
**Project: Development of a National Gender Affirmative Monitoring Database Prototype.**

**BY**

Chesia Anyika - 665567

Joshua Ruheni Wambugu - 661446

Melissa Kariuki - 664819

Ian Andy Njagi - 665174

**School of Science and Technology, United States International University - Africa**

**APT1050 - Database Systems**

**Prof. Elisha Toyne O. Omulo**

**Spring 2023**

## ABSTRACT

Gender equality is a critical objective for sustainable development, necessitating robust monitoring mechanisms to track progress and inform policy interventions. This study presents the development and implementation of a **National Gender Affirmative Monitoring Database**, designed to systematically track key educational and employment metrics, including enrollment, graduation rates, academic performance, and employment outcomes by gender. The database was developed using **Programiz SQL Compiler**, structured through **SQL queries**, and visually modeled using **Draw.io** for ERD diagramming. Data was generated using **ChatGPT**, then manually entered and normalized up to **Third Normal Form (3NF)** to ensure data integrity and reduce redundancy. Various reports were generated to analyze gender representation, salary disparities, and hiring trends, providing insights into the effectiveness of gender-affirmative policies. By leveraging structured data analysis, this project enhances evidence-based decision-making, supporting efforts to bridge gender disparities in education and employment. Future advancements may incorporate AI-driven analytics for more precise monitoring and policy recommendations.

**TABLE OF CONTENTS**

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background

Gender equality remains a fundamental goal for social and economic development, necessitating systematic mechanisms to monitor progress and enforce affirmative action policies. National Gender Affirmative Monitoring (NGAM) refers to a structured process of assessing and ensuring gender equity in policies, institutions, and societal practices (UN Women, 2021). This process involves collecting gender-disaggregated data, evaluating policy effectiveness, and recommending strategies to address disparities (OECD, 2020).

Governments and international organizations have established NGAM frameworks to track gender representation in various sectors, including political participation, education, and employment. These monitoring systems are often mandated by legal and institutional frameworks, such as gender quotas in leadership and equal pay policies (World Economic Forum, 2022). The aim is to measure the impact of affirmative action initiatives and inform evidence-based policymaking to achieve gender parity.

Monitoring efforts typically involve data collection through national statistics agencies, non-governmental organizations (NGOs), and research institutions. The data is analyzed to assess progress and identify gaps in gender equity, leading to policy adjustments where necessary (European Institute for Gender Equality, 2021). Additionally, annual reports and gender audits are conducted to ensure accountability and transparency in policy implementation.

The significance of NGAM lies in its role in reinforcing international commitments, such as the United Nations Sustainable Development Goal 5 (SDG 5), which advocates for gender equality and the empowerment of women and girls (United Nations, 2015). By systematically tracking gender-related indicators, NGAM contributes to the reduction of gender-based discrimination and fosters inclusive development (World Bank, 2021). However, challenges such as inconsistent data collection, political resistance, and inadequate funding hinder the full realization of gender monitoring objectives (UNDP, 2019).

To address these challenges, various countries have adopted digital platforms and artificial intelligence tools to enhance gender data collection and analysis (International Labour Organization, 2022). These technological advancements improve the accuracy and accessibility of gender monitoring reports, facilitating more effective interventions. Future research should explore how integrating AI-driven gender analysis tools can enhance policy effectiveness and drive long-term gender equality outcomes.

### 1.2 Project overview

This project proposes the development of a National Gender Affirmative Monitoring Database prototype that focuses on key educational and employment aspects by gender. The database will systematically track and analyze data on enrollment, graduation rates, academic performance, and employment outcomes to assess the efficacy of gender-affirmative policies. By consolidating gender-disaggregated data, this initiative will provide policymakers, researchers, and institutions with reliable insights to enhance gender equity efforts. By bridging gaps in gender-related statistics, this initiative will contribute to evidence-based decision-making and the refinement of gender equality policies at national and institutional levels

### 1.3 Objective

The primary objective of this project is to develop a relational database system to monitor gender-related policies by tracking:

1. Educational enrollment and graduation rates per gender.
2. Employment rates by gender across different industries.
3. Salary disparities and hiring patterns by gender.

### 1.4 Scope

The system will cover:

1. Counties as administrative regions.
2. Educational institutions from primary schools to universities.
3. Employment organizations across different industries.
4. Data related to scholarships and salary distributions.

### 1.5 Definition of Terms

#### 1.5.1 National Gender Affirmative Monitoring (NGAM)

National Gender Affirmative Monitoring refers to a structured approach to tracking and evaluating gender-related policies, representation, and participation in various sectors to promote equity and inclusivity. It involves collecting and analyzing data on gender representation to inform policy decisions (UN Women, 2021).

#### 1.5.2 Database

A database is an organized collection of structured information or data that is stored electronically in a computer system. Databases allow for efficient retrieval, management, and modification of data (Elmasri & Navathe, 2015).

### 1.5.3 Relational Database

A relational database is a type of database that stores data in tables (relations) that are structured with rows and columns, using unique keys to establish relationships between different tables (Codd, 1970).

### 1.5.4 Relational Database Management System (RDBMS)

An RDBMS is software that enables users to create, update, and manage relational databases by supporting structured query language (SQL) operations, enforcing constraints, and ensuring data integrity (Date, 2019).

### 1.5.5 Schema

A schema is the logical structure that defines the organization of data in a database, including tables, relationships, and constraints (Elmasri & Navathe, 2015).

### 1.5.6 Conceptual, External, and Internal Schema

1. **Conceptual schema** represents the overall structure of the database from a high-level perspective, defining entities, relationships, and constraints (Elmasri & Navathe, 2015).
2. **External schema** is the user-specific view of the database, defining how data is presented to different user groups (Elmasri & Navathe, 2015).
3. **Internal schema** defines the physical storage structure, including indexing and data organization on storage devices (Elmasri & Navathe, 2015).

### 1.5.7 Entity-Relationship Diagram (ERD)

An entity-relationship diagram (ERD) is a graphical representation of entities, their attributes, and the relationships between them in a database system (Chen, 1976).

### 1.5.8 Structured Query Language (SQL)

SQL is a standardized programming language used for managing and manipulating relational databases, including querying, inserting, updating, and deleting data (Chamberlin & Boyce, 1974).

### 1.5.9 Unified Modeling Language (UML)

UML is a standardized modeling language used in software engineering to visualize system architecture, including database structures and application behaviors (Booch, Rumbaugh, & Jacobson, 2005).

### 1.5.10 Large Language Model (LLM)

A large language model (LLM) is an advanced AI model trained on vast amounts of text data to generate human-like text, answer questions, and perform language-related tasks (Brown et al., 2020).

### 1.5.11 ChatGPT

ChatGPT is a conversational AI model developed by OpenAI, based on the GPT (Generative Pre-trained Transformer) architecture, designed for generating human-like text responses in various applications (OpenAI, 2023).

## 1.6 List of Abbreviations

**1.6.1 NGAM** – National Gender Affirmative Monitoring

**1.6.2 ERD** – Entity Relationship Diagram

**1.6.3 1NF** – First Normal Form

**1.6.4 2NF** – Second Normal Form

**1.6.5 3NF** – Third Normal Form

**1.6.6 RDBMS** – Relational Database Management System

**1.6.7 SQL** – Structured Query Language

**1.6.8 UML** – Unified Modeling Language

**1.6.9 LLM** – Large Language Model

# CHAPTER TWO

# DATABASE DESIGN

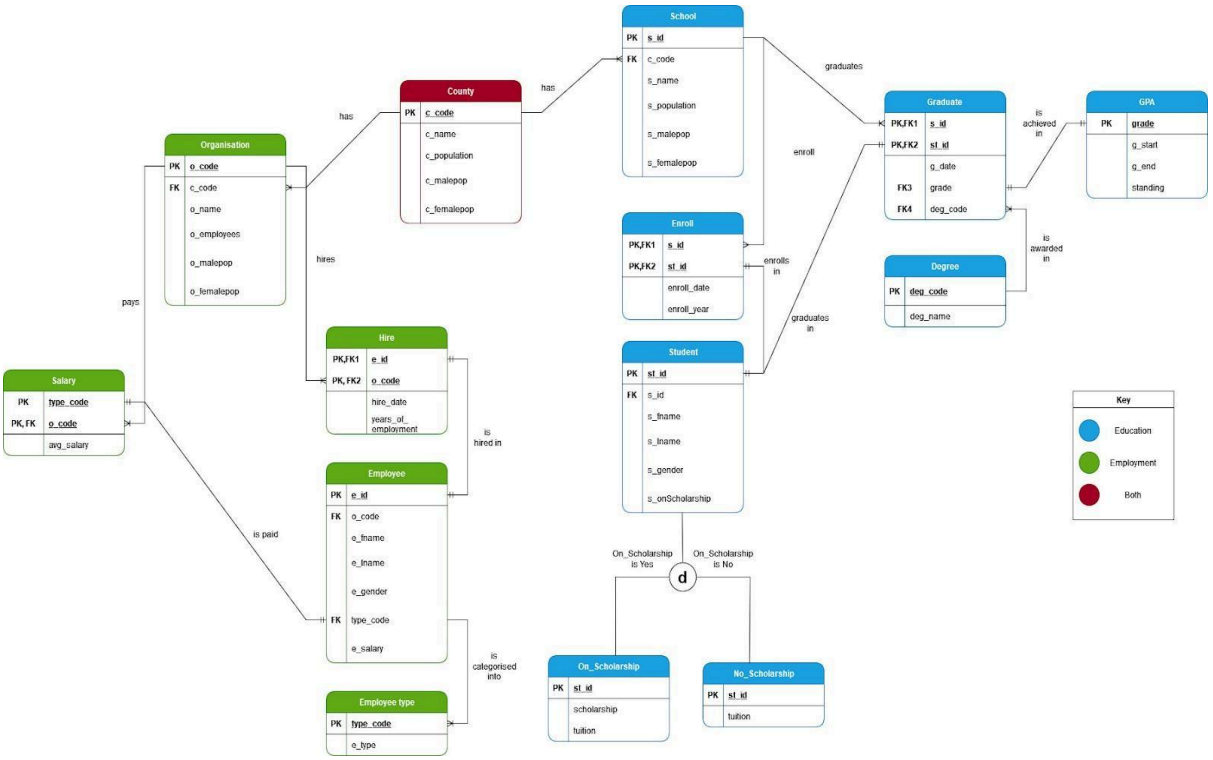
## 2.1 Conceptual Schema

The conceptual schema represents the overall logical structure of the database, independent of physical storage or specific user applications. It defines entities, relationships, constraints, and rules that ensure data integrity and consistency (Elmasri & Navathe, 2016). This level provides a unified view of the entire database, abstracting implementation details while supporting different user perspectives (Connolly & Begg, 2020).

The key entities defined by the conceptual schema are:

1. **County**: Represents different administrative regions.
2. **School**: Tracks educational institutions.
3. **Student**: Holds student information, including gender and scholarship status.
4. **Degree**: Represents different educational programs.
5. **Graduate**: Tracks student graduation details.
6. **Organization**: Represents employment organizations.
7. **Employee**: Tracks employment details including gender, job title, and salary.

These key entities are represented in an Entity Relationship Diagram (ERD) as follows:



## 2.2 External Schema

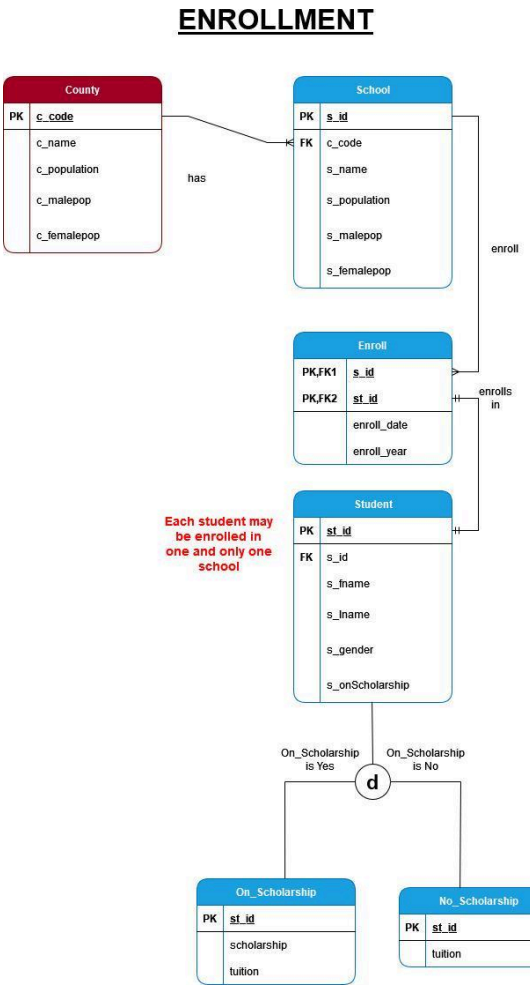
The external schema represents different user views of the database, tailored to specific application needs. It defines how data is presented to end-users and applications, ensuring that they access only the relevant subset of data (Ramez & Shamkant, 2021). Multiple external schemas can exist for a single database, allowing different departments or users to interact with data based on their requirements (Connolly & Begg, 2020).

The system supports the following key processes:

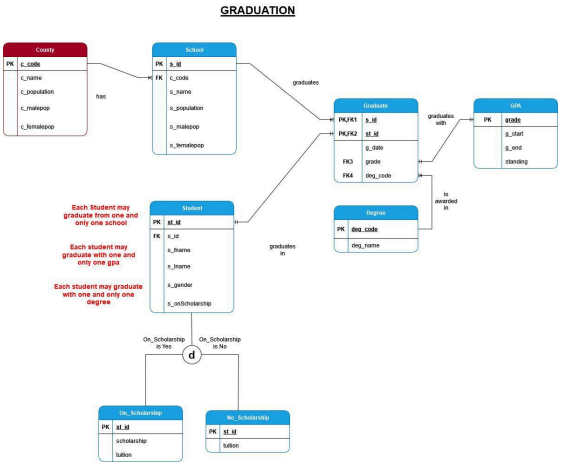
1. **Enrollment**: Tracks student enrollment by gender.
2. **Graduation**: Stores graduation details, including degree obtained and gender representation.
3. **Scholarship Awarding**: Monitors gender distribution in scholarships.
4. **Hiring**: Tracks employment rates by gender.
5. **Paying**: Monitors salary distribution to identify gender pay gaps.

These processes and their entities are each represented in the following entity relationship diagrams:

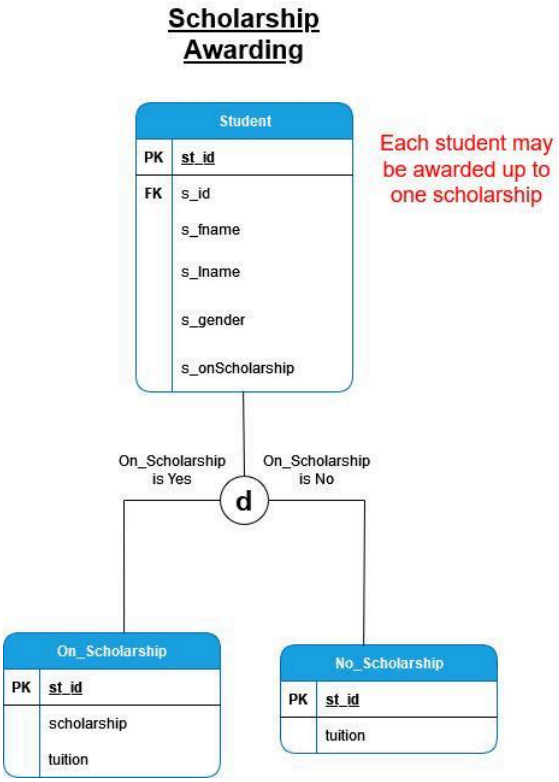
### 2.2.1 Enrollment



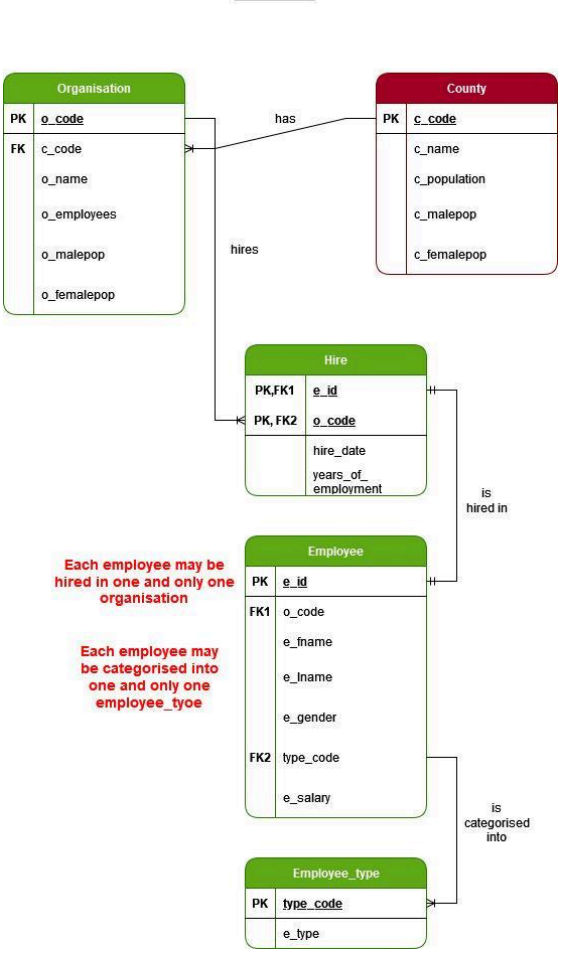

### 2.2.2 Graduation



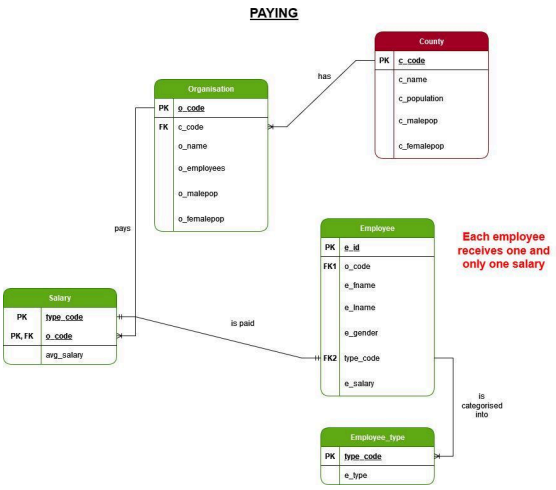
## 2.2.3 Scholarship Awarding




## 2.2.4 Hiring




## 2.2.5 Paying



### 2.3 Internal Schema

The internal schema describes the physical storage structure of the database, including file organization, indexing strategies, and access paths. It is concerned with optimizing data retrieval and storage efficiency (Silberschatz, Korth, & Sudarshan, 2019). This schema ensures that data is stored in a way that minimizes redundancy and enhances performance by defining how tables, indexes, and records are physically allocated (Date, 2019).

The internal schema includes the SQL script to create the following tables and relations:

1. **County Table**: Stores county details with gender-specific population data.
2. **School Table**: Tracks schools with male/female student populations.
3. **Student Table**: Includes gender and scholarship details.
4. **Degree Table**: Stores degree programs.
5. **Graduate Table**: Links students to graduation data.
6. **Organization Table**: Holds organization details with gender-specific employment.
7. **Employee Table**: Stores employee details, including job type and gender.
8. **Hire Table**: Tracks employment history.
9. **Salary Table**: Monitors salary data by gender.

The complete SQL script of the internal schema has been attached as an appendix (*Marked 'Appendix 1'*). The relational database, based on the internal schema, is implemented using **Programiz SQL Compiler –** screenshots of this implementation are attached as an appendix (*Marked 'Appendix 2'*).

### 2.4 Data Normalization

#### 2.4.1 Definition of Normalization

Data normalization is a process in database design that organizes data to minimize redundancy and improve data integrity. It involves structuring a database according to a set of normal forms, which define specific rules to eliminate data anomalies (Elmasri & Navathe, 2016). Normalization ensures that data is stored efficiently, reduces data duplication, and enhances consistency across the database (Silberschatz, Korth, & Sudarshan, 2019).

The levels of normalization are as follows:

1. **First Normal Form (1NF):** A table is in 1NF if it contains only atomic (indivisible) values and each column contains values of a single type. There should be no repeating groups or arrays within a row (Connolly & Begg, 2020).
2. **Second Normal Form (2NF):** A table is in 2NF if it meets the requirements of 1NF and ensures that all non-key attributes are fully functionally dependent on the primary

key. This eliminates partial dependencies where a non-key attribute depends on only part of a composite key (Date, 2019).

3. **Third Normal Form (3NF):** A table is in 3NF if it satisfies 2NF and eliminates transitive dependencies. This means that all non-key attributes must depend only on the primary key, not on other non-key attributes (Ramez & Shamkant, 2021).

### 2.4.2 Normalization of the NGAM Database

The database for this project has been normalized up to **3NF** to reduce redundancy and ensure data integrity:

1. **1NF Implementation:** All tables store atomic values without repeating groups. For example, student records include separate fields for first name, last name, gender, and enrollment year, rather than storing multiple values in a single column.
2. **2NF Implementation:** All non-key attributes are fully dependent on the primary key. For instance, in the Enrollment Table, student information is linked to a unique student ID, ensuring that attributes such as enrollment year and course name are not partially dependent on a composite key.
3. **3NF Implementation:** Transitive dependencies have been eliminated. For example, employment statistics are stored in a separate Employment Table rather than being embedded in the Graduation Table. This prevents attributes like salary range and job sector from depending on graduation year rather than directly on the individual graduate's ID.

This normalization process enhances data integrity by preventing anomalies, such as inconsistent updates or deletions, and facilitates efficient querying for gender-related educational and employment analysis.

# CHAPTER 3

# DATABASE CREATION & IMPLEMENTATION

## 3.1 Tools Used

### 3.1.1 Database Management System: Programiz SQL Compiler

The database was implemented using Programiz SQL Compiler, an online platform for writing and executing SQL queries. It was used to create and manage the database schema, run queries, and enforce data integrity through constraints and normalization techniques. While Programiz does not provide full database management features like an enterprise-grade RDBMS, it serves as a practical tool for developing and testing SQL-based database structures.

### 3.1.2 SQL Queries: Data Retrieval and Analysis

Structured Query Language (SQL) was used to interact with the database, enabling efficient data retrieval, modification, and management. SQL queries were written for tasks such as:

1. Data definition (creating tables, defining primary and foreign keys).
2. Data manipulation (inserting, updating, deleting records).
3. Data retrieval (using SELECT statements for reporting and analysis).
4. Data integrity enforcement (through constraints like UNIQUE, NOT NULL, and FOREIGN KEY).

### 3.1.3 ERD Diagramming: Draw.io (diagrams.net)

To design the conceptual schema and visualize entity relationships, Draw.io (diagrams.net) was used as a UML diagramming tool. Entity-Relationship Diagrams (ERD) were created to define entities, their attributes, and the relationships between them. This step ensured proper database normalization and structure before implementation in the Programiz SQL Compiler.

## 3.2 Data Generation & Population

The data used in this project was randomly generated using a Large Language Model (LLM), specifically ChatGPT, to simulate realistic records for the database. This approach ensured a diverse dataset while maintaining relevance to the key educational and employment metrics being tracked. The generated data included attributes such as student enrollment numbers, graduation rates, academic performance (grades), and employment statistics categorized by gender.

After generation, the data was manually input into the database by group members using SQL INSERT statements within the Programiz environment. This manual entry process allowed

for careful validation and adjustment of records to align with the database schema and normalization standards, ensuring consistency and accuracy in the dataset.

### 3.3 Report Generation

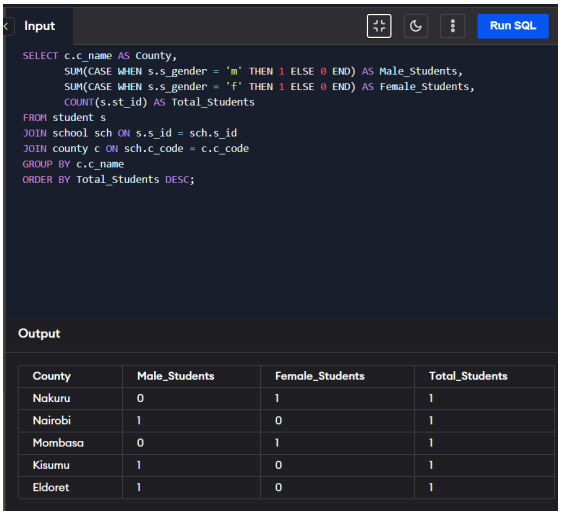
Report generation in SQL databases refers to the process of extracting, formatting, and presenting data in a structured manner to provide insights into various metrics. SQL queries are used to retrieve data from tables, aggregate information, and generate summaries that aid decision-making. These reports help organizations analyze trends, identify disparities, and support evidence-based policy formulation.

#### 3.3.1 Gender Representation Reports

Gender representation reports provide insights into the distribution of individuals across different categories based on gender. These reports help policymakers and institutions assess the effectiveness of gender affirmative policies in education and employment.

**Enrollment by Gender per County**

This report shows how student enrollment is distributed across different counties by gender.

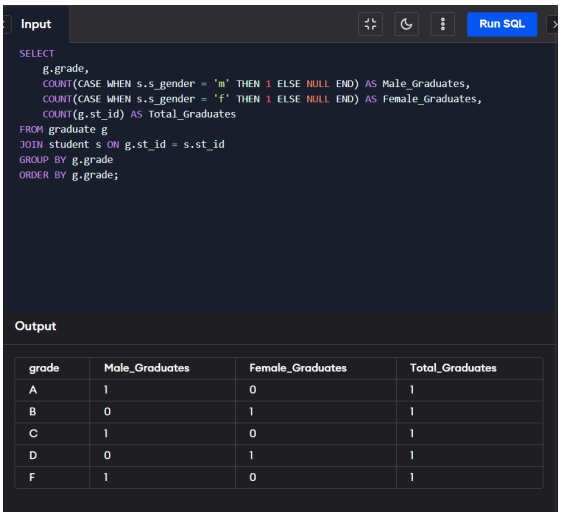



| County | Male_Students | Female_Students | Total_Students |
|---|---|---|---|
| Nakuru | 0 | 1 | 1 |
| Nairobi | 1 | 0 | 1 |
| Mombasa | 0 | 1 | 1 |
| Kisumu | 1 | 0 | 1 |
| Eldoret | 1 | 0 | 1 |

A significant disparity in enrollment may indicate gender-based barriers to education in specific regions, requiring targeted interventions. The report shows marginally more male than female students in total, but a generally equal distribution.

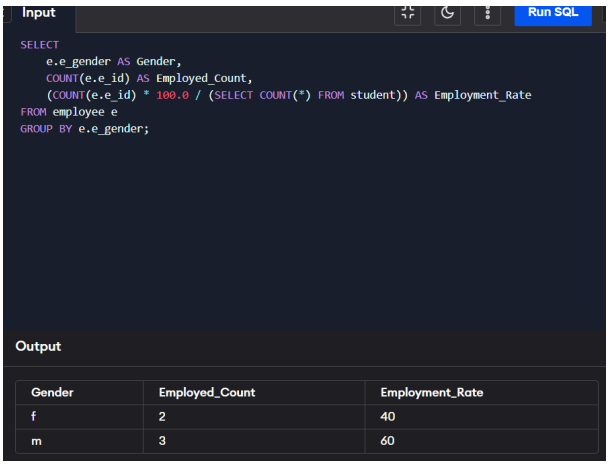
**Graduation Rates per Gender**

This report tracks the percentage of enrolled students who successfully graduate, disaggregated by gender.




| grade | Male_Graduates | Female_Graduates | Total_Graduates |
|---|---|---|---|
| A | 1 | 0 | 1 |
| B | 0 | 1 | 1 |
| C | 1 | 0 | 1 |
| D | 0 | 1 | 1 |
| F | 1 | 0 | 1 |

A lower graduation rate for one gender could signal systemic challenges such as lack of support, financial barriers, or societal expectations affecting educational attainment. The report shows a higher rate of graduation for men in general, but a generally equal graduation rate.




**Employment Rates per Gender**

This report examines post-graduation employment rates for different genders.

If one gender has consistently lower employment rates, it may indicate bias in hiring practices or a need for additional training and support programs.

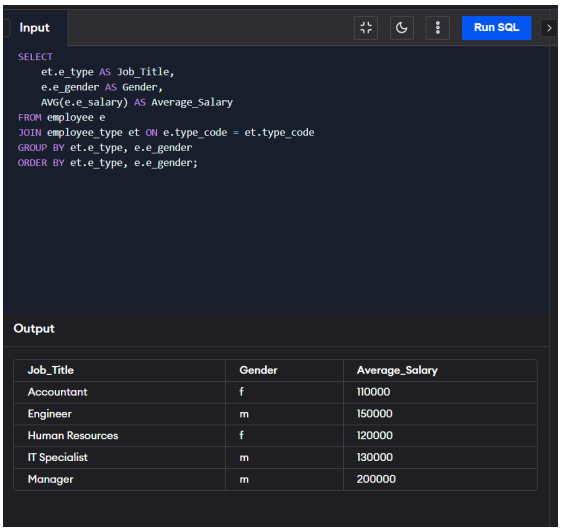
The report shows a lower percentage of employment for women than for men.

### 3.3.2 Salary Disparity Reports

Salary disparity reports analyze differences in earnings across genders and job titles. These reports highlight potential wage gaps and inform policies to ensure equal pay for equal work.

**Average Salary by Gender and Job Title**

This report compares average salaries across various job titles for different genders.




If one gender earns significantly more than the other for the same roles, this indicates a gender pay gap that may require policy adjustments or organizational reforms. The database created has singular employee entries in unique roles, thus no comparison can be made.

**Salary Distribution Across Different Sectors**

This report categorizes salary ranges across industries, highlighting gender-based earnings distribution.

Input

Run SQL

```
SELECT
    o.o_name AS Organization,
    et.e_type AS Job_Title,
    AVG(e.e_salary) AS Average_Salary,
    MIN(e.e_salary) AS Min_Salary,
    MAX(e.e_salary) AS Max_Salary
FROM employee e
JOIN organisation o ON e.o_code = o.o_code
JOIN employee_type et ON e.type_code = et.type_code
GROUP BY o.o_name, et.e_type
ORDER BY o.o_name, et.e_type;
```

Output

| Organization | Job_Title | Average_Salary | Min_Salary | Max_Salary |
|---|---|---|---|---|
| Kenya Ports Authority | Human Resources | 120000 | 120000 | 120000 |
| Kisumu Breweries | Manager | 200000 | 200000 | 200000 |
| Moi University | IT Specialist | 130000 | 130000 | 130000 |
| Nakuru County Gov. | Accountant | 110000 | 110000 | 110000 |
| Safaricom | Engineer | 150000 | 150000 | 150000 |

If one gender has consistently lower employment rates, it may indicate bias in hiring practices or a need for additional training and support programs. Due to the singular entries, no comparison can be made.

### 3.3.3 Hiring Trends

Hiring trend reports provide insights into employment patterns across organizations, helping to assess diversity and inclusion efforts in recruitment.

**Employment Rates by Gender Across Different Organizations**

This report examines hiring patterns in various industries, disaggregated by gender.

Input

Run SQL

```
SELECT
    o.o_name AS Organization,
    COUNT(CASE WHEN e.e_gender = 'm' THEN 1 ELSE NULL END) AS Male_Employees,
    COUNT(CASE WHEN e.e_gender = 'f' THEN 1 ELSE NULL END) AS Female_Employees,
    COUNT(e.e_id) AS Total_Employees
FROM employee e
JOIN organisation o ON e.o_code = o.o_code
GROUP BY o.o_name
ORDER BY Total_Employees DESC;
```

Output

| Organization | Male_Employees | Female_Employees | Total_Employees |
|---|---|---|---|
| Safaricom | 1 | 0 | 1 |
| Nakuru County Gov. | 0 | 1 | 1 |
| Moi University | 1 | 0 | 1 |
| Kisumu Breweries | 1 | 0 | 1 |
| Kenya Ports Authority | 0 | 1 | 1 |

If women or men are underrepresented in specific organizations, it may indicate implicit biases in hiring processes or barriers to entry that need addressing.

The report shows more employment of men as compared to women in total, but a generally balanced pattern of employment per organisation.

# CHAPTER FOUR

## Conclusion

This study underscores the importance of structured gender data monitoring in driving equitable educational and employment opportunities. By implementing a well-structured National Gender Affirmative Monitoring Database, this project enables policymakers and stakeholders to make informed decisions based on comprehensive, gender-disaggregated data. The database's adherence to Third Normal Form (3NF) ensures minimal redundancy and maximum efficiency in data retrieval, fostering transparency in gender-related policies. The insights drawn from the generated reports offer a clear view of disparities in gender representation, salary distributions, and employment trends, serving as a foundation for future gender equality interventions. Moving forward, integrating AI-driven analytics and machine learning techniques can further enhance predictive analysis, providing deeper insights and proactive policy recommendations for sustained gender equity advancements.

**References**


Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Addison-Wesley.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & others. (2020). *Language Models are Few-Shot Learners*. arXiv preprint arXiv:2005.14165.

Chamberlin, D. D., & Boyce, R. F. (1974). *SEQUEL: A Structured English Query Language*. Proceedings of the ACM SIGFIDET Workshop on Data Description, Access and Control.

Chen, P. P. (1976). *The Entity-Relationship Model—Toward a Unified View of Data*. ACM Transactions on Database Systems, 1(1), 9-36.

Codd, E. F. (1970). *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 13(6), 377-387.

Connolly, T., & Begg, C. (2020). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.

Date, C. J. (2019). *An Introduction to Database Systems* (8th ed.). Pearson.

Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.

European Institute for Gender Equality. (2021). *Gender Equality Index 2021*. EIGE Publications.

International Labour Organization. (2022). *Gender Equality in the Workforce: Trends and Challenges*. ILO Reports.

OECD. (2020). *Measuring Gender Equality: OECD Gender Data Portal*.

OpenAI. (2023). *Introducing ChatGPT*. Retrieved from https://openai.com.

Ramez, E., & Shamkant, N. (2021). *Database Management Systems*. McGraw-Hill.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill.

UN Women. (2021). *Gender Equality and Monitoring Frameworks*. Retrieved from https://www.unwomen.org.

UN Women. (2021). *The Role of Gender Monitoring in Achieving Equity*. UN Women Publications.

UNDP. (2019). *Gender and Development: Challenges in Implementation*. United Nations Development Programme.

United Nations. (2015). *Sustainable Development Goals: Goal 5 - Gender Equality*. United Nations.

World Bank. (2021). *Gender Data for Development: A Global Perspective*. World Bank Group.

World Economic Forum. (2022). *Global Gender Gap Report 2022*. World Economic Forum.

# APPENDICES

## I. APPENDIX 1 - Internal Schema

```
/* create county table
*/

CREATE TABLE county (

  c_code INT,

  c_name VARCHAR(20),

  c_population INT,

  c_malepop INT,

  c_femalepop INT,

  PRIMARY KEY (c_code)

);
```

```
/* create organisation
table */

CREATE TABLE
organisation (

  o_code INT,

  c_code INT,

  o_name VARCHAR(50),

  o_employees INT,

  o_malepop INT,

  o_femalepop INT,

  PRIMARY KEY (o_code),

  FOREIGN KEY (c_code)
REFERENCES
county(c_code)

);
```

```
/* create employee type
table */

CREATE TABLE
employee_type (

  type_code CHAR(3),
/*abbreviation of job
title, like engineer =
ENG */

  e_type VARCHAR(20),

  PRIMARY KEY
(type_code)

);
```

```
/* create employee
table */

CREATE TABLE employee (

  e_id INT,

  o_code INT,

  e_fname VARCHAR(20),

  e_lname VARCHAR(20),

  type_code VARCHAR(3),
/*abbreviation of job
title, like engineer =
ENG */

  e_gender CHAR(1), /*
m or f */

  e_salary
NUMBER(10,2),

  PRIMARY KEY (e_id),

  FOREIGN KEY (o_code)
REFERENCES
organisation(o_code),

  FOREIGN KEY
(type_code) REFERENCES
employee_type(type_code
)
```

```
  CONSTRAINT
unique_values CHECK
(e_gender in ('m','f'))

);
```

```
/* create hire table */

CREATE TABLE hire (

  e_id INT,

  o_code INT,

  hire_date DATE,

  years_of_employment
INT,

  PRIMARY KEY (e_id,
o_code),

  FOREIGN KEY (e_id)
REFERENCES
employee(e_id),

  FOREIGN KEY (o_code)
REFERENCES
organisation(o_code)

);
```

```
/* create salary table
*/

CREATE TABLE salary (

  type_code INT,

  o_code INT,

  avg_salary NUMBER(10,
2),

  PRIMARY KEY
(type_code, o_code),

  FOREIGN KEY (o_code)
REFERENCES
organisation(o_code)
```

```
);


/* create school table
*/

CREATE TABLE school (

  s_id       INT,

  c_code INT,

  s_name VARCHAR(20),

  s_population INT,

  s_malepop INT,

  s_femalepop INT,

  PRIMARY KEY (s_id),

  FOREIGN KEY (c_code)
REFERENCES
county(c_code)

);


/* create student table
*/

CREATE TABLE student (

  st_id INT,

  s_id INT,

  s_fname VARCHAR(20),

  s_lname VARCHAR(20),

  s_gender CHAR(1),

  s_onScholarship
CHAR(1),

  PRIMARY KEY (st_id)

  CONSTRAINT
unique_values CHECK
(s_gender in
('m','f')),

  CONSTRAINT
unique_values CHECK
(s_onScholarship in
('y','n'))

);


/* create
on_scholarship table */

CREATE TABLE
on_scholarship (

  st_id INT,

  scholarship
VARCHAR(20),

  tuition NUMBER(10,
2),

  PRIMARY KEY (st_id),

  FOREIGN KEY (st_id)
REFERENCES
student(st_id)

);


/* create
no_scholarship table */

CREATE TABLE
no_scholarship (

  st_id INT,

  tuition NUMBER(10,
2),

  PRIMARY KEY (st_id),

  FOREIGN KEY (st_id)
REFERENCES
student(st_id)

);


/* create enroll table
*/

CREATE TABLE enroll (

  st_id INT,

  s_id INT,

  enroll_date DATE,

  enroll_year
VARCHAR(10),
/*freshman, sophomore,
junior, senior*/

  PRIMARY KEY (st_id,
s_id),

  FOREIGN KEY (st_id)
REFERENCES
student(st_id),

  FOREIGN KEY (s_id)
REFERENCES school(s_id)

);


/* create degree table
*/

CREATE TABLE degree (

  deg_code CHAR(3),
/*Abbreviation of
degree name, e.g
Accounting = ACC*/

  deg_name VARCHAR(30),

  PRIMARY KEY
(deg_code)

);


/* create gpa table */

CREATE TABLE gpa (

  grade CHAR(1), /* A,
B, C, D, F */

  g_start INT, /* e.g A
starts at 90 */

  g_end INT, /*eng A
ends at 100*/
```

```
  standing VARCHAR(10),
/*good or bad*/

  PRIMARY KEY (grade)

  CONSTRAINT
unique_values CHECK
(standing in
('good','bad'))

);



/* create graduate
table */

CREATE TABLE graduate (

  st_id INT,

  s_id INT,

  g_date DATE,

  grade CHAR(1), /* A,
B, C, D, F */

  deg_code CHAR(3),
/*abbreviation of
degree name, e.g
Psychology = PSY */

  PRIMARY KEY (st_id,
s_id),

  FOREIGN KEY (st_id)
REFERENCES
student(st_id),

  FOREIGN KEY (s_id)
REFERENCES
school(s_id),

  FOREIGN KEY
(deg_code) REFERENCES
degree(deg_code),

  FOREIGN KEY (grade)
REFERENCES gpa(grade)

);




Records
```

```
County table

INSERT INTO county
(c_code, c_name,
c_population,
c_malepop, c_femalepop)

VALUES

(001, 'Mombasa',
1208333, 609157,
599176),

(002, 'Kwale', 866820,
429328, 437492),

(003, 'Kilifi',
1453787, 708480,
745307),

(004, 'Tana River',
315943, 159364,
156579),

(005, 'Lamu', 143920,
69767, 74153),

(006, 'Taita Taveta',
340671, 168091,
172580),

(007, 'Garissa',
841353, 440610,
400743),

(008, 'Wajir', 781263,
402527, 378736),

(009, 'Mandera',
867457, 459358,
408099),

(010, 'Marsabit',
459785, 234567,
225218);




Employment
tables(organisation,
employee_type,
employee, hire, salary)

/*insert records into
organisation table*/
```

```
INSERT INTO
organisation VALUES

(1001, 001, 'ABC Inc.',
50, 25, 25),

(1002, 002, 'XYZ
Corp.', 100, 60, 40),

(1003, 003, 'Acme Co.',
200, 100, 100),

(1004, 004, 'Beta
Corp.', 75, 30, 45),

(1005, 005, 'Gamma
LLC', 150, 75, 75),

(1006, 006, 'Delta
Corp.', 80, 40, 40),

(1007, 007, 'Epsilon
Inc.', 30, 10, 20),

(1008, 008, 'Theta
Corp.', 125, 70, 55),

(1009, 009, 'Iota LLC',
50, 20, 30),

(1010, 010, 'Kappa
Co.', 90, 45, 45);



/*insert records into
employee_type table*/

INSERT INTO
employee_type VALUES

('ACC', 'Accountant'),

('ADM',
'Administrator'),

('ENG', 'Engineer'),

('EXC', 'Company
Executive'),

('HRM', 'Human
Resources Manager'),

('INT', 'Intern'),

('MRK', 'Marketing’),

('PRG', 'Programmer'),
```

```
('SAL', 'Salesperson'),

('OPR', 'Operations
manager');



/*insert records into
employee table*/

INSERT INTO employee
VALUES

(600001, 1001, 'John',
'Doe', 'ENG', 'm',
70000.00),

(600002, 1003, 'Jane',
'Smith', 'HRM', 'f',
85000.00),

(600003, 1002, 'David',
'Brown', 'INT', 'm',
90000.00),

(600004, 1001, 'Linda',
'Johnson', 'ACC', 'f',
55000.00),

(600005, 1002,
'Michael', 'Wilson',
'ENG', 'm', 80000.00),

(600006, 1005, 'Sarah',
'Taylor', 'MRK', 'f',
65000.00),

(600007, 1004,
'Matthew', 'Anderson',
'OPR', 'm', 95000.00),

(600008, 1003, 'Emily',
'Clark', 'SAL', 'f',
75000.00),

(600009, 1005,
'Daniel', 'Martinez',
'PRG', 'm', 85000.00),

(600010, 1004,
'Amanda', 'Lee', 'EXC',
'f', 70000.00);



/*insert records into
hire table*/

INSERT INTO hire VALUES

(600001, 1001,
'2010-05-12', 11),

(600002, 1003,
'2018-09-20', 3),

(600003, 1002,
'2015-02-28', 6),

(600004, 1001,
'2012-07-11', 9),

(600005, 1002,
'2019-04-30', 2),

(600006, 1005,
'2016-11-18', 5),

(600007, 1004,
'2014-03-15', 8),

(600008, 1003,
'2017-06-22', 4),

(600009, 1005,
'2013-08-08', 8),

(600010, 1004,
'2021-01-10', 1);



/*insert records into
salary*/

INSERT INTO salary
VALUES

('ACC', 1001,
85000.00),

('ADM', 1001,
125000.00),

('ENG', 1002,
65000.00),

('EXC', 1002,
55000.00),

('HRM', 1003,
45000.00),

('INT', 1003,
60000.00),

('MRK', 1004,
80000.00),

('PRG', 1004,
130000.00),

('SAL', 1005,
70000.00),

('OPR', 1005,
40000.00);



Education
tables(school, student,
on_scholarship,
no_scholarship, enroll,
degree, gpa, graduate)



                /*insert
into school*/

INSERT INTO school
(s_id, c_code, s_name,
s_population,
s_malepop, s_femalepop)

VALUES

(1, 001, 'Springfield
High School', 1000,
500, 500),

(2, 002, 'Oakland
Elementary', 750, 350,
400),

(3, 002, 'Maplewood
Middle School', 900,
450, 450),

(4, 003, 'Northridge
High School', 1100,
600, 500),

(5, 004, 'Lincoln
Elementary', 600, 300,
300),

(6, 005, 'Roosevelt
Middle School', 800,
400, 400),

(7, 006, 'Jefferson
Elementary', 650, 300,
350),
```

```
(8, 008, 'Westfield
High School', 950, 500,
450),

(9, 009, 'Hillside
Elementary', 500, 250,
250),

(10, 010, 'Fairview
Middle School', 700,
350, 350);



/*insert into student*/

INSERT INTO student
(st_id, s_id, s_fname,
s_lname, s_gender,
s_onScholarship)

VALUES

  (1, 1, 'John', 'Doe',
'm', 'y'),

  (2, 2, 'Jane',
'Smith', 'f', 'n'),

  (3, 3, 'Bob',
'Johnson', 'm', 'n'),

  (4, 4, 'Alice',
'Williams', 'f', 'y'),

  (5, 5, 'Mark',
'Davis', 'm', 'n'),

  (6, 6, 'Sara', 'Lee',
'f', 'y'),

  (7, 7, 'David',
'Brown', 'm', 'n'),

  (8, 8, 'Emily',
'Taylor', 'f', 'n'),

  (9, 9, 'Michael',
'Wilson', 'm', 'y'),

  (10, 10, 'Jennifer',
'Thomas', 'f', 'y')

  (11, 1, 'Oliver',
'Doe', 'm', 'y'),

  (12, 2, 'Olivia',
'Smith', 'f', 'n'),

  (13, 3, 'Bill',
'Johnson', 'm', 'n'),

  (14, 4, 'Alayna',
'Williams', 'f', 'y'),

  (15, 5, 'Matt',
'Davis', 'm', 'n'),

  (16, 6, 'Sarah',
'Lee', 'f', 'y'),

  (17, 7, 'Dave',
'Brown', 'm', 'n'),

  (18, 8, 'Emilia',
'Taylor', 'f', 'n'),

  (19, 9, 'Mikey',
'Wilson', 'm', 'y'),

  (20, 10, 'Jeanine',
'Thomas', 'f', 'y')



/*insert into enroll*/

INSERT INTO enroll
(st_id, s_id,
enroll_date,
enroll_year)

VALUES

(1, 1, '2021-09-01',
'freshman'),

(2, 2, '2021-09-01',
'junior'),

(3, 3, '2021-09-01',
'sophomore'),

(4, 4, '2021-09-01',
'senior'),

(5, 5, '2021-09-01',
'freshman'),

(6, 6, '2021-09-01',
'sophomore'),

(7, 7, '2021-09-01',
'freshman'),

(8, 8, '2021-09-01',
'senior'),

(9, 9, '2021-09-01',
'junior'),

(10, 10, '2021-09-01',
'freshman');



/*insert into
on_scholarship*/

INSERT INTO
on_scholarship (st_id,
scholarship, tuition)

VALUES

  (1, 'Merit
Scholarship', 5000.00),

  (2, 'Athletic
Scholarship', 7500.00),

  (4, 'Need-based
Scholarship',
10000.00),

  (5, 'Merit
Scholarship', 5000.00),

  (6, 'Athletic
Scholarship', 7500.00),

  (7, 'Need-based
Scholarship',
10000.00),

  (8, 'Merit
Scholarship', 5000.00),

  (9, 'Athletic
Scholarship', 7500.00),

  (10, 'Need-based
Scholarship',
10000.00),

  (3, 'Merit
Scholarship', 5000.00);



/*insert into
no_scholarship*/

INSERT INTO
no_scholarship (st_id,
tuition)
```

```
VALUES
  (11, 10000.00),
  (12, 12000.00),
  (13, 9000.00),
  (14, 11000.00),
  (15, 8000.00),
  (16, 13000.00),
  (17, 9500.00),
  (18, 11500.00),
  (19, 7500.00),
  (20, 14000.00);


/*insert into degree table*/
INSERT INTO degree (deg_code, deg_name)
VALUES
  ('CSI', 'Computer Science'),
  ('ENG', 'English'),
  ('BIO', 'Biology'),
  ('GDE', 'Graphic Design'),
  ('PSY', 'Psychology'),
  ('HIS', 'History'),
  ('MTH', 'Mathematics'),
  ('CHM', 'Chemistry'),
  ('PSC', 'Political Science'),
  ('ENV', 'Environmental Science');


/*insert into gpa table*/
INSERT INTO gpa (grade, g_start, g_end, standing)
VALUES
  ('A', 90, 100, 'good'),
  ('B', 80, 89, 'good'),
  ('C', 70, 79, 'good'),
  ('D', 60, 69, 'bad'),
  ('F', 50, 59, 'bad'),
  ('G', 40, 49, 'bad'),
  ('H', 30, 39, 'bad'),
  ('I', 20, 29, 'bad'),
  ('J', 10, 19, 'bad'),
  ('K', 0, 9, 'bad');


/*insert into graduate*/
/*option 1*/
INSERT INTO graduate (st_id, s_id, g_date, grade, deg_code)
VALUES
  (1, 1, '2022-05-20', 'A', 'CHM'),
  (2, 2, '2021-12-15', 'B', 'ENG'),
  (3, 3, '2022-05-20', 'A', 'BIO'),
  (4, 4, '2021-12-15', 'C', 'HIS'),
  (5, 5, '2022-05-20', 'B', 'ENV'),
  (6, 6, '2021-12-15', 'A', 'CSI'),
  (7, 7, '2022-05-20', 'C', 'PSC'),
  (8, 8, '2021-12-15', 'D', 'PSC'),
  (9, 9, '2022-05-20', 'B', 'ENG'),
  (10, 10, '2021-12-15', 'F', 'BIO');
```

# II. APPENDIX 2 - SCREENSHOTS OF DATABASE CREATION

## A. CREATING THE TABLES

County table created

```
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.

/* create county table */
CREATE TABLE county (
  c_code INT,
  c_name VARCHAR(20),
  c_population INT,
  c_malepop INT,
  c_femalepop INT,
  PRIMARY KEY (c_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Organisation table created

```
/* create organisation table */
CREATE TABLE organisation (
  o_code INT,
  c_code INT,
  o_name VARCHAR(50),
  o_employees INT,
  o_malepop INT,
  o_femalepop INT,
  PRIMARY KEY (o_code),
  FOREIGN KEY (c_code) REFERENCES county(c_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Employee type table created

```
/* create employee type table */
CREATE TABLE employee_type (
  type_code CHAR(3), /*abbreviation of job title, like engineer = ENG */
  e_type VARCHAR(20),
  PRIMARY KEY (type_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Employee table created

```
/* create employee table */
CREATE TABLE employee (
  e_id INT,
  o_code INT,
  e_fname VARCHAR(20),
  e_lname VARCHAR(20),
  type_code VARCHAR(3), /*abbreviation of job title, like engineer = ENG */
  e_gender CHAR(1), /* m or f */
  e_salary NUMBER(10,2),
  PRIMARY KEY (e_id),
  FOREIGN KEY (o_code) REFERENCES organisation(o_code),
  FOREIGN KEY (type_code) REFERENCES employee_type(type_code)
  CONSTRAINT unique_values CHECK (e_gender in ('m','f'))
```

Output

SQL query successfully executed. However, the result set is empty.

Hire table created

```
/* create hire table */
CREATE TABLE hire (
  e_id INT,
  o_code INT,
  hire_date DATE,
  years_of_employment INT,
  PRIMARY KEY (e_id, o_code),
  FOREIGN KEY (e_id) REFERENCES employee(e_id),
  FOREIGN KEY (o_code) REFERENCES organisation(o_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Salary table created

```
/* create salary table */
CREATE TABLE salary (
  type_code INT,
  o_code INT,
  avg_salary NUMBER(10, 2),
  PRIMARY KEY (type_code, o_code),
  FOREIGN KEY (o_code) REFERENCES organisation(o_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

School table created

```
/* create school table */
CREATE TABLE school (
  s_id INT,
  c_code INT,
  s_name VARCHAR(20),
  s_population INT,
  s_malepop INT,
  s_femalepop INT,
  PRIMARY KEY (s_id),
  FOREIGN KEY (c_code) REFERENCES county(c_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Student table created

```
/* create student table */
CREATE TABLE student (
  st_id INT,
  s_id INT,
  s_fname VARCHAR(20),
  s_lname VARCHAR(20),
  s_gender CHAR(1),
  s_onScholarship CHAR(1),
  PRIMARY KEY (st_id)
  CONSTRAINT unique_values CHECK (s_gender in ('m','f')),
  CONSTRAINT unique_values CHECK (s_onScholarship in ('y','n'))
);
```

Output

SQL query successfully executed. However, the result set is empty.

On_scholarship table created

```
/* create on_scholarship table */
CREATE TABLE on_scholarship (
  st_id INT,
  scholarship VARCHAR(20),
  tuition NUMBER(10, 2),
  PRIMARY KEY (st_id),
  FOREIGN KEY (st_id) REFERENCES student(st_id)
);
```

Output

SQL query successfully executed. However, the result set is empty.

No_scholarship table created

```
/* create no_scholarship table */
CREATE TABLE no_scholarship (
  st_id INT,
  tuition NUMBER(10, 2),
  PRIMARY KEY (st_id),
  FOREIGN KEY (st_id) REFERENCES student(st_id)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Enroll table created

```
/* create enroll table */
CREATE TABLE enroll (
  st_id INT,
  s_id INT,
  enroll_date DATE,
  enroll_year VARCHAR(10), /*freshman, sophomore, junior, senior*/
  PRIMARY KEY (st_id, s_id),
  FOREIGN KEY (st_id) REFERENCES student(st_id),
  FOREIGN KEY (s_id) REFERENCES school(s_id)
);
```

Output

SQL query successfully executed. However, the result set is empty.

Degree table created

```
/* create degree table */
CREATE TABLE degree (
  deg_code CHAR(3), /*Abbreviation of degree name, e.g Accounting = ACC*/
  deg_name VARCHAR(30),
  PRIMARY KEY (deg_code)
);
```

Output

SQL query successfully executed. However, the result set is empty.

GPA table created

```
/* create gpa table */
CREATE TABLE gpa (
  grade CHAR(1), /* A, B, C, D, F */
  g_start INT, /* e.g A starts at 90 */
  g_end INT, /*eng A ends at 100*/
  standing VARCHAR(10), /*good or bad*/
  PRIMARY KEY (grade)
  CONSTRAINT unique_values CHECK (standing in ('good','bad'))
);
```

Output

SQL query successfully executed. However, the result set is empty.

Graduate table created

```
/* create graduate table */
CREATE TABLE graduate (
  st_id INT,
  s_id INT,
  g_date DATE,
  grade CHAR(1), /* A, B, C, D, F */
  deg_code CHAR(3), /*abbreviation of degree name, e.g Psychology = PSY */
  PRIMARY KEY (st_id, s_id),
  FOREIGN KEY (st_id) REFERENCES student(st_id),
  FOREIGN KEY (s_id) REFERENCES school(s_id),
  FOREIGN KEY (deg_code) REFERENCES degree(deg_code),
  FOREIGN KEY (grade) REFERENCES gpa(grade)
);
```

Output

SQL query successfully executed. However, the result set is empty.

## B. PUTTING RECORDS INTO THE TABLES

Records into county table

```
INSERT INTO county (c_code, c_name, c_population, c_malepop, c_femalepop)
VALUES
(001, 'Mombasa', 1208333, 609157, 599176),
(002, 'Kwale', 866820, 429328, 437492),
(003, 'Kilifi', 1453787, 708488, 745307),
(004, 'Tana River', 315043, 159364, 156579),
(005, 'Lamu', 143920, 69767, 74153),
(006, 'Taita Taveta', 340671, 168091, 172580),
(007, 'Garissa', 841353, 440818, 400743),
(008, 'Wajir', 781263, 402527, 378736),
(009, 'Mandera', 867457, 459358, 408099),
(010, 'Marsabit', 459785, 234567, 225218);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into organisation table

```
/*insert records into organisation table*/
INSERT INTO organisation VALUES
(1001, 001, 'ABC Inc.', 50, 25, 25),
(1002, 002, 'XYZ Corp.', 100, 60, 40),
(1003, 003, 'Acme Co.', 200, 100, 100),
(1004, 004, 'Beta Corp.', 75, 30, 45),
(1005, 005, 'Gamma LLC', 150, 75, 75),
(1006, 006, 'Delta Corp.', 80, 40, 40),
(1007, 007, 'Epsilon Inc.', 30, 10, 20),
(1008, 008, 'Theta Corp.', 125, 70, 55),
(1009, 009, 'Iota LLC', 50, 20, 30),
(1010, 010, 'Kappa Co.', 90, 45, 45);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into employee type table

```
/*insert records into employee_type table*/
INSERT INTO employee_type VALUES
('ACC', 'Accountant'),
('ADM', 'Administrator'),
('ENG', 'Engineer'),
('EXC', 'Company Executive'),
('HRM', 'Human Resources Manager'),
('INT', 'Intern'),
('MRK', 'Marketing'),
('PRG', 'Programmer'),
('SAL', 'Salesperson'),
('OPR', 'Operations manager');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into employee table

```
/*insert records into employee table*/
INSERT INTO employee VALUES
(600001, 1001, 'John', 'Doe', 'ENG', 'm', 70000.00),
(600002, 1001, 'Jane', 'Smith', 'HRM', 'f', 85000.00),
(600003, 1002, 'David', 'Brown', 'INT', 'm', 50000.00),
(600004, 1001, 'Linda', 'Johnson', 'ACC', 'f', 55000.00),
(600005, 1002, 'Michael', 'Wilson', 'ENG', 'm', 80000.00),
(600006, 1001, 'Sarah', 'Taylor', 'MRK', 'f', 65000.00),
(600007, 1004, 'Matthew', 'Anderson', 'OPR', 'm', 95000.00),
(600008, 1003, 'Emily', 'Clark', 'SAL', 'f', 75000.00),
(600009, 1005, 'Daniel', 'Martinez', 'PRG', 'm', 85000.00),
(600010, 1004, 'Amanda', 'Lee', 'EXC', 'f', 70000.00);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into hire table

```
/*insert records into hire table*/
INSERT INTO hire VALUES
(600001, 1001, '2010-05-12', 11),
(600002, 1003, '2018-09-20', 3),
(600003, 1002, '2015-02-28', 6),
(600004, 1001, '2012-07-11', 9),
(600005, 1002, '2019-04-30', 2),
(600006, 1005, '2016-11-18', 5),
(600007, 1004, '2014-03-15', 8),
(600008, 1003, '2017-06-22', 4),
(600009, 1005, '2013-08-08', 8),
(600010, 1004, '2021-01-10', 1);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into salary table

```
/*insert records into salary*/
INSERT INTO salary VALUES
('ACC', 1001, 85000.00),
('ADM', 1001, 125000.00),
('ENG', 1002, 65000.00),
('EXC', 1001, 55000.00),
('HRM', 1003, 45000.00),
('INT', 1003, 60000.00),
('MRK', 1004, 80000.00),
('PRG', 1004, 130000.00),
('SAL', 1005, 70000.00),
('OPR', 1005, 40000.00);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into school table

```
/*insert into school*/
INSERT INTO school (s_id, c_code, s_name, s_population, s_malepop, s_
VALUES
(1, 001, 'Springfield High School', 1000, 500, 500),
(2, 002, 'Oakland Elementary', 750, 350, 400),
(3, 003, 'Maplewood Middle School', 900, 450, 450),
(4, 003, 'Northridge High School', 1100, 600, 500),
(5, 004, 'Lincoln Elementary', 600, 300, 300),
(6, 005, 'Roosevelt Middle School', 800, 400, 400),
(7, 006, 'Jefferson Elementary', 650, 300, 350),
(8, 008, 'Westfield High School', 950, 500, 450),
(9, 009, 'Hillside Elementary', 500, 250, 250),
(10, 010, 'Fairview Middle School', 700, 350, 350);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into student table

```
/*insert into student*/
INSERT INTO student (st_id, s_id, s_fname, s_lname, s_gender, s_onSc
VALUES
  (1, 1, 'John', 'Doe', 'm', 'y'),
  (2, 2, 'Jane', 'Smith', 'f', 'n'),
  (3, 3, 'Bob', 'Johnson', 'm', 'n'),
  (4, 4, 'Alice', 'Williams', 'f', 'y'),
  (5, 5, 'Mark', 'Davis', 'm', 'n'),
  (6, 6, 'Sara', 'Lee', 'f', 'y'),
  (7, 7, 'David', 'Brown', 'm', 'n'),
  (8, 8, 'Emily', 'Taylor', 'f', 'n'),
  (9, 9, 'Michael', 'Wilson', 'm', 'y'),
  (10, 10, 'Jennifer', 'Thomas', 'f', 'y'),
  (11, 1, 'Oliver', 'Doe', 'm', 'y'),
  (12, 2, 'Olivia', 'Smith', 'f', 'n');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into enroll table

```
/*insert into enroll*/
INSERT INTO enroll (st_id, s_id, enroll_date, enroll_year)
VALUES
(1, 1, '2021-09-01', 'freshman'),
(2, 2, '2021-09-01', 'junior'),
(3, 3, '2021-09-01', 'sophomore'),
(4, 4, '2021-09-01', 'senior'),
(5, 5, '2021-09-01', 'freshman'),
(6, 6, '2021-09-01', 'sophomore'),
(7, 7, '2021-09-01', 'freshman'),
(8, 8, '2021-09-01', 'senior'),
(9, 9, '2021-09-01', 'junior'),
(10, 10, '2021-09-01', 'freshman');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into on_scholarship table

```
/*insert into on_scholarship*/
INSERT INTO on_scholarship (st_id, scholarship, tuition)
VALUES
  (1, 'Merit Scholarship', 5000.00),
  (2, 'Athletic Scholarship', 7500.00),
  (4, 'Need-based Scholarship', 10000.00),
  (5, 'Merit Scholarship', 5000.00),
  (6, 'Athletic Scholarship', 7500.00),
  (7, 'Need-based Scholarship', 10000.00),
  (8, 'Merit Scholarship', 5000.00),
  (9, 'Athletic Scholarship', 7500.00),
  (10, 'Need-based Scholarship', 10000.00),
  (3, 'Merit Scholarship', 5000.00);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into no_scholarship table

```
/*insert into no_scholarship*/
INSERT INTO no_scholarship (st_id, tuition)
VALUES
  (11, 10000.00),
  (12, 12000.00),
  (13, 9000.00),
  (14, 11000.00),
  (15, 8000.00),
  (16, 13000.00),
  (17, 9500.00),
  (18, 11500.00),
  (19, 7500.00),
  (20, 14000.00);
```

Output

SQL query successfully executed. However, the result set is empty.

Records into degree table

```
/*insert into degree table*/
INSERT INTO degree (deg_code, deg_name)
VALUES
  ('CSI', 'Computer Science'),
  ('ENG', 'English'),
  ('BIO', 'Biology'),
  ('GDE', 'Graphic Design'),
  ('PSY', 'Psychology'),
  ('HIS', 'History'),
  ('MTH', 'Mathematics'),
  ('CHM', 'Chemistry'),
  ('PSC', 'Political Science'),
  ('ENV', 'Environmental Science');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into gpa table

```
/*insert into gpa table*/
INSERT INTO gpa (grade, g_start, g_end, standing)
VALUES
  ('A', 90, 100, 'good'),
  ('B', 80, 89, 'good'),
  ('C', 70, 79, 'good'),
  ('D', 60, 69, 'bad'),
  ('F', 50, 59, 'bad'),
  ('G', 40, 49, 'bad'),
  ('H', 30, 30, 'bad'),
  ('I', 20, 29, 'bad'),
  ('J', 10, 19, 'bad'),
  ('K', 0, 9, 'bad');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into graduate table

```
/*insert into graduate table*/
INSERT INTO graduate (st_id, g_date, grade, deg_code)
VALUES
  (1, 1, '2022-05-20', 'A', 'CHM'),
  (2, 2, '2021-12-15', 'B', 'ENG'),
  (3, 3, '2022-05-20', 'A', 'BIO'),
  (4, 4, '2021-12-15', 'C', 'HIS'),
  (5, 5, '2022-05-20', 'B', 'ENV'),
  (6, 6, '2021-12-15', 'A', 'CSI'),
  (7, 7, '2022-05-20', 'C', 'PSC'),
  (8, 8, '2021-12-15', 'D', 'PSC'),
  (9, 9, '2022-05-20', 'B', 'ENG'),
  (10, 10, '2021-12-15', 'F', 'BIO');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into graduate table

```
/*insert into graduate*/
/*option 2*/
INSERT INTO graduate (st_id, s_id, g_date, grade, deg_code)
VALUES
  (1, 1, '2022-05-20', 'A', 'CHM'),
  (2, 2, '2021-12-15', 'B', 'ENG'),
  (3, 3, '2022-05-20', 'A', 'BIO'),
  (4, 4, '2021-12-15', 'C', 'HIS'),
  (5, 5, '2022-05-20', 'B', 'ENV'),
  (6, 6, '2021-12-15', 'A', 'CSI'),
  (7, 7, '2022-05-20', 'C', 'PSC'),
  (8, 8, '2021-12-15', 'D', 'PSC'),
  (9, 9, '2022-05-20', 'B', 'ENG'),
  (10, 10, '2021-12-15', 'F', 'BIO');
```

Output

SQL query successfully executed. However, the result set is empty.

Records into gpa table

```
/*insert into gpa table*/
INSERT INTO gpa (grade, g_start, g_end, standing)
VALUES
  ('A', 90, 100, 'good'),
  ('B', 80, 89, 'good'),
  ('C', 70, 79, 'good'),
  ('D', 60, 69, 'bad'),
  ('F', 50, 59, 'bad'),
  ('G', 40, 49, 'bad'),
  ('H', 30, 30, 'bad'),
  ('I', 20, 29, 'bad'),
  ('J', 10, 19, 'bad'),
  ('K', 0, 9, 'bad');
```

Output

SQL query successfully executed. However, the result set is empty.