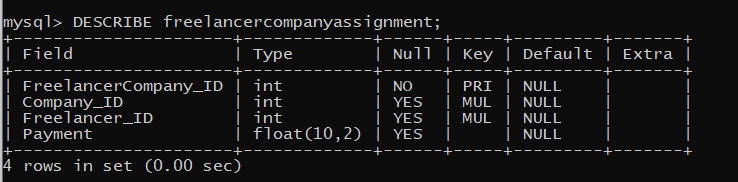
### Output:



### Querry:

DESCRIBE freelancercompanyassignment;

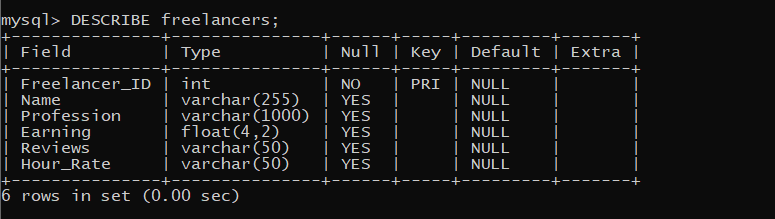
### Output:



### Querry:

DECRIBE freelancers;

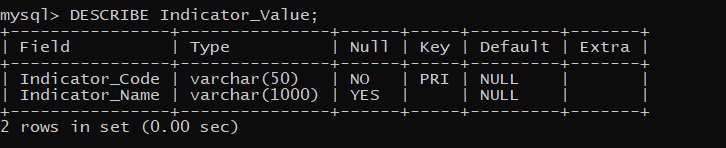
### Output:



### Querry:

DESCRIBE Indicator\_Value;

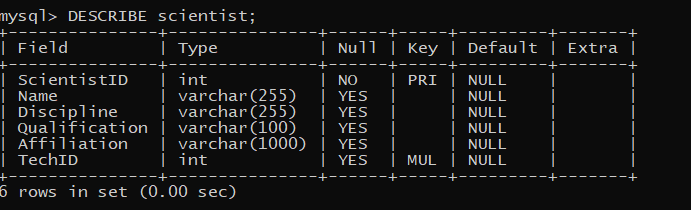
### Output:



### Querry:

DESCRIBE Scientist;

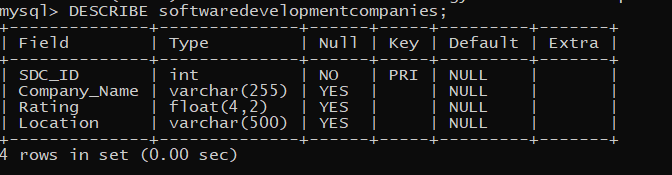
### Output:



### Querry:

DESCRIBE SoftwareDevelopmentCompanies;

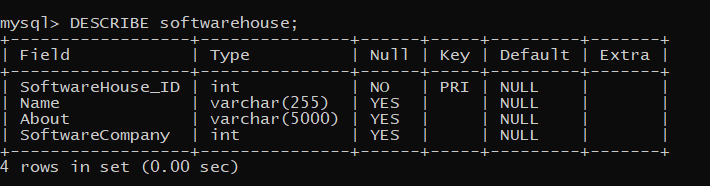
### Output:



### Querry:

DESCRIBE SoftwareHouse;

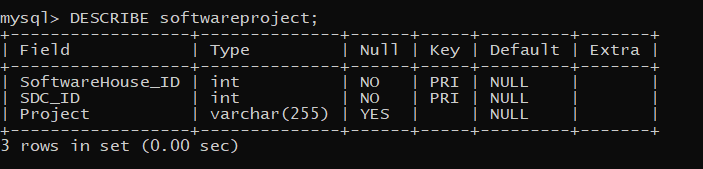
### Output:



### Querry:

DESCRIBE SoftwareProject;

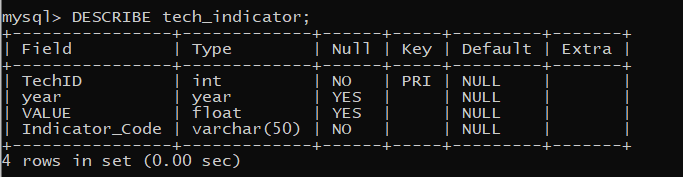
### Output:



### Querry:

DESCRIBE tech\_indicator;

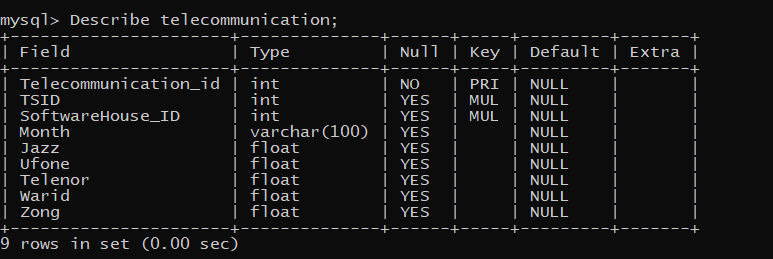
### Output:



Querry:

DESCRIBE Telecommunication;

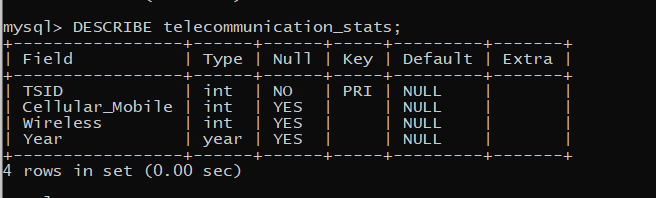
### Output:



Querry:

DESCRIBE Telecommunication\_Stats;

### Output:

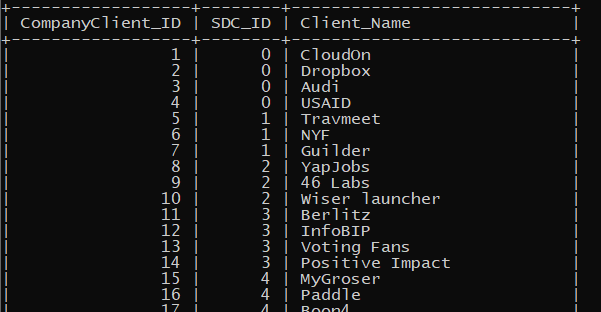


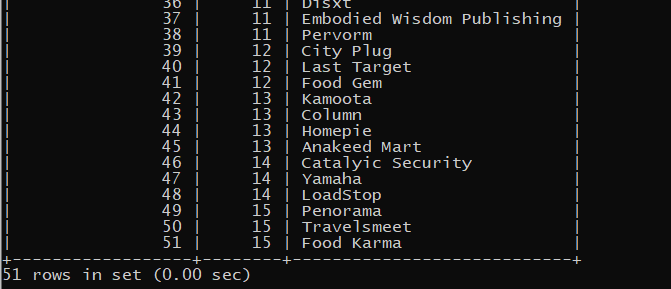
* 1. **DATA SAMPLES INSIDE TABLES:**

### Querry:

SELECT \* FROM Clients;

### Output:

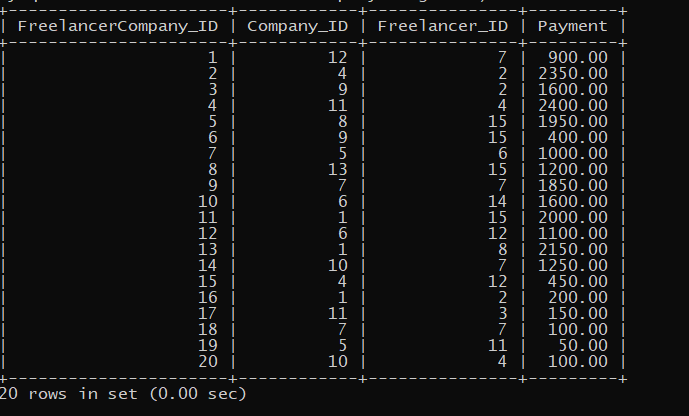




### Querry:

SELECT \* FROM freelancercompanyassignment;

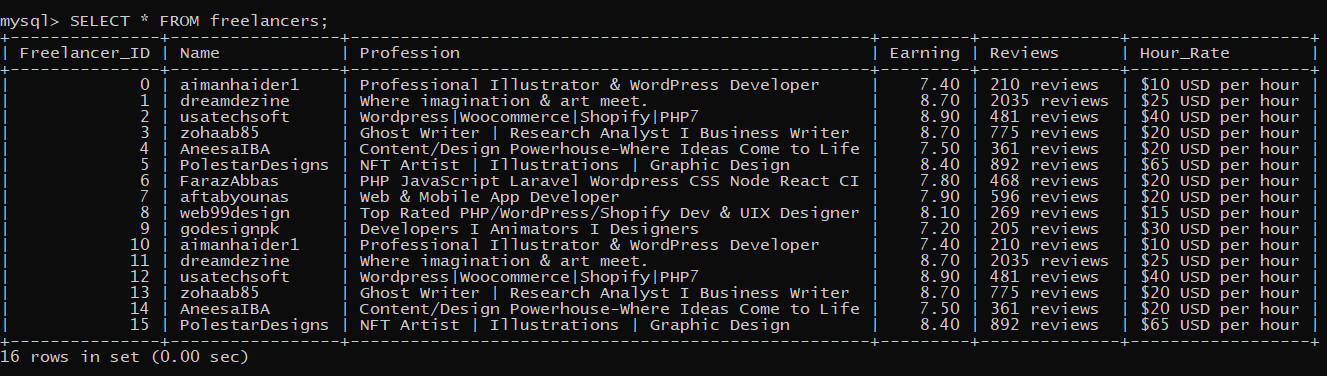
### Output:



### Querry:

SELECT \* FROM freelancers;

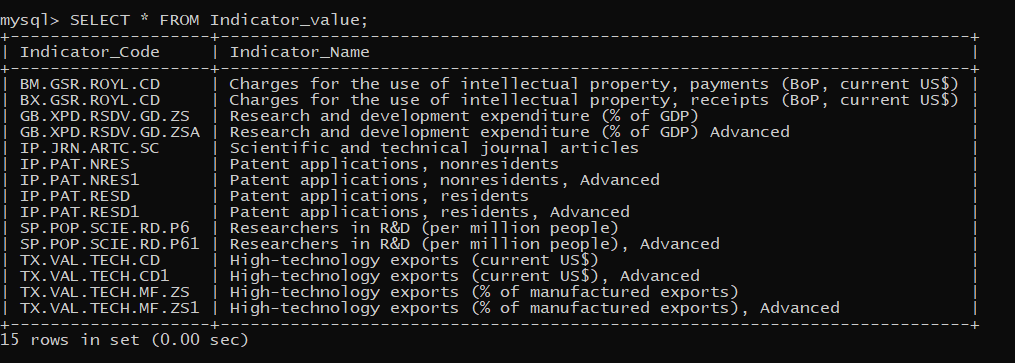
### Output:



### Querry:

SELECT \* FROM Indicator\_Value;

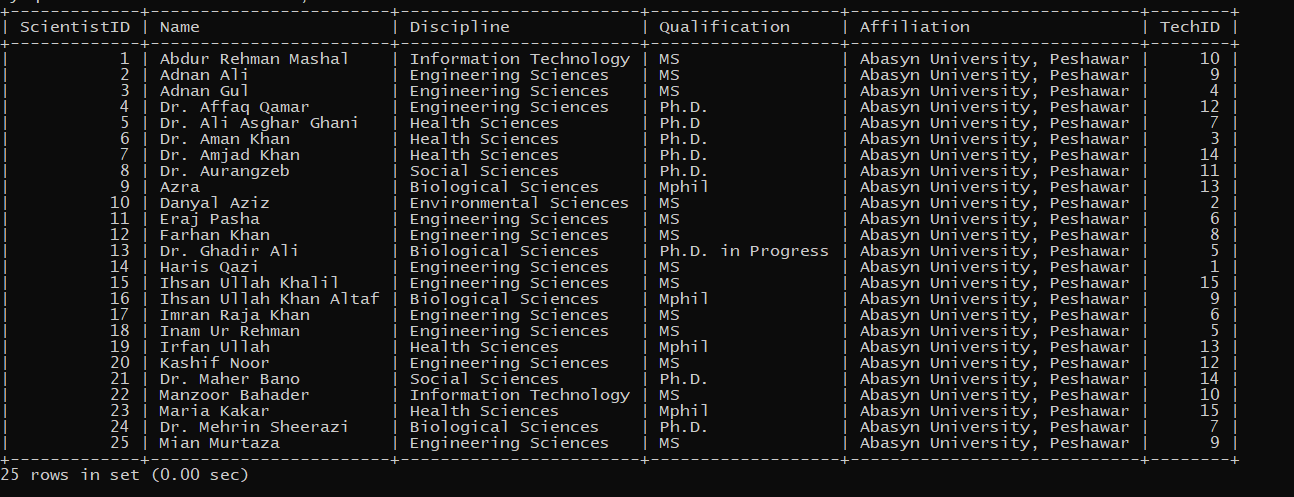
### Output:



### Querry:

SELECT \* FROM Scientist;

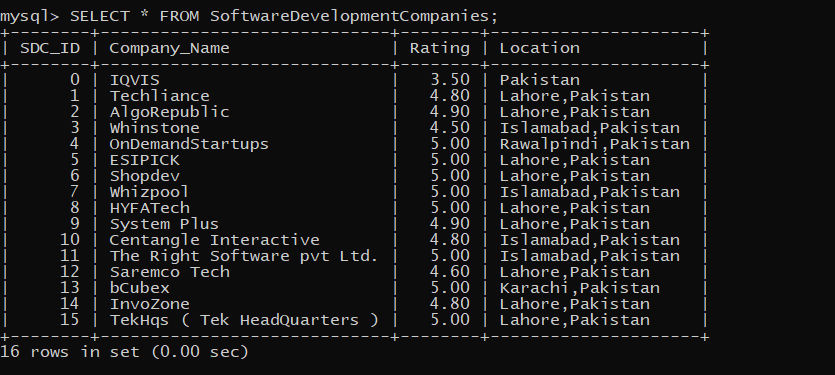
### Output:



### Querry:

SELECT \* FROM SoftwareDevelopmentCompanies;

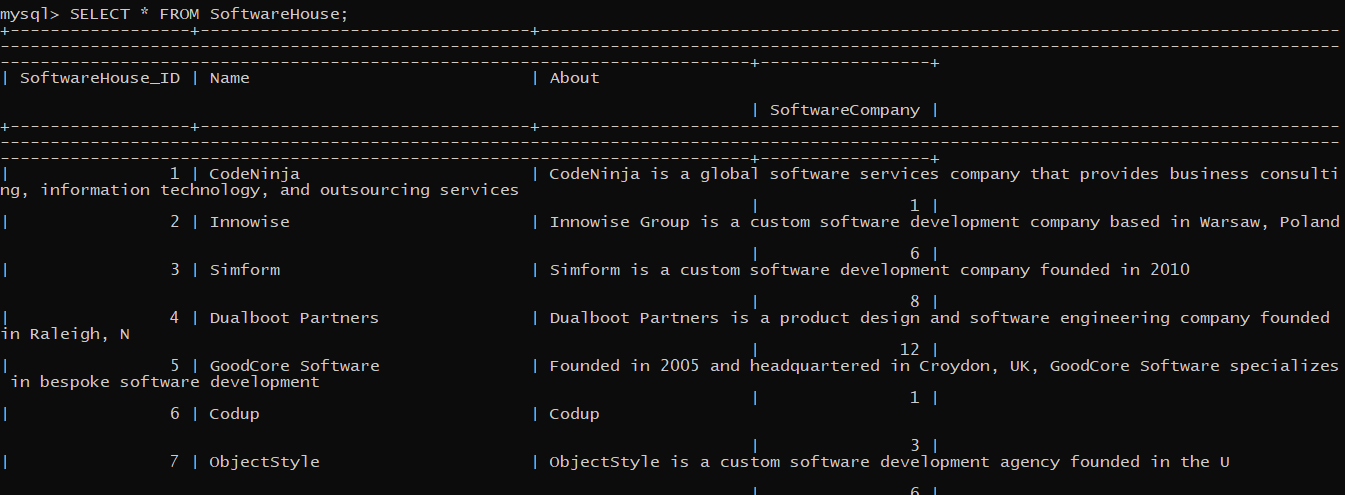
### Output:

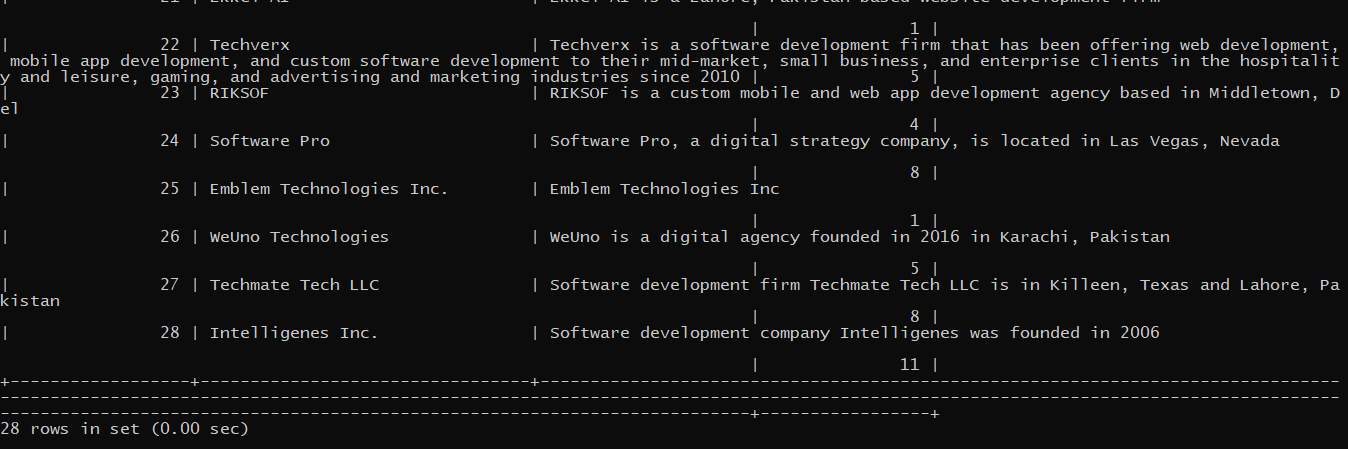


### Querry:

SELECT \* FROM SoftwareHouse;

### Output:

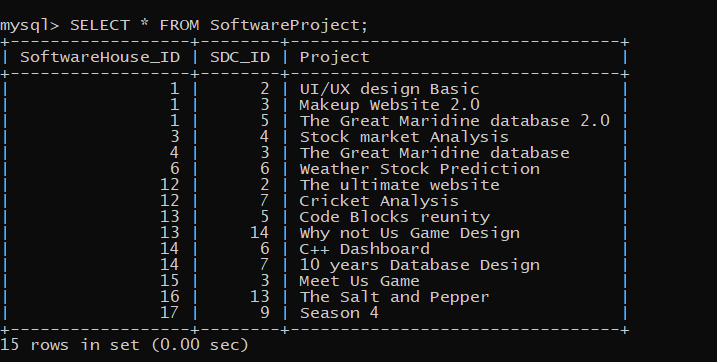




### Querry:

SELECT \* FROM SoftwareProject;

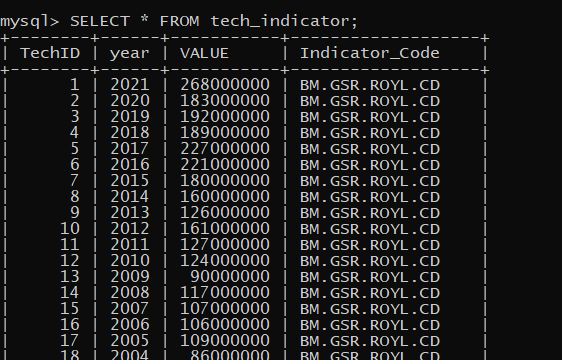
### Output:

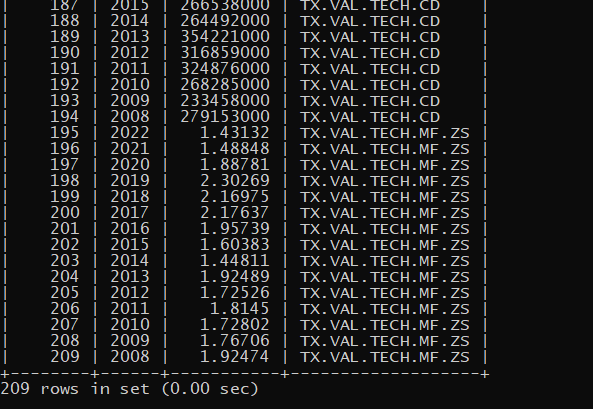


### Querry:

SELECT \* FROM tech\_indicator;

### Output:

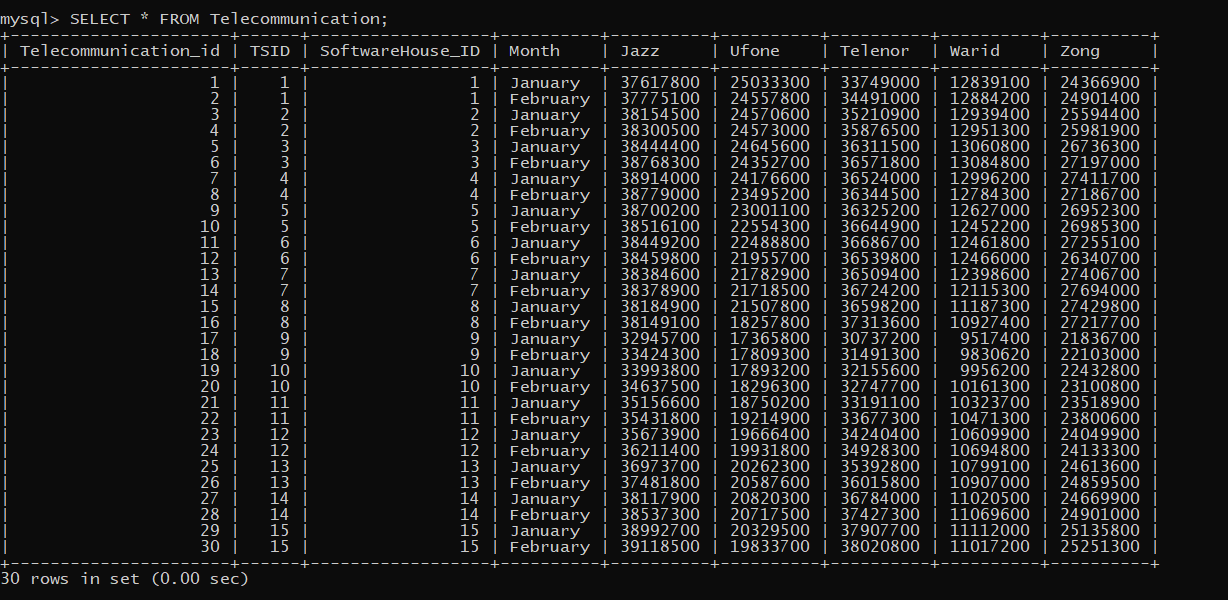




Querry:

SELECT \* FROM Telecommunication;

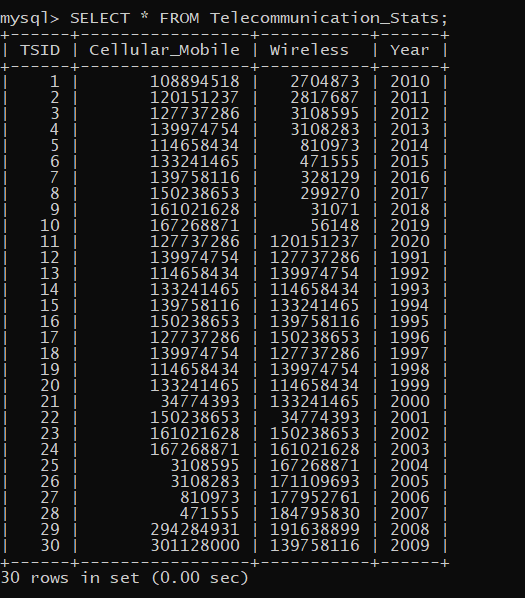
### Output:



Querry:

SELECT \* FROM Telecommunication\_Stats;

### Output:

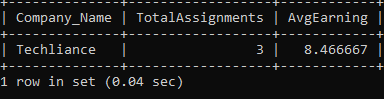


* 1. **QUERIES RESULTS:**

### Querry:

SELECT sdc.Company\_Name, COUNT(fca.FreelancerCompany\_ID) AS TotalAssignments, AVG(f.Earning) AS AvgEarning FROM SoftwareDevelopmentCompanies sdc JOIN FreelancerCompanyAssignment fca ON sdc.SDC\_ID = fca.Company\_ID JOIN Freelancers f ON fca.Freelancer\_ID = f.Freelancer\_ID GROUP BY sdc.Company\_Name HAVING COUNT(fca.FreelancerCompany\_ID) > 2 AND AVG(f.Earning) > (SELECT AVG(Earning) FROM Freelancers) ORDER BY TotalAssignments DESC;

### Output:



### Querry:

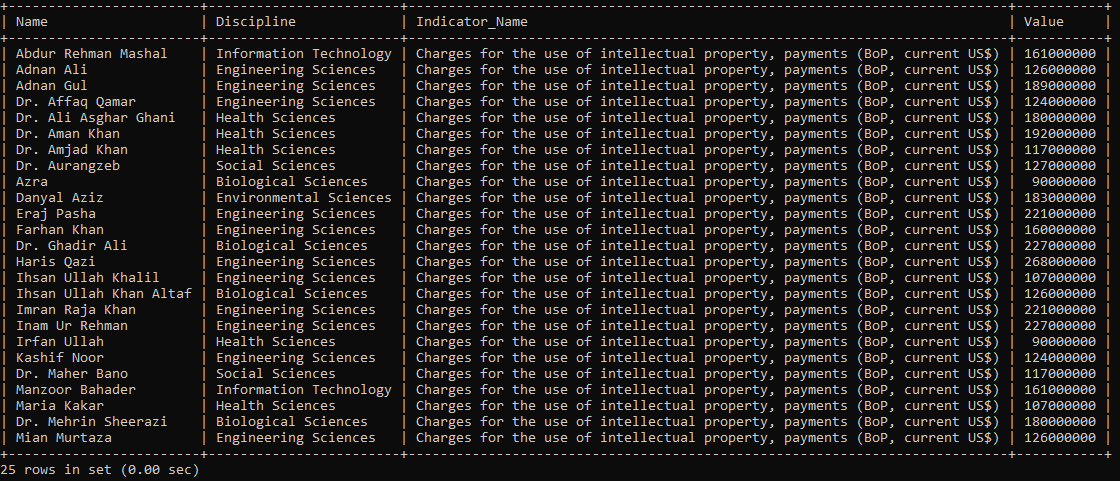
SELECT s.Discipline, COUNT(s.ScientistID) AS ScientistCount FROM Scientist s GROUP BY s.Discipline HAVING COUNT(s.ScientistID) > 4;

### Output:

### Querry:

SELECT s.Name, s.Discipline, i.Indicator\_Name, t.Value FROM Scientist s JOIN Tech\_Indicator t ON s.TechID = t.TechID JOIN Indicator\_Value i ON t.Indicator\_Code = i.Indicator\_Code;

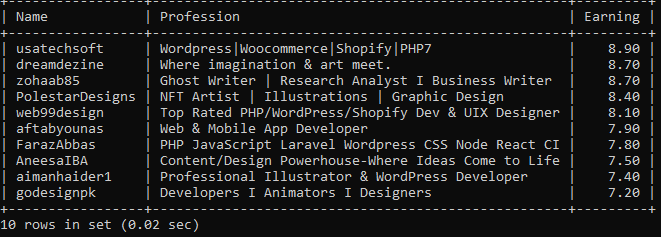
### Output:



### Querry:

### SELECT Name, Profession, MAX(Earning) AS Earning FROM Freelancers GROUP BY Name, Profession ORDER BY Earning DESC;

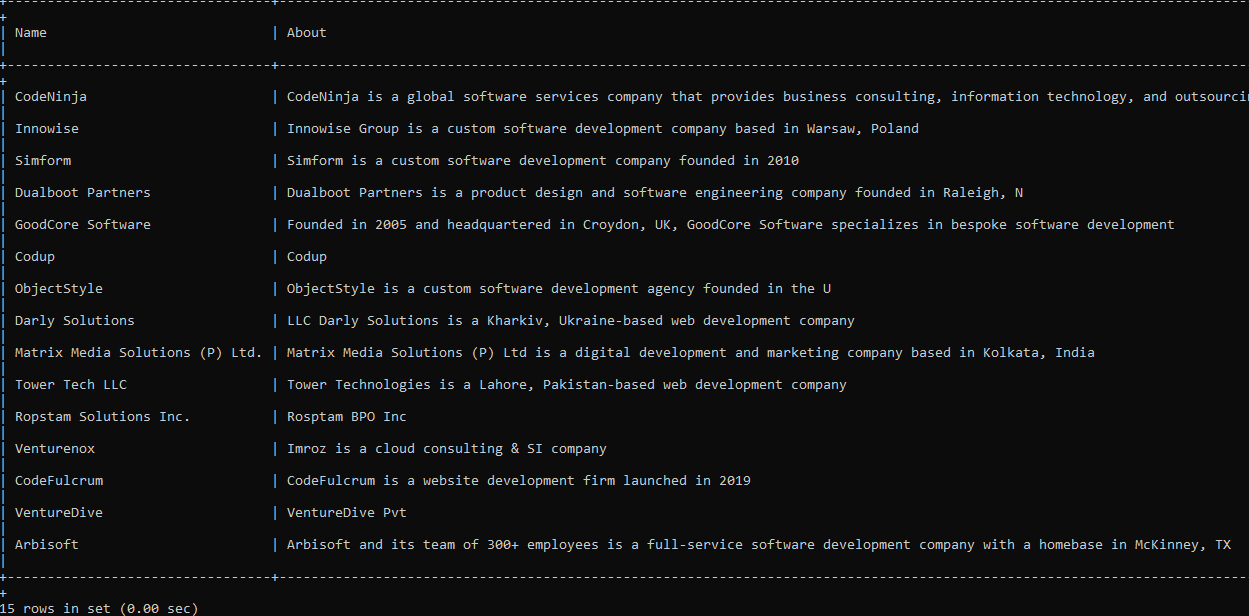
### Output:



### Querry:

SELECT Name, About FROM SoftwareHouse WHERE SoftwareHouse\_ID IN (SELECT SoftwareHouse\_ID FROM Telecommunication WHERE Jazz > 50);

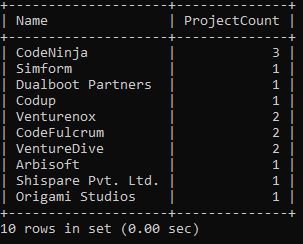
### Output:



### Querry:

SELECT sh.Name, COUNT(sp.Project) AS ProjectCount FROM SoftwareHouse sh JOIN SoftwareProject sp ON sh.SoftwareHouse\_ID = sp.SoftwareHouse\_ID GROUP BY sh.Name;

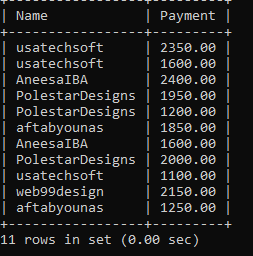
### Output:



### Querry:

SELECT f.Name, fc.Payment FROM Freelancers f JOIN FreelancerCompanyAssignment fc ON f.Freelancer\_ID = fc.Freelancer\_ID WHERE fc.Payment > 1000;

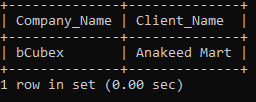
### Output:



### Querry:

SELECT SDC.Company\_Name, c.Client\_Name FROM SoftwareDevelopmentCompanies SDC JOIN Clients c ON SDC.SDC\_ID = c.SDC\_ID WHERE SDC.Rating > 4.5 AND c.Client\_Name LIKE 'A%';

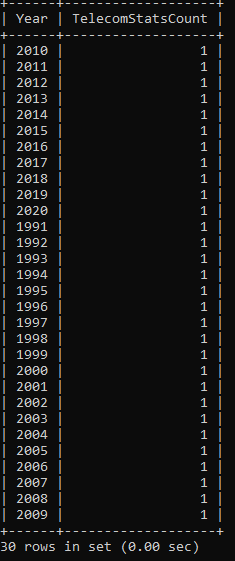
### Output:



### Querry:

SELECT t.Year, COUNT(t.TSID) AS TelecomStatsCount FROM Telecommunication\_Stats t GROUP BY t.Year;

### Output:



### Querry:

SELECT sc.Name,sdc.Rating FROM SoftwareDevelopmentCompanies sdc JOIN SoftwareProject sp ON sdc.SDC\_ID = sp.SDC\_ID JOIN SoftwareHouse sc ON sp.SoftwareHouse\_ID = sc.SoftwareHouse\_ID WHERE sdc.Rating > 4.0;

### Output:



### Querry:

SELECT Name, (SELECT COUNT(\*) FROM FreelancerCompanyAssignment fca WHERE fca.Freelancer\_ID = f.Freelancer\_ID) AS AssignmentCount FROM Freelancers f;

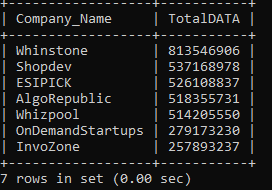
### Output:



### Querry:

SELECT sdc.Company\_Name, SUM(t.Jazz + t.Ufone + t.Telenor + t.Warid + t.Zong) AS TotalDATA FROM SoftwareDevelopmentCompanies sdc JOIN SoftwareProject sp ON sdc.SDC\_ID = sp.SDC\_ID JOIN SoftwareHouse sh ON sp.SoftwareHouse\_ID = sh.SoftwareHouse\_ID JOIN Telecommunication t ON sh.SoftwareHouse\_ID = t.SoftwareHouse\_ID GROUP BY sdc.Company\_Name HAVING SUM(t.Jazz + t.Ufone + t.Telenor + t.Warid + t.Zong) > 1000 ORDER BY TotalSubscribers DESC;

### Output:



: Interface Design

## LANGUAGE/FRAMEWORK:

For the development of the GUI client in this project, we have chosen **Python** with the **Flask** framework, integrated with HTML, CSS, and JavaScript for the frontend.

### Technologies Used:

1. **Python**: The main programming language used for the backend.
2. **Flask**: A lightweight WSGI web application framework for Python.
3. **HTML/CSS**: Used for structuring and styling the web pages.
4. **JavaScript**: Adds interactivity to the web pages.
5. **Bootstrap/CSS**: A CSS framework that helps in designing responsive web pages quickly.

### Reasons for Choosing Python and Flask:

1. **Ease of Use**: Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced developers. Its straightforward syntax allows for rapid development and maintenance.
2. **Flask**: Flask is a lightweight web framework for Python that provides the necessary tools and libraries for building web applications. It is known for being simple yet powerful, offering flexibility without imposing a lot of structure, which is ideal for projects that require customization.
3. **Extensive Libraries**: Python’s rich ecosystem of libraries and frameworks allows for quick integration of various functionalities.
4. **Community and Support**: Both Python and Flask have large, active communities that contribute to a wealth of documentation, tutorials, and third-party modules.

### Key Features and Their Contributions:

* **Flask**: Provides a simple and flexible framework to build web applications, with powerful routing and templating.
* **HTML/CSS/JavaScript**: Enables the creation of dynamic and responsive web interfaces. We use these technologies to insert/manipulate data by forms.
* **Bootstrap/CSS**: Facilitates responsive design and ensures the application looks good on various devices and screen sizes.
  1. **DATABASE CONNECTIVITY:**

*s*The GUI client connects to the MySQL database using the mysql.connector library. This library provides a robust way to connect to a MySQL database from a Python application. The connection is established using the mysql.connector.connect method, which requires database credentials and other parameters.

### Code Snippet for Database Connection

from flask import Flask, render\_template, request, redirect, url\_for, flash, jsonify

import mysql.connector

from mysql.connector import Error

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

# Database connection

try:

dataBase = mysql.connector.connect(

host='your\_host',

user='your\_username',

password='your\_password',

database='your\_database\_name'

)

if dataBase.is\_connected():

cursor = dataBase.cursor(buffered=True)

print("Successfully connected to the database")

except Error as e:

print(f"Error while connecting to MySQL: {e}")

# Close database connection when app stops

@app.teardown\_appcontext

def close\_connection(exception):

if dataBase.is\_connected():

cursor.close()

dataBase.close()

print("MySQL connection is closed")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

if dataBase.is\_connected():

cursor.close() # Close the cursor object

dataBase.close() # Close the database connection

print("MySQL Connection is Closed.")

### In this setup:

1. The mysql.connector.connect method establishes the connection using the provided host, username, password, and database name.
2. If the connection is successful, a cursor is created using dataBase.cursor(buffered=True).
3. Error handling is implemented using a try-except block to catch and print any connection errors.
4. The @app.teardown\_appcontext decorator ensures the database connection is closed when the Flask application context ends.
   1. **STORED PROCEDURES AND FUNCTIONS:**

#### Procedure: **InsertClient**

@app.route('/insert/<table>', methods=['POST'])

def process\_insert(table):

data = request.form

if table == 'Clients':

company\_client\_id = data["company\_client\_id"]

sdc\_id = data['sdc\_id']

client\_name = data['client\_name']

try:

query = f"INSERT INTO Clients (CompanyClient\_ID, SDC\_ID, Client\_Name) VALUES ({company\_client\_id}, {sdc\_id}, '{client\_name}');"

cursor.execute(query)

dataBase.commit()

result = "Data Inserted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occured, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'insert\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Inserts a new client into the Clients table.
* **Implementation:** Uses input data (company\_client\_id, sdc\_id, client\_name) to construct and execute an INSERT SQL query.
* **Enhancement:** Encapsulates insertion logic, ensuring data integrity and reducing SQL injection risks.

#### Procedure: **UpdateClient**

@app.route('/update/<table>', methods=['POST'])

def process\_update(table):

data = request.form

if table == 'Clients':

id = data["company\_client\_id"]

sdc\_id = data['sdc\_id']

client\_name = data['client\_name']

try:

query = f"UPDATE Clients SET SDC\_ID = {sdc\_id}, Client\_Name = '{client\_name}' WHERE CompanyClient\_ID = {id};"

cursor.execute(query)

dataBase.commit()

result = "Data Updated Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occured, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'update\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Updates client details in the Clients table.
* **Implementation:** Constructs and executes an UPDATE SQL query based on input data (id, sdc\_id, client\_name).
* **Enhancement:** Simplifies update operations, ensuring consistency and data accuracy.

#### Procedure: **DeleteClient**

@app.route('/delete/<table>', methods=['POST'])

def process\_delete(table):

data = request.form

if table == 'Clients':

id = data["company\_client\_id"]

try:

query = f"DELETE FROM Clients WHERE CompanyClient\_ID = {id};"

cursor.execute(query)

dataBase.commit()

result = "Data Deleted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occured, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'delete\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Deletes a client from the Clients table.
* **Implementation:** Executes a DELETE SQL query based on the company\_client\_id.
* **Enhancement:** Ensures secure deletion and maintains data consistency.

#### Function: process\_search

@app.route('/process\_search/<table>', methods=['POST'])

def process\_search(table):

data = request.form

id = data["client\_id"]

if table == 'Clients':

query = f"SELECT \* FROM clients Where CompanyClient\_ID = {id};"

cursor.execute(query)

rows = cursor.fetchall()

col\_names = [desc[0] for desc in cursor.description]

# Other table checks and queries omitted for brevity

return render\_template(f'search/AllTables.html', table = table, rows=rows, col\_names=col\_names)

**Explanation:**

* **Purpose:** Retrieves client data from the Clients table based on the client ID.
* **Implementation:** Constructs and executes a SELECT SQL query.
* **Enhancement:** Facilitates searching and displaying client information within the GUI client.

#### **InsertFreelancerCompanyAssignment Procedure**

@app.route('/insert/<table>', methods=['POST'])

def process\_insert(table):

data = request.form

if table == 'FreelancerCompanyAssignment':

fc\_assignment\_id = data['fc\_assignment\_id']

company\_id = data['company\_id']

freelancer\_id = data['freelancer\_id']

payment = data['payment']

try:

query = f"INSERT INTO FreelancerCompanyAssignment (FreelancerCompany\_ID, Company\_ID, Freelancer\_ID, Payment) VALUES ({fc\_assignment\_id}, {company\_id}, {freelancer\_id}, {payment});"

cursor.execute(query)

dataBase.commit()

result = "Data Inserted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occurred, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'insert\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Inserts a new assignment record into the FreelancerCompanyAssignment table.
* **Implementation:** Constructs and executes an INSERT SQL query with input data (fc\_assignment\_id, company\_id, freelancer\_id, payment).
* **Enhancement:** Provides a structured way to add assignments, ensuring accurate record keeping and data integrity.

#### **InsertScientist Procedure**

@app.route('/insert/<table>', methods=['POST'])

def process\_insert(table):

data = request.form

if table == 'Scientist':

scientist\_id = data['scientist\_id']

name = data['name']

discipline = data['discipline']

qualification = data['qualification']

affiliation = data['affiliation']

tech\_id = data['tech\_id']

try:

query = f"INSERT INTO Scientist (ScientistID, Name, Discipline, Qualification, Affiliation, TechID) VALUES ({scientist\_id}, '{name}', '{discipline}', '{qualification}', '{affiliation}', {tech\_id});"

cursor.execute(query)

dataBase.commit()

result = "Data Inserted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occurred, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'insert\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Inserts a new scientist record into the Scientist table.
* **Implementation:** Constructs and executes an INSERT SQL query with input data (scientist\_id, name, discipline, qualification, affiliation, tech\_id).
* **Enhancement:** Facilitates adding new scientists to the database, supporting the application's functionality in managing scientific personnel.

#### **DeleteFreelancerCompanyAssignment Procedure**

@app.route('/delete/<table>', methods=['POST'])

def process\_delete(table):

data = request.form

if table == 'FreelancerCompanyAssignment':

fc\_assignment\_id = data['fc\_assignment\_id']

try:

query = f"DELETE FROM FreelancerCompanyAssignment WHERE FreelancerCompany\_ID = {fc\_assignment\_id};"

cursor.execute(query)

dataBase.commit()

result = "Data Deleted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occurred, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'delete\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Deletes an assignment record from the FreelancerCompanyAssignment table.
* **Implementation:** Constructs and executes a DELETE SQL query based on the fc\_assignment\_id.
* **Enhancement:** Provides a controlled way to remove assignments, ensuring data integrity and application consistency.

#### **DeleteSoftwareDevelopmentCompanies Procedure**

@app.route('/delete/<table>', methods=['POST'])

def process\_delete(table):

data = request.form

if table == 'Software\_Development\_Companies':

sdc\_id = data['company\_id']

try:

query = f"DELETE FROM SoftwareDevelopmentCompanies WHERE SDC\_ID = {sdc\_id};"

cursor.execute(query)

dataBase.commit()

result = "Data Deleted Successfully."

except Error as e:

result = f"Error occurred: {e}"

result = f"Error occurred, {e}. "

else:

result = "Table not found"

flash(result)

return render\_template(f'delete\_forms/{table.lower()}.html')

**Explanation:**

* **Purpose:** Deletes a software development company record from the SoftwareDevelopmentCompanies table.
* **Implementation:** Constructs and executes a DELETE SQL query based on the sdc\_id.
* **Enhancement:** Supports management of software development companies within the application, allowing for removal of outdated or unnecessary records.

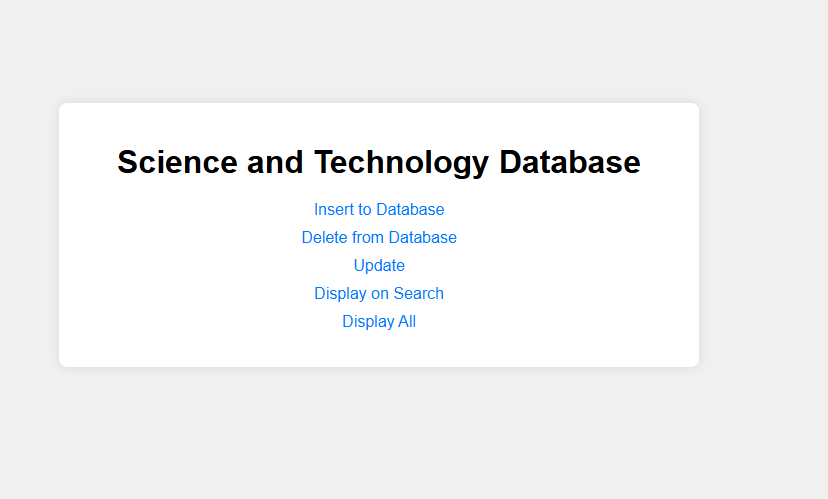
These stored procedures enhance the functionality of the GUI client by providing encapsulated and optimized routines for inserting data into specific database tables. They streamline the process of adding new records by encapsulating complex SQL queries into reusable procedures, which are invoked directly from the GUI client. This approach improves application efficiency and reduces redundancy by centralizing database interaction logic. Additionally, these procedures ensure data integrity and consistency by implementing error handling and validation checks within the database itself, thereby enhancing the overall reliability of the GUI client's data management capabilities.

**Note:**  
There are to many procedures in our database so we add a few above in this section.

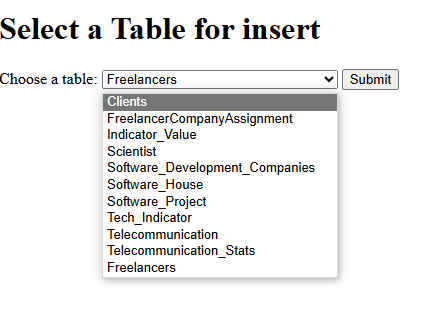
* 1. **INTERFACES:**

**Here are some screenshots providing visual documentation of the GUI.**

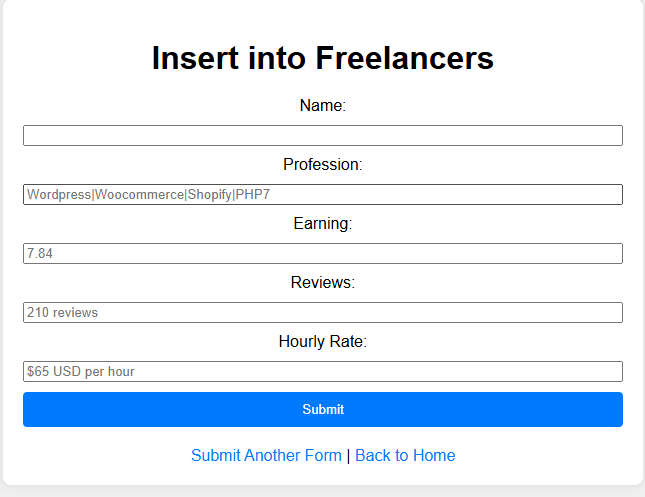
**Home page of our database/GUI which gives the option to insert delete update data from the database.**

**

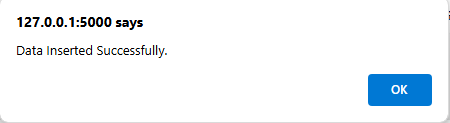
**For inserting the data into the database. Here is the list of the tables to be selected to insert data to it.**

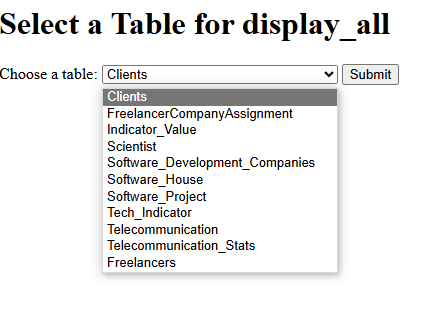
**

**To add a new freelancer, all required details must be provided**

**

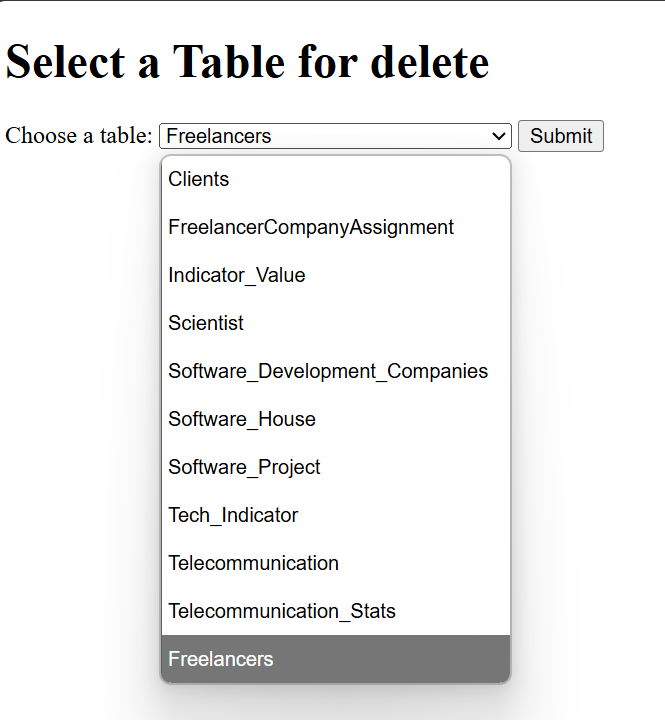
**Successfully inserted the freelancer.**

**

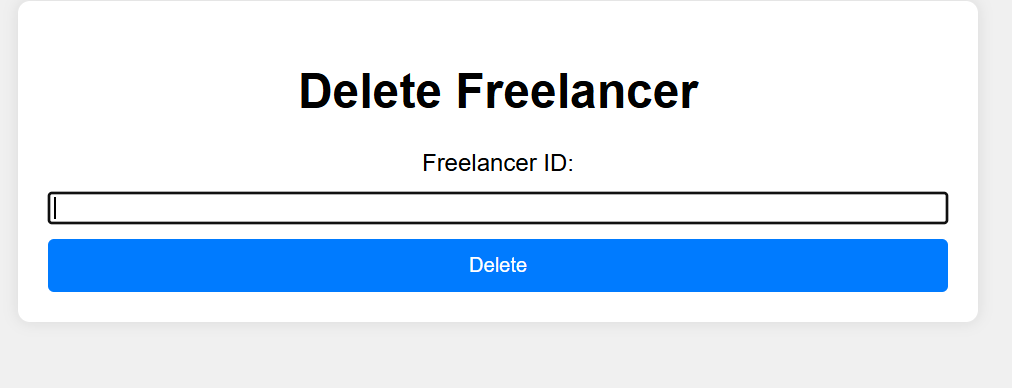
**

**

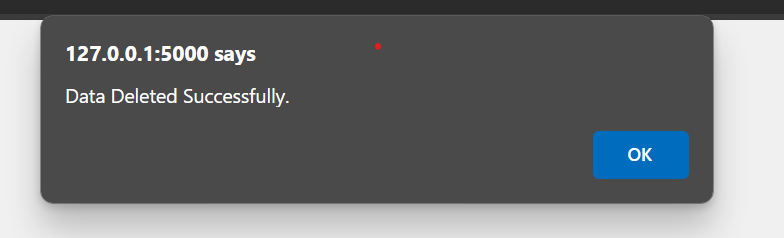
**For Deleting the data into the database. Here is the list of the tables to be selected to delete data to it.**

**

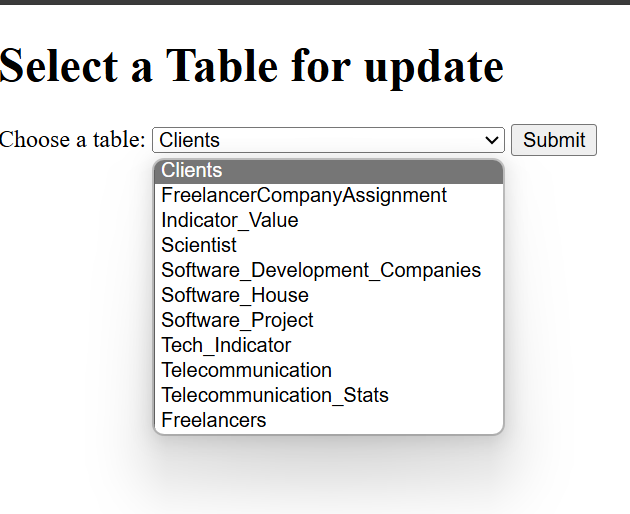
**Deleting a new freelancer requires providing their Freelancer ID.**

**

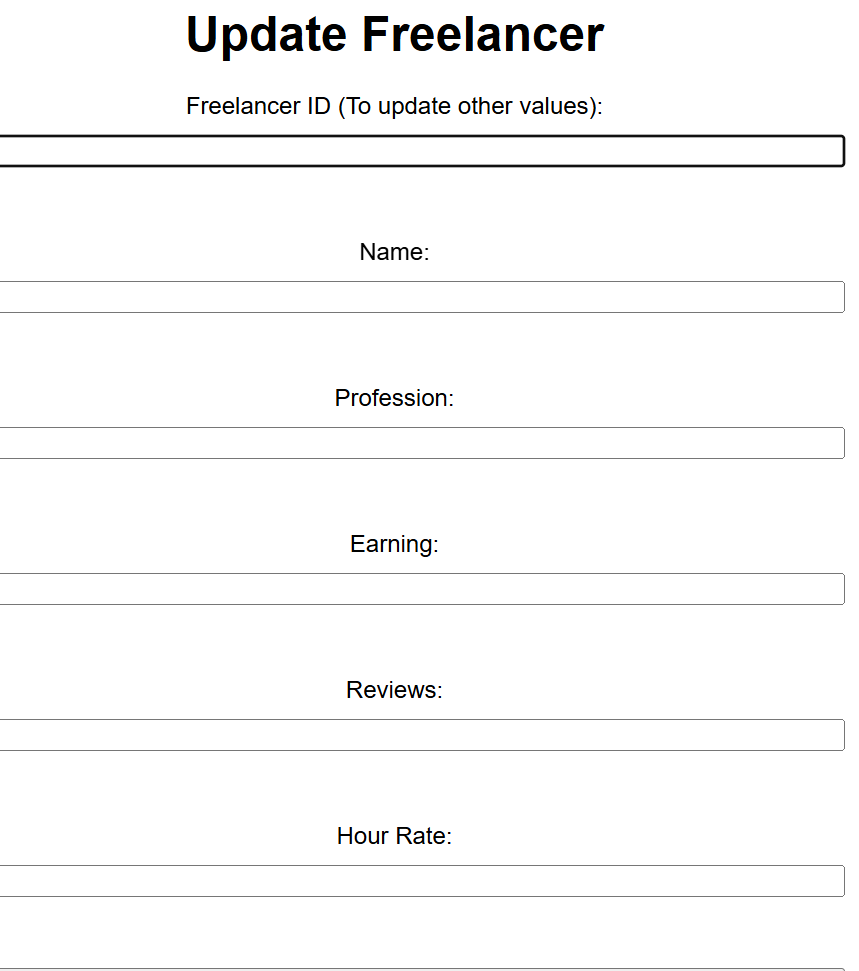
**Data is deleted successfully**

**

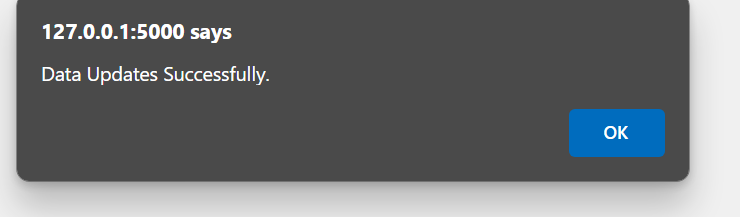
**For Updating the data into the database. Here is the list of the tables to be selected to update data to it.**

**

**To update a freelancer's profile, all necessary details must be provided.**

**

**Data is deleted successfully**

**

**To display data from the database, the user can select a table from a provided list of available tables.***A screenshot of a computer

Description automatically generated*

**The system requires the Freelancer ID to display a new freelancer's information.**

*A screenshot of a computer

Description automatically generated*

*A white box with black text

Description automatically generated*

# : CONCLUSION

## LESSONS LEARNED:

**This project has been a valuable learning experience in the realm of database management and software development.**

**The process began with a detailed Entity-Relationship Diagram (ERD), highlighting the crucial role of thorough initial planning. We learned that a well-structured ERD is essential for building a robust database foundation. Applying normalization techniques to our database design emphasized the importance of balancing data redundancy and efficiency, optimizing both performance and integrity.**

**Integrating MySQL with Flask for the GUI demonstrated the critical nature of seamless backend/frontend communication, reliable database connections, and effective error handling. Throughout the project, we encountered various challenges that required us to develop robust debugging and error-handling skills, proving pivotal in maintaining system stability.**

**Iterative testing and feedback loops were instrumental in refining our database and GUI, underscoring the significance of user-centric design and continuous improvement. Furthermore, the project enhanced our teamwork and communication skills, teaching us how to leverage each member's strengths and coordinate effectively.**

**Overall, this project has deepened our understanding of database management, improved our technical abilities, and prepared us for more complex future projects.**

* 1. **CHALLENGES AND SOLUTIONS:**

The project's development presented a series of challenges, which the team successfully addressed. Design complexities emerged during Entity-Relationship Diagram (ERD) development and normalization, necessitating iterative refinement to achieve an optimal schema that balanced data integrity and performance. Implementation challenges centered around fine-tuning SQL queries and ensuring robust error handling mechanisms, crucial for maintaining system reliability. Rigorous testing phases focused on validating data insertion, retrieval, and Graphical User Interface (GUI) functionality to meet user expectations. Solutions involved collaborative problem-solving, leveraging team expertise to effectively overcome design intricacies and technical hurdles.

**FUTURE WORK AND IMPROVEMENTS:**

Right now, our project helps us understand technology and science. We can make it even better by using special tools to predict future trends. We can also make the information easier to understand by showing it in a visually appealing way. To make sure everyone can use our project, we can move it to the cloud, so it's accessible anywhere. Lastly, we can use our project to figure out how technology will impact Pakistan's economy and society, which will benefit everyone.

* 1. **FINAL THOUGHTS:**

In the end, this project has been a testament to our team's dedication. Beyond developing a robust database-driven GUI application, it has underscored the importance of structured project management, agile methodologies, and continuous learning. Personally, this journey has deepened our understanding of database systems, software development lifecycles, and the transformative potential of integrating technology with real-world applications. As we look ahead, we are confident that the insights gained, and the collaborative spirit fostered will guide us towards future successes in leveraging technology for positive societal impact.

# REFERENCES

|  |  |
| --- | --- |
| [1] | "Science and Technology," 09 2021. [Online]. Available: https://opendata.com.pk/dataset?category=Science+%26+Technology&page=1. [Accessed 13 3 2024]. |
| [2] | "world-bank-science-and-technology-indicators-for-pakistan," 28 02 2024. [Online]. Available: https://data.humdata.org/dataset/world-bank-science-and-technology-indicators-for-pakistan?. [Accessed 12 3 2024]. |