PROJECT DELIVERABLE 02

Group 13

Individual Contribution			
CWID	**Name**	**Contribution (Description)**	**Percent Contribution**
A20593079	Akshada Ranalkar	Created DB, Loaded Tables, Worked on Indexes and Temp Tables queries, Documentation	33.3%
A20563287	Anuja Wavdhane	Created DB, Loaded Tables, Worked on Views and Triggers queries, Relational Schema, Documentation	33.3%
A20560966	Suhasi Gadge	Created DB, Loaded Tables, Worked on Stored Procedures and Functions queries, Documentation	33.3%

Project Title:

ACADEMIC PUBLICATION DATABASE SYSTEM

(REFERENCE WEBSITE: IEEE Website

https://ieeexplore.ieee.org/abstract/document/344065)

1.1 Objectives of Deliverable # 02

1. Database Creation and Data Loading:
    - Create a database with precise DDL statements.
    - Load at least 15 records per table with realistic and coherent data.

2. Implementation of Indexes, Temporary Tables, and Views:
    - Implement indexes to optimize query performance.
    - Use temporary tables for data transformation.
    - Create insightful and well-structured views for specific purposes.

3. Triggers, Stored Procedures, and Functions:
    - Develop triggers, stored procedures, and functions with complex business logic.
    - Ensure triggers and procedures are well-structured and maintain data integrity.

4. Display of Statements and Output:

- Show all SQL statements used (DDL, DML, indexes, views, temporary tables, triggers, procedures, and functions).
- Provide corresponding outputs for each task, including data loading.
- Ensure the presentation is well-organized with results clearly linked to the statements.

1.2 Database Creation

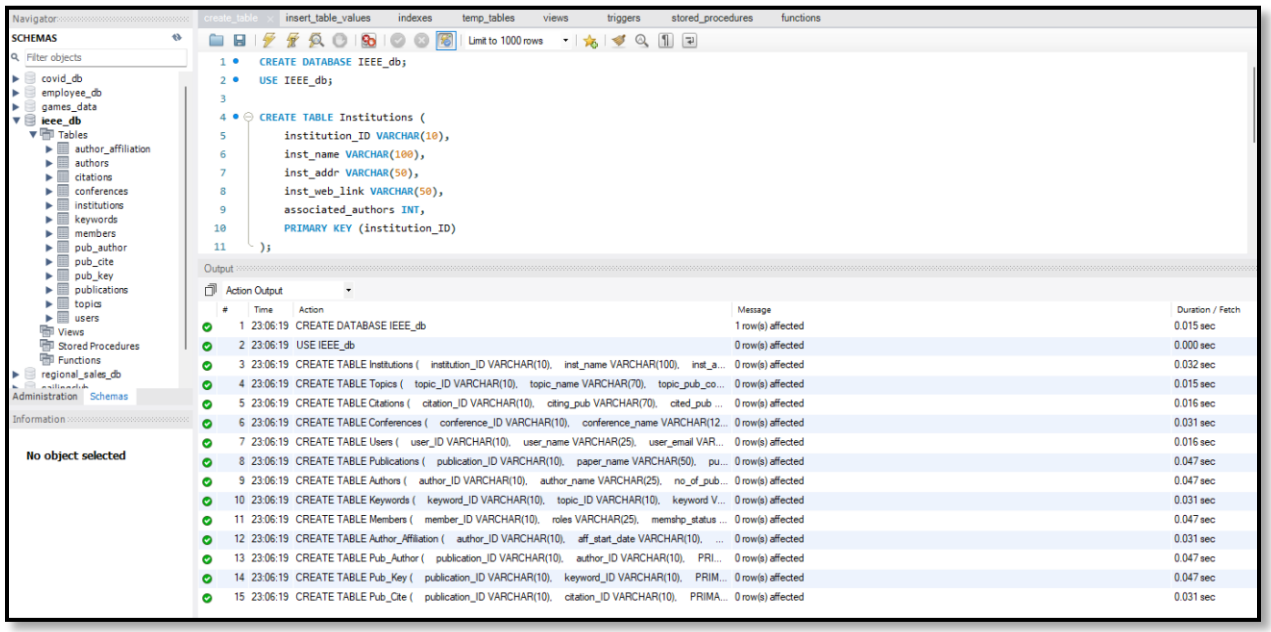
Database Schema: The database consists of several interrelated tables designed to manage publications, authors, conferences, citations, and users. The main entities include `Publications`, `Authors`, `Conferences`, `Users`, `Institutions`, `Topics`, and `Citations`. We have also created the bridge entity tables and inserted relevant values into them.

File name: create_table.sql

DDL Statements: Example of table creation (DDL):

```
CREATE TABLE Institutions (
    institution_ID VARCHAR(10),
    inst_name VARCHAR(100),
    inst_addr VARCHAR(50),
    inst_web_link VARCHAR(50),
    associated_authors INT,
    PRIMARY KEY (institution_ID)
);

CREATE TABLE Publications (
    publication_ID VARCHAR(10),
    paper_name VARCHAR(50),
    publisher VARCHAR(25),
    DOI VARCHAR(10),
    date_of_conference DATE,
    date_of_publication DATE,
```

```
    print_ISSN VARCHAR(10),
    print_ISBN VARCHAR(20),
    conference_ID VARCHAR(10),
    PRIMARY KEY (publication_ID),
    FOREIGN KEY (conference_ID) REFERENCES Conferences(conference_ID)
);
```



1.3 Database Loading

More than 15 records were inserted into each table, ensuring that the data is realistic and coherent. For instance, publications are associated with conferences and authors, while citations link to other papers.

File Name: insert_table_values.sql

Example `INSERT` statements:

INSERT INTO Institutions (institution_ID, inst_name, inst_addr, inst_web_link, associated_authors)

VALUES

('INST001', 'Woods Hole Oceanographic Institution', 'Woods Hole, MA, USA', 'www.whoi.edu', 25),

('INST002', 'Georgia Institute of Technology', 'Atlanta, GA, USA', 'www.cc.gatech.edu', 100),

('INST003', 'Jiangsu University', 'Zhenjiang, China', 'www.ujs.edu.cn', 50),

…..

('INST020', 'Imperial College London, Electrical Engineering', 'London, UK', 'www.imperial.ac.uk', 170);

==========

INSERT INTO Topics (topic_ID, topic_name, topic_pub_count)

VALUES

('TOP001', 'Computing and Processing', 250),

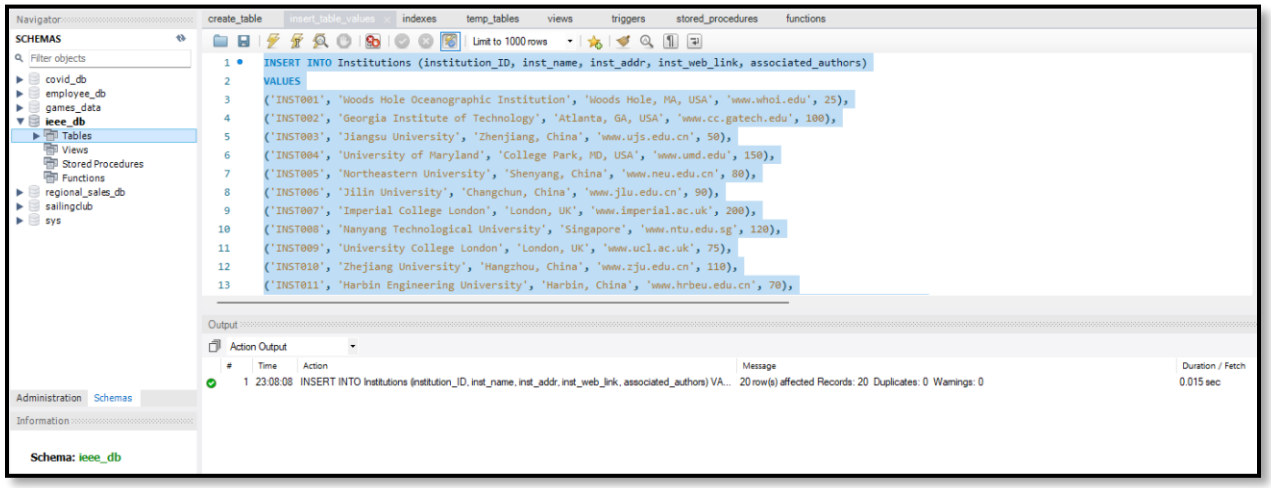
('TOP002', 'Components, Circuits, Devices and Systems', 180),

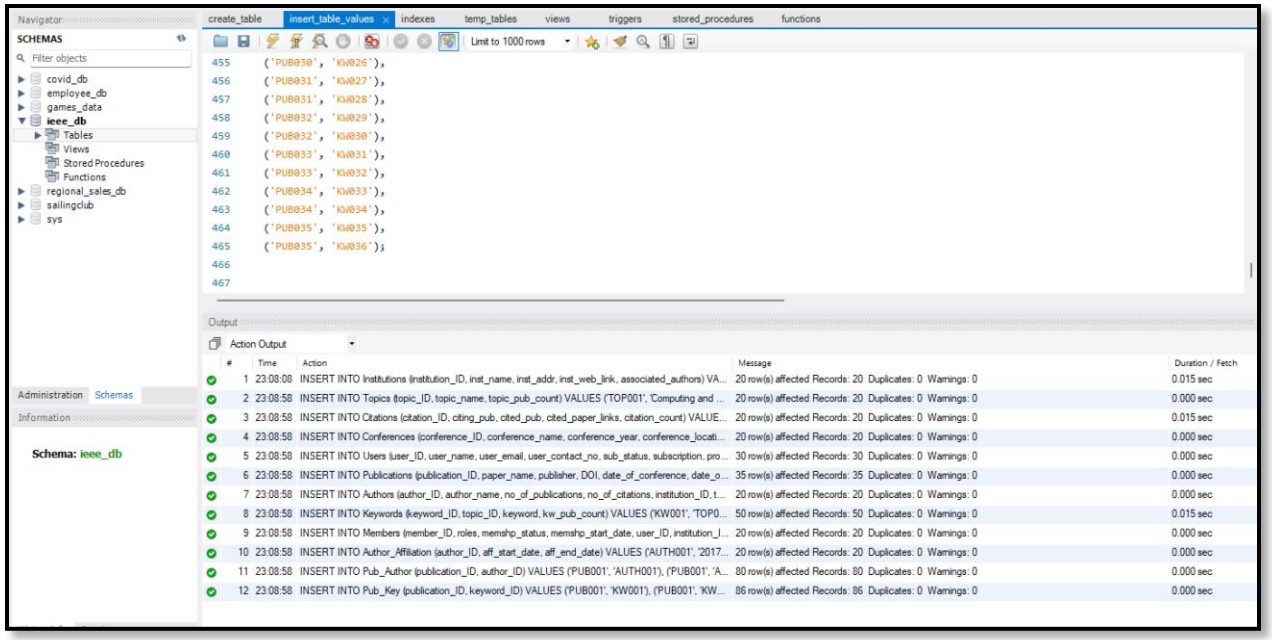
('TOP003', 'Communication, Networking and Broadcast Technologies', 220),

('TOP004', 'Signal Processing and Analysis', 200),

…..

('TOP020', 'Quantum Computing', 170);





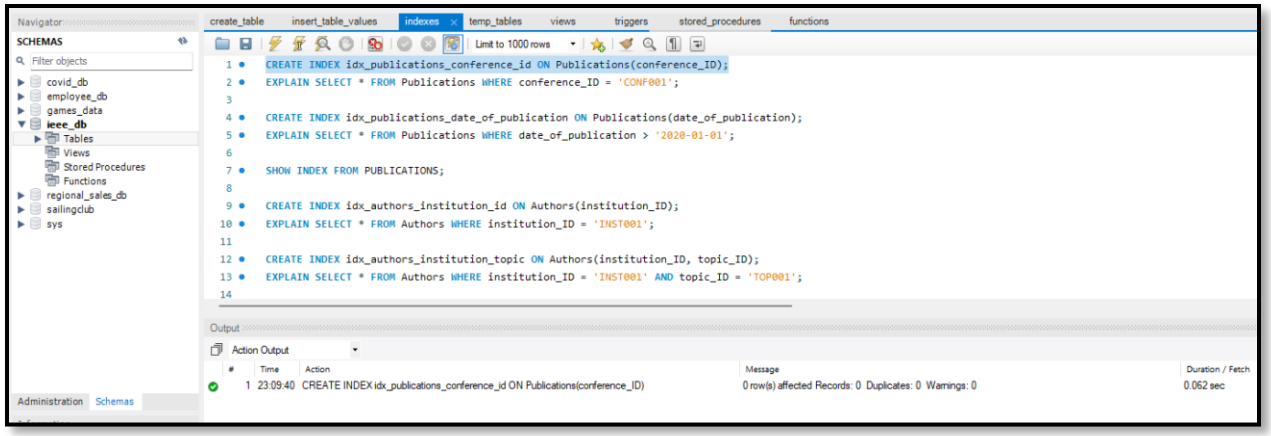
1.4 Indexes Implementation

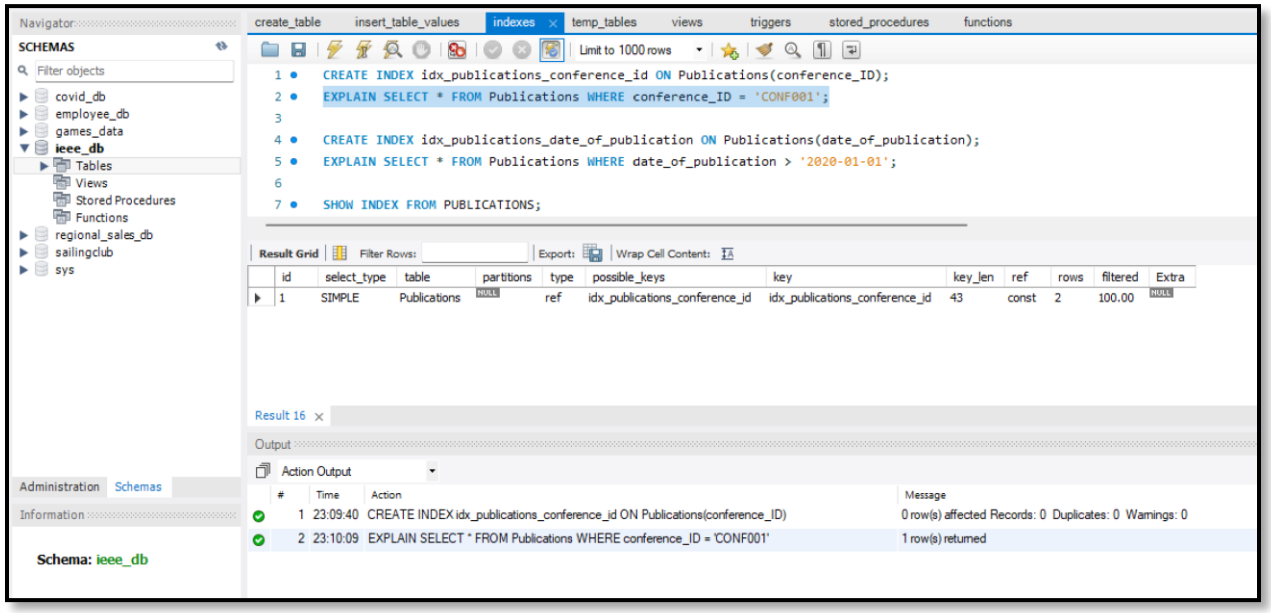
Indexes were implemented to optimize query performance, particularly for frequently queried columns such as `publication_ID`, `author_ID`, and `keyword_ID`.

File Name: indexes.sql

Example of index creation:

```
CREATE INDEX idx_publications_conference_id ON Publications(conference_ID);

EXPLAIN SELECT * FROM Publications WHERE conference_ID = 'CONF001';


CREATE INDEX idx_publications_date_of_publication ON Publications(date_of_publication);

EXPLAIN SELECT * FROM Publications WHERE date_of_publication > '2020-01-01';


SHOW INDEX FROM PUBLICATIONS;
```





1.5 Temporary Table Implementation

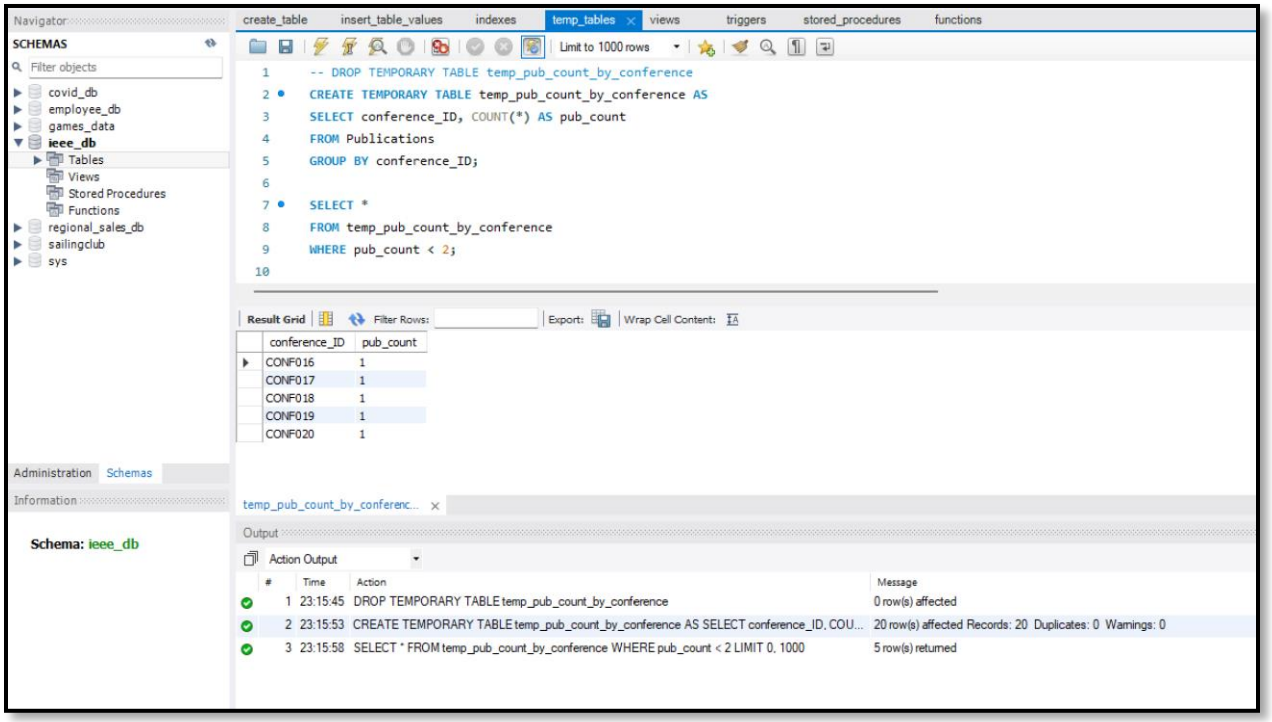
Temporary tables were used to transform data for intermediate results, such as aggregating publication counts for each conference.

File Name: temp_tables.sql

Example of temporary table creation:

```
CREATE TEMPORARY TABLE temp_pub_count_by_conference AS
SELECT conference_ID, COUNT(*) AS pub_count
FROM Publications
GROUP BY conference_ID;
```

```
SELECT *
FROM temp_pub_count_by_conference
WHERE pub_count < 2;
```



1.6 Views Implementation

Views were created to simplify complex queries, such as listing all active users and their associated institutions or identifying popular keywords.

File Name: views.sql

Example of a view:

CREATE VIEW active_users_contributions AS

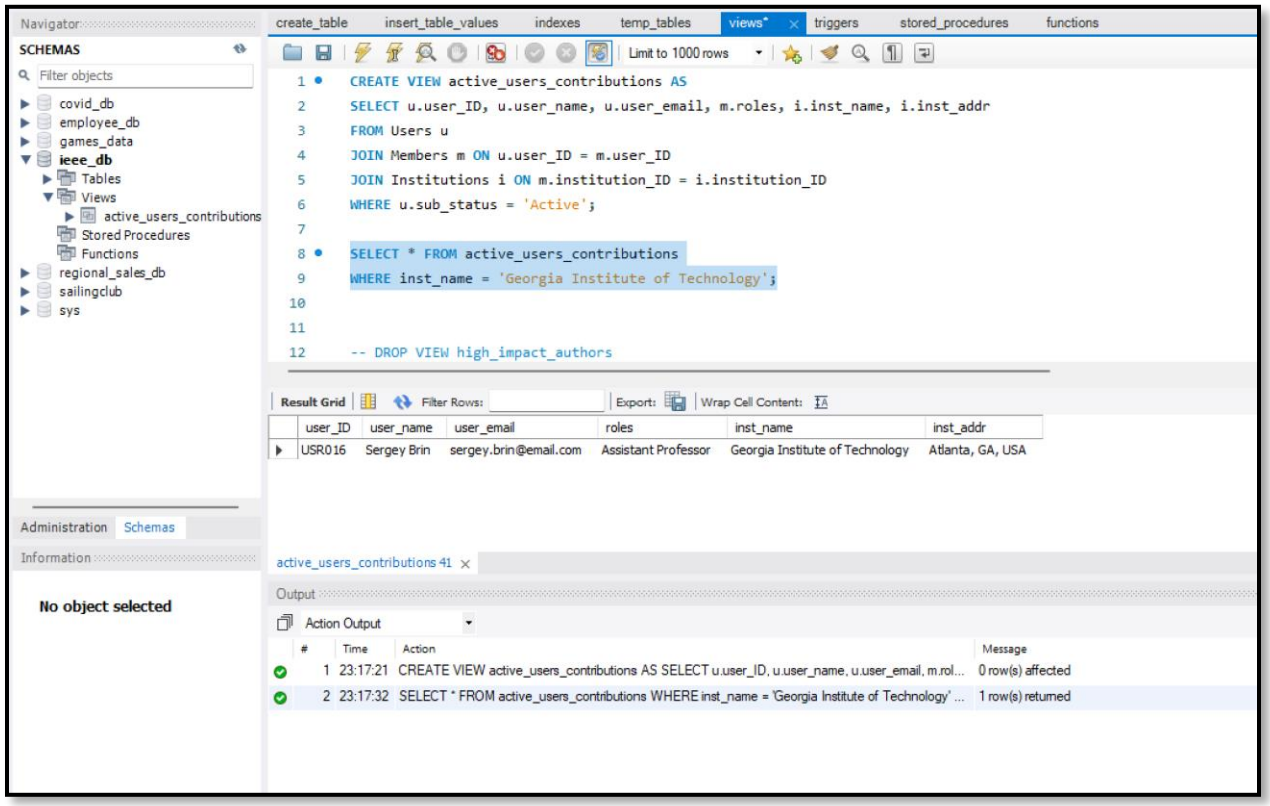
SELECT u.user_ID, u.user_name, u.user_email, m.roles, i.inst_name, i.inst_addr

FROM Users u

JOIN Members m ON u.user_ID = m.user_ID

JOIN Institutions i ON m.institution_ID = i.institution_ID

```
WHERE u.sub_status = 'Active';


SELECT * FROM active_users_contributions
WHERE inst_name = 'Georgia Institute of Technology';
```



1.7 Triggers Implementation

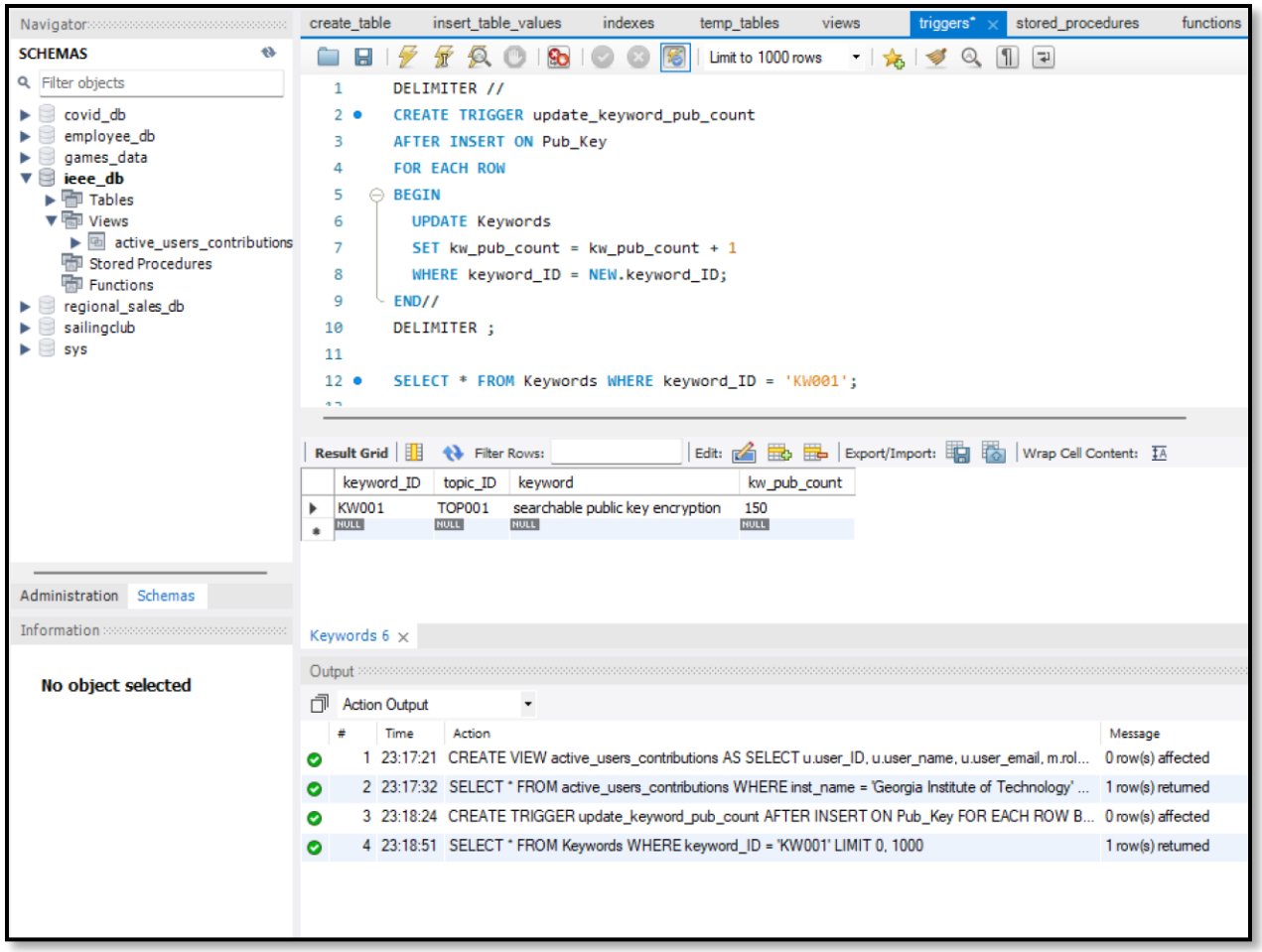
Triggers were designed to maintain data integrity, such as automatically updating citation counts when a new citation is added.

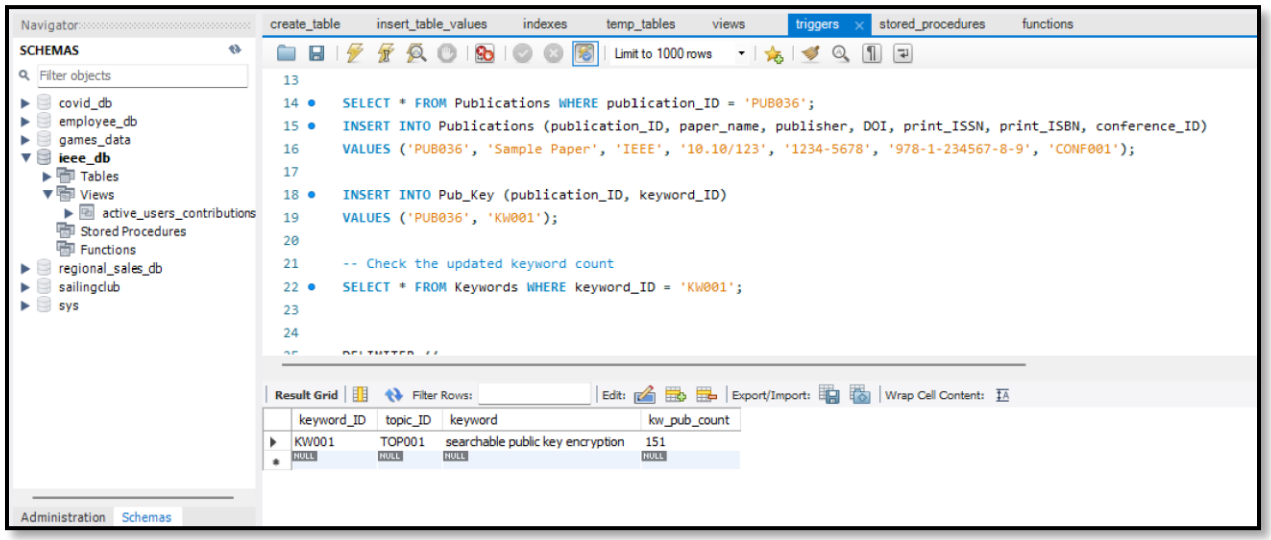
File Name: triggers.sql

Example of Trigger Creation:

```
DELIMITER //
```

```
CREATE TRIGGER prevent_duplicate_email
BEFORE INSERT ON Users
FOR EACH ROW
BEGIN
  IF EXISTS (SELECT 1 FROM Users WHERE user_email = NEW.user_email) THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Duplicate email not allowed.';
  END IF;
END//
DELIMITER ;


-- First insert a user with a unique email
INSERT INTO Users (user_ID, user_name, user_email, user_contact_no, sub_status,
subscription, profession)
VALUES ('USR031', 'John Doe', 'john.doe@example.com', '3284108410', 'Active', 'Premium',
'Engineer');


-- Attempt to insert a duplicate email
INSERT INTO Users (user_ID, user_name, user_email, user_contact_no, sub_status,
subscription, profession)
VALUES ('USR032', 'John Doe', 'john.doe@example.com', '9182374657', 'Active', 'Basic',
'Scientist');
```


Navigator
SCHEMAS
Filter objects
covid_db
employee_db
games_data
ieee_db
Tables
Views
active_users_contributions
Stored Procedures
Functions
regional_sales_db
sailingclub
sys
Administration
Schemas
Information
No object selected
create_table
insert_table_values
indexes
temp_tables
views
triggers*
stored_procedures
functions
Limit to 1000 rows
1 DELIMITER //
2 CREATE TRIGGER update_keyword_pub_count
3 AFTER INSERT ON Pub_Key
4 FOR EACH ROW
5 BEGIN
6 UPDATE Keywords
7 SET kw_pub_count = kw_pub_count + 1
8 WHERE keyword_ID = NEW.keyword_ID;
9 END//
10 DELIMITER ;
11
12 SELECT * FROM Keywords WHERE keyword_ID = 'KW001';
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
keyword_ID topic_ID keyword kw_pub_count
KW001 TOP001 searchable public key encryption 150
NULL NULL NULL NULL
Keywords 6
Output
Action Output
Time Action Message
1 23:17:21 CREATE VIEW active_users_contributions AS SELECT u.user_ID, u.user_name, u.user_email, m.rol... 0 row(s) affected
2 23:17:32 SELECT * FROM active_users_contributions WHERE inst_name = 'Georgia Institute of Technology' ... 1 row(s) returned
3 23:18:24 CREATE TRIGGER update_keyword_pub_count AFTER INSERT ON Pub_Key FOR EACH ROW B... 0 row(s) affected
4 23:18:51 SELECT * FROM Keywords WHERE keyword_ID = 'KW001' LIMIT 0, 1000 1 row(s) returned


Navigator
SCHEMAS
Filter objects
covid_db
employee_db
games_data
ieee_db
Tables
Views
active_users_contributions
Stored Procedures
Functions
regional_sales_db
sailingclub
sys
Administration
Schemas
create_table
insert_table_values
indexes
temp_tables
views
triggers
stored_procedures
functions
Limit to 1000 rows
13
14 SELECT * FROM Publications WHERE publication_ID = 'PUB036';
15 INSERT INTO Publications (publication_ID, paper_name, publisher, DOI, print_ISSN, print_ISBN, conference_ID)
16 VALUES ('PUB036', 'Sample Paper', 'IEEE', '10.10/123', '1234-5678', '978-1-234567-8-9', 'CONF001');
17
18 INSERT INTO Pub_Key (publication_ID, keyword_ID)
19 VALUES ('PUB036', 'KW001');
20
21 -- Check the updated keyword count
22 SELECT * FROM Keywords WHERE keyword_ID = 'KW001';
23
24
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
keyword_ID topic_ID keyword kw_pub_count
KW001 TOP001 searchable public key encryption 151
NULL NULL NULL NULL

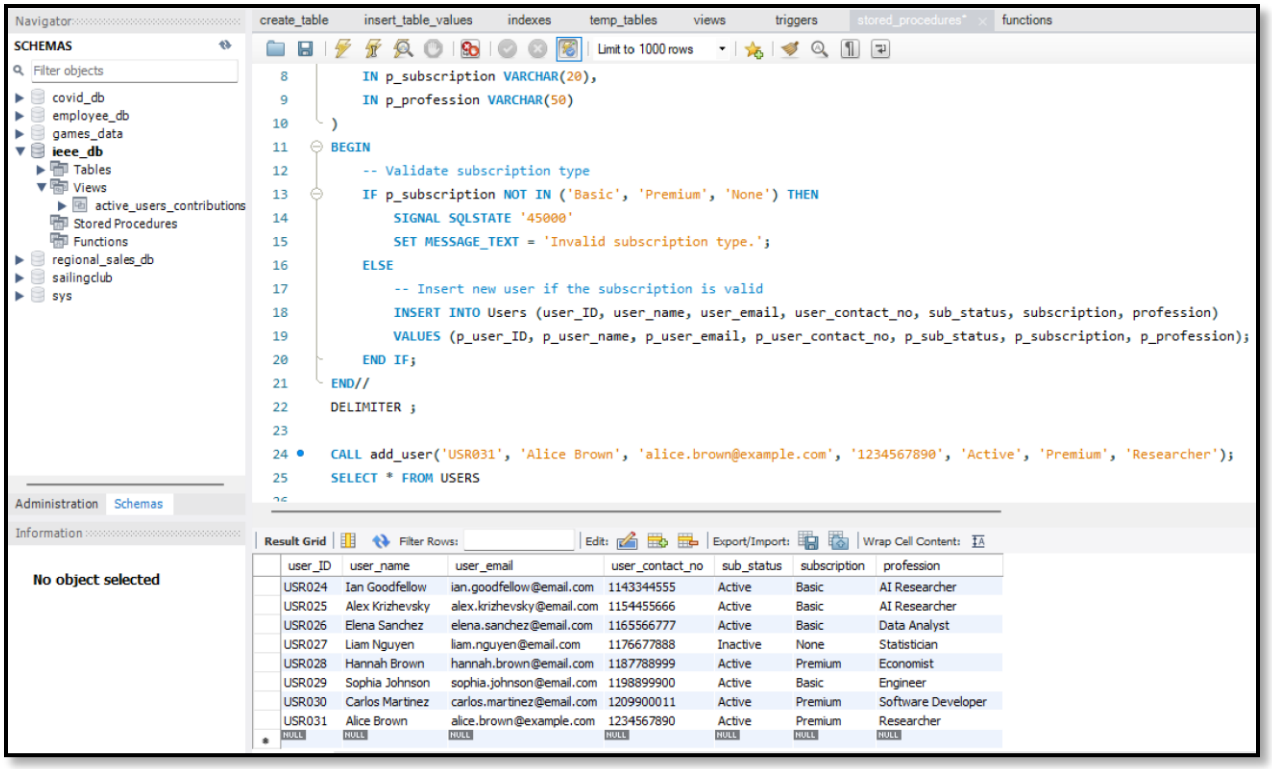
1.8 Stored Procedures Implementation

Stored procedures were implemented to handle common tasks such as adding a new publication or author, which automates tasks and reduces redundancy.

File Name: stored_procedures.sql

Example of Stored Procedure:

```
DELIMITER //

CREATE PROCEDURE get_author_metrics(
   IN p_author_ID VARCHAR(10)
)
BEGIN
   -- Retrieve the author's metrics
   SELECT a.author_name, COUNT(pa.publication_ID) AS num_publications,
        SUM(c.citation_count) AS total_citations
   FROM Authors a
   JOIN Pub_Author pa ON a.author_ID = pa.author_ID
   JOIN Pub_Cite pc ON pa.publication_ID = pc.publication_ID
   JOIN Citations c ON pc.citation_ID = c.citation_ID
   WHERE a.author_ID = p_author_ID
   GROUP BY a.author_name;
END//

DELIMITER ;

CALL get_author_metrics('AUTH002');
```



1.9 Functions Implementation

Functions were created to return specific values like total citations for an author.

File Name: functions.sql

Example of Function Creation:

```
DELIMITER //

CREATE FUNCTION get_author_total_citations(p_author_ID VARCHAR(10))
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE total_citations INT;
```

```
    SELECT SUM(c.citation_count)
    INTO total_citations
    FROM Pub_Author pa
    JOIN Pub_Cite pc ON pa.publication_ID = pc.publication_ID
    JOIN Citations c ON pc.citation_ID = c.citation_ID
    WHERE pa.author_ID = p_author_ID;

    RETURN IFNULL(total_citations, 0);  -- Return 0 if no citations exist
END//

DELIMITER ;

SELECT get_author_total_citations('AUTH003') AS total_citations;
```