National Teachers College
Office of the Vice President for Academic Affairs
School of Teacher Education

"A Generation Falling Behind: The Learnings of Students through the Lens of Corruption"

SDG 4 - QUALITY EDUCATION (TARGET 4.1)

Group Members:

Apolinario, Christian Jolo

Ermitanio, Jerpy Hans

Fabellon, Karyl Hyacinth

Riel, Franz Adrian

Tabios, Jenny

Professor | Ms. Justin Louise R. Neypes

Database Management System

December 2025

TABLE OF CONTENTS

I. Introduction

1.1. Project Overview & UN SDG Target

This study begins with the point of view that the educational quality is greatly determined by the management of school resources and that corruption or mismanagement indirectly influence students' learning experiences negatively. If the money intended for educational materials, classrooms, and teacher training is not only stolen but also wasted or misused, the education system becomes not only weak but also unequal. Students have no choice but to study in classrooms that are either poorly supplied, or are using old materials, or worse, there is no proper equipment at all. These situations are very hard for the students to grasp the lessons, take part in discussions and apply the necessary skills for the future. Gradually, the disadvantages result in a great learning gap, poor academic performance, and the students losing the opportunity to realize their full potential.

This perspective strongly connects with UN SDG Target 4.1, which aims to ensure that all children receive free, equitable, and quality primary and secondary education by 2030. For education to be truly free, students should not be required to pay for or provide basic items that public schools are supposed to supply. However, many families still face hidden costs such as notebooks, school supplies, and learning materials expenses that become heavy especially for households with very low income. These indirect costs become barriers that prevent learners from participating fully in school, making the learning experience unequal and unfair. The lack of free and complete materials contradicts the promise of SDG 4.1 and reveals how financial burdens continue to affect the educational journey of many students.

To achieve equitable and quality education, all students regardless of their family background must have equal access to learning resources. Schools must be equipped with enough supplies, well-trained teachers, safe facilities, and proper funding to support every learner. However, corruption in the education system holds back this goal by reducing the resources schools can use, delaying procurement processes, and limiting the quality of materials bought for students. When funds are not used honestly and responsibly, the entire learning environment weakens, and students are left with fewer opportunities to succeed.

Through examining these issues, this study emphasizes the importance of transparency, accountability, and proper management of school funds as key elements in providing every child with a fair, meaningful, and effective learning experience.

1.2. Problem Statement

UN SDG Target 4.1: Free, Equitable, and Quality Education stresses the importance of offering all children and pupils access to fully and properly equipped learning environments. However, in several schools, this objective is partially breached because of doubtful and unclear information on educational funding. A corruptive practice, inadequate reporting, and inconsistent documentation all together cut down on the basic learning materials that schools are supposed to receive, e.g. books, chairs, laboratory tools, and other supplies. The paper presents a procurement system that enhances the methods of information gathering and providing access and organization of the information, thereby fostering openness and accountability in the management of school funds. It is aimed at overcoming the following real-life issues

- Documentary records of school purchases that are incomplete and unreliable thus resulting in huge difficulty in tracing how funds for books, supplies, and equipment are actually spent.
- Procurement processes that lack transparency and create opportunities for corruption, overpricing, and the misuse of funds meant for educational materials.
- Suppliers and pricing records that are inconsistent and disorganized, leading to delays in acquiring school supplies and the school not being able to select suppliers who are both trustworthy and cost-efficient due to lack of good records.
- Limited access to updated procurement data would be the primary reason for late deliveries of learning materials and thus the students' learning experience being disrupted.

- Inaccessible or poorly maintained records are the main reasons why the detection of irregularities has become difficult, thereby making it hard for the administrators to pinpoint the missing funds, fake transactions, or overpriced items.
- Lack of proper information to identify which departments or classrooms are in need of resources results in unequal distribution of school supplies and widening learning gaps among students.
- The lack of a proper system for monitoring procurement requests results in a mix-up of documents, lost papers, and slow to approve requests which all in turn affect the learning materials availability that is not on time.
- The system, by dealing with these challenges, helps in the achievement of SDG 4.1 by facilitating the management of funds in a clear manner, the delivery of school supplies on time, and by providing an area of study where only the resources that the learners need are the ones that they are able to access.

1.3. Scope and Limitations

Scope

The main goal of this study is to design a Procurement and Inventory Management System (PIMS) that is tailored to the needs of one public school in Manila, and thus helping them to manage and trace school supplies by means of structured database features like tables, triggers, stored procedures, and views. The system will provide management tools for departments, suppliers, items, inventory, procurement requests, and delivery records in order to facilitate transparency and proper resource allocation in the school supporting SDG Target 4.1. The system is designed to handle the purchasing of educational materials by keeping track of requests from departments, suppliers, items, and inventory changes. Users can send procurement requests by a stored procedure in the system, which also ensures data consistency by using transactions and updates the inventory automatically when the status of the requests is changed to Delivered. There are also views that serve as a dashboard for procurement activities.

Limitations

The system performs solely at the database level and necessitates SQL-based interaction since a user interface is not available. Moreover, the research covers only the procurement aspect of a single school and does not take into account broader educational factors like teaching quality, curriculum development, or the physical condition of the school.

II. Requirements & Analysis

2.1. Functional Requirements and Non-Functional Requirements

This part of the paper shows the requirements for the Procurement and Inventory Management System (PIMS).

Here we will show the lists of features using tables. It will be categorized into two (2) categories or database architecture, the Functional Requirements and the Non-Functional Requirements. The Functional Requirements is the one which defines the essential process in the system. While the said Non-Functional Requirements is the one who defines the quality of the system attributes and its constraints. The said two (2) requirements are directly derived from the supplied triggers, stored procedures and most importantly the database schema.

Table 1. Functional Requirements

ID	Features Category	Alignment	Compliance
FR1	***Core Entity Management :*** This is where the system gives or must provide a comprehensive management (CRUD operation) for our primary reference data	Defined by the standalone tables: *departments, suppliers, and items.*	**Fully Met:** This is verified by the successful creation of the tables and manipulation of base entity records.

	entities which are the following, **Departments**, **Suppliers**, and the **Inventory Items**.		
FR2	***Procurement Initiation:*** The system must allow them—the users—to create and submit a new procurement request which captures the requesting department, request dates, and initially the status of the system into '**Pending**'.	Core function of the *procurement_requests* table and the *submit_procurement* stored procedure.	**Fully Met:** The procedure successfully captured the request and initialized status correctly.
FR2.1	***Detailed Item Specification:*** The system must allow users to link specific inventory items,designated suppliers, and required quantities to be submitted procurement requests.	Defined and enforced by the structure and foreign keys in the *procurement_details* table linking *request_id, item_id,* and *supplier_id.*	**Fully Met:** Referential integrity ensures proper linking of all transaction details that the system has.
FR3	***Status Workflow Management:*** The system must of course support the sequential update of a procurement request's status using the mandated states which are the **'Pending'**, **'Approved'**, and the **'Delivered'**	Defined by the *status ENUM* constraint in the *procurement_requests* table.	**Fully Met:** The status updates are constrained to defined and mandated sequence. A validation trigger also enforces that users cannot skip the "Approved" status. Any attempt to change a request directly from "Pending" to "Delivered" will

			automatically be blocked by the system.
FR4	***Automated Inventory Update:*** The system must automatically trigger an update to item stock levels in the **'inventory'** table. This is whenever a linked procurement request is officially marked as **'Delivered'**	Directly implemented by the *trg_update_inventory* trigger.	**Fully Met:** The trigger executed and executes automatically upon the status change to *'Delivered'*.
FR5	***Reporting: Pending Requests:*** The system must provide a view that will summarize all the currently **'Pending'** procurement requests, detailing the Request ID, Requesting Department Name, and Request Date.	Implemented via the 'CREATE VIEW' *vw_pending_requests* database view.	**Fully Met:** The view returns required filtered data for reporting.
FR5.1	***Reporting: Item Consumption Analysis:*** *The system must provide an aggregated view showing the total quantity procured for each specific item across all requests.*	Implemented via the 'CREATE VIEW' *vw_item_procurement_summary* database view.	**Fully Met:** The view has successfully aggregated consumption data per item that will be input in the system.
FR5.2	***Reporting: Supplier Volume:*** The system must report the total number of individual transaction details processed	This is implemented via the 'CREATE VIEW' *vw_supplier_transactions* database view.	**Fully Met:** The view has accurately reported transaction volume per supplier.

:	with each supplier.		

Functional Requirements Lists

The table shows the Functional Requirements, where it is defining what the PIMS or the Procurement and Inventory Management System must **do** to manage the given procurement and inventory lifecycle processes.

Table 2. Non-Functional Requirements

ID	Requirement	Metric	Compliance
NFR1	***The Transaction Atomicity:*** This is the multi-step process for the submission or the process of submitting a new procurement request must adhere to the atomicity. This is to ensure that all succeed or fail as a unified single transaction	Enforced by the **START TRANSACTION; ... COMMIT;** block in the *submit_procurement* stored procedure.	**Fully Met:** ACID properties enforced by transaction control.
NFR1.1	***Data Consistency:*** The system must enforce strict business constraints, such as ensuring all quantities **(quantity)** and prices **(unit_price)** are positive, and available inventory stock (**quantity_available**) is never negative.	Enforced by specific **CHECK** constraints on *items*, *inventory*, and *procurement_details* tables.	**Fully Met:** Data integrity validated by table-level CHECK constraints.

NFR2	***Referential Integrity:*** The system must maintain a strong link between the entities, preventing orphaned data by ensuring that the given related records cannot be deleted without first removing dependent child records.	Enforced by all defined **FOREIGN KEY** constraints.	**Fully Met:** Strong data linkage ensured by FOREIGN KEY constraints. The constraints enforce reliable and robust data linkage across all tables.
NFR3	***The Process Automation:*** This is the critical inventory management where its logic must reside within the database layer it has (**via triggers)**, of course guaranteeing consistent execution regardless of the application interface used to update the request status.	It is implemented via the *trg_update_inventory* trigger.	**Fully Met:** Critical logic isolated to the DBMS via the trigger. A BEFORE UPDATE trigger ensures that all procurement requests follow the mandatory workflow sequence (“Pending → Approved → Delivered”) by blocking any early transition to the Delivered state.
NFR4	***Auditability:*** The system must automatically do the tracking and record of the precise timestamp when the available quantity for any item in the inventory was last modified.	Directly Implemented by the ***DEFAULT CURRENT_TIMESTAMP ON* UPDATE *CURRENT_TIMESTAMP*** attribute on the *inventory.last_updated* column.	**Fully Met:** System automatically updates timestamp upon any change to inventory quantity.
NFR5	***Data Accessibility:*** The common and recurring analytical data sets are	Implemented via the use of **CREATE VIEW**	**Fully Met:** Views provide optimized, secure, and

Non-Functional Requerements Lists

The table shows the NFR or the Non-Functional Requests, which specifically specify the quality attributes and operational constraints for the system's effectiveness and reliability later.

2.2. Data Requirements

The system manages data for the procurement of school supplies in public schools in Manila. It uses a structured dataset that includes all the main parts of the procurement process. The key tables departments, items, suppliers, **procurement_requests**, and **procurement_details** contain around 50 sample records each. These tables are essential because they keep track of supply requests, item prices, supplier information, and the needs of each department. The supportive tables such as inventory, which lists the available stock, and users, which holds the accounts for Admins, Approvers, and Staff, contribute to the system's organization and efficiency. Through these tables, schools can timely receive their supplies thus providing the students with a proper learning environment equipped with necessary materials. The system backs SDG 4.1: Free, Equitable, and Quality Education by allowing students access to the required materials free of charge.

Some tables contain fewer records but still reflect real conditions. For instance, the departments table corresponds to the actual academic units located in Manila public schools while the suppliers table consists of real suppliers with their respective names and emails. The use of legitimate data not only ensures the system's operation to be similar to that of real ones but also avoids the creation of non-existent data. Consequently, it becomes easier to monitor supplies justly and to guarantee that all schools, even those in a disadvantaged area, receive the necessary resources. The system's realistic data organization contributes to providing equitable education by allowing all students equal access to books, learning materials, and even classroom equipment.

In order to achieve accuracy and reliability, the system is strict and implements rules and validations. The prices of items must always be greater than zero, the quantities must be positive and the inventory cannot be minus. The dates that indicate the request, the approval and the delivery are checked to prevent items from being delivered before they are approved. The procurement requests can only change between **'Pending', 'Approved'** or **'Delivered'** statuses. These rules help to make sure that the transactions are monitored, recorded accurately and are not hidden. The system keeps procurement organized and trustworthy, therefore, it supports SDG 4.1, which calls for free, fair and high-quality education for public school students in Manila. All learners can get their supplies, the cost barriers can be removed and the learning outcomes can be improved.

Table 1. Department

Attribute Name	Data Type	Description
department_id (PK)	INT	Unique identifier for each department.
department_name	VARCHAR(100)	Name of the requesting department.

Table 2. Suppliers

Attribute Name	Data Type	Description
supplier_id (PK)	INT	Unique identifier for each supplier.
supplier_name	VARCHAR(100)	Name of the suppliers
contact_email	VARCHAR(100)	Email address for the transactions and

		communication

Table 3. Items

Attribute Name	Data Type	Description
item_id	INT	Unique ID for each item.
item_name	VARCHAR(100)	Name of school supply (e.g., book, chair).
unit	VARCHAR(20)	Name of school supply (e.g., book, chair).
unit_price	VARCHAR(10,2)	Price per unit must be

Table 4. Inventory

Attribute Name	Data Type	Description
item_id	INT	Item in stock.
quantity_available	INT	Number of items available
last_updated	DATETIME	Timestamp of last update.

Table 5. Procurement Requests

Attribute Name	Data Type	Description
request_id	INT	Unique ID of the request.
department_id	INT	Department requesting items.

request_date	DATE	A date request was made.
status	ENUM('Pending','Approved','Delivered')	**Status of the request.**

Table 6. Procurement Details

Attribute Name	Data Type	Description
detail_id	INT	Unique ID for the detail.
request_id	INT	Links to parent requests.
item_id	INT	Item requested.
supplier_id	INT	Supplier providing the item.
quantity	INT	Number of units requested

2.3. Schema Normalization Analysis

To make sure that the data in our procurement system is clean and reliable, especially when studying how corruption affects students, we normalize the main tables up to **BCNF** and the **procurement request**, table reached this level because everything in it depends only on the RequestID and details like supplier or officer information were moved to their own table to avoid confusion or repetition and the **approval record** table also benefit in **BCNF** since all information depends on the ApprovalID and the officer details are kept separately for clarity, the **officer and supplier** tables are simple and also follow **BCNF** because each one is organized around a single, clear, identifier by normalizing the data this way we reduces errors and keep the information consistent and making it easier to spot any suspicious procurement patterns that might be harming students learning.

III. Design Specification

3.1 Core DBMS Concepts Used:

For each of the required DBMS, include a section detailing:

- Justification: Why was the specific concept chosen(e.g., Why was a TRIGGER used instead of a CHECK constraint for this business rule?)
- Implementation Details: Provide the exact SQL code snippet for the concept (Stored Procedure, Trigger, View, etc.)

In order to achieve accuracy, transparency, and overall good functioning of the system, the team implemented the key concepts of Database Management System (DBMS) learned through the Prelim, Midterm, and Finals. For the Prelim topics, they made use of the Data Definition Language (DDL) to construct the complete database where the tables, keys, and constraints were created. Meanwhile, the Data Manipulation Language (DML) was used to insert test data into the tables. Thus, all the tables were effectively normalized, assignments of primary keys were made and linking of foreign keys was done properly to create clear relationships among the data. Moreover, the team did create multiple views including vw_pending_requests which is a view that merges the data of several related databases into a single output that is easier to read.

At the midterm stage, the group added some SQL constraints with particular emphasis on CHECK constraints which were meant to apply and thus ensure correct data at the table level. Such restrictions will make certain that the fields of unit_price, quantity, and quantity_available will not only be filled with valid numbers but also be of a positive value thus no cases of negative prices or invalid item quantities throughout the database. Normalization rules up to 3NF/BCNF were also applied in this phase, which means that no repeating information was stored in the database as departments, suppliers, items, requests,

and inventory were all placed in separate tables thus no duplication occurs and all the records are consistent especially when the same item or supplier is involved in different transactions.

The team, for finals concepts, utilized stored procedures and triggers to implement system logic that was more complex. The stored procedure submit_procurement is a guarantee that the adding of a new procurement request along with the corresponding item details will always be treated as one complete transaction. Meanwhile, the trigger trg_update_inventory was utilized such that inventory is automatically updated by the system when a procurement request is marked as "Delivered." A trigger was selected instead of a CHECK constraint because CHECK cannot update another table nor carry out multi-step logic, while a trigger can respond to changes and enforce the business rules immediately. These advanced concepts are of great help in keeping accuracy, preventing errors, and automating the procurement system's essential tasks.

1. Tables

Tables were selected as the method to present all key entities of the procurement system because they are the most efficient way to present data in a structured and logical manner. A separate table is dedicated to each aspect of the system: departments provides information about the school or organization **departments** making **procurement requests**; suppliers includes the details of suppliers providing the items; items holds the product details like name, unit, and price; **inventory** monitors the number of items available; **procurement_requests** keeps a record of each request made by departments; and **procurement_details** keeps a record of the items requested in each procurement transaction along with their quantities and supplier. The use of tables in a database allows the segregation of the data into groups, hence, it is very easy to search for, sort through, and update while the integrity of the data is still maintained. It is hard to imagine the system being able to handle such large amounts of interrelated data effectively without tables.

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(100) NOT NULL
);
```

```
  CREATE TABLE suppliers (
     supplier_id INT PRIMARY KEY AUTO_INCREMENT,
     supplier_name VARCHAR(100) NOT NULL,
     contact_email VARCHAR(100) NOT NULL
  );

  CREATE TABLE items (
     item_id INT PRIMARY KEY AUTO_INCREMENT,
     item_name VARCHAR(100) NOT NULL,
     unit VARCHAR(20) NOT NULL,
     unit_price DECIMAL(10,2) NOT NULL CHECK (unit_price > 0)
);
```

2. Foreign Key

Justification (How it was used)

- Foreign Keys were used to connect related tables and maintain relational integrity, ensuring that procurement details always link to valid departments, items, and suppliers.

A. procurement_requests → departments

FOREIGN KEY (department_id) REFERENCES departments(department_id)

B. procurement_details → procurement_requests

FOREIGN KEY (request_id) REFERENCES procurement_requests(request_id)

C. procurement_details → items

FOREIGN KEY (item_id) REFERENCES items(item_id)

D. procurement_details → suppliers

FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)

E. inventory → items

FOREIGN KEY (item_id) REFERENCES items(item_id)

3. Trigger

- **trg_update_inventory**
- **trg_prevent_skip_approval**

Justification (How it was used)

- **A Trigger was used to automatically update inventory only when a request's status becomes DELIVERED.**
- **Avoids the situation when the stock is updated twice**
- **Updates inventories that are newly added as well as those that already exist**

A trigger was set up so that the quantity of the inventory is updated automatically every time a procurement request changes its status to "Delivered." Unlike manual updates or checking constraints which take time, triggers provide instant reaction of the database during the specific events. In this case, the trigger updates the inventory table only when valid item_id values are present since the foreign key blocks the process if the item being referenced does not exist in the database.

The use of this method not only maintains inventory data and correctness but also eliminates the instance of inserting invalid data. The moment of delivery confirmation, the quantity of that particular item is automatically increased, thus eliminating repetitive tasks

and possible human error. The trigger supports the real-time synchronization between the delivery and the inventory records, thus making the system more reliable and efficient.

One more trigger, trg_prevent_skip_approval, was created to help the company staff follow the right workflow sequence in the procurement process. This trigger is used to prohibit the marking of a request as “Delivered” in case it has not been “Approved” first. Because CHECK constraints are not able to compare old and new row values and cannot stop changes in the workflow, a BEFORE UPDATE trigger was selected to make sure the rule is checked in the database. This is in line with the school’s policy on procurement and assures that the data is consistent.

```
DELIMITER $$

CREATE TRIGGER trg_update_inventory
AFTER UPDATE ON procurement_requests
FOR EACH ROW
BEGIN
   IF NEW.status = 'Delivered' AND OLD.status <> 'Delivered' THEN
      INSERT INTO inventory (item_id, quantity_available)
      SELECT item_id, quantity
      FROM procurement_details
      WHERE request_id = NEW.request_id
      ON DUPLICATE KEY UPDATE
         quantity_available = quantity_available + VALUES(quantity_available);
   END IF;
END$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER trg_prevent_skip_approval
BEFORE UPDATE ON procurement_requests
FOR EACH ROW
```

```
BEGIN
    IF NEW.status = 'Delivered' AND OLD.status <> 'Approved' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: You cannot deliver a request that is not approved.';
    END IF;
END$$

DELIMITER ;
```

4. Primary Key

Justification (How it was used)

- Primary Keys were used to uniquely identify each record (e.g., each department, item, request).

The implementation of primary keys throughout all tables served the purpose of uniquely identifying each record and disallowing duplicates. This is of prime importance in a procurement system since there might be a lot of entries which will look similar but still need to be separated one by one. For instance, in the departments' table, **department_id** guarantees that each department will be isolated and identified. In suppliers, **supplier_id** maintains the dissimilarity of every supplier even if the names of some might be akin. In items, item_id gives a direct indication of each product and allows accurate monitoring of stock levels.

The new inventory table also considers **item_id** as the primary key and includes a last updated column, which automatically logs the date and time of changes done with the item, as well. Also, there is a limit on the **quantity_available** column which is imposed to stop stock values from going negative, so the data is less reliable. As **item_id** acts as a foreign key to the items table, the system will not allow inserting inventory data for non-existing items thus ensuring consistency. This setup guarantees that the inventory represents the actual quantities, and it continues to be in coordination with deliveries and procurement actions.

In **procurement_requests**, **request_id** segregates each request thus no two requests can get mixed up. Eventually, in **procurement_details**, **detail_id** bestows upon each item entry its unique identity inside a request enabling single tracking of multiple items under one order. The system, through the usage of primary keys and foreign keys in conjunction, not only maintains data integrity but also provides relational connections between tables and enables exact monitoring of transactions throughout procurement processing.

department_id INT PRIMARY KEY AUTO_INCREMENT

supplier_id INT PRIMARY KEY AUTO_INCREMENT

item_id INT PRIMARY KEY AUTO_INCREMENT

item_id INT PRIMARY KEY

request_id INT PRIMARY KEY AUTO_INCREMENT

detail_id INT PRIMARY KEY AUTO_INCREMENT

5. Stored Procedure

Justification (How it was used)

- To automate the process of submitting a procurement request a Stored Procedure was employed, thus a manual insertion into two tables was avoided.

- It guarantees ACID compliance as all operations are carried out within a single transaction.

A stored procedure was used to handle the complete submission of procurement requests along with their corresponding details in one transaction. With this technique, it becomes possible for several SQL statements to function as one, consequently, data

consistency is assured and errors are minimized. In case any of the steps do not go through, the transaction can be reversed and thus no incomplete or wrong data will be saved. Also, the stored procedure offers better performance since it enables the users to run one command for data insertion into several tables at the same time, and it also contributes to data security since it regulates the manner of data being inserted. The **Delimiter** command is necessary due to the fact that the default statement delimiter in MySQL is a semicolon **(;)**, but still, the procedure has many statements that are separated by semicolons. Thus, the changing of the delimiter **(to $$)** is a temporary measure that tells MySQL to treat the whole procedure as one single statement, thus avoiding errors. And after the procedure is defined, the delimiter is set back to the semicolon so that the normal SQL commands can be executed. This guarantees that the process of creating and executing the procedure is both correct and safe.

```
DELIMITER $$

CREATE PROCEDURE submit_procurement (
    IN p_department_id INT,
    IN p_request_date DATE,
    IN p_item_id INT,
    IN p_supplier_id INT,
    IN p_quantity INT
)
BEGIN
    DECLARE req_id INT;

    START TRANSACTION;

    INSERT INTO procurement_requests
        (department_id, request_date, status)
    VALUES
        (p_department_id, p_request_date, 'Pending');

    SET req_id = LAST_INSERT_ID();
```

```
    INSERT INTO procurement_details
      (request_id, item_id, supplier_id, quantity)
    VALUES
      (req_id, p_item_id, p_supplier_id, p_quantity);

    COMMIT;
  END$$

  DELIMITER ;
```

- START TRANSACTION ensures atomicity
- Two related inserts are executed as one unit
- COMMIT permanently saves the changes
- Prevents incomplete or orphaned records

6.. Check Constraints

- **CHECK constraints were implemented to maintain data regulations locally at the table level, e.g., ensuring that prices remain positive and quantities that are not negative.**
- **Such constraints inhibit the entry of invalid data into the system.**

```
CREATE TABLE items (
  item_id INT PRIMARY KEY AUTO_INCREMENT,
  item_name VARCHAR(100) NOT NULL,
  unit VARCHAR(20) NOT NULL,
  unit_price DECIMAL(10,2) NOT NULL CHECK (unit_price > 0)
);
```

—----------

quantity INT NOT NULL CHECK (quantity > 0)

7. Views

Concept: Data Abstraction & Reporting

Justification (How it was used)

- Views simplified reporting as professors or staff members could check easily readable summaries without the need of writing complex SQL.

- One view is for pending requests, another for item summaries, and the last one for supplier activity.

1. View of Pending Requests

```
CREATE VIEW vw_pending_requests AS
SELECT
   pr.request_id,
   d.department_name,
   pr.request_date
FROM procurement_requests pr

JOIN departments d ON pr.department_id = d.department_id
WHERE pr.status = 'Pending';
```

- This shows all pending procurement requests

2. View of Item Procurement Summary

```
CREATE VIEW vw_item_procurement_summary AS
SELECT
   i.item_name,
```

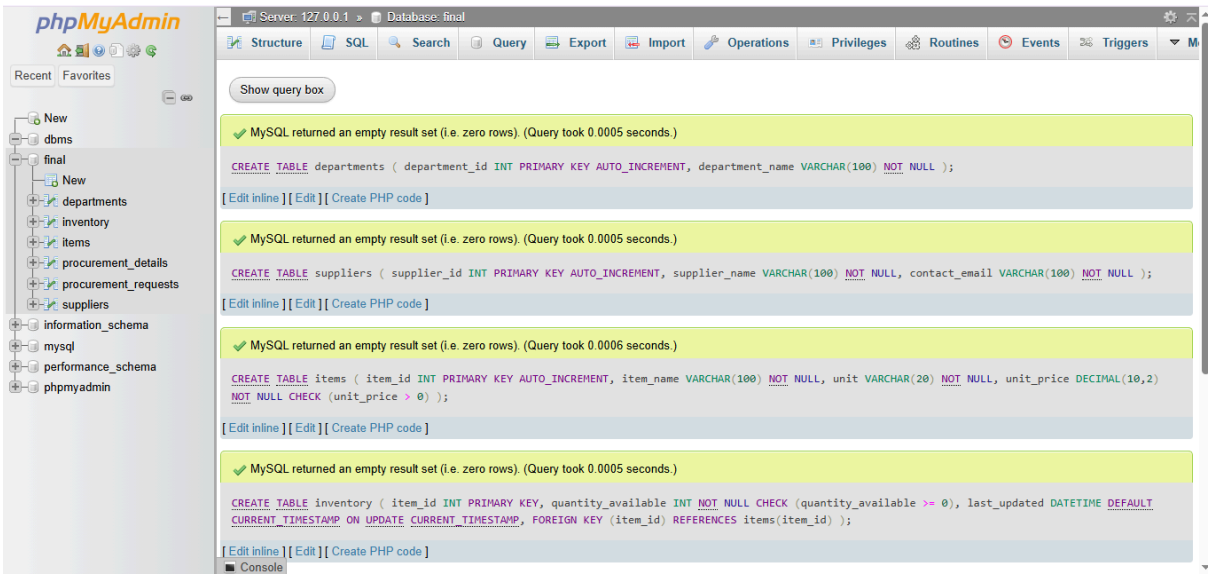
```
    SUM(pd.quantity) AS total_quantity
FROM items i
JOIN procurement_details pd ON i.item_id = pd.item_id
GROUP BY i.item_name;
```

- This summarizes the total quantity requested per item

3. View of Supplier Transactions

```
CREATE VIEW vw_supplier_transactions AS
SELECT
    s.supplier_name,
    COUNT(pd.detail_id) AS total_transactions
FROM suppliers s
JOIN procurement_details pd ON s.supplier_id = pd.supplier_id
GROUP BY s.supplier_name;
```

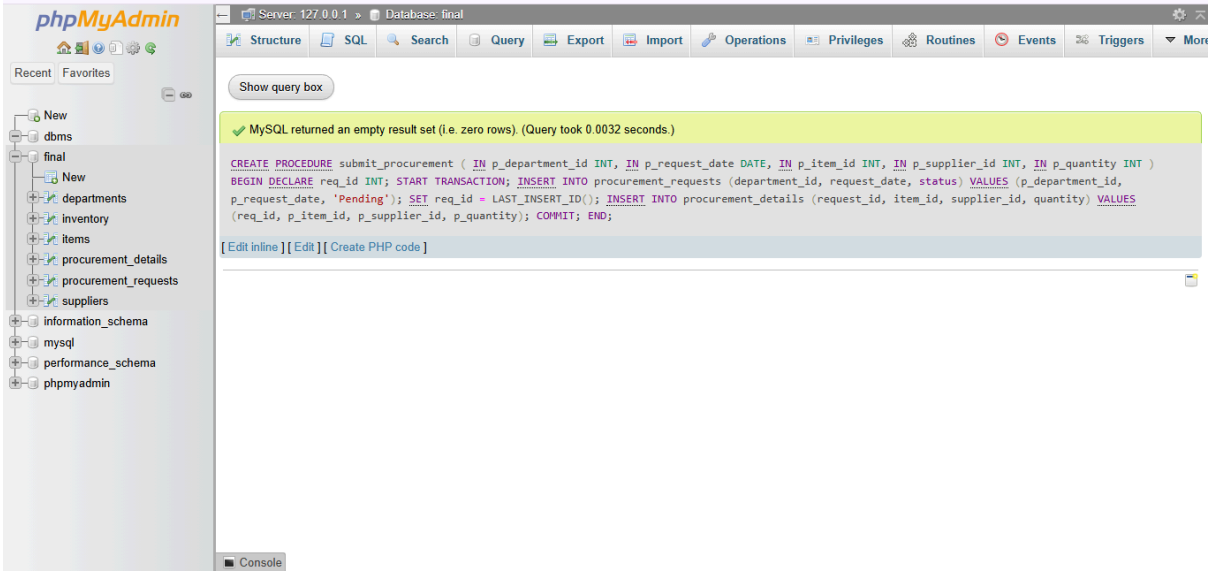
- Shows supplier participation.



Picture 1 – Table Creation (Schema Setup):

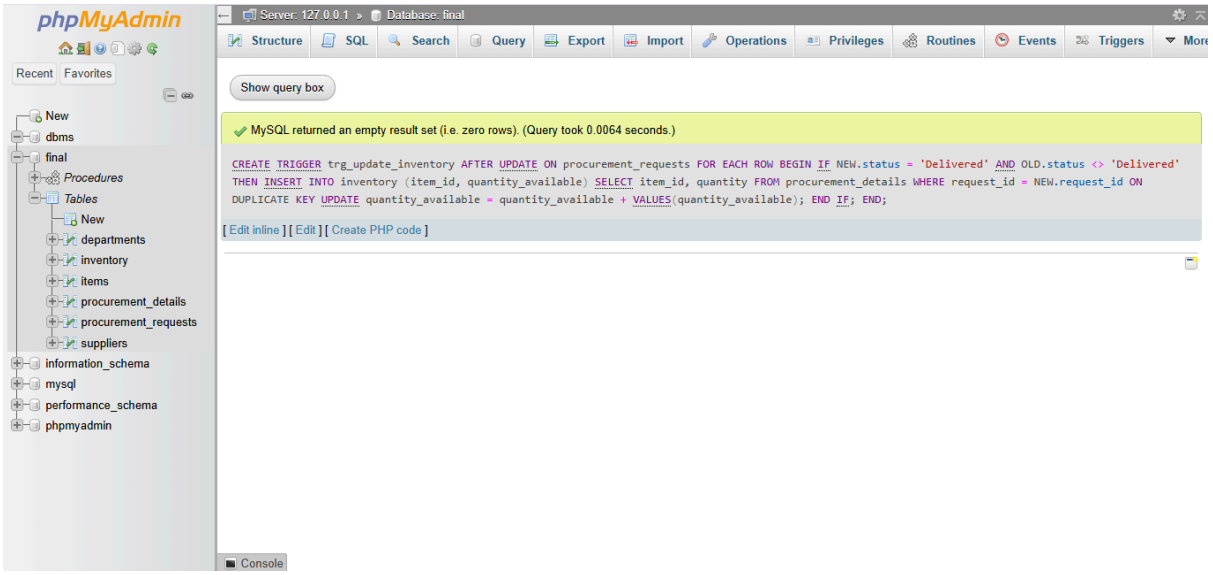
The initial picture illustrates the SQL "CREATE TABLE" statements that were employed to set up the main tables of the database which are "departments, suppliers, items, and inventory". Each of the tables specifies its primary key, the fields that are a must and the types of data to be able to store data in an organized and valid manner. Integrity constraints

such as "NOT NULL", "CHECK", and "FOREIGN KEY" are imposed to control the quality of data, for example, preventing records of negative prices or quantities and incorrectly linking inventory records to items.



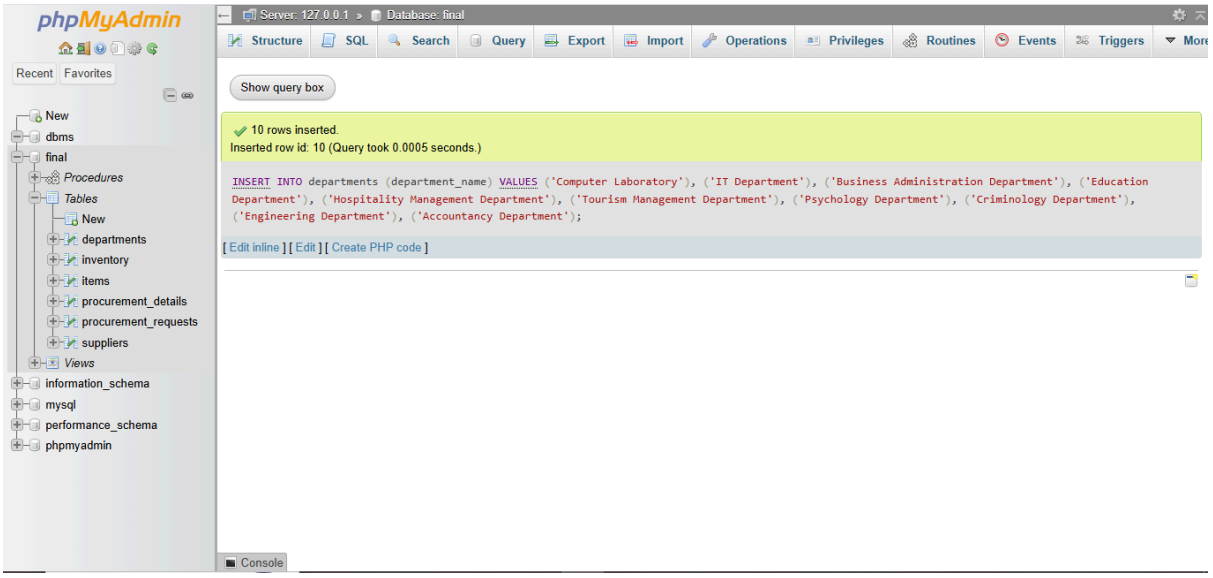
Picture 2 – Stored Procedure (submit_procurement):

The second image shows a stored procedure called "submit_procurement" which makes the process of creating a procurement request to be done automatically. It begins with the transaction, inserts a new record into the "procurement_requests" table with default status "Pending", then it gets the request ID and that is used to insert the details that are related into the "procurement_details" table. A transaction is used to make sure that both the inserts succeed together and thus the data remains consistent.



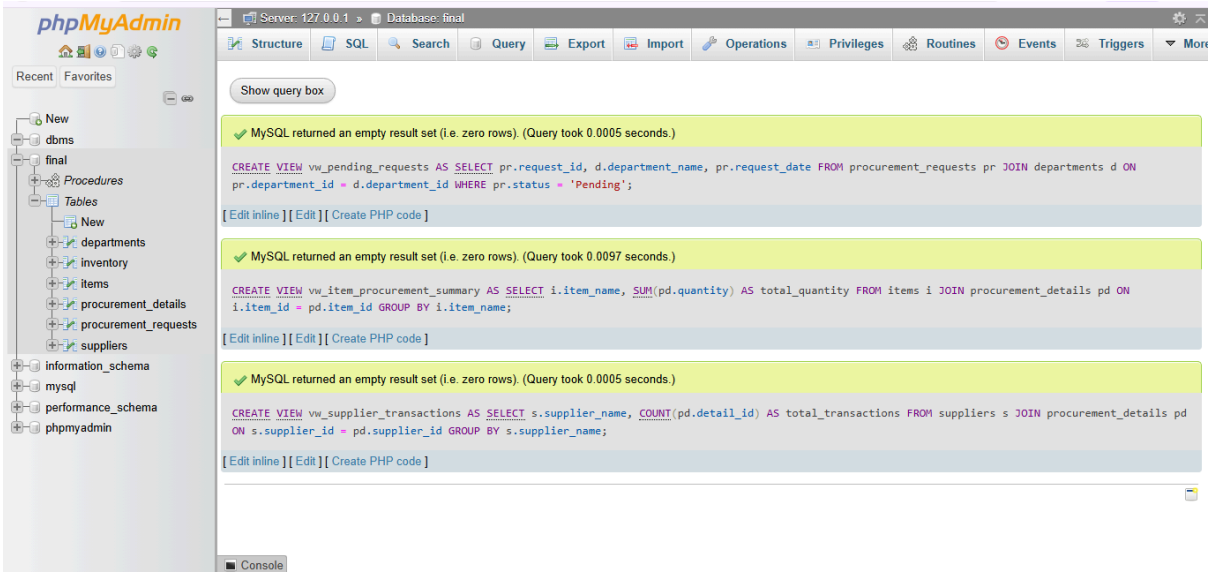
Picture 3 – Trigger (Automatic Inventory Update):

The third image reveals a database trigger known as "trg_update_inventory" that works automatically "after every update" on the "procurement_requests" table. Trigger works in such a way that as soon as the status of a request changes to "Delivered", it adds the delivered quantities to the 'inventory" table or updates existing stock if the item is already in stock. The trigger prevents inventory discrepancies by comparing the old status and the new status, thus, confirming delivery once.



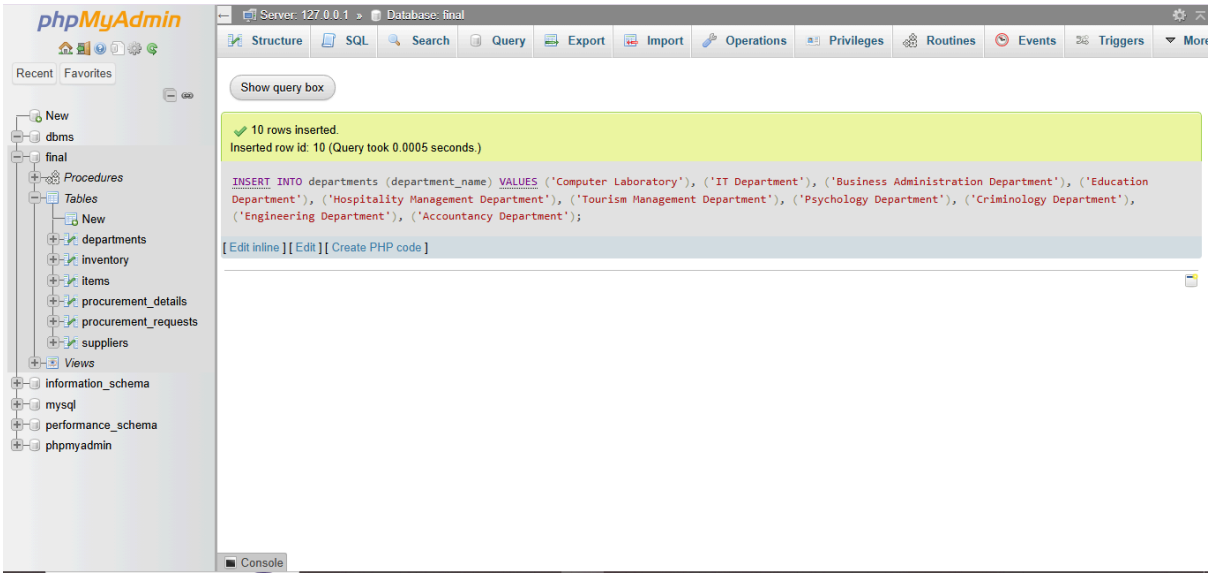
Picture 4 – Trigger Creation (Business Rule Enforcement)

The first image shows a SQL "BEFORE UPDATE trigger" called "trg_prevent_skip_approval" on the `procurement_requests` table, which stops the updates that try to alter the data in the application. This automatic process triggers every status update on a request and imposes the following rule: a request can be marked as "Delivered" only if the previous status was "Approved". If someone tries to avoid the approval process, the trigger terminates the operation and sends a custom error message using "SIGNAL SQLSTATE '45000"`. Thus, definitely, maintaining data accuracy and limiting control over the workflow at the database level, not only in the application.



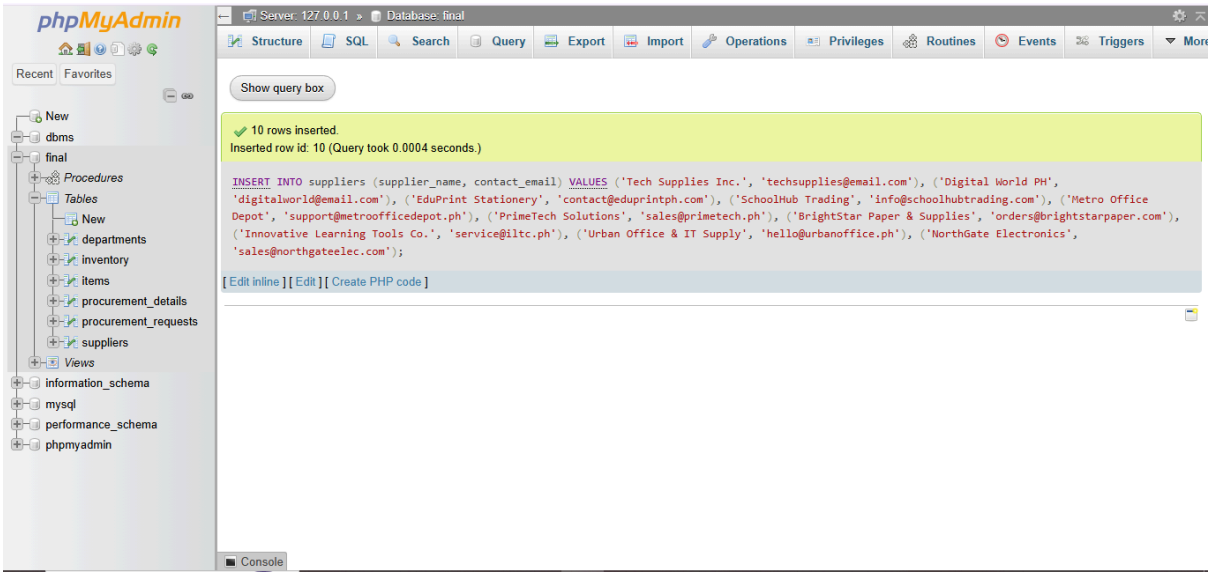
Picture 5 – View Creation (Reporting and Data Analysis)

The second image depicts a situation where three " SQL views "were created for reporting purposes. By "joining" `procurement_requests` and `departments`, the first view "vw_pending_requests" shows all pending procurement requests along with their department names and request dates. The second view "vw_item_procurement_summary" provides a summary of the total quantity requested for each item by analyzing the data from "items" and "procurement_details" through the grouping technique. The third view "vw_supplier_transactions" shows the number of procurement transactions that each supplier has dealt with. All these views present the data in a straightforward manner, hiding the complexity of the underlying queries, and they are almost ready-to-use data sources.



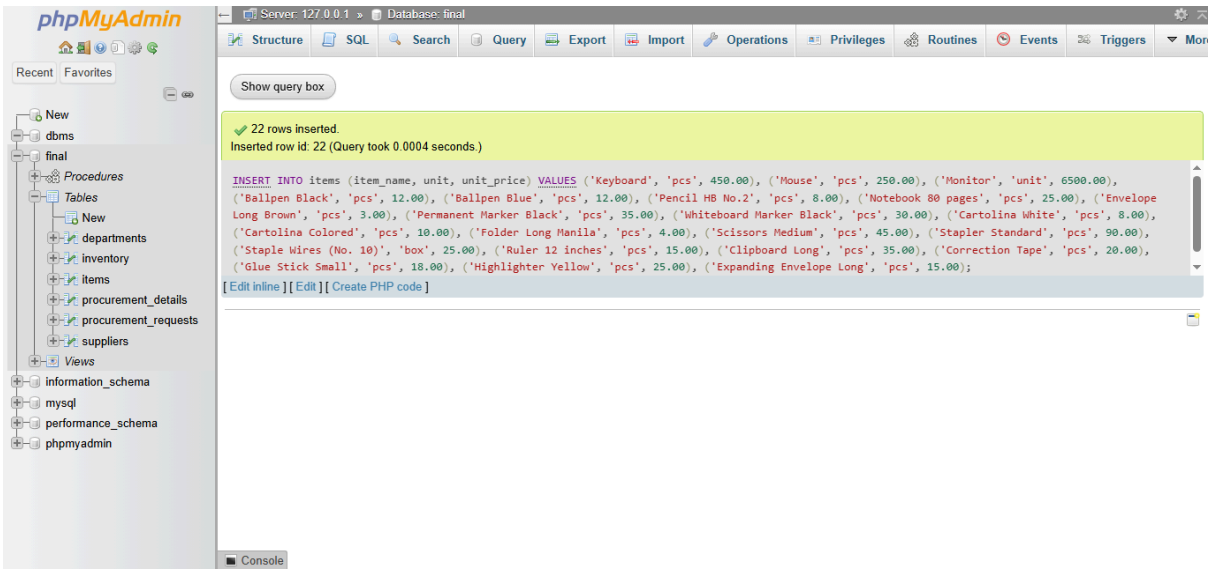
Picture 6 – Data Insertion (Populating the Departments Table)

In the third image, an "INSERT statement" can be seen used to fill the "departments" table with data. It adds ten department names, for example, Computer Laboratory, IT Department, Business Administration, up to others with one query. This action is necessary for the creation of the reference or master data that is through foreign keys, i.e., procurement requests, relied on other tables. The successful insertion of these records is a confirmation of the correctness of the table structure and also the database is ready for actual system operations.



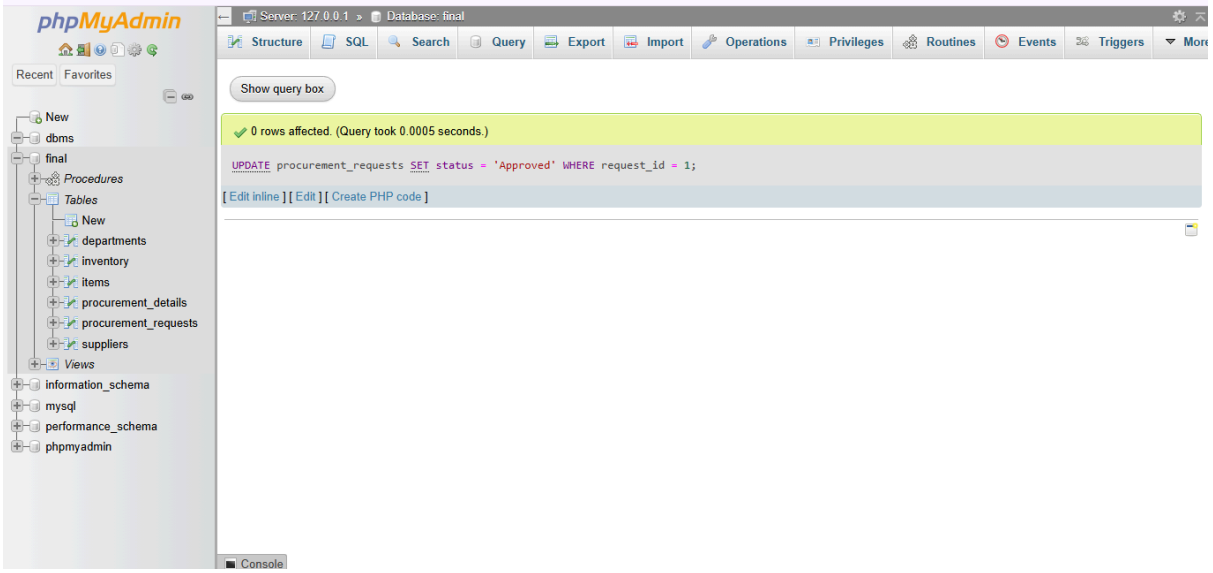
Picture 7 – (INSERT INTO items):

Here is depicted an SQL `INSERT INTO` command that serves the purpose of adding data into the "items" table. The code is inserting in one go several office supply items such as Keyboard, Mouse, Monitor and Ballpen, with unit (like pcs) and unit price also mentioned. The green message “22 rows inserted” indicates that the query was successful and all the records of items were captured in the database. This is done to initially populate or to add eventually new item data so that the system has a clear idea of the items available.



Picture 8 – (UPDATE procurement_requests):

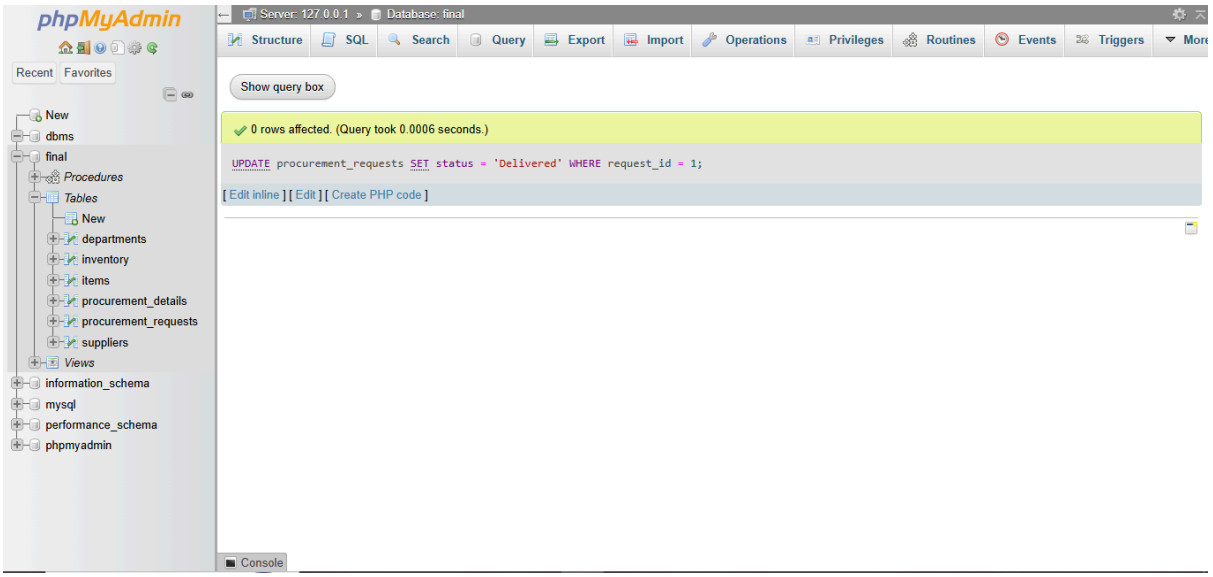
This picture exhibits an SQL `UPDATE` statement that alters the status of a procurement request. The code modifies the "procurement_requests" table and designates the status to "Delivered" for the request with `request_id = 1`. The success message indicates the database has altered the record as per the request. This is a case where an item that was requested has already been received, thus the system can update that the request is completed.



Picture 9 –(CREATE TRIGGER error):

This image depicts the endeavor to set up a “trigger” called `trg_prevent_skip_approval`, but there is an error. The trigger aims at restricting the users to not make the request status change directly to “Delivered” unless the preceding status was “Approved,” thus supporting the workflow rules properly. On the other hand, the error message indicates that the trigger

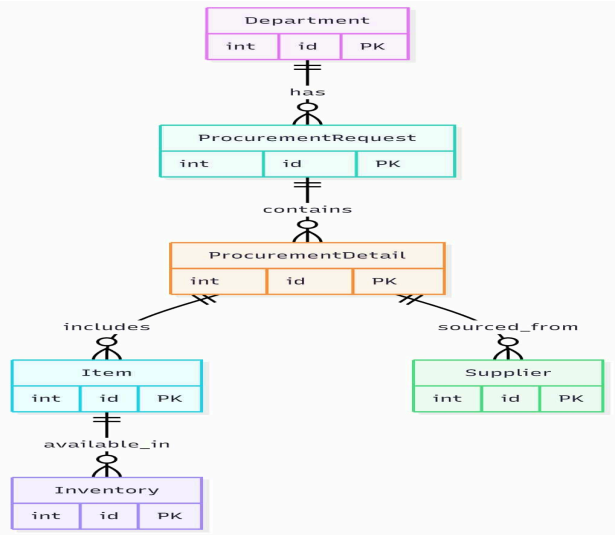
already exists which implies that it was created beforehand. This eliminates the possibility of creating duplicate triggers, and MySQL is shown to be securing the database against conflicts.



3.2. ER Diagram:

"This document consists of the full variety of Entity-Relationship Diagrams (ERDs) of the database, including the Conceptual, Logical, and Physical models. Every model has been developed with utmost care to reflect all entities and their attributes, also to exhibit primary and foreign keys distinctly. The connections among the entities are all clarified with cardinalities, thus all the interrelations and dependencies of the database are depicted precisely. The ERDs then give an overall and conclusive view of the database design along with a blueprint for implementation and future reference."

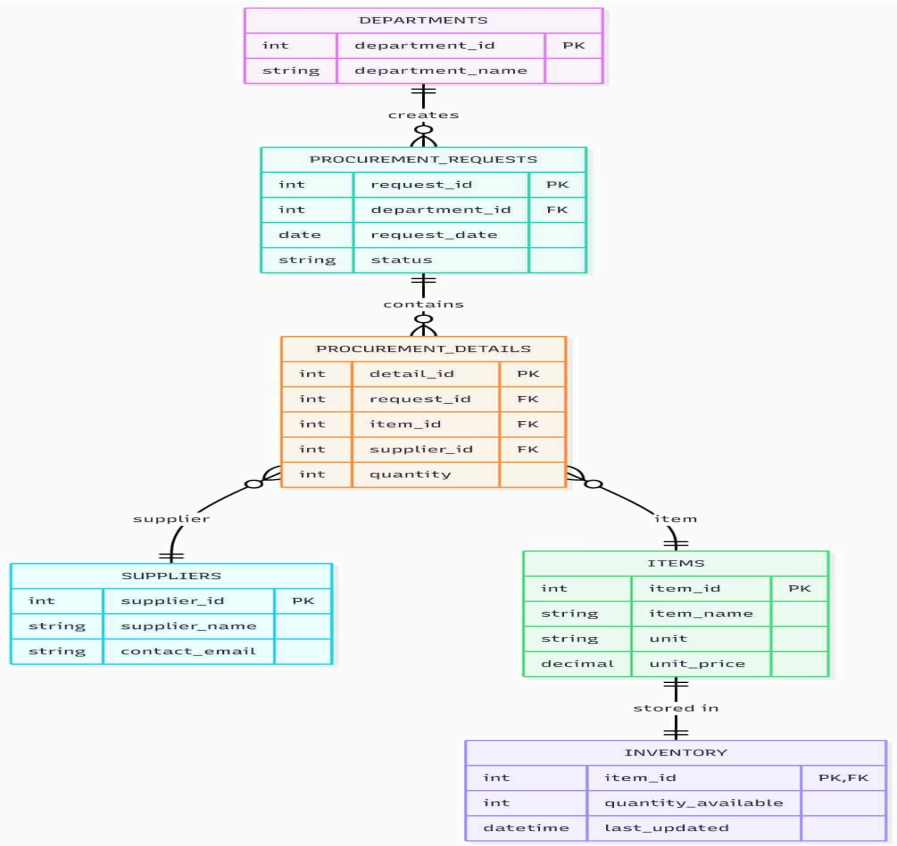
A. Conceptual ERD



View a clearer version at this link: Conceptual ERD

The Conceptual ERD represents the core idea of the system. It emphasizes the main entities, namely Department, Procurement Request, Procurement Details, Item, Supplier, and Inventory, and their relationship to the data types, table design, and the restrictions are not mentioned. It only tells the system's basic requirements like the departments that make request, the requests that contain details, the joining of the details with the suppliers and items and the item's presence in the inventory

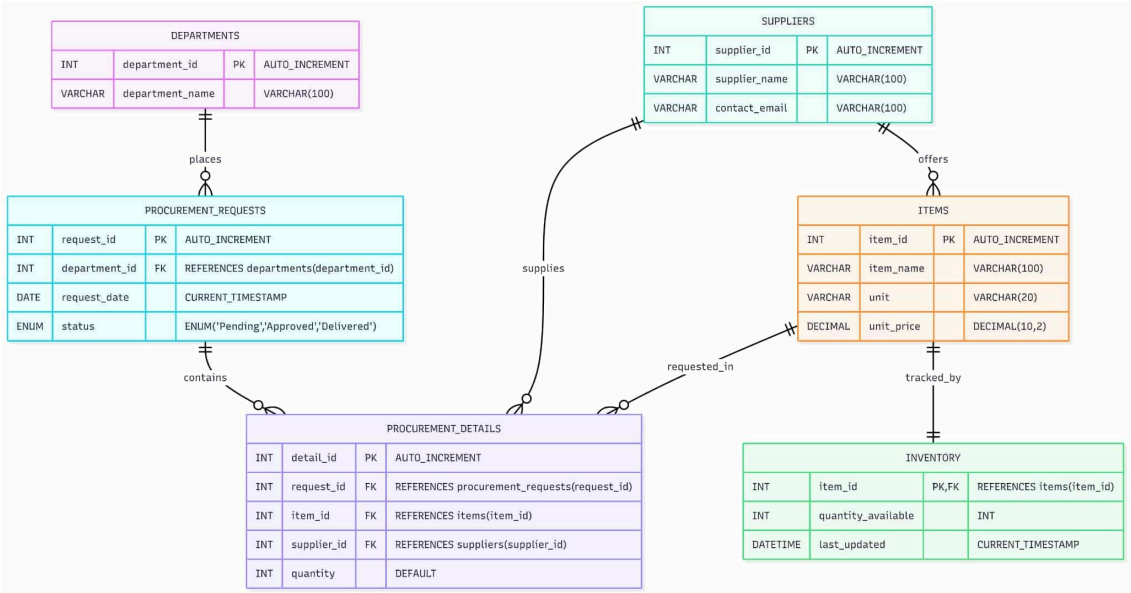
B. Logical ERD



View a clearer version at this link: **Logical ERD**

The Logical ERD introduces more features with regard to the conceptual model. In particular, it presents the attributes (item_name, request_date, quantity) already and also marks the primary and foreign keys, yet at this stage, it is still not attached to any particular database system. It mainly concerns the data organization revealing more relationships and the fields required for the system operation. This ERD depicts the database structure in a quite extensive manner, nevertheless, it still remains independent of SQL syntax or storage specifics.

C. Physical ERD



View a clearer version at this link: **Physical ERD**

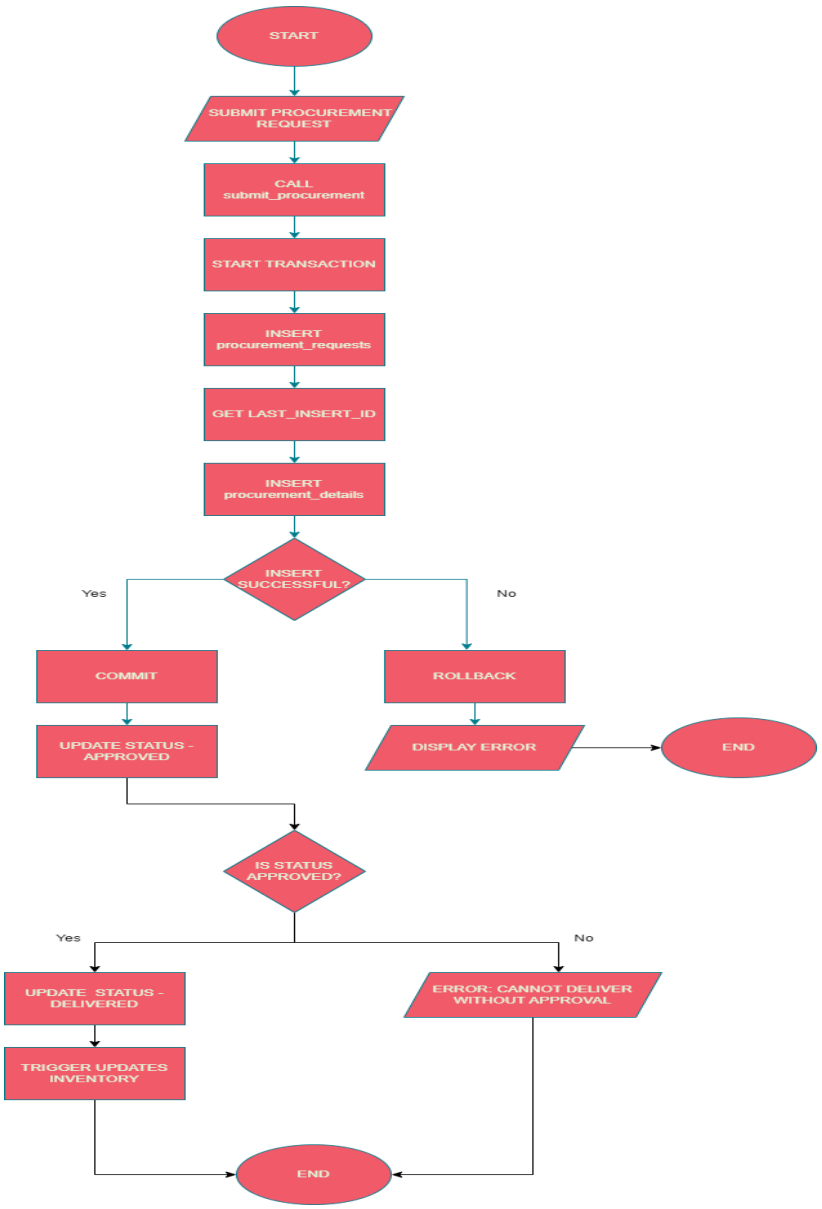
The Physical ERD is the most comprehensive one. It illustrates the database implementation specifics like **AUTO_INCREMENT**, **VARCHAR** sizes, **DECIMAL** formats, **ENUM** values, and even the particular FK connections. This is the version which developers utilize to create the database in MySQL or any other DBMS. It concerns precisely the manner of the database being created, incorporating data types, constraints, and indexing thus, making it ready for the actual execution.

Relationship	Cardinality
Department → Procurement Request	1 : Many
Procurement Request → Procurement Detail	1 : Many
Item → Procurement Detail	1 : Many
Supplier → Procurement Detail	1 : Many
Item → Inventory	1 : 1

3.3. Transaction Flowchart:

The flowchart that is provided here illustrates the complete route from making a procurement request to processing it by means of a transaction-based system. The entire procedure starts with a user making a procurement request which subsequently activates the function submit_production. A database transaction is opened, and the system processes the following actions: it inserts a new record in the procurement_request table, gets the ID of the last inserted record and then uses this ID to insert the related details into the procurement_details table. If all insert actions are successful, then the transaction is committed; otherwise, it is rolled back to maintain data consistency.

The following step is the database where the admin evaluates the request. If the request is approved, it can now be delivered. The system also validates the status transition and prevents a request from being delivered unless it has first been approved. Attempts to skip the approval step will result in an error and the update will not proceed. On the other hand, if the transaction fails, the admin will still be informed about the request status, but instead of continuing with the usual flow, an error message will be displayed. The operation will conclude either by the successful updating of the inventory or by the resolution of any errors that may have occurred.



View a clearer version at this link: FLOWCHART.jpg

V. Conclusion and Contributions

5.1. Conclusion

This research examines the link between school procurement systems and UN SDG Target 4.1 Free, Equitable and Quality Education. The target sets 2030 as the deadline for all children to go through a free and fair education and have significant learning outcomes. In order for education to be completely free, all students will have to pay no hidden bills, such as notebooks, papers, or other classroom materials that the schools normally provide. For education to be equitable, students from poorer families must not only be supplied with the same basic materials as the rich ones but also be treated the same way as their rich counterparts in every other aspect of schooling. Eventually, if procurement is not done correctly, or corrupt practices occur, the aforementioned goals will be thwarted as the students will be the ones deprived of the very materials required for their learning.

The results of the research have revealed that corruption, incomplete records, and improper procurement procedures are among the most significant factors that lower the quality of education. In the case of mishandling of funds or delayed delivery of supplies, the absence of basic learning tools in the classrooms becomes the norm, and the students, particularly those coming from low-income families, are the ones who bear the brunt. Thus, the barriers to learning become bigger and more difficult to cross. To combat this, the Procurement and Inventory Management System was invented to organize the procurement process, make it more clear, and hold people accountable. The use of triggers, stored procedures, foreign keys, and reporting views are some of the features through which the system ensures that data is accurate, monitoring is easier, and there are fewer chances for dishonesty to occur.

Overall, it is the case that the project has demonstrated the significance of the procurement system enhancement for the attainment of SDG 4.1. Schools when managing their supplies well and also having truthful and comprehensive records can therefore provide the students with the necessary tools without any additional costs. Such a system that is both transparent and well-managed would help to establish and maintain a learning environment where no student is deprived of quality materials because of unequal access. Improved procurement means the students can concentrate on learning instead of on what they do not have which causes an educational experience that is fairer, stronger, and more meaningful for everybody.

5.2. Individual Contributions (Detailed breakdown of each member's assigned module/script).

Name	Distribution
Apolinario, Christian Jolo (424005731)	Data Requirements, Schema Normalization Analysis
Fabellon, Karyl Hyacinth (424001382)	Functional Requirements and Non-Functional Requirement, Flowchart
Ermitanio, Jerpy Hans (424005110)	Implementation Details, ER Diagram's, Code, Flowchart
Riel, Franz Adrian (424002474)	Schema Normalization Analysis
Tabios, Jenny (424001421)	Project Overview & UN SDG Target, Core DBMS Concepts Used, Conclusion, Problem Statement, Flowchart, and ERD's