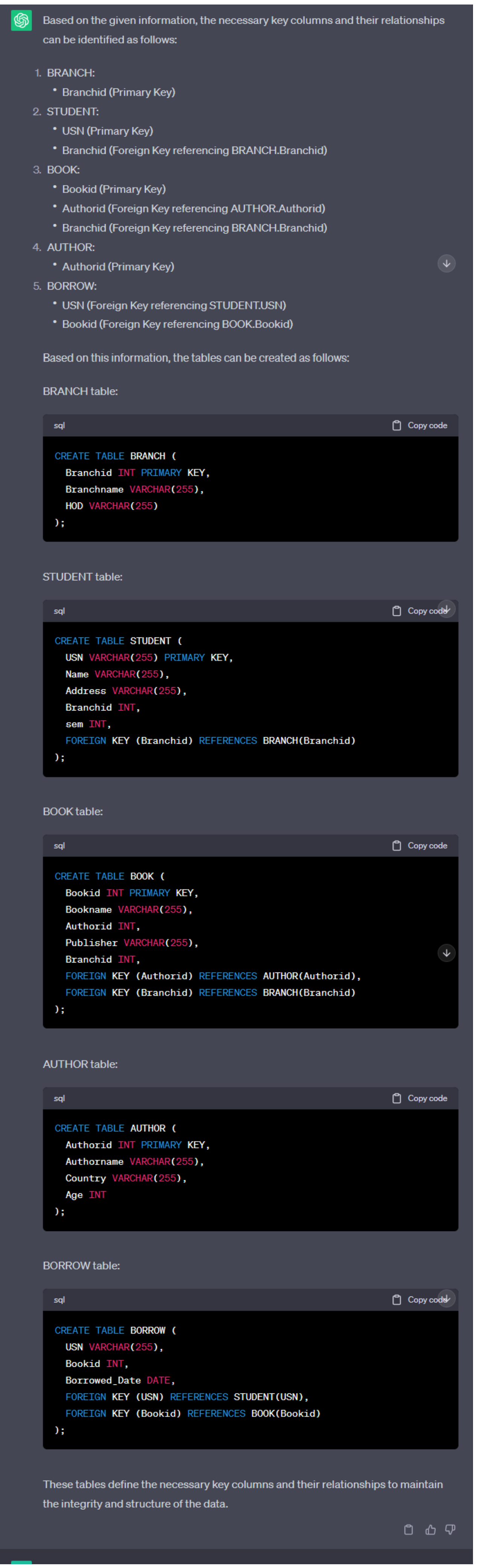
DBMS LAB

1	**Create the following tables with properly specifying Primary keys, Foreign keys and solve the following queries.** • BRANCH (Branchid, Branchname, HOD) • STUDENT (USN, Name, Address, Branchid, sem) • BOOK (Bookid, Bookname, Authorid, Publisher, Branchid) • AUTHOR (Authorid, Authorname, Country, age) • BORROW (USN, Bookid, Borrowed_Date) **Execute the following Queries:** i. List the details of Students who are all studying in 2nd sem MCA. ii. List the students who are not borrowed any books. iii. Display the USN, Student name, Branch_name, Book_name, Author_name, Books_Borrowed_Date of 2nd sem MCA Students who borrowed books. iv. Display the number of books written by each Author. v. Display the student details who borrowed more than two books. vi. Display the student details who borrowed books of more than one Author. vii. Display the Book names in descending order of their names. viii. List the details of students who borrowed the books which are all published by the same publisher.
2	**Consider the following schema:** STUDENT (USN, name, date_of_birth, branch, mark1, mark2, mark3, total, GPA) **Execute the following queries:** i. Update the column total by adding the columns mark1, mark2, mark3. ii. Find the GPA score of all the students. iii. Find the students who born on a particular year of birth from the date_of_birth column. iv. List the students who are studying in a particular branch of study. v. Find the maximumGPA score of the student branch-wise. vi. Find the students whose name starts with the alphabet "S". vii. Findthe students whose name ends with the alphabets "AR". viii. Delete the student details whose USN is given as 1001

SOLUTION 1:

TABLE CREATION



Based on the given information, the necessary key columns and their relationships can be identified as follows:

1. BRANCH:
   - Branchid (Primary Key)
2. STUDENT:
   - USN (Primary Key)
   - Branchid (Foreign Key referencing BRANCH.Branchid)
3. BOOK:
   - Bookid (Primary Key)
   - Authorid (Foreign Key referencing AUTHOR.Authorid)
   - Branchid (Foreign Key referencing BRANCH.Branchid)
4. AUTHOR:
   - Authorid (Primary Key)
5. BORROW:
   - USN (Foreign Key referencing STUDENT.USN)
   - Bookid (Foreign Key referencing BOOK.Bookid)

Based on this information, the tables can be created as follows:

BRANCH table:

```sql
CREATE TABLE BRANCH (
  Branchid INT PRIMARY KEY,
  Branchname VARCHAR(255),
  HOD VARCHAR(255)
);
```

STUDENT table:

```sql
CREATE TABLE STUDENT (
  USN VARCHAR(255) PRIMARY KEY,
  Name VARCHAR(255),
  Address VARCHAR(255),
  Branchid INT,
  sem INT,
  FOREIGN KEY (Branchid) REFERENCES BRANCH(Branchid)
);
```

BOOK table:

```sql
CREATE TABLE BOOK (
  Bookid INT PRIMARY KEY,
  Bookname VARCHAR(255),
  Authorid INT,
  Publisher VARCHAR(255),
  Branchid INT,
  FOREIGN KEY (Authorid) REFERENCES AUTHOR(Authorid),
  FOREIGN KEY (Branchid) REFERENCES BRANCH(Branchid)
);
```

AUTHOR table:

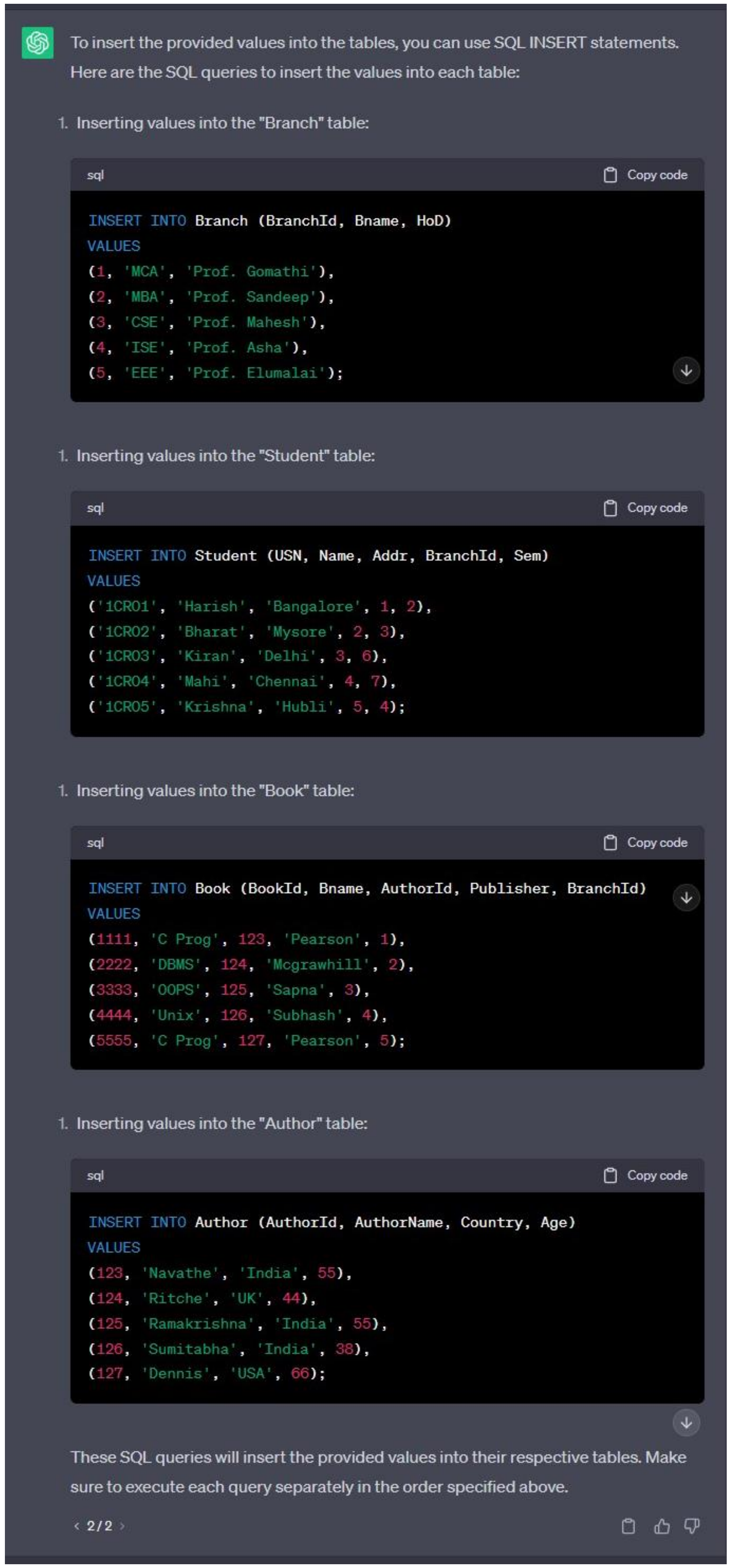
```sql
CREATE TABLE AUTHOR (
  Authorid INT PRIMARY KEY,
  Authorname VARCHAR(255),
  Country VARCHAR(255),
  Age INT
);
```

BORROW table:

```sql
CREATE TABLE BORROW (
  USN VARCHAR(255),
  Bookid INT,
  Borrowed_Date DATE,
  FOREIGN KEY (USN) REFERENCES STUDENT(USN),
  FOREIGN KEY (Bookid) REFERENCES BOOK(Bookid)
);
```

These tables define the necessary key columns and their relationships to maintain the integrity and structure of the data.

VALUES INSERTION



To insert the provided values into the tables, you can use SQL INSERT statements. Here are the SQL queries to insert the values into each table:

1. Inserting values into the "Branch" table:

```sql
INSERT INTO Branch (BranchId, Bname, HoD)
VALUES
(1, 'MCA', 'Prof. Gomathi'),
(2, 'MBA', 'Prof. Sandeep'),
(3, 'CSE', 'Prof. Mahesh'),
(4, 'ISE', 'Prof. Asha'),
(5, 'EEE', 'Prof. Elumalai');
```

1. Inserting values into the "Student" table:

```sql
INSERT INTO Student (USN, Name, Addr, BranchId, Sem)
VALUES
('1CR01', 'Harish', 'Bangalore', 1, 2),
('1CR02', 'Bharat', 'Mysore', 2, 3),
('1CR03', 'Kiran', 'Delhi', 3, 6),
('1CR04', 'Mahi', 'Chennai', 4, 7),
('1CR05', 'Krishna', 'Hubli', 5, 4);
```

1. Inserting values into the "Book" table:

```sql
INSERT INTO Book (BookId, Bname, AuthorId, Publisher, BranchId)
VALUES
(1111, 'C Prog', 123, 'Pearson', 1),
(2222, 'DBMS', 124, 'Mcgrawhill', 2),
(3333, 'OOPS', 125, 'Sapna', 3),
(4444, 'Unix', 126, 'Subhash', 4),
(5555, 'C Prog', 127, 'Pearson', 5);
```

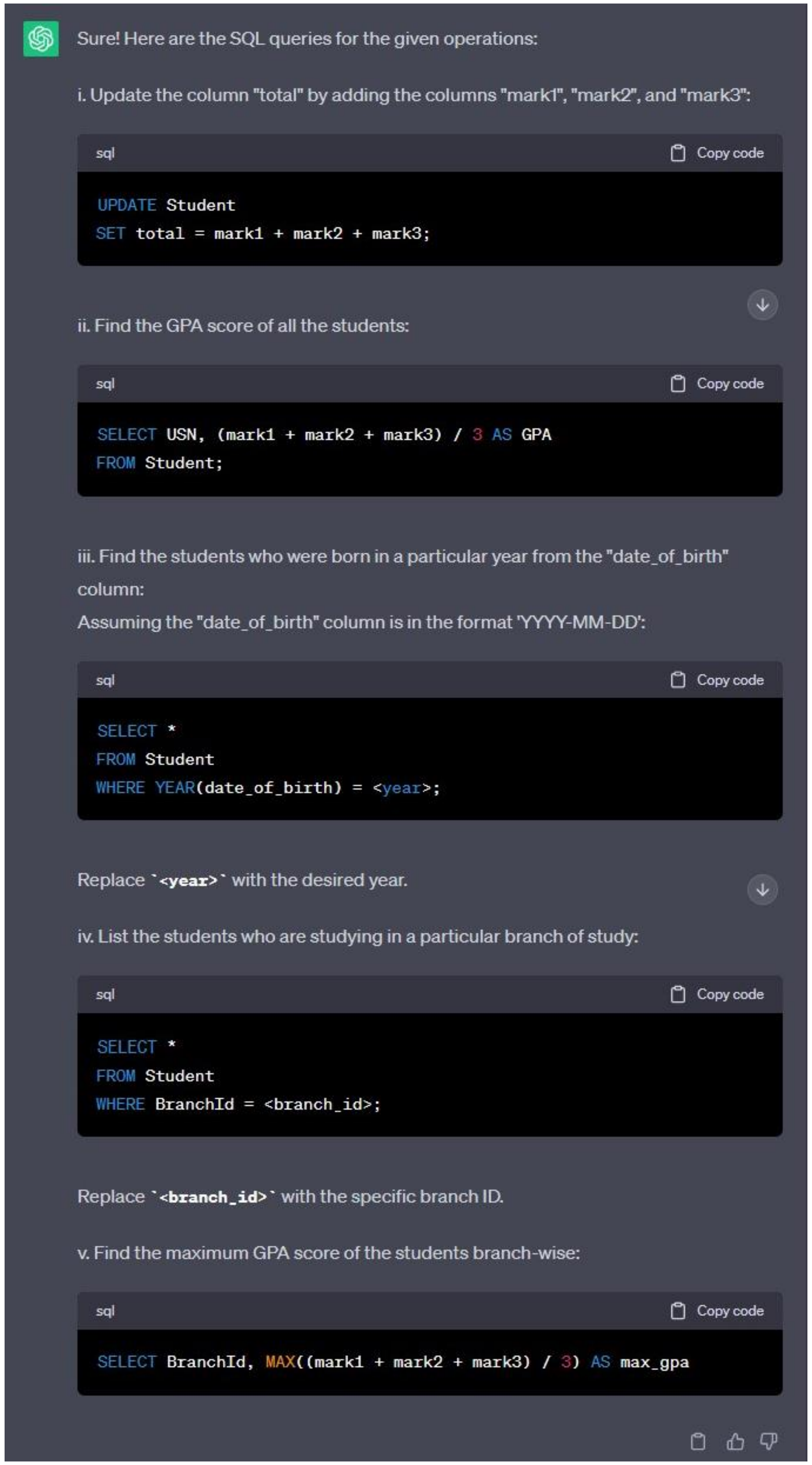
1. Inserting values into the "Author" table:

```sql
INSERT INTO Author (AuthorId, AuthorName, Country, Age)
VALUES
(123, 'Navathe', 'India', 55),
(124, 'Ritche', 'UK', 44),
(125, 'Ramakrishna', 'India', 55),
(126, 'Sumitabha', 'India', 38),
(127, 'Dennis', 'USA', 66);
```

These SQL queries will insert the provided values into their respective tables. Make sure to execute each query separately in the order specified above.

‹ 2 / 2 ›

QUERIES IMPLEMENTATION



Sure! Here are the SQL queries for the given operations:

i. Update the column "total" by adding the columns "mark1", "mark2", and "mark3":

```sql
UPDATE Student
SET total = mark1 + mark2 + mark3;
```

ii. Find the GPA score of all the students:

```sql
SELECT USN, (mark1 + mark2 + mark3) / 3 AS GPA
FROM Student;
```

iii. Find the students who were born in a particular year from the "date_of_birth" column:
Assuming the "date_of_birth" column is in the format 'YYYY-MM-DD':

```sql
SELECT *
FROM Student
WHERE YEAR(date_of_birth) = <year>;
```

Replace `<year>` with the desired year.

iv. List the students who are studying in a particular branch of study:

```sql
SELECT *
FROM Student
WHERE BranchId = <branch_id>;
```

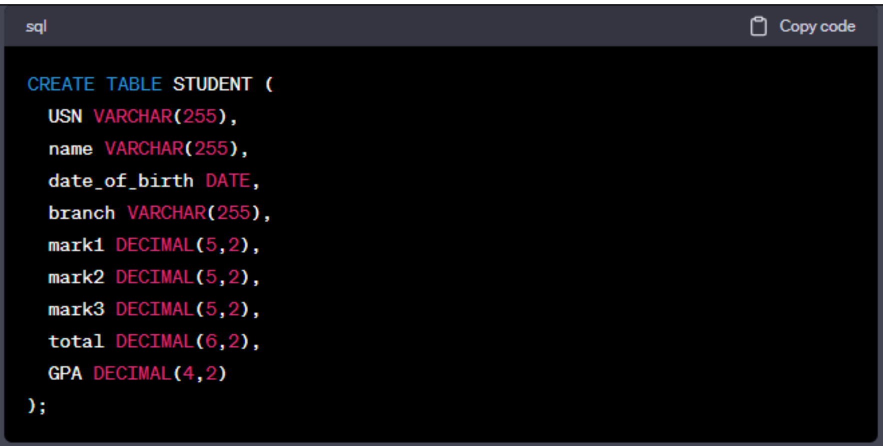
Replace `<branch_id>` with the specific branch ID.

v. Find the maximum GPA score of the students branch-wise:

```sql
SELECT BranchId, MAX((mark1 + mark2 + mark3) / 3) AS max_gpa
```

SOLUTION 2

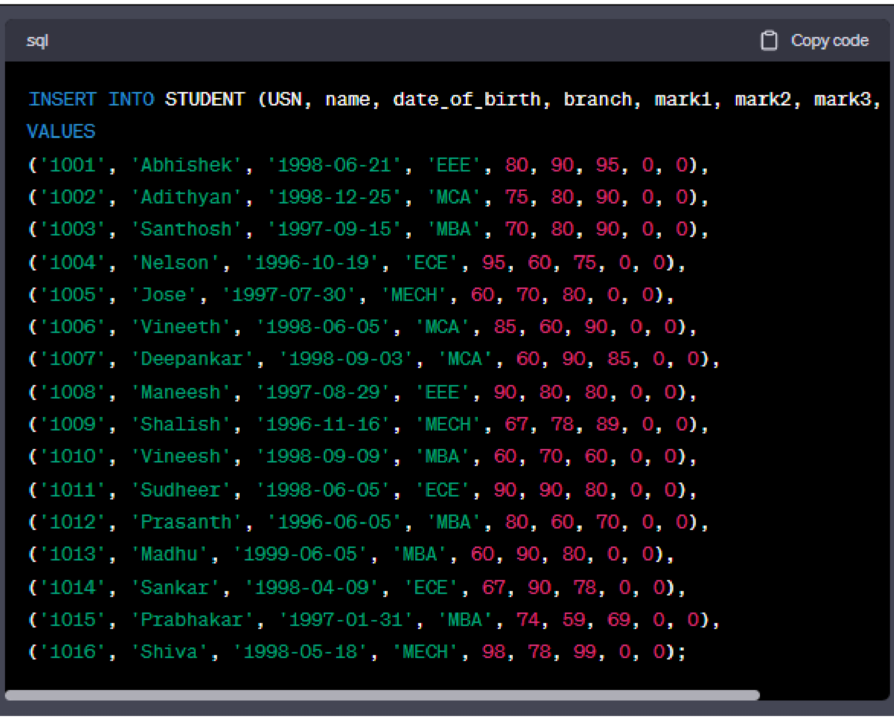
TABE CREATON



sql Copy code

```sql
CREATE TABLE STUDENT (
  USN VARCHAR(255),
  name VARCHAR(255),
  date_of_birth DATE,
  branch VARCHAR(255),
  mark1 DECIMAL(5,2),
  mark2 DECIMAL(5,2),
  mark3 DECIMAL(5,2),
  total DECIMAL(6,2),
  GPA DECIMAL(4,2)
);
```

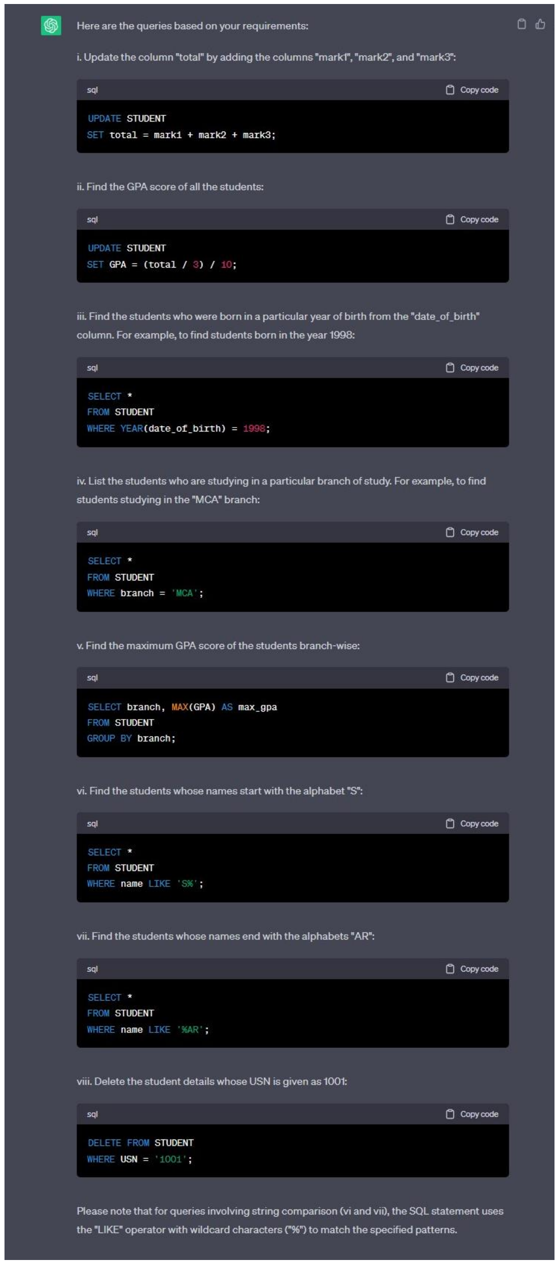
VALUES INSERTION



sql Copy code

```sql
INSERT INTO STUDENT (USN, name, date_of_birth, branch, mark1, mark2, mark3,
VALUES
('1001', 'Abhishek', '1998-06-21', 'EEE', 80, 90, 95, 0, 0),
('1002', 'Adithyan', '1998-12-25', 'MCA', 75, 80, 90, 0, 0),
('1003', 'Santhosh', '1997-09-15', 'MBA', 70, 80, 90, 0, 0),
('1004', 'Nelson', '1996-10-19', 'ECE', 95, 60, 75, 0, 0),
('1005', 'Jose', '1997-07-30', 'MECH', 60, 70, 80, 0, 0),
('1006', 'Vineeth', '1998-06-05', 'MCA', 85, 60, 90, 0, 0),
('1007', 'Deepankar', '1998-09-03', 'MCA', 60, 90, 85, 0, 0),
('1008', 'Maneesh', '1997-08-29', 'EEE', 90, 80, 80, 0, 0),
('1009', 'Shalish', '1996-11-16', 'MECH', 67, 78, 89, 0, 0),
('1010', 'Vineesh', '1998-09-09', 'MBA', 60, 70, 60, 0, 0),
('1011', 'Sudheer', '1998-06-05', 'ECE', 90, 90, 80, 0, 0),
('1012', 'Prasanth', '1996-06-05', 'MBA', 80, 60, 70, 0, 0),
('1013', 'Madhu', '1999-06-05', 'MBA', 60, 90, 80, 0, 0),
('1014', 'Sankar', '1998-04-09', 'ECE', 67, 90, 78, 0, 0),
('1015', 'Prabhakar', '1997-01-31', 'MBA', 74, 59, 69, 0, 0),
('1016', 'Shiva', '1998-05-18', 'MECH', 98, 78, 99, 0, 0);
```

QUERIES IMPLEMENTATION



Here are the queries based on your requirements:

i. Update the column "total" by adding the columns "mark1", "mark2", and "mark3":

sql Copy code

```sql
UPDATE STUDENT
SET total = mark1 + mark2 + mark3;
```

ii. Find the GPA score of all the students:

sql Copy code

```sql
UPDATE STUDENT
SET GPA = (total / 3) / 10;
```

iii. Find the students who were born in a particular year of birth from the "date_of_birth" column. For example, to find students born in the year 1998:

sql Copy code

```sql
SELECT *
FROM STUDENT
WHERE YEAR(date_of_birth) = 1998;
```

iv. List the students who are studying in a particular branch of study. For example, to find students studying in the "MCA" branch:

sql Copy code

```sql
SELECT *
FROM STUDENT
WHERE branch = 'MCA';
```

v. Find the maximum GPA score of the students branch-wise:

sql Copy code

```sql
SELECT branch, MAX(GPA) AS max_gpa
FROM STUDENT
GROUP BY branch;
```

vi. Find the students whose names start with the alphabet "S":

sql Copy code

```sql
SELECT *
FROM STUDENT
WHERE name LIKE 'S%';
```

vii. Find the students whose names end with the alphabets "AR":

sql Copy code

```sql
SELECT *
FROM STUDENT
WHERE name LIKE '%AR';
```

viii. Delete the student details whose USN is given as 1001:

sql Copy code

```sql
DELETE FROM STUDENT
WHERE USN = '1001';
```

Please note that for queries involving string comparison (vi and vii), the SQL statement uses the "LIKE" operator with wildcard characters ("%") to match the specified patterns.