QUALITATIVE ASSESSMENT -4

NAME : -KOWSHIK G

ROLL NO:- 25MSIC107

SUBJECT :-JAVA

GITHUB PROFILE:- https://github.com/kowshik28-G

GITHUB LINK:- https://github.com/kowshik28-G/java_unit4

TITLE: PERSONAL EXPENSE TRACKER USING SQLITE

1.Introduction

The Personal Expense Tracker is a database-based mini project developed using SQLite.

It helps users store, manage, and analyze their daily expenses.

The system stores details about course, college, and student and

uses SQL queries to generate useful reports.

This project demonstrates the use of:

- Table creation
- Foreign keys
- Insert operations
- JOIN queries
- Aggregate functions

The ER diagram represents the logical structure of the Personal Expense Tracker database .It shows the entities, their attributes, primary keys, foreign keys, and relationships between the tables.

This system contains three main entities:

- **COURSE**

ATTRIBUTE	**DESCRIPTION**
course_id (PK)	Unique ID for each course
Name	Name of the course
email	Email address

- **COLLEGE**

ATTRIBUTE	DESCRIPTION
college_id (PK)	Unique ID for each college
college_Name	Name of the college

- **STUDENT**

ATTRIBUTE	DESCRIPTION
student_id (PK)	Unique ID for each student
date	Date of student
Amount	student amount
description	Purpose of student
college_id(FK)	Links to college table
course_id(FK)	Links to course table

RELATIONSHIP	DESCRIPTION
course-> student	One course can have many student
college -> student	One college can have many student

So:

One-to-Many between course
and student One-to-Many
between college and student

2. ER Diagram

COURSE (course_id, name, email)

|

| 1

|

| M

STUDENT (student_id, date, amount,
description, college_id, course_id)

|

| M

|

| 1

college (college_id, college_name)

3. Explanation

Each student must belong to:

- One course
- One college

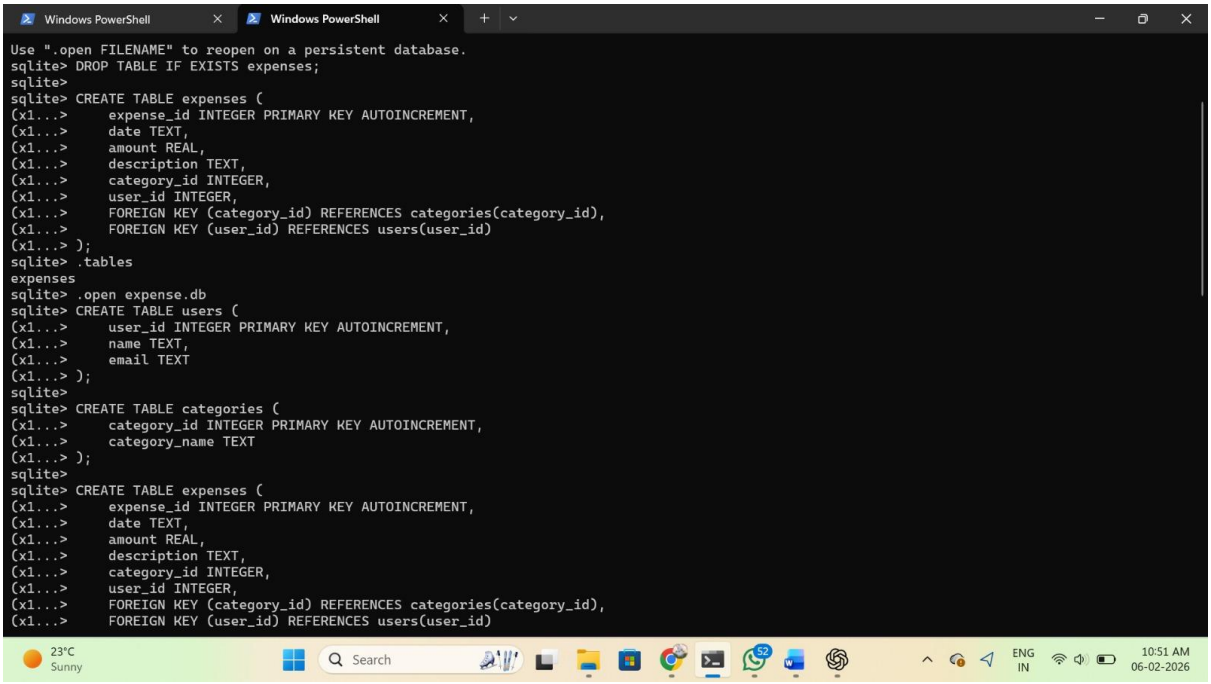
This design avoids duplication and maintains data consistency using foreign keys.

QUERIES

CREATE A TABLE

```
CREATE TABLE student (
   student_id INTEGER PRIMARY KEY AUTOINCREMENT,
   date TEXT,
   amount
   REAL,
   description
   TEXT,
   college_id
   INTEGER,
   course_id
   INTEGER,
   FOREIGN KEY (college_id) REFERENCES
   categories(category_id), FOREIGN KEY (course_id)
   REFERENCES course(course_id)
);
```

OUTPUT:




```
CREATE TABLE users (
    course_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name
    TEXT,
    email
    TEXT
);
CREATE TABLE college (
    college_id INTEGER PRIMARY KEY AUTOINCREMENT,
    college_name TEXT
);
```

```
CREATE TABLE student (
   student_id INTEGER PRIMARY KEY AUTOINCREMENT,
   date TEXT,
   amount
   REAL,
   description
   TEXT,
   college_id
   INTEGER,
   course_id
   INTEGER,
   FOREIGN KEY (college_id) REFERENCES
   college(college_id), FOREIGN KEY (course_id)
   REFERENCES course(course_id)
);
```

INSERTING DATA:

```
INSERT INTO course (course_id, name,
email) VALUES (1, 'kowshik',
'kowshik104@gmail.com');
INSERT INTO college (college_name) VALUES
('Food'); INSERT INTO college (college_name)
VALUES ('Travel');
INSERT INTO college (college_name) VALUES ('Shopping');
```

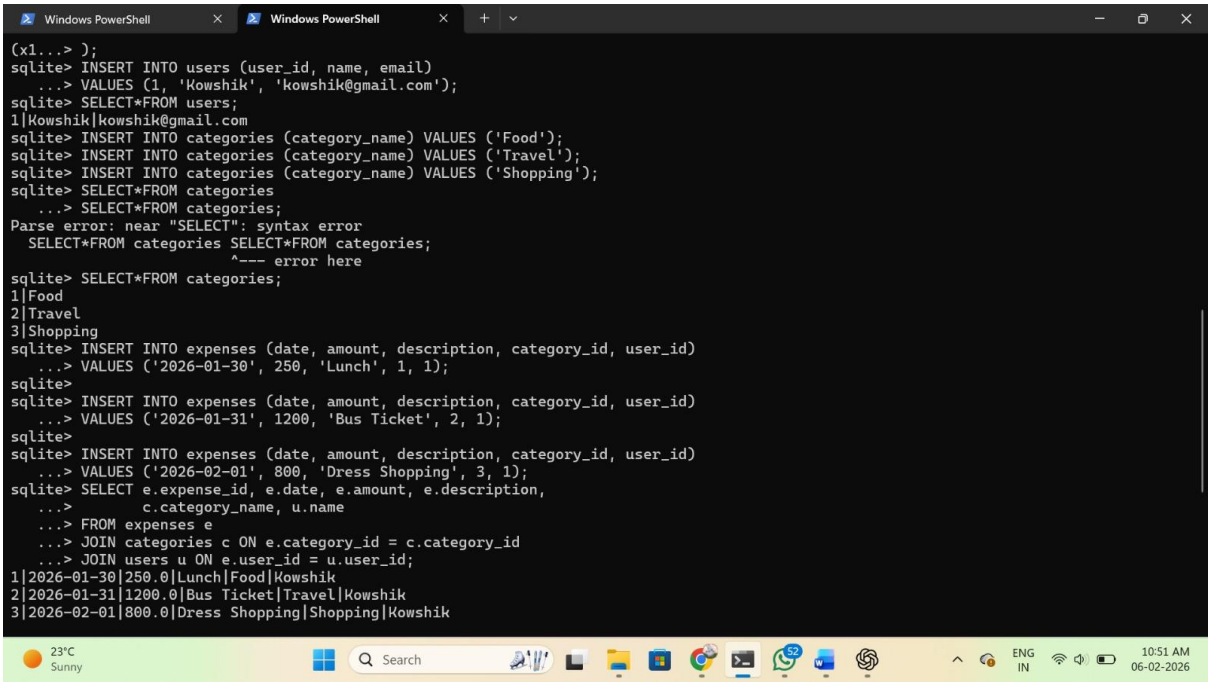
```
INSERT INTO student (date, amount, description,
college_id,course_id) VALUES ('2026-01-30', 250, 'Lunch',
1, 1);
INSERT INTO student (date, amount, description,
college_id, course_id) VALUES ('2026-01-31', 1200, 'Bus
Ticket', 2, 1);
INSERT INTO student (date, amount, description,
college_id, course_id) VALUES ('2026-02-01', 800, 'Dress
Shopping', 3, 1);
SELECT e.student_id, e.date, e.amount,
e.description, c.college_name, u.name
FROM student e
JOIN college c ON e.college_id =
c.college_id JOIN course u ON
e.course_id = u.course_id; SELECT
u.name, SUM(e.amount) AS total_student
FROM student e
JOIN college u ON e.course_id =
u.course_id GROUP BY u.name;
SELECT c.college_name,
SUM(e.amount) AS total FROM student
e
JOIN college c ON e.college_id =
c.college_id GROUP BY c.college_name;
SELECT description, MAX(amount) AS
```

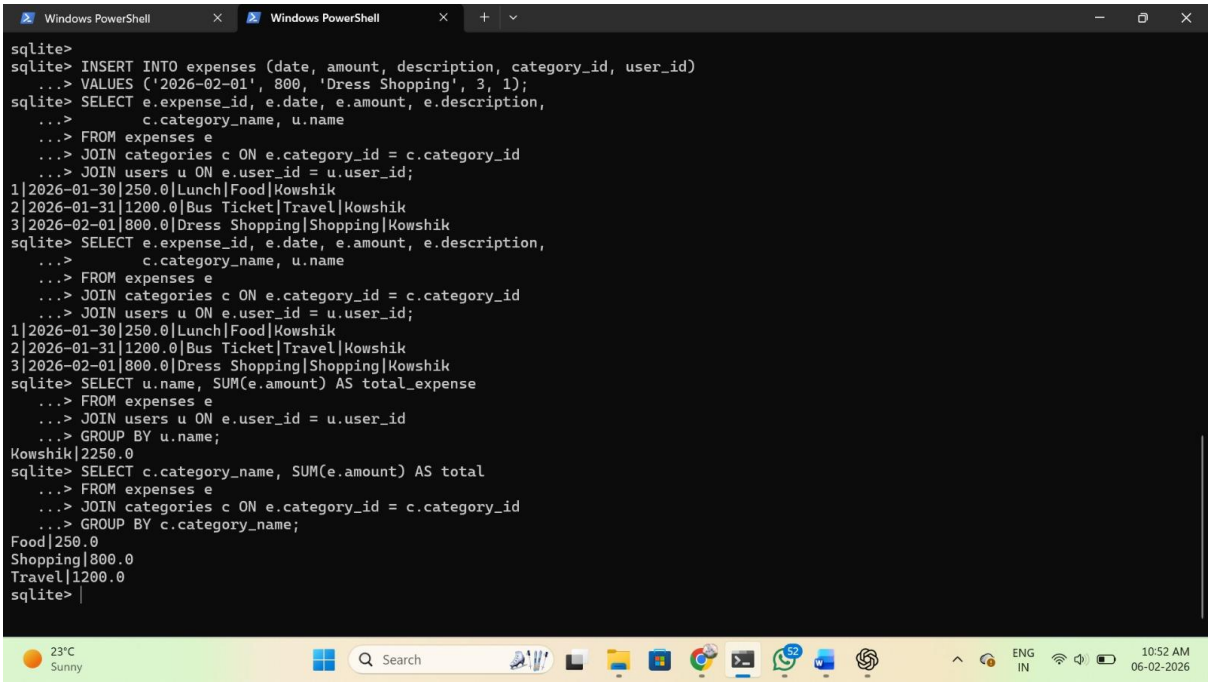
highest_amount FROM student;

SELECT substr(date,1,7) AS month, SUM(amount)

AS total FROM student

GROUP BY month;





CONCLUSION:

The Personal Expense Tracker using SQLite is a simple and effective database project that helps users store and manage their daily expenses.

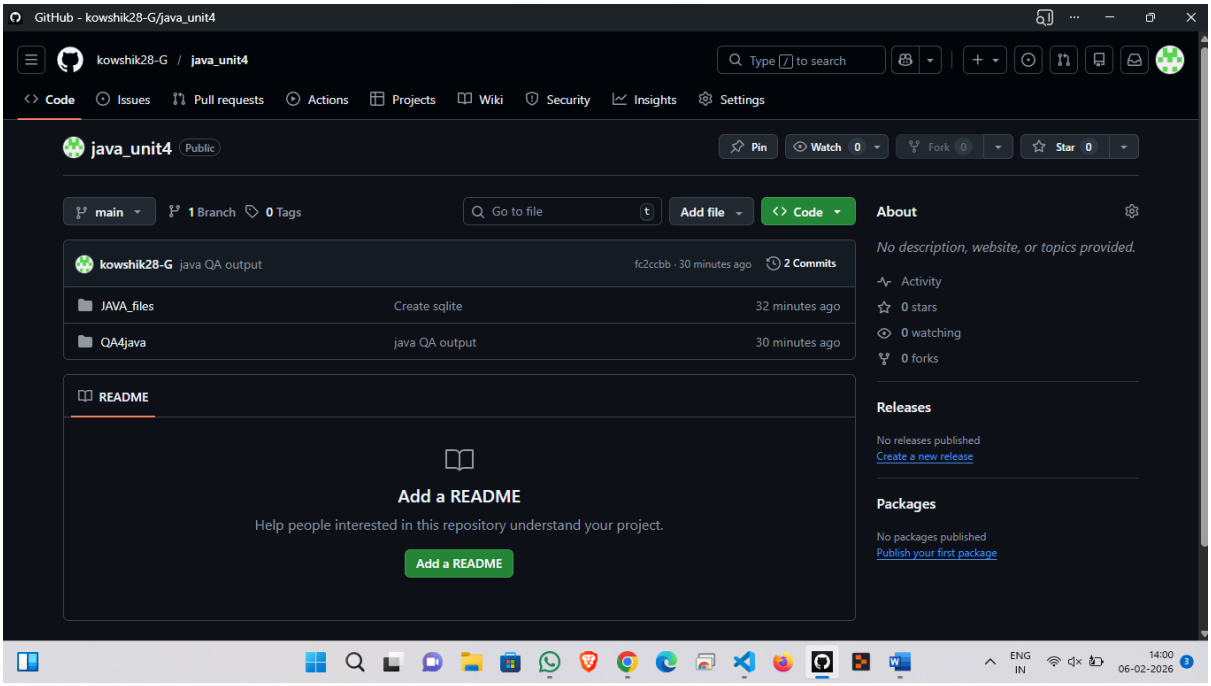
It uses relational database concepts such as primary keys, foreign keys, joins, and aggregate functions.

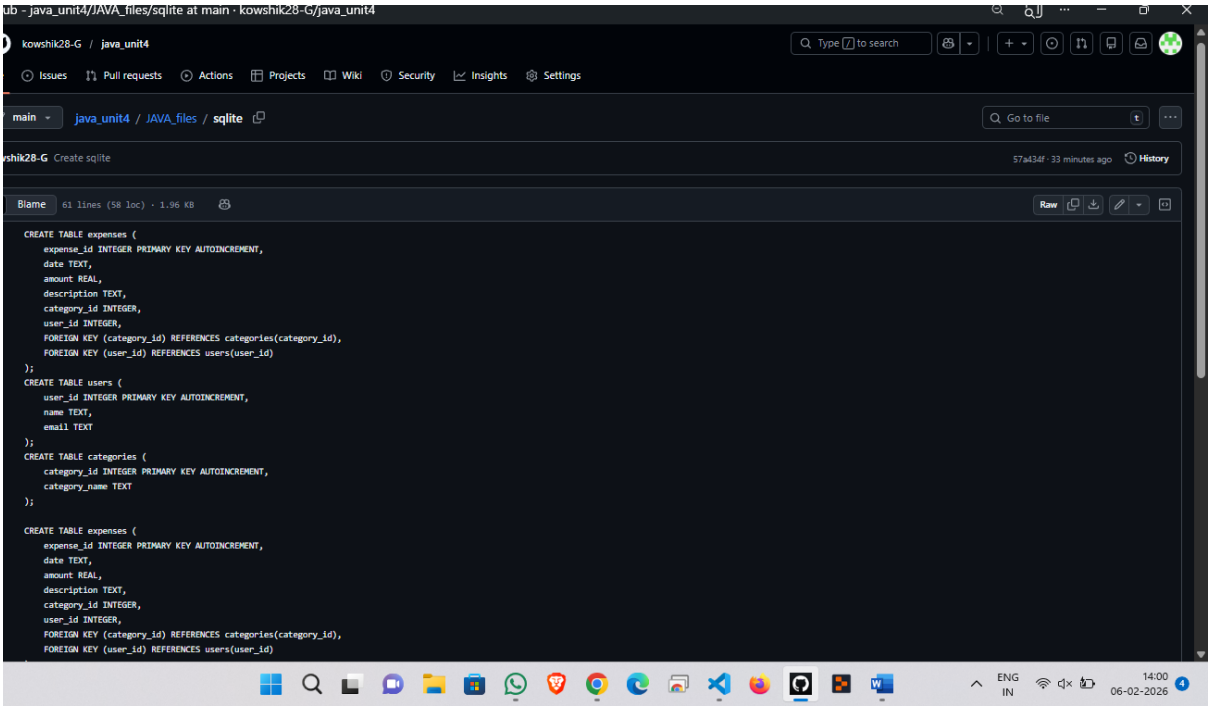
This project successfully demonstrates:

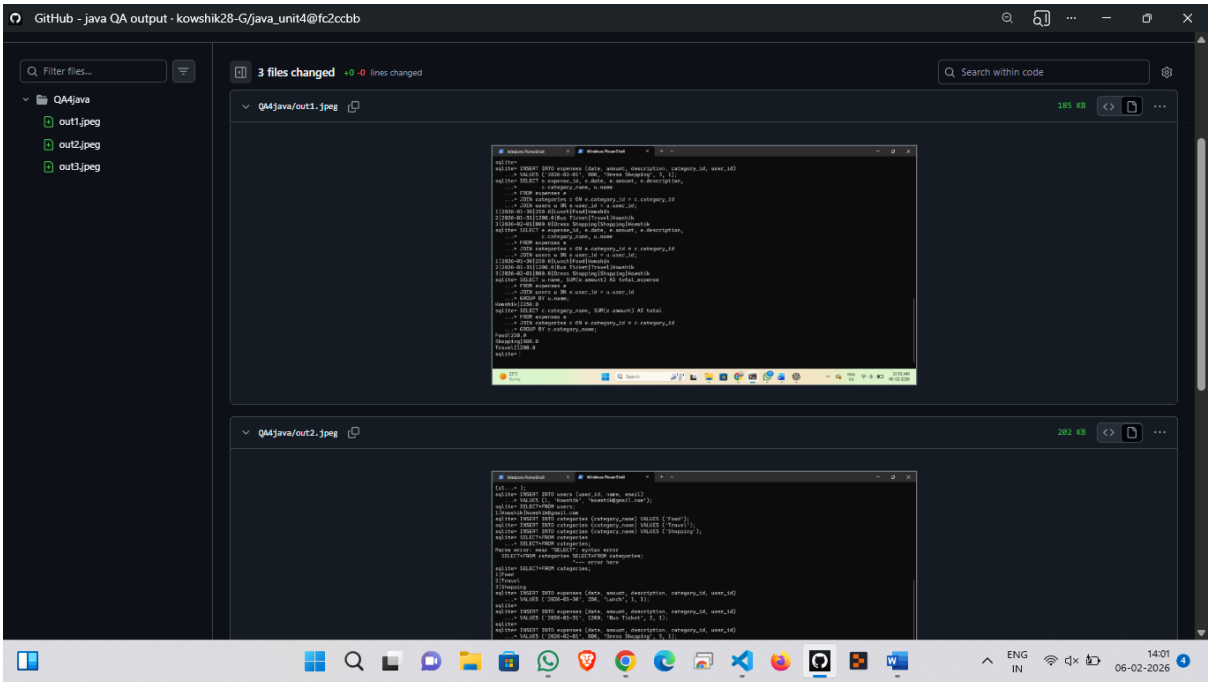
- Table creation
- Data insertion
- Data retrieval
- Report generation

The system is efficient, user-friendly, and easy to maintain

GITHUB:-


GitHub - kowshik28-G/java_unit4
kowshik28-G / java_unit4
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
java_unit4
Public
Pin
Watch 0
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
About
No description, website, or topics provided.
Activity
0 stars
0 watching
0 forks
kowshik28-G java QA output
fc2ccbb · 30 minutes ago
2 Commits
JAVA_files
Create sqlite
32 minutes ago
QA4java
java QA output
30 minutes ago
README
Add a README
Help people interested in this repository understand your project.
Add a README
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
ENG
IN
14:00
06-02-2026


ub - java_unit4/JAVA_files/sqlite at main · kowshik28-G/java_unit4
kowshik28-G / java_unit4
Type / to search
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
java_unit4 / JAVA_files / sqlite
Go to file
shik28-G Create sqlite
57a434f · 33 minutes ago
History
Blame
61 lines (58 loc) · 1.96 KB
Raw
CREATE TABLE expenses (
expense_id INTEGER PRIMARY KEY AUTOINCREMENT,
date TEXT,
amount REAL,
description TEXT,
category_id INTEGER,
user_id INTEGER,
FOREIGN KEY (category_id) REFERENCES categories(category_id),
FOREIGN KEY (user_id) REFERENCES users(user_id)
);
CREATE TABLE users (
user_id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT,
email TEXT
);
CREATE TABLE categories (
category_id INTEGER PRIMARY KEY AUTOINCREMENT,
category_name TEXT
);
CREATE TABLE expenses (
expense_id INTEGER PRIMARY KEY AUTOINCREMENT,
date TEXT,
amount REAL,
description TEXT,
category_id INTEGER,
user_id INTEGER,
FOREIGN KEY (category_id) REFERENCES categories(category_id),
FOREIGN KEY (user_id) REFERENCES users(user_id)
ENG
IN
14:00
06-02-2026


GitHub - java QA output - kowshik28-G/java_unit4@fc2ccbb
Filter files...
QA4java
out1.jpeg
out2.jpeg
out3.jpeg
3 files changed +0 -0 lines changed
Search within code
QA4java/out1.jpeg
185 KB
QA4java/out2.jpeg
202 KB
ENG
IN
14:01
06-02-2026